PHOTOCOPY AND USE AUTHORIZATION

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission for extensive copying of my thesis for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature _____

Date _____

The Development of the Generalized Spallation Model

by

Chase M. Juneau

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Nuclear Science and Engineering

Idaho State University

Summer 2019

COMMITTEE APPROVAL

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of Chase M. Juneau find it satisfactory and recommend that it be accepted.

Dr. Leslie Kerby
Committee Chair

Dr. Clell Solomon
Committee Member (Los Alamos National Laboratory)

Dr. Chad Pope
Committee Member

Dr. David Beard
Graduate Faculty Representative

DEDICATION

This thesis is dedicated to my wife, Kaity.

Words cannot express the unending
love, joy, and support you have
given to me.

Thanks be to God as well for His
faithfulness and everlasting
love.

God is our refuge and strength, a very present help in trouble. Therefore we will not fear though the earth gives way, though the mountains be moved into the heart of the sea, though its waters roar and foam, though the mountains tremble at its swelling.

- Psalms 46:1-3

ACKNOWLEDGMENTS

I would like to thank my major advisor, Dr. Kerby, for her unyielding support of me and this work throughout its duration. I am confident in saying that this thesis would not have come to fruition had it not been for her.

I would like to thank Dr. Stepan Mashnik, Dr. Arnold Sierk, Dr. Konstantin Gudima, and Dr. Leslie Kerby for their thorough understanding of the multitude of complex physics that is discussed in this document. I would also like to thank Dr. Clell Solomon and Dr. Leslie Kerby for their insights and direction regarding how the GSM was to be engineered.

This work was carried out with funding provided by the Idaho State University and, in part, with funding provided by the Los Alamos National Laboratory, under Idaho State University subcontract number 385443 and with funds from the advanced simulation and computing program.

We ask readers of this thesis to contact the author[1] using the e-mail addresses[2] provided in case of questions, comments, or discovered "bugs." I thank them in advance for comments and information about possible problems in using this code and for any information given on bugs that may be discovered.

[1] junechas@isu.edu

[2] Please *cc* Dr. Leslie Kerby (*kerblesl@isu.edu*) and Dr. Clell Solomon (*csolomon@lanl.gov*), if possible, when inquiring about the GSM or providing feedback to ensure it is received.

Table of Contents

List of Figures . . . . . **xiii**

List of Tables . . . . . **xvi**

Obtaining the GSM Code . . . . . **xvii**

Abstract . . . . . **xviii**

I Introduction **1**

II A Review of Spallation Physics in GSM **4**

II.1 A Survey of INC Physics 5

II.1.1 A General INC Physics Simulation Scheme 8

II.2 The Standard Dubna Cascade Model 11

II.2.1 Characterization of the Target Nucleus 12

II.2.2 Intranuclear Interactions 13

II.2.3 Two-Bodied Kinematics 15

II.2.4 Three-Bodied Kinematics 17

II.2.5 The Pauli Exclusion Principle 19

II.2.6 Nuclear Trawling 20

II.2.7 Transition out of the INC Process 20

II.3 The Modified Dubna Cascade Model 22

II.3.1 A Mathematical Description of the Modified DCM 22

II.3.2 The Quark-Gluon String Model 24

II.3.3 Modified DCM Photon Interactions 25

II.4 Coalescence Physics 27

II.4.1 The Coalescence Model 30

II.5 Fermi Break-Up Physics 33

II.5.1 Allowed Fermi Break-Up Channels 33

II.5.2 Fermi Break-Up Channel Probability 34

II.5.3 Fermi Break-Up Channel Selection 36

II.6 Preequilibrium Physics 37

II.6.1 Particle Emission 39

II.6.2 Exciton Transitions 41

II.6.3 Sampling Preequilibrium Emission 43

II.6.4 Angular Distributions of Preequilibrium Progeny . . . 45
II.6.5 Photon Production from Nucleus De-Excitation . . . 46
II.6.6 Preequilibrium Emission off of Light Fragments . . . 46
II.6.7 Flow of Preequilibrium Simulations . . . 47
II.7 The Evaporation and Fission Models . . . 49
II.7.1 The Evaporation Model . . . 49
II.7.1.1 Fragment Decay Width . . . 50
II.7.1.2 Kinetic Energy Selection . . . 54
II.7.1.3 Fragment Selection . . . 55
II.7.1.4 Fragment Excitation . . . 56
II.7.2 The Fission Model . . . 57
II.7.2.1 Fission Probability . . . 57
II.7.2.2 Fission Fragment Mass Distribution . . . 60
II.7.2.3 Fission Fragment Charge Distribution . . . 61
II.7.2.4 Fission Fragment Kinetic Energy Distribution . . . 62
II.7.2.5 Updates to and Multipliers for the Fission Model . . . 63
II.7.3 Photon Production from Nucleus De-Excitation . . . 63
II.7.4 Evaporation of Small Fragments . . . 64
II.8 Photon Emission . . . 64

III Modernization . . . 66
III.1 Context . . . 66
III.2 Preliminary Modernization . . . 71
III.2.1 A Basic Case Study: the “chabs” Procedure . . . 73
III.3 Migrating GSM to an Object-Oriented Architecture . . . 75
III.4 Benefits of the Object-Oriented Approach . . . 79
III.5 The GSM Object . . . 82

IV GSM Usage . . . 88
IV.1 GSM Data Initialization . . . 88
IV.2 GSM Construction . . . 92
IV.3 Simulation Arguments . . . 95
IV.4 The GSM Driver . . . 103
IV.5 The GSM API . . . 107

V Verification of GSM . . . 113

VI Validation of GSM . . . 120
VI.1 Integral Regression Testing of GSM . . . 121
VI.1.1 Proton Induced Reactions . . . 121
VI.1.2 Neutron Induced Reactions . . . 127
VI.1.3 Light-Ion Induced Reactions . . . 133
VI.2 Validation of the Modernized GSM . . . 135
VI.3 Continued Validation Efforts of GSM . . . 145

VII Summary . . . 147

References . . . 156

A Known Bugs in GSM and its Sub-Models . . . 166

B Documentation for the Molnix Class . . . 173
B.1 Build Requirements . . . 173
B.2 Primer . . . 174
B.3 Data Structures . . . 176
B.4 Model Construction . . . 177
B.5 Usage and Results . . . 177
B.6 Tips, Warnings, and Limitations . . . 178

C Documentation for the Fission Barrier Class . . . 180
C.1 Build Requirements . . . 180
C.2 Primer . . . 181
C.3 Data Structures . . . 182
C.4 Model Construction . . . 184
C.5 Usage and Results . . . 185
C.6 Tips, Warnings, and Limitations . . . 186

D Documentation for the GSM . . . 187
D.1 Build Requirements . . . 187
D.2 Data Initialization . . . 188
D.3 Data Structures . . . 188
D.4 GSM Procedures . . . 195
D.5 Model Construction . . . 199
D.6 Usage and Results . . . 199

E Documentation for the Standard Dubna Cascade Model 200
E.1 Build Requirements 200
E.2 Primer 201
E.3 Data Initialization 203
E.4 Data Structures 206
E.5 The Target Data Object 211
E.6 Model Construction 216
E.7 Usage and Results 217
E.8 Tips, Warnings, and Limitations 221

F Miscellaneous Model Documentation 223
F.1 Evaporation and Fission Model Options 223
F.1.1 Evaporation Parameters 223
F.1.1.1 Inverse Reaction Cross Section Parameters 223
F.1.1.2 Level Density Parameters 227
F.1.1.3 CPU Time Saving Simulation 228
F.1.2 Fission Parameters 229

List of Figures

FIGURE II.1: The simulation scheme used by GSM for spallation physics. 6
FIGURE II.2: A general simulation scheme for INC physics. 9
FIGURE II.3: Quantum overlap approximation. 29
FIGURE II.4: A theoretical depiction of preequilibrium particle emission. 38
FIGURE II.5: Calculational flow for preequilibrium emission simulations. 48
FIGURE II.6: The simulation scheme for evaporation and fission physics in the GEM. . 51

FIGURE III.1: A sample modern Fortran module and data object. 69
FIGURE III.2: An example of a Fortran procedure pointer being created and utilized. . . 70
FIGURE III.3: The standard DCM "chabs" procedure preceeding GSM modernization. 74
FIGURE III.4: The standard DCM "chabs" procedure proceeding modernization. 74
FIGURE III.5: The class structure of the GSM "Preequilibrium" sub-model. 77
FIGURE III.6: The procedure pointer interface and its usage for all I/O handling within GSM and its sub-models. 84
FIGURE III.7: The procedure pointer interface and its usage regarding the GSM random number generator. 85
FIGURE III.8: The procedure pointer interface, initialization, and usage within GSM. . 87

FIGURE IV.1: Sample GSM data initialization utilizing both no arguments and all available arguments. 90
FIGURE IV.2: A sample GSM class construction interface. 93
FIGURE IV.3: Construction of the "GSMResults" object. 101
FIGURE IV.4: A simple build of GSM from the command prompt located in the top-level GSM directory. 104
FIGURE IV.5: A simple example of regression testing from a Unix command prompt. . 105
FIGURE IV.6: A sample simulation of GSM utilizing all available command line arguments. 107
FIGURE IV.7: A sample GSM class construction interface. 109

FIGURE VI.1: ^{6}Li production cross section at 20° to the incident beam for the reaction ^{107}Ag *(480 MeV p, X)*. 122
FIGURE VI.2: Neutron production cross section at 10° to the incident beam for the reaction ^{232}Th *(1.2 GeV p, X)*. 123

FIGURE VI.3: ^{6}Li production cross section at 100° to the incident beam for the reaction ^{197}Au (1.9 *GeV p, X)*. 124
FIGURE VI.4: ^{7}Li production cross section at 35° to the incident beam for the reaction ^{27}Al (2.5 *GeV p, X)*. 124
FIGURE VI.5: Proton production cross section at 32° to the incident beam for the reaction ^{9}Be (3 *GeV p, X)*. 125
FIGURE VI.6: ^{4}He production cross section at 90° to the incident beam for the reaction ^{238}U (5.5 *GeV p, X)*. 126
FIGURE VI.7: ^{10}B production cross section for the reaction ^{238}U (5.5 *GeV p, X)*. 127
FIGURE VI.8: ^{12}C production cross section for the reaction ^{12}C (28.15 *MeV n, X)*. . . . 128
FIGURE VI.9: ^{4}He production cross section at 80° to the incident beam for the reaction ^{209}Bi (175 *MeV n, X)*. 129
FIGURE VI.10: ^{3}He production cross section at 40° to the incident beam for the reaction ^{28}Si (175.4 *MeV n, X)*. 130
FIGURE VI.11: Tritium production cross section at 90° to the incident beam for the reaction ^{63}Cu (383 *MeV n, X)*. 131
FIGURE VI.12: Deuterium production cross section at 164° to the incident beam for the reaction ^{209}Bi (542 *MeV n, X)*. 132
FIGURE VI.13: ^{7}Li production cross section for the reaction ^{7}Li (51 *MeV* ^{10}B, *X)*. 134
FIGURE VI.14: ^{4}He production cross section at 90° to the incident beam for the reaction ^{58}Ni (6.7 *GeV d, X)*. 135
FIGURE VI.15: Fragment production cross section for the reaction ^{197}Au (319.00 *MeV* ^{7}Li, *X)*. 137
FIGURE VI.16: ^{6}He production cross section for the reaction ^{107}Ag (480.00 *MeV p, X)*. . 138
FIGURE VI.17: ^{6}He production cross section for the reaction ^{107}Ag (480.00 *MeV p, X)* for multiple simulations. 139
FIGURE VI.18: ^{6}He production cross section for the reaction ^{107}Ag (480.00 *MeV p, X)* with statistical estimates. 140
FIGURE VI.19: Neutron production cross section for the reaction ^{232}Th (1.2 *GeV p, X)*. . 141
FIGURE VI.20: Tritium production cross section for the reaction ^{63}Cu (383.00 *MeV n, X)*. 142
FIGURE VI.21: ^{4}He production cross section for the reaction ^{58}Ni (6.70 *GeV* ^{2}d, *X)*. . . . 142
FIGURE VI.22: Fragment production cross sections for the reaction ^{238}U (42.00 *GeV* ^{20}Ne, *X)*. 143
FIGURE VI.23: Computation times of the modernized GSM event generator. 145

FIGURE VII.1: The general structure utilized for each modernized object within the GSM. 148

FIGURE VII.2: A sample GSM API. 150
FIGURE VII.3: Neutron production cross section for the reaction ^{232}Th (1.2 *GeV p, X)*. . 151
FIGURE VII.4: ^{6}He production cross section for the reaction ^{107}Ag (480.00 *MeV p, X)*. . 152
FIGURE VII.5: Fragment production cross sections for the reaction ^{238}U (42.00 *GeV* ^{20}Ne, *X)*. 152
FIGURE VII.6: Computation times of the modernized GSM event generator. 153

FIGURE B.1: Sample API for the Molnix object. 175

FIGURE C.1: Sample API for the Fission Barrier object. 183

FIGURE E.1: Sample API for the standard DCM object. 204
FIGURE E.2: Sample API for the *StandardDCMData* object. 213
FIGURE E.3: A sample of the standard DCM object construction. 218
FIGURE E.4: A sample construction and usage of the *standardDCMResults* object. . . 220

List of Tables

TABLE II.I: Nucleon and Pion-Induced Interactions Considered by the Standard DCM 13
TABLE II.II: Pion Production Modes for Photonuclear Interactions 15
TABLE II.III: Coefficients a_{nk} for $\gamma + p$ Reactions Involving Isobar Production 18
TABLE II.IV: Channels of Elementary $\gamma + N$ Interactions Accounted for by the Modified DCM 25
TABLE II.V: Coalescence Radii for Composite Particle Formation 32
TABLE II.VI: Final State Interactions Considered by the Coalescence Model 32
TABLE II.VII: Isotopes Considered for Emission in the Preequilibrium and Evaporation Models 56
TABLE II.VIII: Constants $C(Z)$ and $A_0(Z)$ for Eqn. II.83 60

TABLE IV.I: Variable Members of the "GSMProjectile" Object 96
TABLE IV.II: Variable Members of the "GSMTarget" Object 98
TABLE IV.III: Variable Members of the "GSMResults" Object 100
TABLE IV.IV: Variable Members of the "GSMOutput" Object 101
TABLE IV.V: A List of GSM Procedures Available to Clients 111

TABLE VI.I: Computation Times for Proton Induced Reactions 128
TABLE VI.II: Percent Difference and Relative Error of the Modernized GSM Results to Experimental Data (mb/sr/MeV) 131
TABLE VI.III: Computation Times for Neutron Induced Reactions 133
TABLE VI.IV: Computation Times for Light-Ion Induced Reactions 136

TABLE II.I: Molnix Modules 173
TABLE II.II: Compiler Support for the Molnix Class 174
TABLE II.III: Data Types Used by the Molnix Object 176
TABLE II.IV: The *molnixOptions* Data Type 176
TABLE II.V: The *molnixIO* Data Type 176

TABLE III.I: Fission BarrierModules 180
TABLE III.II: Compiler Support for the Fission Barrier Class 181
TABLE III.III: Data Types Used by the Fission Barrier Object 182
TABLE III.IV: The *fissionBarrierOptions* Data Type 182
TABLE III.V: The *fissionBarrierIO* Data Type 184

TABLE IV.I: GSM Modules . . . 187
TABLE IV.II: Compiler Support for the Standard Dubna Cascade Model . . . 188
TABLE IV.III: Data Types Used by the GSM Object . . . 189
TABLE IV.IV: The *gsmOptions* Data Type . . . 191
TABLE IV.V: The *gsmProgeny* Data Type . . . 192
TABLE IV.VI: The *gsmResidual* Data Type . . . 192
TABLE IV.VII: The *gsmIO* Data Type . . . 192
TABLE IV.VIII: The *coalescenceDataOptions* Data Type . . . 193
TABLE IV.IX: The *coalescenceOptions* Data Type . . . 193
TABLE IV.X: The *fermiBreakUpOptions* Data Type . . . 193
TABLE IV.XI: The *preequilibriumOptions* Data Type . . . 194
TABLE IV.XII: The *evaporationDataOptions* Data Type . . . 194
TABLE IV.XIII: The *evaporationOptions* Data Type . . . 194

TABLE V.I: Standard DCM Modules . . . 201
TABLE V.II: Compiler Support for the Standard Dubna Cascade Model . . . 201
TABLE V.III: Parameterized Data Initialization Flags for the Standard DCM . . . 205
TABLE V.IV: Data Types Used by the Standard DCM Object . . . 206
TABLE V.V: The *sDCMOptions* Data Type . . . 206
TABLE V.VI: The *sDCMProjectile* Data Type . . . 207
TABLE V.VII: The *sDCMProgeny* Data Type . . . 208
TABLE V.VIII: The *sDCMCompound* Data Type . . . 209
TABLE V.IX: The *sDCMExcitons* Data Type . . . 209
TABLE V.X: The *standardDCMResults* Data Type . . . 210
TABLE V.XI: The *sDCMPhotonCrossSections* Data Type . . . 210
TABLE V.XII: The *sDCMPauliInfo* Data Type . . . 211
TABLE V.XIII: The *sDCMIO* Data Type . . . 211
TABLE V.XIV: The *sDCMDataOptions* Data Type . . . 214
TABLE V.XV: The *sDCMDataTarget* Data Type . . . 214
TABLE V.XVI: Construction Flag Values . . . 215
TABLE V.XVII: Member Procedures of the *StandardDCMData* Object . . . 216

TABLE VI.I: Precise Inverse Cross Section Parameter Set . . . 224
TABLE VI.II: Inverse Cross Section Parameter Set from Ref. [77] . . . 225
TABLE VI.III: *rcal* Value for Inverse Cross Section Parameters . . . 227
TABLE VI.IV: *alev* Value for Level Density Parameters . . . 228
TABLE VI.V: Criteria for Heavy Ejectile Emission Rejection from Compound Nuclei 228

TABLE VI.VI: *ired* Value for Use of Reduced Computation Time 229
TABLE VI.VII: *ifis* Value for Determination of Fission Parameterization 229

OBTAINING THE GSM CODE

The GSM is presently hosted by the Los Alamos National Laboratory on a private *BitBucket* code repository[3]. The GSM is not made public at this time to users due to licenser and ownership concerns. A public open-source license for the GSM may be requested. Contact the XCP-3 team at the Los Alamos National Laboratory, Chase Juneau[4], or Dr. Leslie Kerby for access to the code, where you will be added to the *gsmdev* group on the *BitBucket* server or be emailed a copy of the code upon receiving all proper approvals.

[3]The GSM code is presently privately hosted on xcp-stash.lanl.gov.

[4] junechas@isu.edu

ABSTRACT

The generalized spallation model (GSM) is a general spallation physics event generator with a physics-rich history dating back to the early 1970s during the development of the CEM [1]. GSM estimates the outcome of high-energy hadron-nucleus and nucleus-nucleus collisions, where collision energies often exceed hundreds of MeV. Such a collision may include a neutron incident on carbon, for example. The thesis presented here outlines the continued development and modernization of GSM, whereby it was migrated from a legacy Fortran code to utilize an advanced and modern Fortran architecture. Verification and validation of GSM also took place throughout the continued development of GSM.

The results of this work have transformed GSM into a highly modern and powerful event generator, where numerous benefits have been fostered and precipitated, including but not limited to portability, modularization, flexibility, robustness, and scalability. GSM is a robust and well developed software library that clients and end-users may confidently use, within the scope outlined here, as a result of this work.

Keywords: Event Generator, GSM, Monte Carlo, Spallation

Chapter I. Introduction

The generalized spallation model (GSM) is a general spallation physics event generator. GSM may estimate the outcome of high-energy hadron-nucleus and nucleus-nucleus collisions, where collision energies often exceed hundreds of MeV. Generality is provided to clients and end-users of GSM regarding the incident and struck particles, the incident or collision energy, and in addition provides control to its clients and end-users of how the event generator behaves and is used for simulations. GSM is recommended to end-users and clients when attempting to predict reaction progeny. It is recommended that GSM be used for hadron-nucleus and nucleus-nucleus collisions where the incident particle has energy greater than $\sim 100\ MeV$, not exceeding $\sim 1\ TeV$ per nucleon. Valid incident particles of GSM include neutrons, protons, charged and uncharged pions, mono-energetic and bremsstrahlung photons, and nearly any nucleus[1], where physical limitations are applied. Particle transport codes, such the Monte Carlo N-Particle® (MCNP®) transport code[2], or others, may utilize GSM for their simulations in place or in addition to their currently available event generators.

Applications of GSM are typically focused in the accelerator, space, and medical industries. Event generators are utilized for a variety of applications within these industries. Such applications may include personnel dose estimation at accelerator facilities, transmutation of nuclear wastes, the generation of neutron spallation sources, proton radiography, radiation therapy, shielding development for each of the industries, electronics protection, single event-upset (SEU) estimations, and many others. The GSM event generator is highly scalable for its clients to aid in accurate and precise estimation of these numerous processes.

[1] GSM, when utilizing the standard DCM for its IntraNuclear Cascade simulations requires that target nuclei be larger than ^{4}He.

[2] MCNP® and Monte Carlo N-Particle® are registered trademarks owned by Los Alamos National Security, LLC, manager and operator of Los Alamos National Laboratory. Any third party use of such registered marks should be properly attributed to Los Alamos National Security, LLC, including the use of the ® designation as appropriate. Any questions regarding licensing, proper use, and/or proper attribution of Los Alamos National Security, LLC marks should be directed to trademarks@lanl.gov.

Preliminary development of GSM was undergone [2] in mid to late 2017, being developed based on the CEM [1] event generator, incorporating portions of the LAQGSM [3] event generator into its code, namely the modified Dubna Cascade Model (DCM). Much of GSM at that time consisted of almost solely legacy Fortran code, using mostly Fortran66 and Fortran77 code standards. Many poor constructs existed in the GSM code, where great technical debt was present. GSM was also a highly monolithic model with the addition of the modified DCM to its code.

The GSM event generator code was modernized to allow its further development, usage, and implementation of itself into the near and far future. The modernization of GSM allowed for the addition of best practices, making the code more robust in its implementation. Protection to the coded numerical data, as well as client usage, was utilized to aid in its modernization. Modernization of the GSM has allowed for the additional implementation of third party libraries as a result of its modernization, such as implementing a cross-platform build system[3], to simplify the code and maintenance of the code itself. Modernization of GSM required a significant amount of work to implement and test given the size and age of the legacy code. Modern syntax, implementations, and structures were utilized as a result of modernizing GSM to utilize current and supported compilers.

The GSM, and five of its six sub-models, was migrated to an object-oriented architecture. Migrating GSM to utilize an object-oriented framework has numerous benefits. Such benefits include: providing end-users and clients with clear and precise simulation control, allowing the usage and testing of specific components or sub-libraries, providing immense flexibility to the GSM and its clients and end-users, and provides great scalability. Note that many other benefits to migrating GSM to an object-oriented architecture exist, however these are not listed here for brevity.

Migrating GSM to an object-oriented framework additionally allowed for the creation of a robust application programming interface (API) to the GSM library, whereby clients of GSM may use it to estimate and test a multitude of physics processes. The flexibility granted to GSM and its sub-models as a result of its modernization and the object-oriented migration is great. Clients may modify behaviors of specific sub-models, select from a variety of numerical parameterizations,

[3]The *CMake* cross-platform build system was implemented in the GSM.

modify numerics, progeny considered for emission, bias progeny creation, and even omit physics if desired, in addition to other options. The available client scope is additionally highly limited by the object-oriented architecture in that access to certain features is limited, aiding in the robustness of the code.

Few event generators exist, if any, that allow clients and end-users to utilize high performance computing (HPC) resources. Simulations by GSM, like most event generators, require significant computation time due to the complex and multi-faceted phenomena experienced at the applicable incident energies. The need for high performance computing is necessary regarding event generators to provide precise and accurate results as event generators may provide end-users and clients with information regarding collision cross sections, progeny created, particle yields, and much more. The prior lack of HPC compatibility in the GSM event generator required clients and end-users to tolerate slow and imprecise simulations. The GSM migration transformed the GSM into a highly scalable code, providing clients and end-users with a more robust and potentially lower client simulation times.

Verification of GSM was undertaken as a result of its modernization, object-oriented migration, and its improved scalability. Coding standards compliance was ensured within GSM as well during its continued development. Validation of GSM was considered, where continuous regression testing was performed during the development of GSM. Resolved coding errors were tested during the validation of GSM throughout its development, where GSM was analyzed for numerical and simulation changes during its modernization. Unit tests were utilized to validate GSM against other event generators, namely the CEM and LAQGSM event generators, as well as experimentally available data.

Chapter II. A Review of Spallation Physics in GSM

High-energy spallation reactions are highly complex multi-faceted phenomena. Many event generators, such as GSM, simulate these reactions via Monte Carlo techniques to estimate fragment emissions and residual nuclei as a result of a high-energy collision. The spallation process is often considered as a two-step stage, being composed of the precompound and compound stages. Some event generators, such as GSM, separate the precompound stage further by considering the equilibration of the excited residual nucleus, where the precompound stage of the reaction consists of the IntraNuclear Cascade (INC) and the equilibration, also known as preequilibrium decay, processes. The compound stage of the reaction consists of particle emission via evaporation and fission from the compound nucleus that remains as a result of the precompound stage of the reaction.

Fig. II.1 details the flow chart of spallation physics simulated by GSM and its predecessor event generators, CEM and LAQGSM[1, 4, 5]. GSM simulates spallation physics when provided valid incident and target nuclei. Note that GSM is recommended when incident energies exceed $\sim 100\ MeV$, up to $\sim 1\ TeV$ per nucleon. The nuclei upon colliding undergo the fast, energetic INC stage of the reaction, where the incident energy is sufficient that collisions may be approximated to occur between nucleons of the incident and target nuclei and where collisions are approximately independent of one another. Many INC progeny, being primarily nucleons, are emitted during the INC reaction, where the incident and target nuclei have been depleted, existing in an excited residual state [6].

INC progeny may coalesce when they are produced within a small difference in position space, $\Delta\mathbf{r}$, and in momentum-space, $\Delta\mathbf{p}$, of one another, being emitted at approximately the same time during the INC process. The theory of particle coalescence requires that the phase-space distribution of the progeny overlap, yielding a probability of compound formation [7, 8]. Many coalescence models however approximate particle compounding for progeny meeting this criteria as lying within a small coalescence radius, $\Delta\mathbf{p}$, of one another. The depleted incident and target nuclei

that remain after the INC process, herein referred to as residual nuclei, then undergo transitions and subsequent particle emission as a result of their high excitation energies and due to their instability after the INC process. GSM estimates this fragment emission using its Fermi break-up or preequilibrium model. The preequilibrium model estimates fragment emissions as a result of the residual nucleus's equilibration, undergoing various exciton and subsequent particle emission until reaching an equilibrated state [9, 10]. The residual nuclei are considered to be compound nuclei upon reaching an equilibrated state.

Compound nuclei are simulated by GSM to undergo particle emission via the Fermi break-up process or the evaporation and fission of the compound [11–13]. Note that all fission fragments created during the evaporation and fission process will subsequently undergo their own evaporation and fission processes. The Fermi break-up model is utilized to predict disintegration of a nucleus. The Fermi break-up model [14], unlike the preequilibrium and evaporation models, does not utilize the statistical assumption of the Weisskopf-Ewing statistical evaporation model [15, 16], thus any excited nucleus not applicable to the preequilibrium and evaporation processes may instead be modeled as undergoing the Fermi break-up process. The Fermi break-up model instead assumes the excitation energy of a nucleus is beyond that of a nucleon's binding energy, thus it disintegrates. The precompound stage of the GSM reaction consists of the IntraNuclear Cascade (INC), preequilibrium decay of the residual, and coalescence of INC progeny. The compound stage of the simulation follows preequilibrium decay of the residual, consisting solely of a compound nucleus's evaporation and fission.

II.1 A SURVEY OF INC PHYSICS

The Dubna Cascade Model (DCM) [17, 18] predicts the emission of nucleons and complex particles from the fast stage of spallation interactions. Excited residual and compound nuclei are often formed as byproducts of the fast stage of spallation reactions [6]. Many spallation models will use statistical deexcitation models in conjunction with an INC model to predict the potential

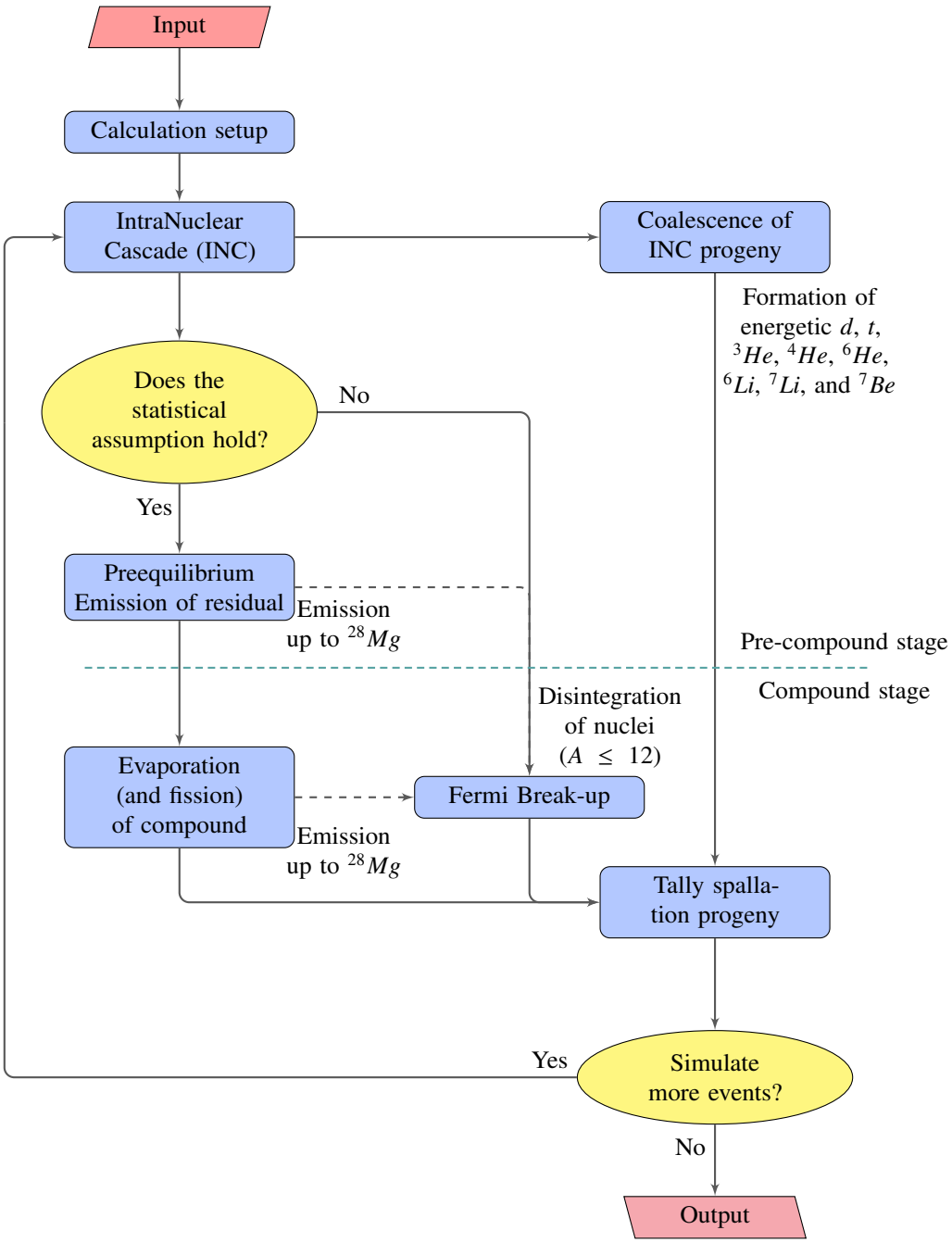

FIGURE II.1. The simulation scheme used by GSM for spallation physics.

particle emissions from the deexcitation and decay of the excited residual and compound nuclei. The estimation of fragment emission during the spallation reaction is split into three primary phases: the fast INC stage, the equilibration of the excited residual nucleus, often referred to as preequilibrium,

and the evaporation of particles from, or the fission[1] of, the compound nucleus. The INC and equilibration processes occur in the precompound stage of the reaction, while evaporation and fission processes occur during the compound stage. Progeny that are created as a result of the particle cascade will undergo final-state interactions to coalesce and form high-energy composite nuclei. Residuals that are sufficiently small will undergo Fermi break-up emission instead of other statistical deexcitation processes. The Fermi break-up model is used to describe emission from these light nuclei because the model, unlike most preequilibrium and evaporation models, does not assume that a large number of nucleons are present to be statistically accurate.

INC models often require that these spallation reactions occur at very high energies, being tens of *MeV* or more. This requirement is necessary because the INC process assumes that nucleons of each nucleus have a sufficiently small wavelength, λ. The wavelength λ should to be smaller than the distance between intranuclear nucleons. Treatment in this manner allows the INC models to treat nucleon collisions approximately, where particle trajectory can be utilized and two-particle collisions can occur inside the nucleus [17, 18]. This approximation typically requires these large energies. Additionally, the standard DCM discussed here requires that the average time in which an individual two-particle intranuclear collision occurs, being $\tau \approx 10^{-23}$ seconds, be less than the time interval between two such consecutive interactions. The minimum time between two consecutive interactions can be described approximately by the quotient of the mean range, l, of the cascade particle prior to the interaction and the velocity of light, c, as described below.

$$\Delta t = \frac{l}{c} \gtrsim \frac{\frac{4\pi r_0^3}{3\sigma}}{c} \gtrsim \frac{3 \times 10^{-22}}{\sigma} \text{ [sec]}$$

In describing the time between intranuclear collisions, it is important to note that $r_0 \approx 1.2$ fm and σ is the cross section, in *mb*, for the interaction with an intranuclear nucleon. Satisfying the time condition, such that collisions occur faster than the time between collisions, permits abstraction of the INC process as individual statistically independent intranuclear collisions. Satisfying the

[1] Note if the compound nucleus fissions, the resulting fission fragments may also undergo evaporation, depending on their excited states.

condition $\tau \leq \Delta t$ is also equivalent to requiring that the intranuclear collision reaction cross section be small, such that $\sigma \lesssim 100\xi$ mb, with the coefficient $\xi \approx 1$ [4].

The assumption can be made that the same characteristics may be used for interaction of cascade particles inside the nucleus as for the interaction of free particles due to the nucleon's energy being significantly larger than its binding energy. The effect of other intranuclear nucleons is taken into account by the introduction of an average potential V, and also by the application of the Pauli exclusion principle [4]. The Pauli exclusion principle is applied by modeling the nucleus as a degenerate Fermi gas of nucleons being enclosed in some nuclear volume. These nucleons must have an energy above the Fermi energy after a collision, otherwise the nucleon was produced from a forbidden interaction. Inclusion of the Pauli principle in effect leads to an increase of the mean free path of fast participant particles inside the nucleus [4]. Assuming the nucleus acts as a degenerate Fermi gas, high energy particles having entered the nucleus will pass through the nucleus and create a cascade of secondary particles. The produced secondary particles may then either leave the nucleus or be absorbed by it, exciting the nucleus to some excitation energy, E^*. Some INC models, such as the INCL, allow for creation of loosely bound nucleons as well, being located on the edges of the degenerate Fermi gas [19].

II.1.1 A General INC Physics Simulation Scheme

Simulation of the INC process is carried out according to the scheme depicted by Fig. II.2, following Ref. [17, 18]. Typical INC model inputs requires only information about the colliding particles and the energy at which they collide. Products from most INC models include produced secondary particles and an excited residual or compound nucleus. Exciton information may also be produced by exciton-based INC models, such as the standard DCM.

The INC process is first modeled by determination of the primary particle's characteristics. Additionally, changes in the primary particle's momentum is accounted for due to the effect of the intranuclear potential, V, and due to refraction and reflection of the DeBroglie wave of the particle

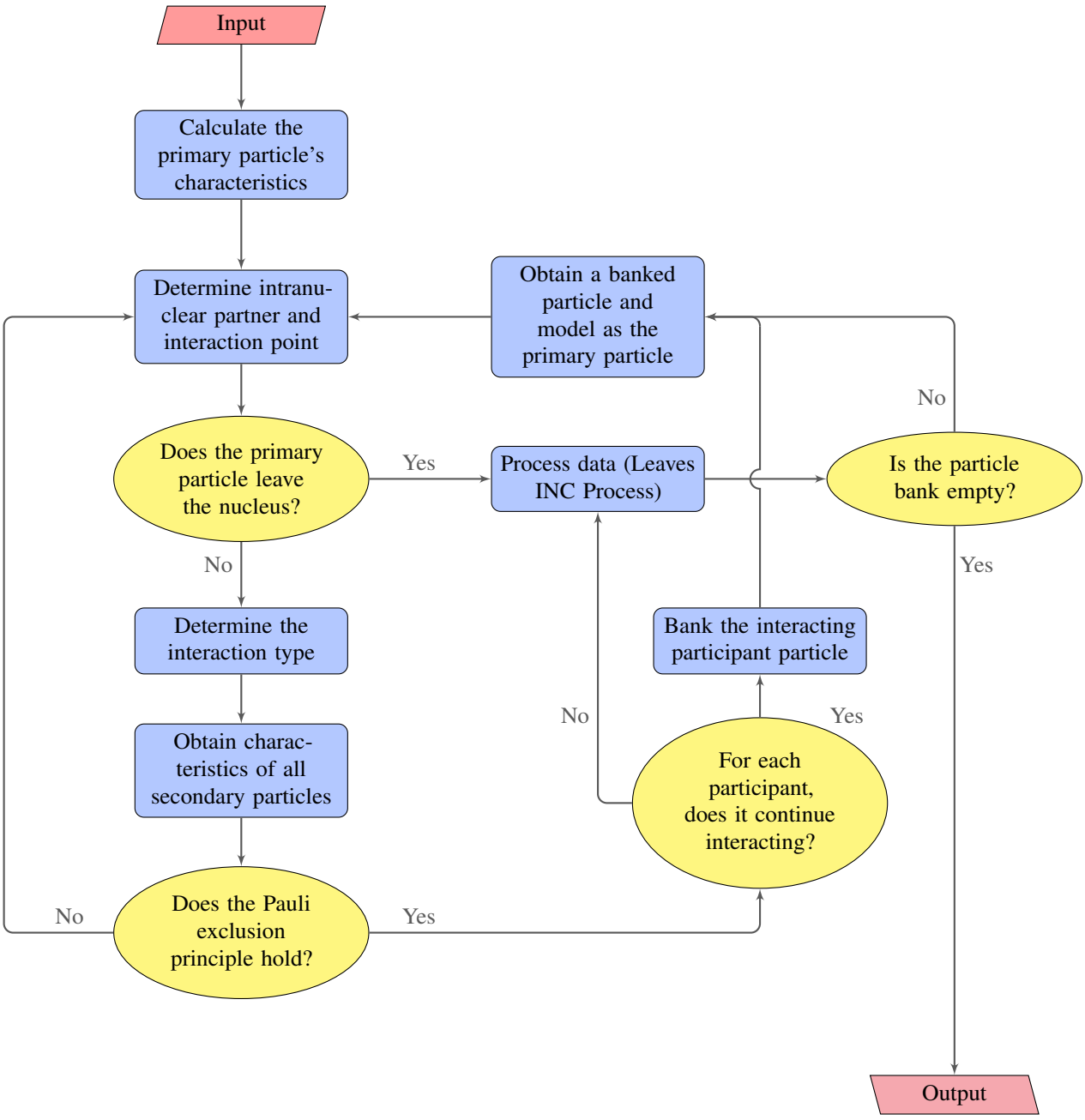

FIGURE II.2. A general simulation scheme for INC physics.

at the nuclear boundary. The primary particle's intranuclear interaction partner is then determined by choosing its momentum and isospin. The point of interaction is also determined by use of the energy-dependent elementary cross section, $\sigma_{tot}(E)$, to determine the mean free path of the primary particle in the nuclear matter. The energy of the interaction, E, is the relative energy of the primary particle and the interacting partner taking part in the intranuclear motion.

The interaction is then determined to be inside or outside the nucleus based upon the interaction point of the primary particle and its interacting partner. The primary particle is assumed to have passed through the nucleus without interacting if its interaction point is found to be outside of the nucleus. The energy-dependent inelastic reaction cross section, $\sigma_{in}(E)$, is characterized by the ratio of the number of such non-interacting primary particles and the total number of interactions considered with the nucleus. The particles that no longer interact with the nucleus may undergo coalescence, where the secondary particles may coalesce to form a compound nucleus, such as ^{4}He. Interactions occurring within the nucleus are determined to be elastic or inelastic based upon the computed elastic and inelastic cross section, $\sigma_{el}(E)$ and $\sigma_{in}(E)$, respectively.

The characteristics of the secondary particle are then determined based upon the primary particle's characteristics, the interacting partner's characteristics, and the reaction type. The Pauli Exclusion principle is verified to hold after determination of the secondary particle's energy. The reaction is forbidden if the secondary particle's energy falls below the Fermi energy. Occurrence of forbidden reactions results in a re-determination of the interacting partner and interaction point with the primary particle, further following the flow of the INC calculation shown by Fig. II.2.

The secondary particle is then allowed to continue interacting if it has sufficient energy, being above some prespecified cut-off energy. Secondary particles with energy below the cut-off are absorbed by the nucleus, and those with energies above the cut-off energy may interact further to create more secondary particles as if they were a primary particle, interacting as is described above. Secondary particles that are absorbed by the target nucleus will increase the excitation energy of the nucleus and be further tracked in the model as excitons. All participant particles that may continue interacting will have their characteristics stored in memory, or "banked." Characteristics of a participant particle will then be obtained from the particle bank at the end of a collision such that the participant is modeled as if it were a primary particle, and the collision of the participant particle with an intranuclear nucleon will be simulated. Note that it is assumed that produced secondary particles have negligible influence on one another's emission, *i.e.* the interacting participants do not interact with one another as they have left the interacting system.

The particle bank will be searched for further interacting particles in the case of non-interacting primary particles or intranuclear collisions ending with only participant particles that do not interact further. Intranuclear collisions of this nature will generally occur at or near the end of the INC process. The INC process is assumed complete when an intranuclear collision of this nature occurs and when no more particles exist in the particle bank. The produced excitons and resulting excited residual nucleus are typically treated by the Modifed-Exciton Model (MEM) during the excited nucleus's equilibration, being simulated after completion of the simulation of the fast, INC portion of the reaction.

The depiction provided in Fig. II.2 is applicable to all INC calculations, regardless of energy, as long as it is still possible to neglect many-particle interactions and nuclear depletion, coined as the "trawling" effect [17]. The specific form of each of the operations discussed above is highly dependent on the models utilized and desired complexity at each of the calculation steps. The INC model discussed here, the standard DCM, utilizes the methods and algorithms discussed in the following section, Sec. II.2. More details on these processes can be found in Ref. [17, 20, 21].

II.2 THE STANDARD DUBNA CASCADE MODEL

The standard Dubna Cascade Model (standard DCM) can be used to describe the intranuclear cascade occurring between primary and participant nucleons during the fast stage of high-energy spallation calculations. Use of the standard DCM model assumes the DeBroglie wavelength of the individual nucleons be smaller than the intranuclear distance between nucleons and that the time between intranuclear collisions is larger than the time elapsed during an intranuclear collision. The INC process undergone during these high-energy collisions can be simulated by the standard DCM using only information about the colliding nuclei mass and atomic numbers, and the energy at which they collide, in MeV. The INC process modeled by the standard DCM tracks exciton holes, charged and uncharged excitons, an excited residual nucleus, and a list of emitted secondary particles. Note that many details and formulas regarding the standard DCM, being based primarily on the standard,

time-independent Dubna Cascade Model [17, 18, 20, 21], can be found in Ref. [1, 4, 5]. The theory presented in Ref. [1, 4, 5] is quoted here for completeness. The projectile particle for INC collisions is considered to be the high-energy incident particle, and the target is the struck nucleus.

II.2.1 Characterization of the Target Nucleus

All INC calculations with the standard DCM are carried out utilizing a three-dimensional geometry. The nuclear matter density, $\rho(r)$, is assumed to vary only radially, being described by a Fermi distribution utilizing two parameters that are based on the analysis of electron-nucleus scattering. The nuclear matter density is assumed to be a superposition of the neutron and proton densities. Eqn. II.1 describes the nuclear matter density modeled.

$$\rho(r) = \rho_p(r) + \rho_n(r) = \rho_0\left[1 - e^{(r-c)/a}\right] \tag{II.1}$$

Eqn. II.1 utilizes the values $c = 1.07A^{1/3}\,fm$, where A is the atomic number of the target nucleus, and $a = 0.545\,fm$. The nuclear matter density is further simplified by dividing the nucleus with concentric spheres to create seven nuclear density zones, where the nuclear density is approximated as constant in each of the respective zones. Reflection and refraction of the interacting particles at the boundaries of these nuclear zones may be included in the simulation of standard DCM physics. Sec. E.4 details this more. The energy spectrum of nucleons within the target nucleus is estimated by utilizing the local Fermi energy, being estimated within the Fermi-gas approximation, as shown in Eqn. II.2.

$$T_F(r) = \hbar^2 \frac{\left[2\pi^2\rho(r)\right]^{2/3}}{2m_N} \tag{II.2}$$

Eqn. II.2 utilizes m_N as the nucleon mass. Ref. [5] provides an example of the nuclear density and Fermi-gas energy distributions.

The influence of intranuclear nucleons on the incoming projectile is accounted for by adding to

its laboratory kinetic energy an effective potential V, as well as by considering the Pauli exclusion principle. The inclusion of the Pauli exclusion principle effectively increases the mean free path of participant particles inside the target nucleus as it forbids certain intranuclear collisions. The incident particle's nucleons are modeled with $V \equiv V_N = T_F(r) + \epsilon$, where $T_F(r)$ is the corresponding Fermi energy and ϵ is the binding energy of the nucleons. Incident pions instead utilize a constant square well of depth $V_\pi \simeq 25\ MeV$, as was done in the initial Dubna Cascade Model [17, 18], where effects of nucleus and pion energies are neglected.

II.2.2 Intranuclear Interactions

The interaction of the incident particle with the target nucleus is approximated as a series of successive quasi-free collisions of the fast cascade particles with intranuclear nucleons. The neutron-, proton-, and pion-induced interactions allowed by the standard DCM are shown in Table II.I. The fast cascade particles considered by the standard DCM include nucleons, N, pions, π, and photons, γ. The standard DCM also accounts for pion absorption on nucleon pairs regarding interactions of incident pions with the target nucleus. The momenta of the participant nucleons regarding pion absorption are chosen via random sampling of the Fermi distribution. The energy of the absorbed pion is equally distributed between these nucleons in the center-of-mass system for the 3-bodied interaction. The direction of motion for the resulting participant nucleons is assumed to be isotropic in space. The effective pion-absorption cross section is related to the experimental pion-absorption cross section for deuterons.

TABLE II.I. Nucleon and Pion-Induced Interactions Considered by the Standard DCM

Number	Interaction
1	$\mathrm{N\,N} \rightarrow \mathrm{N\,N}$
2	$\mathrm{N\,N} \rightarrow \pi\,\mathrm{N\,N}$
3	$\mathrm{N\,N} \rightarrow \pi_1, \cdots, \pi_i\,\mathrm{N\,N}$
4	$\pi\,\mathrm{N} \rightarrow \pi\,\mathrm{N}$
5	$\pi\,\mathrm{N} \rightarrow \pi_1, \cdots, \pi_i\,\mathrm{N}\ (i \geq 2)$
6	$\pi\,\mathrm{N\,N} \rightarrow \mathrm{N\,N}$

Photonuclear reactions in the standard DCM [22] utilize the ideas of the photonuclear version of the Dubna INC model, presented originally by Ref. [23]. The photonuclear Dubna INC model is used to described photonuclear reactions at energies above the Giant Dipole Resonance (GDR) region [24]. For energies below that of the GDR, the incident photons have a DeBroglie wavelength larger than the distance between intranuclear nucleons, thus the photons interact with the nucleus more as a whole and the INC process cannot be applied. The Dubna INC model considers photon absorption on only 'quasi-deuteron" pairs for interactions below the pion-production threshold according to the Levinger model [25], as shown in Eqn. II.3.

$$\sigma_{\gamma A} = L\frac{Z\,(A-Z)}{A}\sigma_{\gamma d} \tag{II.3}$$

The terms A and Z in Eqn. II.3 represent the mass and atomic numbers of the nucleus, with $L \approx 10$ and the total photon-absorption cross section on deuterons, $\sigma_{\gamma d}$, is defined from available experimental data. The Dubna INC model considers the production of one or two pions for interacting photons with energies above the pion-production threshold. The mode of the interaction is chosen randomly according to the partial cross section from available experimental data. The modes considered for pion production during photonuclear interactions, based on charge and mass conservation laws, are shown in Table II.II and are chosen by the Monte Carlo method according to partial cross sections, being defined by experimentally available data. Note that isotopic invariance is assumed for $\gamma + n$ and $\gamma + p$ interactions, *i.e.* $\sigma\,(\gamma + n) \approx \sigma\,(\gamma + p)$.

Compton effects on intranuclear nucleons is neglected, as its cross section is less than $\approx 2\%$ of other reaction modes considered by the INC[4, 26]. Note that the Dubna INC model is limited in that it only considers processes yielding the production of up to two pions during photonuclear collisions, restricting the model to photon energy regimes of $T_\gamma \lesssim 1.5$ GeV. The Dubna INC model approximates photonuclear interactions that occur above the three-pion threshold by estimating the three-pion production cross section as the remainder of the total inelastic pion production cross section, as shown by Eqn. II.4.

TABLE II.II. Pion Production Modes for Photonuclear Interactions

Number	Interaction		
1	$\gamma + p$	$\rightarrow$	$p + \pi^0$
2		$\rightarrow$	$n + \pi^+$
3		$\rightarrow$	$p + \pi^+ + \pi^-$
4		$\rightarrow$	$p + \pi^0 + \pi^0$
5		$\rightarrow$	$n + \pi^0 + \pi^+$
6	$\gamma + n$	$\rightarrow$	$n + \pi^0$
7		$\rightarrow$	$p + \pi^-$
8		$\rightarrow$	$n + \pi^+ + \pi^-$
9		$\rightarrow$	$n + \pi^0 + \pi^0$
10		$\rightarrow$	$p + \pi^0 + \pi^-$

$$\sigma_{3-\pi\ prod.} \approx \sigma_{in,\gamma+p} - \left(\sigma_{2-\pi\ prod.} + \sigma_{1-\pi\ prod.}\right) \tag{II.4}$$

Integral cross sections for the free NN, πN, and γN interactions, as shown by Table II.I and Table II.II, are approximated in the Dubna INC [17] by using a special interpolation and extrapolation algorithm through a number of picked points, mapping available experimental data [22, 27, 28]. Approximations for 34 different types of elementary cross sections for the interactions of Table II.I and Table II.II have been developed. The standard DCM determines integral cross sections for other types of interactions by considering isospin, using the developed approximations for the reactions in the tables. Results by the standard DCM using these various approximations can be found in Ref. [5, 28].

II.2.3 Two-Bodied Kinematics

Kinematics of two-bodied elementary interactions and absorption of photons and pions by a pair of interacting nucleons is defined completely by a direction of emission, θ, of one of the secondary particles. The direction of emission in θ is randomly sampled using Eqn. II.5 according to Monte Carlo theory.

$$\cos(\theta) = 2\xi^{1/2}\left[\sum_{n=0}^{N} a_n \xi^n + \left(1 - \sum_{n=0}^{N} a_n\right)\xi^{N+1}\right] - 1 \tag{II.5}$$

The coefficients a_n are given by Eqn. II.6. Note also that $N = 3 = M$.

$$a_n = \sum_{k=0}^{M} a_{nk} T_i^k \tag{II.6}$$

The coefficients a_{nk} from Eqn. II.6 were fitted to available experimental data at the time for various incident kinetic energies, T_i, and then interpolated and extrapolated to other energies. Ref. [17, 23, 24] provides further details. The distribution of secondary particles over the azimuthal angle ϕ is assumed to be equiprobable. Elementary interactions involving more than two particles in the final state utilize a statistical model to simulate the angles and energies of products [17]. The approximations for Eqn. II.5 and Eqn. II.6 utilize those of the INCL [19] below incident energies up to 2 GeV [29].

The standard DCM utilizes the Moscow INC [30] for $\gamma + p$ and $\gamma + n$ interactions that result in two products[2] instead of fitting experimental data to Eqn. II.5 and Eqn. II.6. The Moscow INC model utilizes a data file for $\gamma + p$ and $\gamma + n$ reactions with smooth approximations for experimental data. The data utilized provides experimental data available for 60 different reaction channels at 50 different photon energies, ranging from $\sim$ 117 MeV to $\sim$ 6 GeV. The data is utilized for such reactions in the system where the photon is at rest and for the center of mass angular distributions ($d\sigma/d\omega$) of secondary particles as functions their emission angle (θ). The standard DCM uses a part of this data file, alongside data for the interactions, to unambiguously simulate the angular distribution and choose the corresponding emission angle θ and photon energy, E_γ. The emission angle θ is sampled randomly according to Eqn. II.7, being a smooth monotonic function. Interpolation is utilized where values of the photon energy, E_γ, differ.

[2]These interactions are shown by Table II.II as interactions one, two, six, and seven.

$$\xi(\cos\theta) = \frac{\int_{-1}^{\cos\theta} \frac{d\sigma}{d\omega} d(\cos\theta)}{\int_{-1}^{1} \frac{d\sigma}{d\omega} d(\cos\theta)} \tag{II.7}$$

II.2.4 Three-Bodied Kinematics

Channel three of Table II.II, a two-pion production channel, was found to proceed through the decay of the Δ^{++} isobar, with mass $E_M = 1.232$ GeV. In this interaction, the decay of the Δ^{++} isobar acts as an intermediate step.

$$\gamma + p \rightarrow \Delta^{++} + \pi^-$$

$$\Delta^{++} \rightarrow p + \pi^+$$

The production of other isobar components $\left(\frac{3}{2}, \frac{3}{2}\right)$, being small, is considered negligible. The standard DCM utilizes the Lindenbaum-Sternheimer resonance model [31] to simulate the decay of the Δ^{++} isobar. The model of Ref. [31] determines the mass of the isobar from the probability distribution of Eqn. II.8, where E is the total energy of the system, F the two-body phase space for the isobar and π^- meson, and σ the isobar production cross section, being assumed to be isotopically invariant from the $\pi^+ p$ elastic scattering.

$$\frac{dW}{dM} \approx F(E, M) \times \sigma(M) \tag{II.8}$$

The emission angle, in the center of mass system, is approximated using Eqn. II.5 and Eqn. II.6, where the coefficients a_{nk} have the values quoted in Table II.III, where the order of the normalized number is given in brackets[3]. Decay of the isobar is assumed isotropic.

Kinematics of the non-resonant part of the third interaction channel of Table II.II, as well as the other three-pion production interactions, in the standard DCM are performed with the

[3] *e.g.*, $40693(0) = 0.40693 \times 10^0$, $-36799(+1) = -0.36799 \times 10^1$, etc.

TABLE II.III. Coefficients a_{nk} for $\gamma + p$ Reactions Involving Isobar Production

n k	0	1	2	3
	$\gamma + p \rightarrow N^{*++} + \pi^-$		$0.45 < T_\gamma < 0.985$ GeV	
0	-10306(+1)	79586(+1)	-14797(+2)	82309(+1)
1	32849(+2)	-12572(+3)	16590(+3)	-67871(+2)
2	-75052(+2)	25604(+3)	-27991(+3)	85762(+2)
3	60255(+2)	-16547(+3)	11333(+3)	59727(+1)
	$\gamma + p \rightarrow N^{*++} + \pi^-$		$T_\gamma \geq 0.985$ GeV	
0	-23722(+3)	65800(+3)	-60653(+3)	18604(+3)
1	96890(+3)	-26941(+4)	24983(+4)	-76933(+3)
2	-16219(+4)	45480(+4)	-42498(+4)	13166(+4)
3	13637(+4)	-38469(+4)	36136(+4)	-11243(+4)

statistical model. The energies of the two pions in the center of mass system are determined from the probability distribution of Eqn. II.9, and that of the nucleon, N, is determined from energy conservation.

$$\frac{dW}{dE_{\pi_1} dE_{\pi_2}} \approx (E - E_{\pi_1} - E_{\pi_2}) \frac{E_{\pi_1} E_{\pi_2}}{E} \tag{II.9}$$

Pion energies are randomly sampled by determining the energy of the first pion, as shown in Eqn. II.10, with regards for the maximum allowed pion energy, shown in Eqn. II.11.

$$E_{\pi_1} = m_{\pi_1} + \xi \left(E_{\pi_1}^{max} - m_{\pi_1} \right) \tag{II.10}$$

$$E_{\pi_1}^{max} = \frac{E^2 + m_{\pi_1}^2 - (m_{\pi_2} + m_N)^2}{2E} \tag{II.11}$$

The energy of the second pion, E_{π_2}, is then simulated according to Eqn. II.9 using a Monte Carlo rejection method. The nucleon energy is calculated according to Eqn. II.12, where energy conservation is applied.

$$E_N = E - E_{\pi_1} - E_{\pi_2} \tag{II.12}$$

The physicality of the particle's energy is validated by verifying that the "triangle" law of momentum is fulfilled, being represented by Eqn. II.13. Rejection of the particles' energy occurs and re-sampling is performed if the condition shown by Eqn. II.13 is not met.

$$\left|p_{\pi_1} - p_{\pi_2}\right| \leq p_N \leq \left|p_{\pi_1} + p_{\pi_2}\right| \tag{II.13}$$

The emission angles θ and ϕ of the produced pions are sampled assuming an isotropic distribution of particles in the center of mass system according to Eqn. II.14 and Eqn. II.15, respectively. The emission angles of the nucleon are defined according to momentum conservation, as shown by Eqn. II.16. Ref. [22, 32] provide further detail for differential elementary cross sections. Note that the standard DCM imposes momentum and energy conservation laws for each interaction throughout the simulation.

$$\cos\theta_\pi = 2\xi - 1 \tag{II.14}$$

$$\phi_\pi = 2\pi\xi \tag{II.15}$$

$$\mathbf{p_N} = -\left(\mathbf{p}_{\pi_1} + \mathbf{p}_{\pi_2}\right) \tag{II.16}$$

II.2.5 The Pauli Exclusion Principle

The Pauli exclusion principle is handled within the standard DCM by assuming that nucleons of the target occupy all the energy levels up to the Fermi energy. Each elastic and inelastic interaction is considered to be forbidden if any of the secondary nucleons have energies smaller than that of the Fermi energy, as the nucleons of the target nucleus are assumed to occupy all of the available energy levels. Failure to meet the criteria established by the Pauli exclusion principle requires that the trajectory of the incident particle is traced further from the forbidden point and a new interaction

point, partner, and mode are sampled. This process is repeated until the Pauli exclusion principle is satisfied, or until the incident particle leaves the nucleus.

II.2.6 Nuclear Trawling

Nuclear matter will become depleted throughout the progression of the INC. As a result of the nuclear depletion, neutron and proton density distributions will decrease. This depletion of nuclear matter during the INC process is known as the "trawling" effect. The standard DCM does not, at present, account for changes in the characteristics of the target nucleus during the INC process, neglecting any effects due to nuclear depletion. Detailed analyses of different characteristics of nucleon- and pion-induced reactions for targets ranging from carbon to americium have shown that this effect may be neglected at incident energies below about 5 GeV in the case of heavy targets[4], and below about 1 GeV for light targets[5]. The effects of nuclear depletion become more prominent as the energy per nucleon of the system is increased. The progressive depletion of nuclear matter throughout the INC process has a strong influence on the simulated secondary particles and residual nucleus that are produced by the INC process, and thus must be accounted for at larger energies.

II.2.7 Transition out of the INC Process

The transition out of the fast portion of the reaction plays an important role in spallation reactions. Most conventional cascade-evaporation models, such as the Bertini INC model [33, 34], traces fast particles to some minimal cutoff energy, being $T_{cut} \lesssim 10$ MeV above the Fermi energy, below which the produced particles are absorbed by the target nucleus. Some time-like INC models, such as the INCL model [19], compare the duration of the cascade stage of a reaction with a cut off time instead, limiting the duration of the INC process. The standard DCM utilizes a different criteria, where its criteria is based upon when primary particles leave the cascade.

[4] *e.g.* actinides
[5] *e.g.* carbon

An effective local optical absorptive potential, $W_{opt,\ mod}(r)$, is defined from the local interaction cross section of the particle, including the Pauli blocking effects. The potential is compared to one defined by a phenomenological global optical model, $W_{opt,\ exp}(r)$. The standard DCM characterizes the degree of similarity or difference of these imaginary potentials by the parameter ζ, as defined by Eqn. II.17. The standard DCM considers the primary particle to have left the system as an exciton when the parameter ζ is increased above an empirically chosen value. This approach provides a smoother transition from the fast stage of the reaction to the slow or intermediate stages of the reaction. Utilizing this transition criteria better represents calculated and experimental spectra of secondary nucleons, particularly at low energies and backward angles. Ref. [5, 35] provides more details on this transition.

$$\zeta = \left| \frac{W_{opt,\ mod} - W_{opt,\ exp}}{W_{opt,\ exp}} \right| \tag{II.17}$$

The standard DCM utilizes $\zeta = 0.3$ at incident energies below 150 MeV, however this value was found to not well predict results at larger energies. As a result, a sharp transition, as is done in the Bertini INC model, is utilized at larger energies with a cut-off energy of $T_{cut} = 1$ MeV. To help smooth out the transition around the threshold of 150 MeV, a blend of the two methods is utilized for incident energies ranging within 75 MeV of the transition threshold energy.

The standard DCM contains information on the simulated excited residual nucleus at this transition regarding its number of nucleons and protons, excitation energy E^*, linear momentum $\mathbf{p}$, angular momentum $\mathbf{L}$, and the associated number of excitons, *i.e.* excited particles and holes. Note that the distributions of residual nuclei after the INC process simulated by the standard DCM are observed to be very broad with respect to these characteristics. Completion of the INC process with this model results in a residual nucleus, exciton information, and information regarding the secondary particles that left the system. Appendix E describes how to use the standard DCM in a client program to obtain information regarding the resulting residual nucleus, the associated exciton information, and the produced secondary particles after the simulation of the interaction of an energetic neutron, proton, pion, or photon incident on a target nucleus.

II.3 THE MODIFIED DUBNA CASCADE MODEL

The modified DCM is an improved version [36, 37] of the time-dependent INC model developed at JINR, Dubna. The modified DCM simulates interactions of fast-cascade particles with nucleon spectators of both the target and projectile nuclei and with other fast-cascade particles. In contrast to the standard DCM, the modified DCM utilizes experimental cross sections at energies below $\sim$ 4.5 GeV/nucleon, and the quark-gluon string model (QGSM) [38–43] otherwise to simulate angular and energy distributions of the cascade particles, accounting for the Pauli exclusion principle to verify the validity of an interaction. Note that much of the theory utilized for the modified DCM corresponds to that of the standard DCM. The theory presented below follows that of Ref. [3, 4] and the references therein.

The modified DCM performs its simulations similarly to the standard DCM, however with several distinct differences. The modified DCM utilizes a continuous nuclear density distribution, where the standard DCM approximates the nuclear density as constant in seven concentric nuclear "zones". The intranuclear collision time and the nuclear depletion are also tracked within the modified DCM, providing greater detail than its standard DCM counterpart. The modified DCM lastly tracks the hadron formation time within its INC simulation.

II.3.1 A Mathematical Description of the Modified DCM

An intranuclear cascade is described by the modified DCM via a modified relativistic Boltzmann equation, Eqn. II.18, for a mixture of gases of stable hadrons and short-lived resonances, where a one-particle distribution function for hadrons of type i, $f_i(x, p)$, is used.

$$p^{\mu}\partial_{\mu}f_i(x,p) = \sum_j C_{coll}\left(f_i, f_j\right) + \sum_k R_{k\to i}(f_k) \tag{II.18}$$

The C_{coll} summation of Eqn. II.18 accounts for all two-body collisions and the $R_{k\to i}$ term describes the resonances decaying into particles of type i with 4-momentum, $p = (E, \mathbf{p})$, and 4-coordinate,

$x = (t, \mathbf{x})$. Separate equations are considered for the spectator nucleons of the target, labeled T, the projectile nucleus, labeled P, for participating cascade particles and stable hadrons, labeled S, and also for resonances, labeled R. The system of equations corresponding to each of these is then represented by Eqn. II.19 for the projectile (P) and nucleons of the target (T) and by Eqn. II.20 for the stable hadrons (S) and resonances (R), respectively. Note that the equations for the projectile may be obtained by swapping each occurrence of T and P in Eqn. II.19, where the equation for resonances is obtained similarly regarding Eqn. II.20.

$$p^{\mu}\partial_{\mu}f_{T}(x,p) = -f_{T}(x,p)\left[\sum_{j=R,S,P}\int f_{j}\left(x,p_{j}\right)Q_{Tj}\sigma^{Tj}d\omega_{j}\right] \tag{II.19}$$

$$\begin{aligned} p^{\mu}\partial_{\mu}f_{S}(x,p) = & \\ & -f_{S}(x,p)\left[\sum_{j=T,P,R,S'}\int f_{j}\left(x,p_{j}\right)Q_{Sj}\sigma^{S,j}d\omega_{j}\right] \\ & +\iint\Phi\left(p_{P},p_{T}|x,p,\tau_{f}\right)d\omega_{P}d\omega_{T} \\ & +\sum_{j=T,P}\sum_{k=R,S'}\iint\Phi\left(p_{j},p_{k}|x,p,\tau_{f}\right)d\omega_{j}d\omega_{k} \\ & +\sum_{j=R',S''}\sum_{k=R,S'}\iint\Phi\left(p_{j},p_{k}|x,p,\tau_{f}\right)d\omega_{j}d\omega_{k} \\ & +\sum_{R}\iint f_{R}(x,p_{R})\,\Gamma^{R\to S+S'}\delta^{(4)}(p_{R}-p_{S'}-p)\,d\omega_{S'}d\omega_{R} \end{aligned} \tag{II.20}$$

The quantity $Q_{ij} = \sqrt{\left(p_i p_j\right)^2 - p_i^2 p_j^2}$ in the state equations is related to the relative velocity of the colliding hardons, $\left|\mathbf{v_i} - \mathbf{v_j}\right| = \frac{Q_{ij}}{E_i E_j}$ and $d\omega = d^3\frac{p}{E}$ is the invariant volume element in momentum space. The hadron formation rate, Φ, entering Eqn. II.20 can be expressed in terms of the probability of collision, per unit time, between the colliding hadrons, i and j, as shown by Eqn. II.21, where $\phi\left(x'|x,p,\tau_f\right)$ is the transition probability of detecting a hadron at the point x if the collision took place at the space-time point x' [44, 45].

$$\Phi\left(p_i,p_j|x,p,\tau_f\right) = \int f_i(x',p_i)\,f_j\left(x',p_j\right)Q_{ij}\sigma^{ij}\phi\left(x'|x,p,\tau_f\right)dx' \tag{II.21}$$

II.3.2 The Quark-Gluon String Model

Both the initial and final states of nuclear reactions occurring at ultra-relativistic energies are hadronic states. The intermediate state of said reactions however has the various quark-gluon degrees of freedom unfrozen, thus yielding an uncertain displaced quark-hadron state, event including the possible formation of a quark-gluon plasma [45]. It is this uncertain intermediate state that creates the need for the modified DCM to be merged with a quark-gluon string model, describing both the evolution of the hadronic and quark phase as well as the processes of deconfinement and hadronization. The Quark-Gluon String Model (QGSM) describes nuclear collisions in the approximation of independent quark-gluon strings [44–48], where the collisions of said quark-gluon strings, in the quasi-classical treatment, may yield the formation of bends in the strings. The QGSM simulates these bends by replacing the leading edges of bent strings with energetic hadrons, corresponding to minimal inclusion of quark dynamics [44–48], allowing for the kinetic equations to be mostly written in terms of hadronic states, *i.e.* the equations of Sec. II.3.1 [3]. The inclusion of the QGSM to the modified DCM introduces the concept of a hadron formation time as a result of passing the quark-gluon strings through nuclear matter.

The transition probability, ϕ, describes the evolution of the quark-gluon system, or of a string-like object being formed as a result of the collision, including the process of subsequent hadronization of the system with the hadrons on the mass shell. The set of parameters characterizing this process is denoted by τ_f. Note the transition probability must be obtained by solving an evolution equation allowing for the influence of the nuclear environment and for the possibility of interaction between the formed strings. The hadron formation time, τ_f, is used in the simplest approximation in Eqn. II.22 to characterize this process in the QGSM, assuming that a hadron with spectra distribution, $(1/\sigma)(d\sigma/d\omega)$, is "prepared" at the collision point x', moving freely to the point x and becoming observable, being in the mass-shell state, only after passage of a time τ_f, being generally distributed in accordance with some law $F\left(\tau_f\right)$. The QGSM estimates the distribution of $F\left(\tau_f\right) \approx 1$ [3].

$$\phi\left(x'|x,p,\tau_f\right)=\frac{1}{\sigma}\frac{d\sigma}{d\omega}\delta\left(t-t'-\tau_f\right)\delta^{(3)}\left(\mathbf{x}-\mathbf{x}'-\frac{\mathbf{p}}{E}\left(t-t'\right)\right)F\left(\tau_f\right) \tag{II.22}$$

The prepared hadron does not interact during the formation time, τ_f, with its surroundings. The formation time, τ_f, may be related to a hadron of mass m via the proper time of hadron formation, τ_f^0, as $\tau_f = (E/m)\,\tau_f^0$, and τ_f^0 may be fixed as a result of experimental data parameterizations. The QGSM uses the model of quark-gluon strings to estimate spectral distributions. More details on the QGSM, being utilized by the modified DCM, may be found in Ref. [3] and references therein.

II.3.3 Modified DCM Photon Interactions

Similarly to the standard DCM, the modified DCM utilizes for its $\gamma + n$ and $\gamma + p$ interactions the high-energy Moscow INC event generator [30], allowing the modified DCM to simulate these interactions well with incident energies up to tens of GeV. The modified DCM additionally expands the various allowed photon-induced interactions within the modified DCM to include 56 channels for $\gamma + n$ and $\gamma + p$ interactions each, consisting primary of pion emission, being those listed in Table II.IV. The modified DCM utilizes for its two-body channels, being channels one to 14 of Table II.IV, smooth data approximations through presently available experimental data associated with the photon event generator, where the angular distribution, $d\sigma/d\Omega$, is sampled unambiguously for any E_γ according to the Monte Carlo technique in the emitted particles center-of-mass frame. The three- and four-pion production channels, channels 15 through 21, utilize this same photon event generator within the modified DCM, however the modified DCM uses its own interpolation for integral cross sections to sample the products' angular distribution, $d\sigma/d\Omega$. The modified DCM describes multi-pion channels, channels 22 through 56 of Table II.IV, utilizing the isospin statistical model as realized in the $\gamma + n$ and $\gamma + p$ photon event generator without modification.

TABLE II.IV: Channels of Elementary $\gamma + N$ Interactions Accounted for by the Modified DCM

Channel	$\gamma + p$ Interactions	$\gamma + n$ Interactions
1	$\gamma + p \rightarrow \pi^+ + n$	$\gamma + n \rightarrow \pi^- + p$

Table II.IV, continued

Channel	$\gamma + p$ Interactions	$\gamma + n$ Interactions
2	$\gamma + p \rightarrow \pi^0 + p$	$\gamma + n \rightarrow \pi^0 + n$
3	$\gamma + p \rightarrow \Delta^{++} + \pi^-$	$\gamma + n \rightarrow \Delta^+ + \pi^-$
4	$\gamma + p \rightarrow \Delta^+ + \pi^0$	$\gamma + n \rightarrow \Delta^0 + \pi^0$
5	$\gamma + p \rightarrow \Delta^0 + \pi^+$	$\gamma + n \rightarrow \Delta^- + \pi^+$
6	$\gamma + p \rightarrow \rho^0 + p$	$\gamma + n \rightarrow \rho^0 + n$
7	$\gamma + p \rightarrow \rho^+ + n$	$\gamma + n \rightarrow \rho^- + n$
8	$\gamma + p \rightarrow \eta + p$	$\gamma + n \rightarrow \eta + n$
9	$\gamma + p \rightarrow \omega + p$	$\gamma + n \rightarrow \omega + n$
10	$\gamma + p \rightarrow \Lambda + K^+$	$\gamma + n \rightarrow \Lambda + K^0$
11	$\gamma + p \rightarrow \sum^0 + K^+$	$\gamma + n \rightarrow \sum^0 + K^0$
12	$\gamma + p \rightarrow \sum^+ + K^0$	$\gamma + n \rightarrow \sum^- + K^+$
13	$\gamma + p \rightarrow \eta' + p$	$\gamma + n \rightarrow \eta' + n$
14	$\gamma + p \rightarrow \phi + p$	$\gamma + n \rightarrow \phi + n$
15	$\gamma + p \rightarrow 0\pi^0 + 1\pi^+ + 1\pi^- + p$	$\gamma + n \rightarrow 0\pi^0 + 1\pi^+ + 1\pi^- + n$
16	$\gamma + p \rightarrow 1\pi^0 + 1\pi^+ + 0\pi^- + n$	$\gamma + n \rightarrow 1\pi^0 + 0\pi^+ + 1\pi^- + p$
17	$\gamma + p \rightarrow 2\pi^0 + 0\pi^+ + 0\pi^- + p$	$\gamma + n \rightarrow 2\pi^0 + 0\pi^+ + 0\pi^- + n$
18	$\gamma + p \rightarrow 3\pi^0 + 0\pi^+ + 0\pi^- + p$	$\gamma + n \rightarrow 3\pi^0 + 0\pi^+ + 0\pi^- + n$
19	$\gamma + p \rightarrow 1\pi^0 + 1\pi^+ + 1\pi^- + p$	$\gamma + n \rightarrow 1\pi^0 + 1\pi^+ + 1\pi^- + n$
20	$\gamma + p \rightarrow 2\pi^0 + 1\pi^+ + 0\pi^- + n$	$\gamma + n \rightarrow 2\pi^0 + 0\pi^+ + 1\pi^- + p$
21	$\gamma + p \rightarrow 0\pi^0 + 2\pi^+ + 1\pi^- + n$	$\gamma + n \rightarrow 0\pi^0 + 1\pi^+ + 2\pi^- + p$
22	$\gamma + p \rightarrow 4\pi^0 + 0\pi^+ + 0\pi^- + p$	$\gamma + n \rightarrow 4\pi^0 + 0\pi^+ + 0\pi^- + n$
23	$\gamma + p \rightarrow 2\pi^0 + 1\pi^+ + 1\pi^- + p$	$\gamma + n \rightarrow 2\pi^0 + 1\pi^+ + 1\pi^- + n$
24	$\gamma + p \rightarrow 0\pi^0 + 2\pi^+ + 2\pi^- + p$	$\gamma + n \rightarrow 0\pi^0 + 2\pi^+ + 2\pi^- + n$
25	$\gamma + p \rightarrow 3\pi^0 + 1\pi^+ + 0\pi^- + n$	$\gamma + n \rightarrow 3\pi^0 + 0\pi^+ + 1\pi^- + p$
26	$\gamma + p \rightarrow 1\pi^0 + 2\pi^+ + 1\pi^- + n$	$\gamma + n \rightarrow 1\pi^0 + 1\pi^+ + 2\pi^- + p$
27	$\gamma + p \rightarrow 5\pi^0 + 0\pi^+ + 0\pi^- + p$	$\gamma + n \rightarrow 5\pi^0 + 0\pi^+ + 0\pi^- + n$
28	$\gamma + p \rightarrow 3\pi^0 + 1\pi^+ + 1\pi^- + p$	$\gamma + n \rightarrow 3\pi^0 + 1\pi^+ + 1\pi^- + n$
29	$\gamma + p \rightarrow 1\pi^0 + 2\pi^+ + 2\pi^- + p$	$\gamma + n \rightarrow 1\pi^0 + 2\pi^+ + 2\pi^- + n$
30	$\gamma + p \rightarrow 4\pi^0 + 1\pi^+ + 0\pi^- + n$	$\gamma + n \rightarrow 4\pi^0 + 0\pi^+ + 1\pi^- + p$
31	$\gamma + p \rightarrow 2\pi^0 + 2\pi^+ + 1\pi^- + n$	$\gamma + n \rightarrow 2\pi^0 + 1\pi^+ + 2\pi^- + p$
32	$\gamma + p \rightarrow 0\pi^0 + 3\pi^+ + 2\pi^- + n$	$\gamma + n \rightarrow 0\pi^0 + 2\pi^+ + 3\pi^- + p$
33	$\gamma + p \rightarrow 6\pi^0 + 0\pi^+ + 0\pi^- + p$	$\gamma + n \rightarrow 6\pi^0 + 0\pi^+ + 0\pi^- + n$
34	$\gamma + p \rightarrow 4\pi^0 + 1\pi^+ + 1\pi^- + p$	$\gamma + n \rightarrow 4\pi^0 + 1\pi^+ + 1\pi^- + n$
35	$\gamma + p \rightarrow 2\pi^0 + 2\pi^+ + 2\pi^- + p$	$\gamma + n \rightarrow 2\pi^0 + 2\pi^+ + 2\pi^- + n$
36	$\gamma + p \rightarrow 0\pi^0 + 3\pi^+ + 3\pi^- + p$	$\gamma + n \rightarrow 0\pi^0 + 3\pi^+ + 3\pi^- + n$
37	$\gamma + p \rightarrow 5\pi^0 + 1\pi^+ + 0\pi^- + n$	$\gamma + n \rightarrow 5\pi^0 + 0\pi^+ + 1\pi^- + p$
38	$\gamma + p \rightarrow 3\pi^0 + 2\pi^+ + 1\pi^- + n$	$\gamma + n \rightarrow 3\pi^0 + 1\pi^+ + 2\pi^- + p$
39	$\gamma + p \rightarrow 1\pi^0 + 3\pi^+ + 2\pi^- + n$	$\gamma + n \rightarrow 1\pi^0 + 2\pi^+ + 3\pi^- + p$
40	$\gamma + p \rightarrow 7\pi^0 + 0\pi^+ + 0\pi^- + p$	$\gamma + n \rightarrow 7\pi^0 + 0\pi^+ + 0\pi^- + n$
41	$\gamma + p \rightarrow 5\pi^0 + 1\pi^+ + 1\pi^- + p$	$\gamma + n \rightarrow 5\pi^0 + 1\pi^+ + 1\pi^- + n$
42	$\gamma + p \rightarrow 3\pi^0 + 2\pi^+ + 2\pi^- + p$	$\gamma + n \rightarrow 3\pi^0 + 2\pi^+ + 2\pi^- + n$

Table II.IV, continued

Channel	$\gamma + p$ Interactions	$\gamma + n$ Interactions
43	$\gamma + p \rightarrow 1\pi^0 + 3\pi^+ + 3\pi^- + p$	$\gamma + n \rightarrow 1\pi^0 + 3\pi^+ + 3\pi^- + n$
44	$\gamma + p \rightarrow 6\pi^0 + 1\pi^+ + 0\pi^- + n$	$\gamma + n \rightarrow 6\pi^0 + 0\pi^+ + 1\pi^- + p$
45	$\gamma + p \rightarrow 4\pi^0 + 2\pi^+ + 1\pi^- + n$	$\gamma + n \rightarrow 4\pi^0 + 1\pi^+ + 2\pi^- + p$
46	$\gamma + p \rightarrow 2\pi^0 + 3\pi^+ + 2\pi^- + n$	$\gamma + n \rightarrow 2\pi^0 + 2\pi^+ + 3\pi^- + p$
47	$\gamma + p \rightarrow 0\pi^0 + 4\pi^+ + 3\pi^- + n$	$\gamma + n \rightarrow 0\pi^0 + 3\pi^+ + 4\pi^- + p$
48	$\gamma + p \rightarrow 8\pi^0 + 0\pi^+ + 0\pi^- + p$	$\gamma + n \rightarrow 8\pi^0 + 0\pi^+ + 0\pi^- + n$
49	$\gamma + p \rightarrow 6\pi^0 + 1\pi^+ + 1\pi^- + p$	$\gamma + n \rightarrow 6\pi^0 + 1\pi^+ + 1\pi^- + n$
50	$\gamma + p \rightarrow 4\pi^0 + 2\pi^+ + 2\pi^- + p$	$\gamma + n \rightarrow 4\pi^0 + 2\pi^+ + 2\pi^- + n$
51	$\gamma + p \rightarrow 2\pi^0 + 3\pi^+ + 3\pi^- + p$	$\gamma + n \rightarrow 2\pi^0 + 3\pi^+ + 3\pi^- + n$
52	$\gamma + p \rightarrow 0\pi^0 + 4\pi^+ + 4\pi^- + p$	$\gamma + n \rightarrow 0\pi^0 + 4\pi^+ + 4\pi^- + n$
53	$\gamma + p \rightarrow 7\pi^0 + 1\pi^+ + 0\pi^- + n$	$\gamma + n \rightarrow 7\pi^0 + 0\pi^+ + 1\pi^- + p$
54	$\gamma + p \rightarrow 5\pi^0 + 2\pi^+ + 1\pi^- + n$	$\gamma + n \rightarrow 5\pi^0 + 1\pi^+ + 2\pi^- + p$
55	$\gamma + p \rightarrow 3\pi^0 + 3\pi^+ + 2\pi^- + n$	$\gamma + n \rightarrow 3\pi^0 + 2\pi^+ + 3\pi^- + p$
56	$\gamma + p \rightarrow 1\pi^0 + 4\pi^+ + 3\pi^- + n$	$\gamma + n \rightarrow 1\pi^0 + 3\pi^+ + 4\pi^- + p$

Upon being absorbed by two nucleons or having an inelastic interaction with a nucleon according to one of the channels listed in Table II.IV, the modified DCM simulates several cascade nucleons, pions, or other meson and resonance progeny listed in Table II.IV, depending on the sampled reaction channel according to the corresponding cross sections at the given photon energy. The cascade progeny are simulated to interact further with intranuclear nucleons or leave the nucleus, depending on the progeny's sampled characteristics, namely their momenta and coordinates relative to the depleted target nucleus. The continued interaction of cascade progeny with intranuclear nucleons is concurrently simulated in the same manner as all other incident nucleons or light- to heavy-ions, where the incident photon is assumed to interact no further, being removed from the simulation.

II.4 COALESCENCE PHYSICS

The coalescence model aids in the simulation of spallation reactions by estimating the formation of composite nuclei via pseudo-deterministic techniques. Progeny created during the cascade

process may undergo final-state interactions to coalesce, forming high-energy composite nuclei[6]. It is then assumed that the INC progeny are created nearly simultaneously and within a small distance, $\delta\mathbf{r}$, of each other. Few processes may make such assumptions, however high-energy spallation reactions may utilize these assumptions for their INC progeny as the INC process occurs very quickly as a result of the incident particle's energy.

It is commonly assumed that nucleons emitted during nuclear collisions have low interaction probabilities. This assumption becomes less justified as the density of cascade nucleons increases [7], *i.e.* in the presence of high nucleon multiplicities [5], resulting in a dense nucleonic field. The emitted nucleons within this high-energy field have a non-zero interaction probablility, where possible final-state interactions result in the formation of composite fragments. The coalescence model accounts for the formation of these composites via allowable final-state interactions. Experimental results have also shown the creation of high-energy deuterium and tritium composites, where the formation of these fragments is suspected to be caused by the coalescence of the emitted cascade nucleons [5]. The coalescence model plays an important role in reactions involving large nucleon multiplicities, such as in high-energy heavy-ion reactions.

The coalescence model discussed here predicts the formation of light fragments via final-state interactions by a sudden approximation in quantum mechanics. This is accomplished by predicting an approximate overlap in the nucleon's wave functions in momentum space. Fig. II.3 demonstrates an overlap in two particles' wave function probabilities, and in addition demonstrates the approximations discussed later in Sec. II.4.1.

The wave function of the interacting nucleons is described by Eqn. II.23 [8], representing the initial state of the nucleons, and the wave function of the resulting composite, in its initial state ψ_i^C, is described by the overlap in, or multiplication of, the initial N nucleons' wave functions, as shown in Eqn. II.24.

$$\psi_i\left(\mathbf{p},\mathbf{r}\right) = e^{\frac{i}{\hbar}\mathbf{p}\cdot\mathbf{r}} \tag{II.23}$$

[6]Composites formed may also be referred to as light fragments.

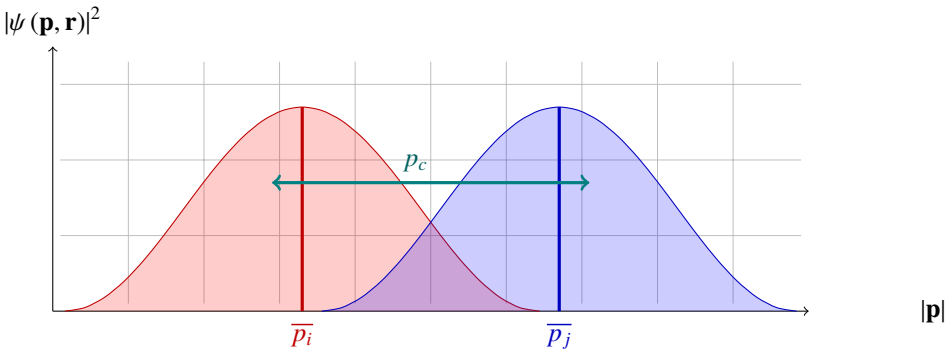

FIGURE II.3. Demonstration of the quantum wave overlap for a two-body interaction and the approximation made by the Coalescence model.

$$\psi_i^C(\mathbf{p},\mathbf{r}) = \prod_{i=1}^{N} \psi_i(\mathbf{p},\mathbf{r}) \tag{II.24}$$

The final-state of the resulting composite is described by Eqn. II.25. Note that Eqn. II.25 is only valid for composites moving in a free space, being unaffected by the surrounding medium, where the term ϕ_0^C, a Bethe-Goldstone type equation, represents the Pauli-quenching effect that could prevent formation of the composite [8]. The term ϕ_0^C may also include in-medium corrections, as well as other corrections, to ensure validity of Eqn. II.25. Further details regarding these equations can be found in Ref. [8]

$$\psi_f^C(\mathbf{p_c},\mathbf{r_c}) = e^{\frac{i}{\hbar}\mathbf{p_c}\cdot\mathbf{r_c}}\phi_0^C(\mathbf{r_c}) \tag{II.25}$$

The formation probability of composites from N interacting particles is then determined by Eqn. II.26 [8]. The term $\rho^C(\mathbf{p},b)$ represents the particle distribution, or particle density, in momentum space, being described by Eqn. II.27 [7]. The particle density in momentum space is related to the one-particle distribution function, integrated over all space.

$$W_c(\mathbf{p_c},b) = \int_{All\ \mathbf{p_i}} \cdots \int \left|\left\langle \psi_i^C | \psi_f^C \right\rangle\right|^2 \prod_{i=1}^{N} \rho_i^C(\mathbf{p},b)\, d\mathbf{p_i} \tag{II.26}$$

$$\rho^C(\mathbf{p}, b) = \int f^C(\mathbf{r}, \mathbf{p}, b)\, d\mathbf{r} \tag{II.27}$$

II.4.1 The Coalescence Model

The coalescence model discussed here performs the above process by approximating the overlap in the nucleons' wave function as a single-valued parameter for each composite being considered. Using the above theory with this approximation, the formation probability of a single composite, A, given an allowed final-state interaction[7] $i + j + ... + N \rightarrow A$ consisting of N interacting particles, with momentum $\mathbf{p_A}$ and impact parameter b [1], is then described in a general sense by Eqn. II.28 [1, 7].

$$W_A(\mathbf{p_A}, b) = \int_{All\ \mathbf{p_i}} \cdots \int \prod_{i=1}^{N} \left(\prod_{j=2,\ j\neq i}^{N} \Theta\left(p_c - \left|\mathbf{p_i} - \mathbf{p_j}\right|\right) \right) \rho^C(\mathbf{p_i}, b)\, \delta\left(\sum_{i=1}^{N} \mathbf{p_i} - \mathbf{p_A}\right) d\mathbf{p_i} \tag{II.28}$$

The Θ term enforces the approximation that the momentum difference of interacting nucleons will fall within a small momentum sphere, p_c, often referred to as the coalescence radii for the particular composite. This term represents the quantum wave approximation being made, where the interacting particles' average moment all fall within some difference of each other smaller than p_c. The coalescence radii, p_c, effectively represents the interaction range of nucleons when forming a composite nuclei, thus restricting or "weighting" the number of formable composites [8]. The $\rho^C(\mathbf{p_i}, b)$ term shown in Eqn. II.28 represents the density of interacting particles being considered for coalescence, and after integration yields the total number of particles. The δ term in Eqn. II.28 enforces the conservation of momentum, ensuring only allowed composite states can occur. Note that the integration over all interacting particles' momenta space in Eqn. II.28 integrates out angular dependence of the nucleons, meaning that the particles could be moving in

[7] This "general" final-state interaction could be a neutron coalescing with a proton to form a deuteron, an already coalesced deuteron coalescing with a proton to form helion, etc.

nearly opposite directions when they coalesce. The disregard for the interacting particles' angular directions may result in an over-estimation of composite formation, however adjustment of the coalescence radii can be used to limit this over-estimation and reduce the error introduced by this assumption.

The coalescence radii, p_c, are chosen such that effects due to in-medium corrections and other potential corrections to theory, as discussed above, are accounted for when used in conjunction with the INC model of the latest version of the Cascade Exciton Model, CEM03.03F. The coalescence radii for each of the composites considered by the model discussed here are shown in Table II.V, where the default set of coalescence radii is chosen for all incident energies of the reaction except when the incident energy lies within the range of 300 MeV and 1 GeV, at which energy the special set is chosen. The coalescence radii are used in the quantum wave approximation, where particles that have differences in their mean momenta smaller than p_c are assumed to coalesce. Fig. II.3 demonstrates the approximation for a two-particle final-state interaction, where the curves presented are the "actual" quantum waves of the interacting particles.

Note that use of the coalescence model with different INC models may require a reparameterization of the coalescence radii for each of the considered composites due to a potential difference in the resulting nucleon distributions. Utilizing the coalescence model with a different set of coalescence radii may easily be done by providing the object with a `coalescenceOptions` object. The values chosen here are based off a parameterization for comparisons to the interaction of 1.04 *GeV/nucleon Ne* incident on a U target [1], and some other reactions [7]. The parameterized coalescence radii appear to satisfactorily describe high-energy complex particle production [1]. Parameterization of the coalescence radii revealed little dependence on the energy of the nuclear collisions and little to no dependence on the nucleonic composition of the convoled nuclei [7],
The final-state interactions considered by the coalescence model for each composite are shown in Table II.VI. The coalescence model considers each of the final-state interactions by flagging nucleons that lie within the composite's respective coalescence radii p_c and with appropriate isotopic composition. Composites are "created" when the group of nucleons has met this criteria, upon

TABLE II.V. Coalescence Radii for Composite Particle Formation

Composite	Default Set (*MeV/c*)	Special Set (*MeV/c*)
Deuteron	90	150
Triton	108	175
Helion	108	175
Helium	130	205
Light Fragments (^6He, ^6Li, ^{7}Li, ^7Be)	175	250

which the composite may or may not be used to further predict the formation of light fragments. Note that the expanded model was developed in recent years in Ref. [49, 50], providing significant improvements to the prediction of light fragment production.

TABLE II.VI. Final State Interactions Considered by the Coalescence Model

ID	Final State Interaction		
	(Base Model)		
1	$n + p$	$\rightarrow$	d
2	$d + d$	$\rightarrow$	^4He
3	$d + n$	$\rightarrow$	t
4	$d + p$	$\rightarrow$	^3He
5	$t + p$	$\rightarrow$	^4He
6	$^3He + n$	$\rightarrow$	^4He
	(Expanded Model)		
7	$^4He + t$	$\rightarrow$	^7Li
8	$^4He + ^3He$	$\rightarrow$	^7Be
9	$^4He + d$	$\rightarrow$	^6Li
10	$t + t$	$\rightarrow$	^6He
11	$t + ^3He$	$\rightarrow$	^6Li

The model uses real data [1] about all emitted cascade nucleons[8], namely their momenta and nucleonic composition, to determine the formation of composites. As depicted in Table II.VI, the coalescence model first considers the formation of high-energy deuterium, then considers the formation of helium, tritium, and helion. Use of the expanded coalescence model [49, 50] further considers the formation of ^{6}He, ^{6}Li, ^{7}Li, and ^{7}Be composites according to how the final-state

[8]Note that the nucleons considered for coalescence do *not* need to originate from the INC process, but rather from can originate from *any* process [8].

interactions are presented in Table II.VI, respectively.

II.5 FERMI BREAK-UP PHYSICS

The Fermi break-up model [14] predicts particle emission from energetic light nuclei. The Fermi break-up model is traditionally used to describe the de-excitation of residual and compound nuclei. These nuclei are often the byproducts of the fast and intermediate stages of high energy spallation reactions [6]. Residuals that are sufficiently small[9] will undergo Fermi break-up emission instead of other de-excitation processes. The Fermi break-up model is used to describe emission from these light nuclei because the model does not require a large number of nucleons to be statistically accurate, as is assumed in many emission models, such as the Generalized Evaporation Model (GEM), as implemented in GEM2 [51]. The Fermi break-up model is necessary to describe progeny emission from light nuclei to account for the unique structure of the nuclei and potential alpha clustering that may occur in these light nuclei, such as in carbon and oxygen [4].

The Fermi break-up model relies on the assumption that the excited nucleus has a kinetic energy that is comparable to its nucleon's binding energy. If this condition is met, the nucleus is assumed to disintegrate into N highly energetic fragments to predict various final states of the nucleus [14]. Note that all details and formulas regarding the Fermi break-up model used by the CEM, GSM, and LAQGSM event generators can be found in Ref. [52]. The theory of Ref. [52] is quoted here for documentation purposes. Note also that the disintegration of an excited nucleus can be predicted with only information regarding its nucleonic composition and kinetic energy.

II.5.1 Allowed Fermi Break-Up Channels

Allowed final state channels are determined based on the total kinetic energy of all fragments. The total kinetic energy, E_k, of all produced fragments in a given channel is depicted by Eqn. II.29

[9] Such residuals have a low number of nucleons. The Fermi break-up model, by default, assumes nucleis wit h $A \leq 12$ are small, however clients may alter this threshold as desired.

[52].

$$E_{kin} = E_{total} - E_{Coulomb} - \sum_{i=1}^{N} (m_i + \epsilon_i) \tag{II.29}$$

Where E_{tot} is the total energy[10] of the nucleus in the rest frame, $E_{coulomb}$ the coulomb barrier of the channel, and m_i and ϵ_i the masses and excitation energies of all N produced fragments in the disintegration channel. The disintegration channel is considered if the total kinetic energy of all N fragments is physically allowable upon disintegration of the nucleus[11]. The coulomb barrier is approximated by the Fermi break-up model with Eqn. II.30 [52].

$$E_{Coulomb} = \frac{3}{5}\frac{e^2}{r_0}(1+\kappa)^{-1/3}\left(\frac{Z^2}{A^{1/3}} - \sum_{i=1}^{N}\frac{Z_i^2}{A_i^{1/3}}\right) \tag{II.30}$$

Where κ is a ratio of the decaying to the normal, ground-state system volumes[12], Z and Z_i the number of protons in the nucleus and each fragment produced, respectively, A and A_i the number of nucleons in the nucleus and each fragment produced, respectively, $e = \sqrt{4\pi\alpha\epsilon_0\hbar c}$ with α as the fine-structure constant, and the radius parameter $r_0 = 1.3$. Note that the radius parameter can be modified prior to simulation with the new object-oriented Fermi break-up model.

II.5.2 Fermi Break-Up Channel Probability

The total probability per unit time for an excited nucleus to break up into N fragments in the final state is given by Eqn. II.31 [52].

$$W(E,N) = \left(\frac{V}{\Omega}\right)^{N-1}\rho_N(E) \tag{II.31}$$

[10]The total energy of the nucleus is the sum of its kinetic energy and ground state mass.

[11]This requirement enforces that the total kinetic energy of all fragments is positive. Note the total kinetic energy represents the kinetic energy available for all of the produced fragments in a specific channel.

[12]Note that a value of $\kappa = 1$ is assumed in the Fermi break-up simulation used by CEM, GSM, and LAQGSM. Other codes that use the Fermi break-up model may use different values of κ. GEANT4, for example, uses $\kappa = 6$ [53].

Where V is the volume of the decaying system[13], $\Omega = (2\pi\hbar)^3$ is a type of normalization volume, and $\rho_N(E)$ is the density of final states. The density of the final states can be defined as a product of the phase space factor, M_N, the spin factor, S_N, and the permutation factor, G_N, as shown in Eqn. II.32.

$$\rho_N(E) = M_N(E) S_N G_N \tag{II.32}$$

The phase space, spin, and permutation factors are represented by Eqn. II.33, Eqn. II.34, and Eqn. II.35, respectively.

$$M_N(E) = \int_{\text{All } \mathbf{p_i}} \cdots \int \delta\left(\sum_{i=1}^{N} \mathbf{p_i}\right) \delta\left(E - \sum_{i=1}^{N} \sqrt{|\mathbf{p_i}|^2 + m_i^2}\right) \prod_{i=1}^{N} \mathbf{p_i} \tag{II.33}$$

$$S_N = \prod_{i=1}^{N} (2s_i + 1) \tag{II.34}$$

$$G_N = \prod_{j=1}^{k} \frac{1}{n_j!} \tag{II.35}$$

The $\mathbf{p_i}$ terms in the phase space factor, represented by Eqn. II.33, represent the momentum vector of the individual fragments produced in the disintegration channel. The spin factor, shown by Eqn. II.34, yields the number of states with different spin orientations that must be considered for each of the possible fragment spin states [14], and the permutation factor, shown by Eqn. II.35, accounts for the indistinguishable [54] identical particles in the final state. The n_j term in Eqn. II.35 represents the number of j-type particles produced[14], and the k index represents the number of different particle types produced in the channel[15]. Eqn. II.33 reduces to Eqn. II.36 when approximating the Fermi break-up channels in a non-relativistic frame [52]. The phase space factor further reduces to Eqn. II.37 when analyzing the integration of Eqn. II.36 analytically [55].

[13]The volume of the decaying system is calculated by $V = \kappa V_0$ where V_0 is the volume of the nucleus in its ground state, estimated by a sphere, given a radius of $r = r_0 A^{1/3}$, where r_0 is that described in Sec. II.5.1.

[14]*E.g.* if four particles ($N = 4$) reach a final state, consisting of two helium atoms, one deuterium atom, and one lithium atom ($k = 3$), the permutation factor will be $G_4 = \frac{1}{2!\ 1!\ 1!} = \frac{1}{2}$.

[15]In equation form, k is defined when the condition $N = \sum_{j=1}^{k} n_j$ is satisfied.

$$M_N(E) = \int_{All\ \mathbf{p_i}} \cdots \int \delta\left(\sum_{i=1}^{N} \frac{|\mathbf{p_i}|^2}{2m_i} - E\right) \prod_{i=1}^{N} \mathbf{p_i} \tag{II.36}$$

$$M_N(E) = \left(\frac{1}{\sum_{i=1}^{N} m_i} \prod_{i=1}^{N} m_i\right)^{3/2} \left(\frac{(2\pi)^{\xi/2}}{\Gamma(\xi/2)}\right) E^{\xi/2-1} \tag{II.37}$$

Where $\xi = 3(N-1)$, representing the phase space dimensionality of the state [14], and $\Gamma(\xi)$ is the gamma function. Back substituting Eqn. II.32, Eqn. II.34, Eqn. II.35, and Eqn. II.37 can be done to obtain the channel probability. The resulting equation from these substitutions, where the energy of the channel is the kinetic energy from Eqn. II.29, is shown in Eqn. II.38, depicting the probability for an excited nucleus to break into N fragments with masses m_i, where i ranges from 1 to N.

$$W(E_{kin}, N) = S_N G_N \left(\frac{V}{\Omega}\right)^{N-1} \left(\frac{1}{\sum_{i=1}^{N} m_i} \prod_{i=1}^{N} m_i\right)^{3/2} \left(\frac{(2\pi)^{\xi}}{\Gamma(\xi)}\right) E_{kin}^{\xi-1} \tag{II.38}$$

II.5.3 Fermi Break-Up Channel Selection

Fermi break-up channels are selected randomly according to the Monte Carlo method, where selection in based on the condition that the available kinetic energy in Eqn. II.29 is greater than zero and based on the probability given by Eqn. II.38. The kinematics for each produced fragment are then determined using Kopylov's method [52, 56] according to the *relativistic N*-body phase space distribution shown in Eqn. II.33. Note that the Fermi break-up channel is selected by approximating the fragments in a non-relativistic frame, while their kinematics are treated relativistically. This distinction is not particularly alarming considering relativistic treatment reduces to non-relativistic treatment at low to mid-range kinetic energies, *i.e.* the inconsistent treatment of fragment kinematics does not affect results much for low E_{kin} in Eqn. II.29. This inconsistency can nonetheless be corrected for because the phase space factor in Eqn. II.33 can be calculated numerically [52]. Note

also that a coulomb repulsion is accounted for regarding charged fragment kinematics.

Fragments are assumed to be emitted isotropically in the center of mass system of the disintegrating nucleus [1, 4] and are statistically independent [14]. Kinetic energies of each fragment are chosen such that momentum and energy are conserved. Fragments produced via Fermi break-up are most often formed in their ground state or low-lying energy levels. In rare cases, use of Fermi break-up model may create unstable fragments. Produced unstable fragments[16] are subsequently decayed in the version of the Fermi break-up model discussed here [4, 5]. Note that although angular momentum ought to be conserved in this process, it was found that there are few numerical differences when neglecting its conservation, allowing for simplified modeling [14]. The Fermi break-up code documentation further describes how to use the Fermi break-up model in a client program to obtain information regarding the predicted fragments, and additionally how meaningful results from the Fermi break-up simulation are obtained.

II.6 PREEQUILIBRIUM PHYSICS

Preequilibrium physics is utilized to predict the emission of particles during the decay of highly excited residual nuclei, often being formed as the byproduct of the IntraNuclear Cascade stage in high-energy spallation simulations. preequilibrium physics are often utilized to describe possible nuclear transitions, whereby the number of excitons n may change, with $\Delta n = 0$ and $\Delta n = \pm 2$, as well as all subsequent emissions of n, p, *etc.*, being modeled up to ^{28}Mg may take place. Fig. II.4 provides a rough depiction of this process, where the particle emission may be observed as a result of exciton transitions. Much of the preequilibrium physics discussed here is based primarily on the Modified Exciton Model (MEM) [9, 10], utilizing a Monte Carlo technique for its simulations [57]. Much of the physics discussed here follows that described by Ref. [58], however further detail regarding portions of the model may be found in Ref. [4] and Ref. [5], each well-documenting features of the preequilibrium model prior to the upgrades discussed in Ref. [58].

[16] Such fragments include ^{5}He, ^{5}Li, ^{8}Be, ^{9}B, *etc.* [5]

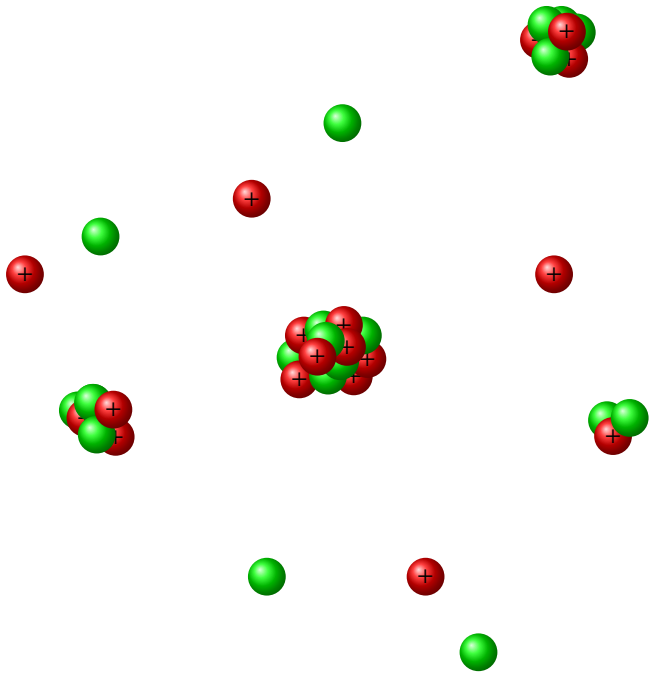

FIGURE II.4. A theoretical depiction of preequilibrium particle emission that occurs as a result of exciton transitions in an excited residual nucleus.

The Modified Exciton Model (MEM) assumes that the probability, P, of finding the nuclear system at a moment in time, t, in the $E\alpha$ state is given by Eqn. II.39, where $\lambda(E\alpha, E\alpha')$ is the transition probability rate. Note that Eqn. II.39 assumes the "memory" time, τ_{mem}, of the system is small compared to the characteristic time for intranuclear transitions, τ_{coll}, being $\tau_{coll} \approx \hbar/\lambda(E\alpha, E\alpha')$, being applicable also when the time moment is $t >> \hbar/\lambda(E\alpha, E\alpha')$, resulting in a Markovian process [58] due to the condition of Eqn. II.39.

$$\frac{\delta}{\delta t} P(E, \alpha, t) = \sum_{\alpha \neq \alpha'} \lambda(E\alpha, E\alpha') P(E, \alpha', t) - \lambda(E\alpha', E\alpha) P(E, \alpha, t) \tag{II.39}$$

The energy-conserving transition probability rate, $\lambda(E\alpha, E\alpha')$ established in time-dependent perturbation theory to the first order is given by Eqn. II.40.

$$\lambda(E\alpha, E\alpha') = \frac{2\pi}{\hbar} |< E\alpha|V|E\alpha' >|^2 \omega_\alpha(E) \tag{II.40}$$

The matrix element of Eqn. II.40, $< E\alpha|V|E\alpha' >$, is believed to be a smooth function in energy,

and $\omega_\alpha(E)$ is defined as the density of the final state of the system. The MEM uses effectively the relationship of the master equation, Eqn. II.39, with the Markovian random processes. Transition probabilities are assumed to be time-independent, thus the waiting time for the system to discontinuously jump from the $E\alpha$ state to the $E\alpha'$ state follows an exponential distribution with average lifetime $\hbar/\Lambda(\alpha, E) = \hbar/\sum_{\alpha}^{'} \lambda(E\alpha, E\alpha')$. It is possible to generalize the exciton model in this treatment of the system to all nuclear transitions with $\Delta n = 0$ and $\Delta n = \pm 2$, possibly containing multiple subsequent particle emissions, thus causing the depletion of nuclear states. The system given this treatment is then described by Eqn. II.41. Eqn. II.41 may be utilized to estimate particle emission rates, λ_c^j and the exciton transition rates, λ_+, λ_0, and λ_-.

$$\begin{aligned}\frac{\delta}{\delta t}P(E,\alpha,t) = &-\Lambda(n,E)\,P(E,n,t) + \lambda_+(n-2,E)\,P(E,n-2,t)\\ &+\lambda_0(n,E)\,P(E,n,t) + \lambda_-(n+2,E)\,P(E,n+2,t)\\ &+\sum_j \int\int \lambda_c^j(n,E,T)\,P\left(E',n+n_j,t\right)\delta\left(E'-E-B_j-T\right)dE'dT\end{aligned} \tag{II.41}$$

II.6.1 Particle Emission

The emission width of a nucleon of type j into the continuum, Γ_j, is estimated by the detailed balance principle according to Eqn. II.42, being the sum of all partial emission probabilities in the continuum, λ_c^j, as shown by Eqn. II.43 [4]. The particle type to be emitted is randomly sampled according to the particles' partial emission rate, via the Monte Carlo technique. Note the particles considered for emission follows those of Table II.VII.

$$\Gamma_j(p,h,E) = \int_{V_j^c}^{E-B_j} \lambda_c^j(p,h,E,T)\,dT \tag{II.42}$$

$$\lambda_c^j(p,h,E,T) = \frac{2s_j+1}{\pi^2\hbar^3}\mu_j \Re(p,h)\,\frac{\omega\left(p-1,h,E-B_j-T\right)}{\omega(p,h,E)}\,T\sigma_{inv}(T) \tag{II.43}$$

The p, h, E, B_j, V_j^c, T, s_j, μ_j, ω, and σ_{inv} terms of Eqn. II.42 and Eqn. II.43 each represent the number of particle excitons, number of hole excitons, internal energy of the excited nucleus, binding energy, Coulomb barrier, kinetic energy of the nucleus, spin, reduced mass of the emitted particle, the level density of the n-exciton state, and the inverse reaction cross section, respectively. A hybdrid of Kalbach-NASA systematics are utilized for the inverse reaction cross section, whose NASA systematics are described by Eqn. II.44. The incorporated hybrid Kalbach-NASA systematics are discussed in greater detail in Ref. [58].

$$\sigma_{inv}^{NASA} = \pi r_0^2 \left(A_p^{1/3} + A_T^{1/3} + \delta_T\right)^2 \left(1 - R_c \frac{B_T}{T_{cm}}\right) X_m \tag{II.44}$$

The $\mathfrak{R}$ term of Eqn. II.43 ensures the condition for the exciton chosen to be the particle of type j, being easily calculated by the Monte Carlo technique. Eqn. II.43 is utilized for both neutron and proton emission. Emission of other complex particles utilizes again Eqn. II.43, however the level density of the n-exciton state is scaled, introducing a new factor to its equation, γ_j. The γ_j scaling factor is described roughly by Eqn. II.45, where it is estimated by assuming the p_j excited nucleons (excitons) are able to condense with probability γ_j to form a complex particle. Note Eqn. II.45 is a rough estimation of a "condensation" probability that is defined by a parameterization over a mesh of residual nuclei energies and mass numbers [1]. Determinations of the F_j scaling factor to provide better complex preequilibrium progeny emission was performed in Ref. [58], being specifically designed for use in this preequilibrium model when utilized by the predecessor to GSM, the CEM event generator.

$$\gamma_j = F_j p_j^3 \left(\frac{p_j}{A}\right)^{p_j - 1} \tag{II.45}$$

The level density of the n-exciton state ω follows Eqn. II.46 assuming an equidistant level scheme with the single-particle density, g [59]. Substitution of Eqn. II.46 into Eqn. II.41 may be done to obtain the transmission rates, λ_c^j.

$$\omega(p,h,E) = \frac{g\,(gE)^{p+h-1}}{p!h!\,(p+h-1)!} \tag{II.46}$$

II.6.2 Exciton Transitions

The partial transition probability, $\lambda_{\Delta n}$, changing the exciton number of a preequilibrium nucleus with excitation energy, E, and number of excitons $n = p + h$ by Δn is given by Eqn. II.47, according to Eqn. II.40.

$$\lambda_{\Delta n}(p,h,E) = \frac{2\pi}{\hbar}\,|M_{\Delta n}|^2\,\omega_{\Delta n}(p,h,E) \tag{II.47}$$

The number of states, ω must account for selection rules for intranuclear exciton-exciton scattering for the transition rates of Eqn. II.47. The resulting set of equations, Eqn. II.48, have been developed by Williams [60] and later corrected for the Pauli exclusion principle and indistinguishability of identical excitons [61, 62].

$$\begin{aligned}
\omega_+(p,h,E) =& \frac{1}{2}g\frac{\left[gE-\xi(p+1,h+1)\right]^2}{n+1}\left[\frac{gE-\xi(p+1,h+1)}{gE-\xi(p,h)}\right]^{n-1}\\
\omega_0(p,h,E) =& \frac{1}{2}g\frac{gE-\xi(p,h)}{n}\left[p(p-1)+4ph+h(h-1)\right]\\
\omega_-(p,h,E) =& \frac{1}{2}gph(n-2)
\end{aligned} \tag{II.48}$$

The $\xi(p,h)$ term in the above equations is defined by Eqn. II.49. An estimation of the matrix element, $|M_{\Delta n}|^2$, is made by neglecting elements with different Δn terms, thus $M_+ = M_- = M_0 \equiv M$. The value of M for a given nuclear state provided this estimation is then determined by associating the partial transition probability, $\lambda_+(p,h,E)$, with the probability for quasi-free scattering of a nucleon above the Fermi level on a nucleon of the target nucleus. Eqn. II.50 is the result of this association.

$$\xi(p,h) = \frac{p^2+h^2+p-h}{4} - \frac{h}{2} \tag{II.49}$$

$$\frac{<\sigma(v_{rel})\,v_{rel}>}{V_{int}} = \frac{\pi}{\hbar}|M|^2\,\frac{g\left[gE-\xi(p+1,h+1)\right]}{n+1}\left[\frac{gE-\xi(p+1,h+1)}{gE-\xi(p,h)}\right]^{n-1} \tag{II.50}$$

The V_{int} term of Eqn. II.50 is the interaction volume, estimated as $V_{int} = \frac{4}{3}\pi\left(2r_c + \frac{\lambda}{2\pi}\right)^3$, with the de Broglie wavelength, $\frac{\lambda}{2\pi}$, corresponding to the relative velocity, $v_{rel} = \sqrt{2\frac{T_{rel}}{m_N}}$. A value of $r_c = 0.6$ fm is utilized to estimate the order of the nucleon radius in the model. The averaging utilized to estimate the quasi-free nucleon-nucleon scattering, the left hand side of Eqn. II.50, is carried out over all excited states, accounting for the Pauli exclusion principle in the approximation represented by Eqn. II.51. The energy-dependent cross section utilized for the quasi-free nucleon-nucleon scattering is represented by Eqn. II.52, accounting for the relative velocity, v_{rel}, of the excited nucleon and the target nucleon as well as the kinetic energy, T, of the exciton.

$$<\sigma(v_{rel})\,v_{rel}> \approx <\sigma(v_{rel})><v_{rel}> \tag{II.51}$$

$$\sigma(v_{rel}) = \frac{1}{2}\left(\sigma_{pp}(v_{rel}) + \sigma_{pn}(v_{rel})\right)\eta\left(\frac{T_F}{T}\right) \tag{II.52}$$

The η term of Eqn. II.52 is defined by Eqn. II.53, with its argument x being a fractional value.

$$\eta(x) = \begin{cases} 1-\frac{7}{5}x, & \text{if } x \le 0.5 \\ 1-\frac{7}{5}x+\frac{2}{5}x\left(2-\frac{1}{x}\right)^{5/2}, & \text{if } x > 0.5 \end{cases} \tag{II.53}$$

The averaged cross section, $<\sigma(v_{rel})>$, is randomly sampled by introducing a factor η, effectively accounting for the Pauli exclusion principle as is done in the Fermi-gas model. The free-particle interaction cross sections, $\sigma_{pp}(v_{rel})$ and $\sigma_{pn}(v_{rel})$, utilized by Eqn. II.51 are estimated using the relations suggested by Metropolis *et. al.* [63] shown in Eqn. II.54[17].

[17]The units of $\sigma_{pp,pn}$ and v_{rel} in Eqn. II.54 are *mb* and the fraction of speed of light, respectively

$$\begin{aligned}\sigma_{pp}(v_{rel}) &= \frac{10.63}{v_{rel}^2} - \frac{29.92}{v_{rel}} + 42.9 \\ \sigma_{pn}(v_{rel}) &= \frac{34.10}{v_{rel}^2} - \frac{82.20}{v_{rel}} + 82.20\end{aligned} \tag{II.54}$$

The relative kinetic energy, T_{rel}, of colliding particles necessary to calculated the relative velocity, v_{rel}, and the η factor of Eqn. II.52 and Eqn. II.53 are estimated in the so-called "right-angle-collision" approximation [4, 9]. The relative kinetic energy in this approximation is instead a sum of the mean kinetic energy of an exciton measured from the bottom of the potential well, $T_p = T_F + E/n$, plus the mean kinetic energy of an intranuclear nucleon partner, $T_N = \frac{3}{5}T_F$, such that $T_{rel} = T_p + T_N = \frac{8}{5}T_F + E/n$. The partial exciton transition rates for the various Δn states may then be expressed, according to Eqn. II.55, as a result of combining the primary partial transition rate equation, Eqn. II.47, the level density equations, Eqn. II.48, and the matrix estimation equation, Eqn. II.50.

$$\begin{aligned}\lambda_+(p,h,E) &= \frac{<\sigma(v_{rel})\,v_{rel}>}{V_{int}} \\ \lambda_0(p,h,E) &= \frac{<\sigma(v_{rel})\,v_{rel}>}{V_{int}}\left(\frac{gE-\xi(p,h)}{gE-\xi(p+1,h+1)}\right)^{n+1}\left(\frac{n+1}{n}\right)\times \\ &\qquad \times\left(\frac{p(p-1)+4ph+h(h-1)}{gE-\xi(p,h)}\right) \\ \lambda_-(p,h,E) &= \frac{<\sigma(v_{rel})\,v_{rel}>}{V_{int}}\left(\frac{gE-\xi(p,h)}{gE-\xi(p+1,h+1)}\right)^{n+1}\left(\frac{ph(n+1)(n-2)}{[gE-\xi(p,h)]^2}\right)\end{aligned} \tag{II.55}$$

II.6.3 Sampling Preequilibrium Emission

The equilibration of an excited residual nucleus is reached, by definition, when the partial transition rates for both a positive and negative change in the number of excitons are equal, $\lambda_+\left(n_{eq},E\right) = \lambda_-\left(n_{eq},E\right)$, from which the value of n_{eq}, the number of excitons at equilibration, can be obtained, as in Eqn. II.56.

$$n_{eq} \simeq \sqrt{2gE} \tag{II.56}$$

The behavior of the remaining excited compound nucleus at equilibration, where $t \geq t_{eq}$ or $n \geq n_{eq}$, is described in the framework of both the Weisskopf-Ewing statistical theory of particle evaporation [15, 16] and the Bohr-Wheeler theory for fission competition [64]. The nucleon density parameter, g, is related to the level-density parameter of single-particle states, according to Eqn. II.57.

$$a = \frac{\pi^2}{6} g \tag{II.57}$$

The level-density parameter, a, at the preequilibrium stage of the reaction is calculated according to Eqn. II.58, being an approximation derived in Ref. [65] in the form proposed in Ref. [66, 67] following the method of Ref. [68].

$$a\left(Z, N, E^*\right) = \tilde{a}\left(A\right)\left[1 + \delta W_{gs}\left(Z, N\right) \frac{f\left(E^* - \Delta\right)}{E^* - \Delta}\right] \tag{II.58}$$

The asymptotic Fermi-gas value of the level-density parameter at high excitation energies, $\tilde{a}$, is given by Eqn. II.59, where B_s is the ratio of the surface area of the nucleus to the surface area of a sphere of the same volume. Note that B_s for a ground state of a nucleus is nearly one [4]. The $f\left(E\right)$ term of Eqn. II.58 is given by Eqn. II.60, with E^* being the total excitation energy of the nucleus, being related to the "thermal" energy, U, as $U = E^* - E_R - \Delta$, and where E_R and Δ represent the rotational and pairing energies of the nucleus, respectively.

$$\tilde{a}\left(A\right) = \alpha A + \beta A^{2/3} B_s \tag{II.59}$$

$$f\left(E\right) = 1 - e^{-\gamma E} \tag{II.60}$$

The preequilibrium model utilizes the shell corrections, $\delta W_{gs}\left(Z, N\right)$, as in Möller *et. al.* [69] and utilizes pairing energy shifts from Möller *et. al.* [70]. Note the values of α, β, and γ in Eqn. II.59

and Eqn. II.60 were derived [65] by fitting the data by Iljinov *et. al.* [68], where the values utilized are $\alpha = 0.1463, \beta = -0.0716$ and $\gamma = 0.0542$.

The preequilibrium model was previously found to emit too few particles, however this problem has been addressed as in Veleský [71]. It is instead assumed that the ratio of the number of quasi-particles n, *i.e.* excitons, at each preequilibrium reaction stage to the number of excitons in the equilibrium configuration corresponding to the residual nucleus's excitation energy, n_{eq}, is a crucial parameter for estimating the probability for preequilibrium emission, P_{pre}, as utilized in Eqn. II.61.

$$P_{pre}\left(\frac{n}{n_{eq}}\right) = \begin{cases} 1 - e^{-\frac{(n/n_{eq})-1}{2\sigma_{pre}^2}}, & \text{for } n \leq n_{eq} \\ 0, & \text{for } n > n_{eq} \end{cases} \tag{II.61}$$

The basic assumption underlying the creation of Eqn. II.61 is that the probability of emission depends exclusively on the ratio n/n_{eq}, and where σ_{pre} is a free parameter no dependence on a residual nucleus's excitation energy [71], having a suggested range of values between 0.4 and 0.5.

II.6.4 Angular Distributions of Preequilibrium Progeny

The preequilibrium model predicts forward peaked, in the laboratory system, angular distributions for preequilibrium particles. The momentum direction, Ω, is thus required for specification of a nuclear state with a given excitation energy, E^*, and exciton number, n. Eqn. II.41 may be generalized, following Ref. [72], for this case provided that the angular dependence for the transition rates, $\lambda_{0,\pm}$, of Eqn. II.55 is factorized, being mathematically stated by Eqn. II.62, where the angular distribution term, $F(\Omega)$, is provided by Eqn. II.63.

$$< \sigma > \rightarrow < \sigma > F(\Omega) \tag{II.62}$$

$$F(\Omega) = \frac{\frac{d}{d\Omega}\sigma^{free}}{\int \frac{d}{d\Omega'}\sigma^{free} d\Omega'} \tag{II.63}$$

The scattering cross section, $d\sigma^{free}/d\Omega$, is assumed to be isotropic for nucleons, with similar assumptions for complex fragments [57], in the reference frame of the interacting excitons, thus resulting in an asymmetry in both the center-of-mass and laboratory frames of the nucleus. Kalbach systematics [73], being phenomenologically based, have been found to provide better agreement with measured angular spectra of emitted preequilibrium particles, compared to the approach of Eqn. II.62 and Eqn. II.63. The preequilibrium model thus incorporates the Kalbach systematics [73] to describe angular distributions for all emitted preequilibrium progeny at incident energies up to 210 MeV, as the Kalbach systematics [73] do not provide extrapolated data beyond incident energies of $\sim$ 200 MeV.

II.6.5 Photon Production from Nucleus De-Excitation

The preequilibrium model discussed here does not incorporate into it a model to estimate the emission of low-energy photons. Photon emission is excluded in the preequilibrium due to the low probability, relative to particle emission, of photon production at mid- to high-energies typically seen during spallation reactions [4]. The preequilibrium model discussed here instead allows its client programs to specify a model, such as the PHT model from LAHET [74], by which photon production can be simulated. The preequilibrium model will provide the specified photon production model with characteristics of the excited nucleus after each particle emission, such as its nucleonic composition, kinetic energy, and momentum. It is important to note that the usage of the photon production model by the preequilibrium model assumes that particle emission does not compete with photon emission, *i.e.* particle and photon emission are independent phenomena, where the assumption is made that all particle decay modes have been exhausted prior to photon emission [4].

II.6.6 Preequilibrium Emission off of Light Fragments

The preequilibrium emission off of small compound nuclei may be handled by the preequilibrium model, however statistical assumptions would be violated. The Fermi break-up model is instead

utilized by the preequilibrium model to simulate the emission from the deexcitation of small[18] and energetic nuclei. The Fermi break-up model is necessary to describe emissions from light nuclei to account for the unique structure of the nuclei and potential alpha clustering that may occur in these light nuclei, such as in carbon and oxygen [4]. The Fermi break-up model relies on the assumption that the excited nucleus has a kinetic energy that is comparable to its nucleon's binding energy. If this condition is met, the nucleus is assumed to disintegrate into N highly energetic fragments to predict various final states of the nucleus [14]. Further details on the theory of the Fermi break-up model can be found in Ref. [52]. Further detail on the Fermi break-up model may be found in Sec. II.5.

II.6.7 Flow of Preequilibrium Simulations

Fig. II.5 details the flow of preequilibrium simulations. The preequilibrium model ensures a residual nucleus is highly excited first to ensure its applicability, entering into a banking loop upon determining a residual nucleus's excitation state. At each opportunity for particle emission, the preequilibrium model ensures the assumptions utilized by the Weisskopf-Ewing statistical theory [15, 16] are satisfied, applying Fermi break-up physics in its place to predict the disintegration of the nucleus when the assumption is not satisfied. The preequilibrium model additionally ensures the residual nucleus has sufficient excitation energy prior to considering further particle emission. Sampling of preequilibrium emission then occurs according to Eqn. II.61 after estimating various emission channel quantities. Failure of preequilibrium emission to occur flags the nucleus as having equilibrated, thus reaching a compound state. Exciton transitions and subsequent particle emission is sampled if preequilibrium emission was sampled to occur according to the Monte Carlo method.

[18] By default, the preequilibrium model considers looks to the Fermi break-up model to determine what "small" nuclei are, typically containing no more than 12 nucleons. Client programs of the preequilibrium model may determine this threshold as desired when using the preequilibrium model.

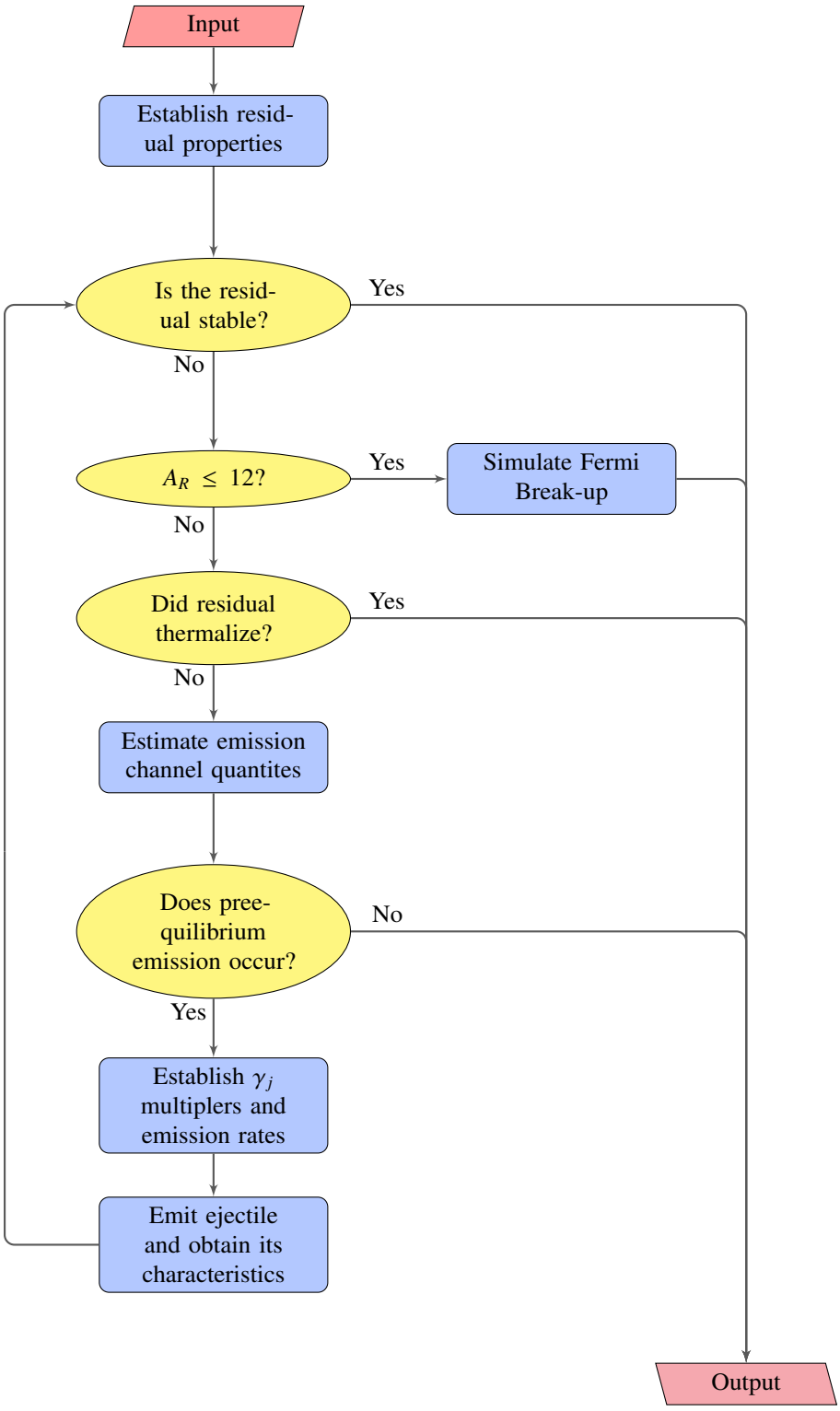

FIGURE II.5. Calculational flow for preequilibrium emission simulations.

II.7 THE EVAPORATION AND FISSION MODELS

The coupled high-energy evaporation and fission model [75–77] predicts the emission of nucleons, complex particles and light-fragments from compound nuclei during their de-excitation. Compound nuclei are often the byproducts of the fast and intermediate stages of spallation reactions [6], where their de-excitation via the evaporation and fission processes occurs after a nucleus's equilibration, during the compound stage of the reaction. The coupled evaporation and fission model discussed here, the Generalized Evaporation Model (GEM) [75–77], describes the evaporation of nucleons, complex particles, and light fragments from compound nuclei, ranging from neutrons, protons, *etc.*, up to ^{28}Mg. Note that highly excited compound nuclei will emit these fragments more strongly than those with low excitation energy.

II.7.1 The Evaporation Model

Use of the evaporation model requires that an excited compound nucleus created during the spallation process satisfies statistical assumptions. These nuclei typically contain more than 12 to 16 nucleons. The evaporated fragments emitted by the compound nucleus can be predicted using only information about the compound's nucleonic composition, kinetic energy, excitation energy, and normalized momentum vector. Nucleons, complex particles, and light fragments are considered for emittance during the evaporation process of GEM2, ranging from neutrons, protons, *etc.*, up to ^{28}Mg. Note that many details and formulas regarding the GEM2 coupled evaporation-fission model, based on the Dostrovsky evaporation [11] and Atchison fission [12, 13] models, can be found in Ref. [76]. The theory presented in Ref. [76] is quoted here for completeness.

Fig. II.6 demonstrates the flow of evaporation and fission physics within the GEM, as utilized by GSM. The evaporation and fission model establishes properties utilized for the compound nucleus upon initially being utilized. The applicability of the evaporation model is verified at the start of each evaporation simulation by first observing the validity of the statistical assumption. The disintegration of the nucleus, simulated by the Fermi break-up model, is utilized when the

statistical assumption is not maintained. Note the evaporation model relies solely on the utilized Fermi break-up model to determine whether or not the statistical assumption is satisfied to ensure de-excitation of the provided nucleus is simulated in some form or another. Disintegration of the compound nucleus via Fermi break-up physics results in an array of progeny and effectively no compound nucleus.

The evaporation model estimates all potentially emitted particle decay widths after ensuring its assumptions are satisfied. Emission of a particle does not occur in the event the sum of all decay widths is zero. The emission of a particle considered for emission is then randomly sampled according to its calculated decay width. The fission process competes with neutron emission, randomly sampling the probability of fission for a given nucleus. The creation of fission fragments will result in the simulation of their subsequent evaporation and fission processes. The compound particle remaining after evaporation, when neither the Fermi break-up or fission processes were underwent, continues to undergo this evaporation and fission process. The simulation ends at the completion of the evaporation and potential fission of the compound and all created fission fragments.

II.7.1.1 Fragment Decay Width

Decay widths for evaporated particles and fragments are estimated using the classical Weisskopf-Ewing statistical model [16]. The Weisskopf-Ewing model estimates the decay probability P_j for the emission of particle j from a parent compound nucleus c, with the total kinetic energy in the center-of-mass system between ϵ and $d\epsilon$ and daughter product d by Eqn. II.64.

$$P_j(\epsilon)\, d\epsilon = g_j \sigma_{inv}(\epsilon) \frac{\rho_d(E - Q - \epsilon)}{\rho_c(E)} \epsilon d\epsilon \tag{II.64}$$

The g_j term represents a multiplication factor, $g_j = \frac{(2S_j+1)m_j}{\pi^2\hbar^2}$, accounting for the various spin states S_j and the reduced mass m_j of the emitted fragment. The $\sigma_{inv}(\epsilon)$ term represents the cross section of the inverse reaction, and the ρ_d and ρ_c terms denote the level density, in MeV^{-1}, of the resulting

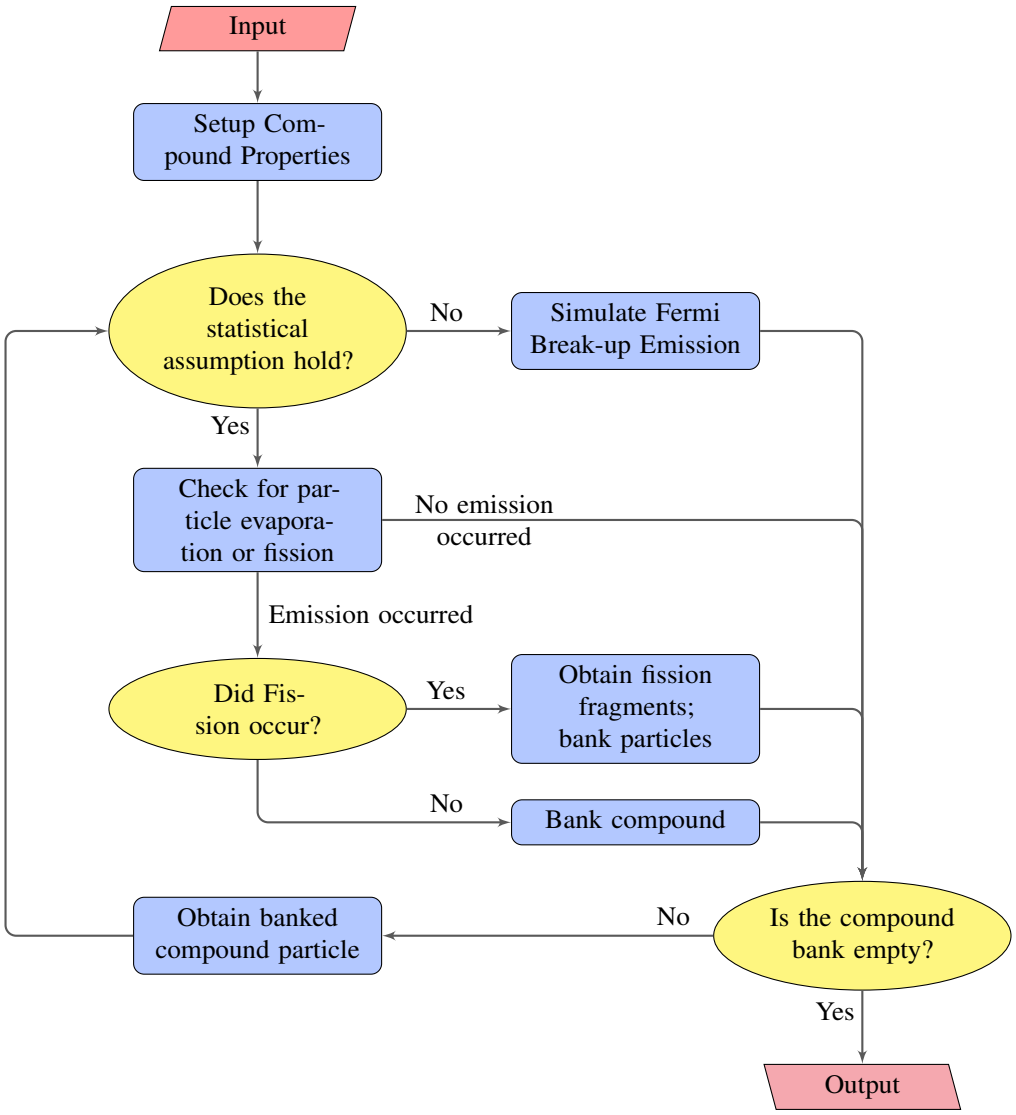

FIGURE II.6. The simulation scheme for evaporation and fission physics in the GEM.

daughter and parent compound nucleus, respectively. Dostrovsky's formula [11], Eqn. II.65, is used to calculate the inverse cross section for all emitted particles and fragments. Note that Eqn. II.65 is often written in the form of Eqn. II.66.

$$\sigma_{inv}(\epsilon) = \sigma_g \alpha \left(1 + \frac{\beta}{\epsilon}\right) \tag{II.65}$$

$$\sigma_{inv}(\epsilon) = \sigma_g \times \begin{cases} c_n \left(1 + \frac{b}{\epsilon}\right), & \text{for neutrons} \\ c_j \left(1 - \frac{V}{\epsilon}\right), & \text{for charged particles} \end{cases} \tag{II.66}$$

Where σ_g is the geometrical cross section, $\sigma_g = \pi R_b^2$, with units of fm^2, and the coulomb barrier, V, with units of MeV, is defined simply by Eqn. II.67.

$$V = k_j e^2 \left(\frac{Z_j Z_d}{R_c} \right) \tag{II.67}$$

Where k_j is the penetrability coefficients calculated in Ref. [11], R_c the sum of the nuclear radii for the daughter and evaporated fragment nuclei, e the charge of an electron, and Z_d and Z_j the atomic number of the daughter and emitted fragment, respectively. The b, c_j, c_n, k_j, R_b, and R_c terms of Eqn. II.66 and Eqn. II.67 may utilize one of four parameter sets, being the simple or precise [11, 78] parameter sets, or two other options from Ref. [77]. By default, the precise parameter set, later modified by Furihata [77, 79], is utilized. The available parameter sets are described more in Sec. F.1.1.1 [76].

The terms E, Q, and ϵ, with units of MeV, in Eqn. II.64 denote the excitation energy of the parent nucleus c, the energy release from the reaction, and the kinetic energy in the center-of-mass system. The energy released, Q, as a result of the evaporation of fragment j from the parent nucleus c is calculated by using the difference in mass excesses of the reaction products and parent nucleus, *i.e.* $Q = \left[M\left(A_j, Z_j\right) + M\left(A_d, Z_d\right) \right] - M\left(A_c, Z_c\right)$. Several data tables are used in GEM2 to calculate these Q-values, being the Audi-Wapstra mass table [80], theoretical masses calculated by Möller *et al.* [69], and the theoretical masses calculated by Comay *et al.* [81], respectively. Mass excesses for nuclei that are outside of the range of the Audi-Wapstra, Möller, and Comay tables utilize the Cameron formula [82] instead.

The total decay width Γ_j is calculated by analytically integrating Eqn. II.64, after substitution of Eqn. II.65 for σ_{inv}, with respect to the total kinetic energy within the valid range permitted by the Coulomb barrier and energy conservation.

$$\Gamma_j\left(E\right) = \frac{g_j \sigma_g \alpha}{\rho_c\left(E\right)} \int_V^{E-Q} \epsilon \left(1 + \frac{\beta}{\epsilon}\right) \rho_d\left(E - Q - \epsilon\right) d\epsilon \tag{II.68}$$

The level density parameter $\rho(E)$ for the parent and daughter nuclides are calculated using a Fermi-gas model approach [83], shown by Eqn. II.69.

$$\rho(E) = \begin{cases} \frac{\pi}{12a^{1/4}(E-\delta)^{5/4}} e^{2\sqrt{a(E-\delta)}}, & \text{for } E \geq E_x \\ \frac{\pi}{12T} e^{(E-E_0)/T}, & \text{for } E < E_x \end{cases} \tag{II.69}$$

Where a is the level-density parameter, in MeV^{-1}, δ is the pairing energy, in MeV, evaluated in Ref. [83, 84], E_0 defined by Eqn. II.71 to define a smooth transition between the energy regimes of Eqn. II.69, and T the nuclear temperature, given by Eqn. II.72. There exists a few options for the level density parameter utilized by the evaporation model. The available options for the level density parameter are detailed in Sec. F.1.1.2. Pairing energies not supplied by Ref. [84] use pairing energies from Ref. [83] instead. The transition between domains shown in Eqn. II.69, denoted as E_x, is given as $E_x = U_x + \delta$, where U_x is denoted by Eqn. II.70.

$$U_x = 2.5 + \frac{150}{A_d} \tag{II.70}$$

$$E_0 = E_x - T\left(log(T) - 0.25log(a) - 1.25log(U_x) + 2\sqrt{aU_x}\right) \tag{II.71}$$

$$\frac{1}{T} = \sqrt{\frac{a}{U_x}} - \frac{1.5}{U_x} \tag{II.72}$$

Substituting the level densities given by Eqn. II.69 for the daughter nuclide into Eqn. II.68 then yields Eqn. II.73.

$$\Gamma_j(E) = \frac{\pi g_j \sigma_g \alpha}{12\rho_c(E)} \times \begin{cases} I_1(t,t) + (\beta + V) I_0(t), & \text{for } (E - Q - V) < E_x \\ I_1(t,t_x) + I_3(s,s_x) e^s + (\beta + V)\left[I_0(t_x) + I_2(s,s_x) e^s\right], & \text{for } (E - Q - V) \geq E_x \end{cases} \tag{II.73}$$

Where t is the fraction of available energy for the fragment, $t = (E - Q - V)/T$, t_x the fraction of available energy at the transition, $t_x = E_x/T$, $s = 2\sqrt{a(E - Q - V - \delta)}$, and $s_x = 2\sqrt{a(E_x - \delta)}$. The transition for Eqn. II.73 defines the point at which the equation can no longer be solved analytically, where an approximation is used when the available energy exceeds that defined by E_x. The terms I_0, I_1, I_2, and I_3 are described below.

$$
\begin{aligned}
I_0(t) &= (e^t - 1)\, e^{-E_0/T} \\
I_1(t, t_x) &= T\left[(t - t_x + 1)\, e^{t_x} - t - 1\right] e^{-E_0/T} \\
I_2(s, s_x) &= 2\sqrt{2}\left[\left(s^{-3/2} + 1.5s^{-5/2} + 3.75s^{-7/2}\right) - \left(s_x^{-3/2} + 1.5s_x^{-5/2} + 3.75s_x^{-7/2}\right) e^{s_x - s}\right] \\
I_3(s, s_x) &= \frac{1}{a\sqrt{2}}\Bigg(2s^{-1/2} + 4s^{-3/2} + 13.5s^{-5/2} + 60s^{-7/2} + 325.125s^{-9/2} - \Big(\left(s^2 - s_x^2\right) s_x^{-3/2} \\
&\quad + \left(1.5s^2 + 0.5s_x^2\right) s_x^{-5/2} + \left(3.75s^2 + 0.25s_x^2\right) s_x^{-7/2} + \left(12.875s^2 + 0.625s_x^2\right) s_x^{-9/2} \\
&\quad + \left(59.0625s^2 + 0.9375s_x^2\right) s_x^{-11/2} + \left(324.8s^2 + 3.28s_x^2\right) s_x^{-13/2}\Big) e^{s_x - s}\Bigg)
\end{aligned}
$$

II.7.1.2 Kinetic Energy Selection

Eqn. II.64 describes the kinetic energy distribution of an ejectile and daughter nucleus. Substitution of the level density, given by Eqn. II.69, into Eqn. II.64 and normalization yields Eqn. II.74. The kinetic energy of the ejectile is then randomly chosen according to the probability given by Eqn. II.74. The excitation of the resulting daughter nucleus is then calculated according to energy conservation laws as $E_d^* = E - Q - \epsilon$, accounting for the kinetic energy of the fragment.

$$
P_j(\epsilon) = \frac{g_j \sigma_g \alpha(\epsilon + \beta)}{\Gamma_j} \times \begin{cases} \frac{1}{T} e^{a(E - Q - \epsilon - E_0)/T}, & \text{for } E - Q - E_x \le \epsilon \le E - Q \\ \frac{e^{2\sqrt{a(E - Q - \delta - \epsilon)}}}{a^{1/4}(E - Q - \delta - \epsilon)^{5/4}}, & \text{for } \kappa V \le \epsilon < E - Q - E_x \end{cases} \tag{II.74}
$$

Ejectiles are assumed to be emitted from the parent nucleus isotropically in the center-of-mass system, given direction of motion **w**. The resulting momentum of the ejectile, in the center-of-mass system, is then determined by the ejectile's resulting kinetic and rest mass energies, shown in

Eqn. II.75.

$$\mathbf{p_{cm}} = \mathbf{w}\sqrt{\epsilon^2 + 2\epsilon\left(\frac{m_j m_d}{m_j + m_d}\right)} \tag{II.75}$$

where m_j and m_d are the masses, in MeV/c^2, of the ejectile and daughter products of the reaction, respectively. The momenta of the ejectile and daughter are obtained by use of a Lorentz transformation.

II.7.1.3 Fragment Selection

The evaporation model allows for the consideration of up to 66 different nuclides in both their ground and excited state energies. The isotopes available for consideration are limited to $Z_j \leq 12$, all of which are naturally occurring or near the stability line and have half-lives longer than 1 ms. The isotopes that may be considered for evaporation are shown in Table II.VII. Fragments are selected randomly, where ejectile j is selected according to the probability distribution calculated by Eqn. II.64, and Γ_j is that given by Eqn. II.73. The probability given by Eqn. II.76 is used to randomly select a fragment j from the N fragments available for emission by comparison to a random number.

$$p_j = \frac{\Gamma_j}{\sum_{j=1}^{N} \Gamma_j} \tag{II.76}$$

Many evaporation models do not presently consider the emission of as many fragments as this model does. As a result of this, the evaporation model requires more computation time than most other evaporation models. To compensate for the increase in computation time, there exists an option to exclude the evaporation of ejectiles heavier than ^4He. This option is discussed in further detail in Sec. F.1.1.3.

TABLE II.VII. Isotopes Considered for Emission in the Preequilibrium and Evaporation Models

Z_j	Ejectiles						
0	n						
1	p	d	t				
2	^{3}He	^{4}He	^{6}He	^{8}He			
3	^{6}Li	^{7}Li	^{8}Li	^{9}Li			
4	^{7}Be	^{9}Be	^{10}Be	^{11}Be	^{12}Be		
5	^{8}B	^{10}B	^{11}B	^{12}B	^{13}B		
6	^{10}C	^{11}C	^{12}C	^{13}C	^{14}C	^{15}C	^{16}C
7	^{12}N	^{13}N	^{14}N	^{15}N	^{16}N	^{17}N	
8	^{14}O	^{15}O	^{16}O	^{17}O	^{18}O	^{19}O	^{20}O
9	^{17}F	^{18}F	^{19}F	^{20}F	^{21}F		
10	^{18}Ne	^{19}Ne	^{20}Ne	^{21}Ne	^{22}Ne	^{23}Ne	^{24}Ne
11	^{21}Na	^{22}Na	^{23}Na	^{24}Na	^{25}Na		
12	^{22}Mg	^{23}Mg	^{24}Mg	^{25}Mg	^{26}Mg	^{27}Mg	^{28}Mg

II.7.1.4 Fragment Excitation

The evaporation model allows ejectiles to have excited states according to Ref. [85] to enhance the yield of heavy fragments. Allowed excited states for ejectiles must have mean resonance lifetimes longer than the decay width of the resonance emission. Excited states of ejectiles are included if its half life, $T_{1/2}$, with units of *seconds*, satisfies the condition presented by Eqn. II.77.

$$\frac{T_{1/2}}{ln\,(2)} > \frac{\hbar}{\Gamma_j^*} \tag{II.77}$$

Γ_j^* in Eqn. II.77 is the decay width of the resonance emission, being calculated in the same way as for ground state emissions. The energy released during the evaporation of an excited ejectile is then instead $Q^* = Q + E_j^*$, with E_j^* being the excitation energy of the resonance. The spin of the excited nucleus is utilized in the calculation of g_j. Note that mass differences due to the excitation of the ejectile are neglected by the evaporation model. An enhancement is made to excited ejectiles to prevent treating them as separate particles such that the decay width of the ejectile, excited to some energy state m, is modified by adding to its ground state decay width Γ_j^0 the decay width of all n lower excited states Γ_j^m, as described by Eqn. II.78.

$$\Gamma_j = \Gamma_j^0 + \sum_{m=1}^{n} \Gamma_j^m \tag{II.78}$$

Excited ejectiles are assumed to have kinetic energies behaving in the same manner as ground state ejectiles, utilizing data from Ref. [86] for the excited spin state, S_j^*, the excitation energy of the resonance, E_j^*, and half life, $T_{1/2}$, of the ejectile.

II.7.2 The Fission Model

Excited compound nuclei evaporate many fragments as they deexcite and may fission as a result of the deexcitation. The fission model discussed here is an extension of the GEM2 fission model to consider fission of excited compound nuclei with $Z_c > 65$. The fission model of GEM2 is based on the Atchison[19] fission model [12, 13]. Fission of the excited compound nucleus is only considered with neutron emission during the evaporation process. It will be seen throughout the proceeding sections that there are two parameterizations for the characterization of fission as a result of the work performed in Ref. [76]. There exists in the fission model an option to utilize the original Atchison or updated Furihata parameterization. Use of these fission parameterizations is discussed further in Sec. F.1.2. Note that all details and formulas regarding the fission model can be found in Ref. [76]. The theory of Ref. [76] is quoted here for documentation purposes.

II.7.2.1 Fission Probability

The extended Atchison fission model considers the fission of excited compound nuclei with $Z_c > 65$, and assumes that fission may occur only during neutron emission. This "competition" of fission with neutron emission is estimated by calculating the probability of fission P_f and comparing it to that for neutron emission. The calculation of the fission probability is treated differently for nuclei with Z_c above and below $Z_c = 89$. This split in fission probability characterization is done primarily because of little experimentally available information for compound nuclei with

[19]The Atchison fission model is also referred to as the Rutherford Appleton Laboratory (RAL) Model.

$85 \leq Z_c \leq 88$, a marked rise in the fission barrier for $Z_c^2/A_c \leq 34$ [13], and a disappearance of asymmetric mass splitting, all of which lead to the hypothesis that there exists a change in the character of the fission process at or near $Z_c^2/A_c \approx 34$.

The fission of compound nuclei with $65 \leq Z_c \leq 88$ uses the original Atchison calculation for both the neutron emission width Γ_n and the fission probability width Γ_f. The fission probability is estimated using these emission widths as depicted by Eqn. II.79.

$$P_f = \frac{\Gamma_f}{\Gamma_f + \Gamma_n} = \frac{1}{1 + \Gamma_n/\Gamma_f} \tag{II.79}$$

The level density [16] of Eqn. II.69, with an energy-independent pre-exponential factor, and the inverse reaction cross section [11] of Eqn. II.65 is used to estimate the neutron emission width Γ_n. Performing the integration of Eqn. II.68 with these substitutions yields Eqn. II.80.

$$\Gamma_n = 0.352\left[1.68J_0 + 1.93A_c^{1/3}J_1 + A_c^{2/3}\left(0.76J_1 - 0.05J_0\right)\right] \tag{II.80}$$

Where J_0 and J_1 are functions of the neutron's level density parameter, a_n. Note that the level density parameter used by the fission model is constant. The equations for J_0 and J_1 are given by the following:

$$\begin{aligned} a_n &= \frac{(A_c - 1)}{8} \\ s_n &= 2\sqrt{a_n\left(E - Q_n - \delta\right)} \\ J_0 &= \frac{(s_n - 1)\,e^{s_n} + 1}{2a_n} \\ J_1 &= \frac{\left(2s_n^2 - 6s_n + 6\right)e^{s_n} + s_n^2 - 6}{8a_n^2} \end{aligned}$$

Note that the neutron level density parameter a_n is kept when calculating the fission probability, although it differs from the GCCI parameterization available to the evaporation model when calculating particle evaporation widths.

The fission width for heavy compound nuclei with $Z_c \leq 88$ is defined by Eqn. II.81, where the fission level density parameter is fitted [13] to be a function of the neutron level density parameter, a_f, and a scaling term, s_f, where s_f utilizes a fission barrier, B_f, that is dependent on only the nucleonic composition of the compounding nucleus.

$$\Gamma_f = \frac{\left(s_f - 1\right) e^{s_f} + 1}{a_f} \tag{II.81}$$

The level density, a_f, and the s_f terms in Eqn. II.81 are described as follows:

$$\begin{aligned} a_f &= a_n \left[1.08926 + 0.01098 \left(\frac{Z_c^2}{A_c} - 31.08551 \right)^2 \right] \\ s_f &= 2\sqrt{a_f \left(E - B_f - \delta \right)} \end{aligned}$$

The fission barrier, B_f, used in the s_f term of Eqn. II.81 is described by Eqn. II.82.

$$B_f = Q_n + 321.2 - 16.7\frac{Z_c^2}{A_c} + 0.218\left(\frac{Z_c^2}{A_c}\right)^2 \tag{II.82}$$

Heavy fissioning compound nuclei with $Z_c \geq 89$ instead determine the ratio of Γ_n to Γ_f from Eqn. II.79 by utilizing a semi-empirical approximation of experimental values published by Ref. [87]. This ratio is estimated according to Eqn. II.83.

$$log\left(\frac{\Gamma_n}{\Gamma_f}\right) = C\left(Z_c\right)\left[A_c - A_0\left(Z_c\right)\right] \tag{II.83}$$

Where $C\left(Z_c\right)$ and $A_0\left(Z_c\right)$ are constants depending only on the nuclear charge of the compound nucleus, Z_c, and are shown in Table II.VIII. Note that use of Eqn. II.83 is independent of the compound nuclei's excitation energy.

TABLE II.VIII. Constants $C(Z)$ and $A_0(Z)$ for Eqn. II.83

Z	$C(Z)$	$A_0(Z)$
89	0.23000	219.40
90	0.23300	226.90
91	0.12225	229.75
92	0.14727	234.04
93	0.13559	238.88
94	0.15735	241.34
95	0.16597	243.04
96	0.17589	245.52
97	0.18018	246.84
98	0.19568	250.18
99	0.16313	254.00
100	0.17123	257.80
101	0.17123	261.30
102	0.17123	264.80
103	0.17123	268.30
104	0.17123	271.80
105	0.17123	275.30
106	0.17123	278.80

II.7.2.2 Fission Fragment Mass Distribution

Fission may occur either symmetrically or asymmetrically regarding the mass of the fission fragments. Only symmetric fission is allowed for compound nuclei with $Z_c^2/A_c \leq 35$. For nuclei with $Z_c^2/A_c > 35$, both asymmetric and symmetric fission are allowed depending on the excitation energy of the fissioning nucleus. An asymmetric fission probability, P_{asy}, is utilized to determine if fission is symmetric or asymmetric according to Eqn. II.84 given the fissioning nucleus's excitation energy, E.

$$P_{asy} = \frac{4870e^{-0.36E}}{1 + 4870e^{-0.36E}} \tag{II.84}$$

A Gaussian distribution is utilized to determine the mass number of the fragments. For asymmetric fission, the mass number of one fission fragment A_1 is selected randomly from the Gaussian given a mean of $A_c = 140$ and width of $\sigma_M = 6.5$. The mass number of the second fragment is

selected such that $A_2 = A_c - A_1$. Selection of A_1 for symmetric fission is chosen by random sampling of a Gaussian curve with a mean equal to half the fissioning nucleus's mass number, $A_c/2$, and two options for the width σ_M of the curve. The original approximation for the width σ_M is utilized as the first option, as shown in Eqn. II.85, and the second was developed by Furihata in Ref. [76], as shown in Eqn. II.86.

$$\sigma_M = \begin{cases} 3.97 + 0.425\left(E - B_f\right) - 0.00212\left(E - B_f\right)^2, & \text{for } \left(E - B_f\right) < 100\ MeV \\ 25.27, & \text{for } \left(E - B_f\right) \geq 100\ MeV \end{cases} \tag{II.85}$$

$$\sigma_M = C_3\left(Z_c^2/A_c\right)^2 + C_4\left(Z_c^2/A_c\right) + C_5\left(E - B_f\right) + C_6 \tag{II.86}$$

All values in both Eqn. II.85 and Eqn. II.86 are given in *MeV*, and the fission barriers of Eqn. II.85 are calculated according to Eqn. II.82 for compound nuclei with $Z_c \leq 88$, and according to Eqn. II.87 [88] for larger nuclei.

$$B_f = C - 0.36\left(\frac{Z_c^2}{A_c}\right) \tag{II.87}$$

The C term in Eqn. II.87 is a constant having values of 18.8, 18.1, 18.1, and 18.5 *MeV* for odd-odd, even-odd, odd-even, and even-even nuclei, respectively. The constants $C_3 = 0.122$, $C_4 = -7.77$, $C_5 = 0.032$, and $C_6 = 134.0$, from Eqn. II.86, were obtained by fitting a collection of experimental fission-fragment mass distributions [89, 90] to the fission model. The fission barrier heights B_f for Eqn. II.86 use that of Ref. [91]

II.7.2.3 Fission Fragment Charge Distribution

The charge distribution of a fission fragment is assumed to be distributed according to a Gaussian distribution of mean Z_f and width of σ_Z. The mean, Z_f, is defined by Eqn. II.88.

$$Z_f = \frac{Z_c + Z_1' - Z_2'}{2} \tag{II.88}$$

The Z_l' term in Eqn. II.88 is defined by Eqn. II.89.

$$Z_l' = \frac{65.5A_l}{131 + A_l^{2/3}} \text{ for } l\text{= 1 or 2.} \tag{II.89}$$

The width of the Gaussian distribution was originally parameterized by Ref. [13] as $\sigma_Z = 2.0$, however an investigation by Ref. [77] suggested that $\sigma_Z = 0.75$ provides better agreement with data. The fission model discussed here utilizes $\sigma_Z = 0.75$ as its default behavior. Selection of Z_1 determines the value of Z_2 in the same manner discussed in Sec. II.7.2.2.

II.7.2.4 Fission Fragment Kinetic Energy Distribution

The kinetic energy of fission fragments is determined by randomly sampling a Gaussian distribution with mean ϵ_f and width σ_{ϵ_f}. The original parameters from Ref. [13] state that the mean, ϵ_f, and width, σ_{ϵ_f}, follows the form of Eqn. II.90 and Eqn. II.91, respectively, however this was later modified by Ref. [76] to that of Eqn. II.92 and Eqn. II.93, respectively. Note that both parameter sets for the kinetic energy selection are available in the fission model.

$$\epsilon_f = 0.133\left(\frac{Z_c^2}{A_c^{1/3}}\right) - 11.4 \tag{II.90}$$

$$\sigma_{\epsilon_f} = 0.084\epsilon_f \tag{II.91}$$

$$\epsilon_f = \begin{cases} 0.131\left(Z_c^2/A_c^{1/3}\right), & \text{for } Z_c^2/A_c^{1/3} \leq 900 \\ 0.104\left(Z_c^2/A_c^{1/3}\right) + 24.3, & \text{for } 900 \leq Z_c^2/A_c^{1/3} \leq 1800 \end{cases} \tag{II.92}$$

$$\sigma_{\epsilon_f} = \begin{cases} C_1 \left(Z_c^2 / A_c^{1/3} - 1000 \right) + C_2, & \text{for } Z_c^2 / A_c^{1/3} \geq 1000 \\ C_2, & \text{for } Z_c^2 / A_c^{1/3} < 1000 \end{cases} \tag{II.93}$$

The values of C_1 and C_2 in Eqn. II.93 are constants fitted to data from Ref. [92], where $C_1 = 0.00057$ and $C_2 = 86.5$.

II.7.2.5 Updates to and Multipliers for the Fission Model

It is important to note that the fission model has been upgraded outside [93] of the model discussed here. The updates discussed in Ref. [93] are not incorporated here. Multipliers to the fission level density, a_f, and the constant $C(Z)$ from Table II.VIII are included in the model [5, 94–97] so that the model can be used to better describe fission when used in conjunction with varying preequilibrium and INC models. The multipliers currently well describe fission of heavy compound nuclei when used in conjunction with the CEM, GSM, and LAQGSM event generators. Using a value of one for each of the multipliers results in use of the original parameterization, such as in LAHET [74].

II.7.3 Photon Production from Nucleus De-Excitation

The evaporation model discussed here does not incorporate into it a model to estimate the emission of low-energy photons. Photon emission is excluded in the evaporation due to the low probability, relative to particle emission, of photon production at mid- to high-energies typically seen during spallation reactions [4]. The evaporation model discussed here instead allows its client programs to specify a model, such as the PHT model from LAHET [74], by which photon production can be simulated. The evaporation model will provide the specified photon production model with characteristics of the excited nucleus after each particle emission, such as its nucleonic composition, kinetic energy, and momentum. It is important to note that the usage of the photon production model by the evaporation model assumes that particle emission does not compete with photon emission,

i.e. particle and photon emission are independent phenomena, where the assumption is made that all particle decay modes have been exhausted prior to photon emission [4].

II.7.4 Evaporation of Small Fragments

The evaporation of small compound nuclei may be handled by the evaporation model, however statistical assumptions would be violated. The Fermi break-up model is instead utilized by the evaporation model to simulate the emission from the deexcitation of small[20] and energetic nuclei. The Fermi break-up model is necessary to describe emissions from light nuclei to account for the unique structure of the nuclei and potential alpha clustering that may occur in these light nuclei, such as in carbon and oxygen [4]. The Fermi break-up model relies on the assumption that the excited nucleus has a kinetic energy that is comparable to its nucleon's binding energy. If this condition is met, the nucleus is assumed to disintegrate into N highly energetic fragments to predict various final states of the nucleus [14]. Further details on the theory of the Fermi break-up model can be found in Ref. [52]. Further detail on the Fermi break-up model may be found in Sec. II.5.

The evaporation and fission model documentation further describes how to use the coupled evaporation and fission models described here in a client program to obtain information regarding the emmitted fragments, resulting nucleus and possible fission fragments, and additionally how general results from the coupled evaporation-fission simulation are obtained.

II.8 PHOTON EMISSION

The GSM model discussed here does not incorporate into it a model to estimate the emission of low-energy photons. Photon emission is excluded in the GSM due to the low probability, relative to particle emission, of photon production at mid- to high-energies typically seen during spallation reactions [4]. The GSM model discussed here instead allows its client programs to specify a model,

[20] By default, the evaporation model considers looks to the Fermi break-up model to determine what "small" nuclei are, typically containing no more than 12 nucleons. Client programs of the evaporation model may determine this threshold as desired when using the evaporation model.

such as the PHT model from LAHET [74], by which photon production can be simulated, where the GSM model provides the appropriate interfaces from the preequilibrium and evaporation . The GSM model will provide the specified photon production model with characteristics of the excited nucleus after each particle emission, such as its nucleonic composition, kinetic energy, and momentum. It is important to note that the usage of the photon production model by the evaporation model assumes that particle emission does not compete with photon emission, *i.e.* particle and photon emission are independent phenomena, where the assumption is made that all particle decay modes have been exhausted prior to photon emission [4]. Sec. III.5 details the inclusion of photon emission during a GSM simulation, whose procedure has the interface shown in Fig. III.8.

Chapter III. Modernization

III.1 CONTEXT

Modernization here is used to describe the adaptation of GSM to use various modern Fortran (namely Fortran2003 and Fortran2008) features and generally accepted best practices. The attempt at modernization of GSM to use these constructs and features has aided in providing protection to GSM internally. Modernization additionally reduces the amount of technical debt contained within GSM, providing developers with more knowledge about the source code and thus requiring less refactoring and other additional work required at a later time. Preliminary modernization of GSM has focused mainly on utilizing argument intent, using intrinsic properties of the modern Fortran language, and creating parameterized data sets within modules for the respective physics models. Further modernization of GSM focused on migrating the various sub-models, and GSM itself, to object-oriented Fortran and utilizing data types, data classes, and procedure pointers to create clear and well-known dependencies, and in addition to limit the scope and required namespace of the various sub-models and GSM itself.

A working knowledge of the Fortran language is help in understanding the benefits of the work presented here in this section. It is important to note that Fortran is a pass-by-reference language, meaning that procedures, being functions and subroutines, are given the memory reference for each argument. The called procedure may then alter the value, or values, represented by the variable. The called procedure may not change the memory reference of the variable, only the value stored in the computer's memory at the argument's memory reference.

Modern Fortran utilizes argument intent, allowing developers to clearly define the usage of an argument within a procedure. An argument with intent `in` is forced to act as a parameter within the procedure, and arguments with intent `out` informs developers that the argument will be given a value within the procedure and returned to the calling procedure with the new value. The other

available argument intent, being `inout`, denotes the argument as being used and altered in the called procedure. The `inout` intent is used most often in GSM regarding nucleus properties and results, particularly regarding particle emission. Utilizing argument intent provides protection for the arguments and provides developers with additional information about the argument.

Much of the GSM code prior to this work utilized `implicit` statments, namely `implicit real*8 (a-h,` to declare variables, allowing variables to "appear" without the developers being aware of the variables' appearance, particularly for large procedures. Additionally, usage of the `*8` and `*4` syntax may provide non-standard data type sizes of different operating systems. Modernization of GSM in this context has removed all of the implicitly declared variables, declared by the `implicit real*8 (a-h, o-z)` and `implicit int*4 (i-n)` statements, in favor of the `implicit none` syntax to require developers to explicitly declare all variables used. GSM also utilizes data type size specifications from the intrinsic `iso_fortran_env` module, providing a standard size for all integer and real, both single and double precision, variables across operating systems. This work involved removal of the `*8` and `*4` syntax to `(real64)`, `(int32)`, `(int64)`, *etc.*, syntax. Requiring procedures to utilize only explicitly and carefully declared variables adds protection to GSM and intrinsically demonstrates that GSM will conform to standard type bit-size for the respective variables across operating platforms and compilers.

Modern Fortran allows for parameterized type names, allowing developers to carefully specify the constants' values, such as 1, 2, π, *etc.*, and the precision of those constants throughout the source code. Parameterized type names may be used for all data types. Usage of the `parameter` keyword provides standardization across the procedures and objects. Fortran compilers substitute parameterized values in place of the type's name to reduce the runtime latency that may have previously occurred when accessing the computer's cache to determine the value of the parameter. Usage of the `parameter` keyword, similarly to the above improvements, provides significant protection in terms of accidental changes that may have been made to the variable prior to becoming a parameter. Usage of the `parameter` keyword ensures developers that the value represented by the type's name is constant and unchanged at all times during program execution.

Parameterized values were placed into data and numerical modules to limit the namespace within each of the modules. For example, the numbers `one`, `two`, `three`, *etc.*, were placed into a parameter module for each sub-model of GSM, including GSM itself, and the module then used by the respective sub-model and its data object only. The same occurred for parameterized data for each of the sub-models. The addition of modules in this way provides layering and less-monolithic code. Additionally, modern Fortran allows developers to specify if a variable or parameter is `public`, `private`, or `protected`, within the module. All parameterized type names are made `public` as they cannot be changed, and non-parameter data for the sub-model is made `protected`, allowing the sub-model to utilize the data without having permission to change the value or values of the data type. For example, photon cross section data read from a file would require both the `public` and `protected` attributes within the module to ensure procedures may utilize the data while simultaneously enforcing the values held by the data to remain unchanged. Usage of these modules containing a model's numerical data, being effectively parameterized, within the procedures requiring the data severely limits the scope accessible to the procedure and clearly defining the namespace contained by the procedure. Usage of these modules clearly defines dependencies within each sub-model as well. Inclusion of the `only` keyword was also specified for each usage of the module to help limit the namespace contained within each procedure and to ensure the declared contents of each procedure are well-defined and obvious to all developers. Fig. III.1 demonstrates some of these concepts.

GSM was further modernized from the above by creating Fortran derived types to act as data types and objects as well class objects. Each data type and object created provided an easy-to-use container for data regarding a single entity, such as a particle within an array or bank of particles, and in addition provided many easy methods clients and users can utilize to query the object. The majority of the work required to modernize GSM took place in creating and utilizing these objects. These objects each take the form shown in Fig. III.1, where data types do *not* utilize member procedures and contain mostly public member variables. Usage of these objects provides many benefits and is described more in-depth in Sec. III.4 and Sec. III.5. Examples of the object-oriented

```
module fooMod
  use, intrinsic:: iso_fortran_env, only: real64
  implicit none

  real(real64), public,  parameter :: barParameterization1 = 1.0_real64
  real(real64), private, parameter :: defaultBar = barParameterization1

  type, [private/public] :: foo
    private
    ! Member variables go here - for example:
    real(real64), private :: bar = defaultBar

   contains
    private
    ! Member procedures go here - for example:
    procedure, private :: setBar
    procedure, public  :: queryBar
  end type foo

 contains

  procedure, public  :: newFoo   ! Constructor for the 'foo' object

end module fooMod
```

FIGURE III.1. A sample modern Fortran module and data object.

nature of GSM are also shown there as a result of the GSM modernization.

Procedure pointers are now also utilized by GSM and its sub-models. Procedure pointers were utilized such that the sub-models now require the client to provide them with a method for obtaining or using information that is needed external to the model. Procedure pointers in modern Fortran require an interface be made and that they be given the `pointer` attribute. Procedure pointers are used within GSM and its sub-models to allow clients to control how messages, such as errors or warnings, are printed during the program execution. Procedure pointers are also used by the various sub-models of GSM to obtain random numbers from a random number generator (RNG) and additional physics not accounted for by GSM, namely photon emission as a result of the spallation process. Fig. III.2 demonstrates these procedure pointers and their use by the sub-models. Procedure pointers in Fortran are used in a similar manner to standard procedures. Inclusion of

```
! "Skeleton" of the procedure:
abstract interface
   function RANDOM() result(rndm)
      use, intrinsic:: iso_fortran_env, only: real64
      implicit none
      real(real64) :: rndm
   end function RANDOM
end interface

! Creation of procedure pointer:
procedure(RANDOM), pointer :: rang => NULL()

! Initialization of procedure pointer:
rang => clientRNG

! Usage:
newRndm = rang()
```

FIGURE III.2. An example of a Fortran procedure pointer being created and utilized.

procedure pointers allows GSM and its sub-models to give clients more control over and flexibility with the object and they also allow the object to be more minimal. Regarding usage of random numbers, clients determine what kind of random number generator is used, *e.g.* a linear congruential generator (LCG) or other, and it allows clients to control or altar the random number generator prior to simulating an event, such as moving the random number generator to the next stride before each

event is simulated, as is done in many particle transport codes. A photon emission procedure is also utilized by GSM and two of its sub-models to allow clients of GSM to simulate photon emission if desired. Photon emission, regarding spallation physics, is of minimal concern and contributes very little to the total tally, whereas users may desire to track photon emission during spallation physics. Inclusion of the message handling procedure pointer allows clients to control how messages are printed. An example of such control could be printing all generated messages to the terminal and, in addition, printing all warnings and errors to an auxiliary file, output file, or both. Clients could use this information for debugging, informing users of issues in a very verbose way, and in many other useful ways.

III.2 PRELIMINARY MODERNIZATION

The benefits of modernizing GSM, given the context stated previously in Sec. III.1, are numerous. The primary benefit from the preliminary modernization focuses on migrating GSM to use modern syntax and considered best practices, and the majority of its modernization focuses on an object-oriented implementation. The preliminary modernization of GSM is considered to comprise usage of argument `intent`, explicit naming of variables, standardization of integer and real numbers across operating systems, and the inclusion of parameterized, or effectively parameterized, parameter and data modules. The naming of the parameter and data modules, each containing numerical data regarding standard and model-specific constants, respectively, was done such that it contains the model's name proceeded by "params" and "data," respectively.

Preliminary modernization of GSM has helped make GSM slightly less monolithic, with full modernization of GSM making it non-monolithic and completely modular. Many procedures within GSM previously utilized outdated data regarding particle rest masses and other miscellaneous data, as well as including in each procedure implicitly declared variables for numerical constants. Preliminary migration enabled the creation of model-specific data sets, helping to reduce the visible namespace of the program. Currently no data constructs exist outside of data modules and the

models are required to `use` their respective data and parameter module. GSM, being based on CEM, was previously very monolithic in that the namespace for GSM consisted of the namespace for every sub-model of GSM, and also having very little known dependencies on the other models or data objects. For example, the *Molnix* data class within GSM is utilized by the *Fission Barrier* data class and the *Standard DCM*, *Preequilibrium*, and *Evaporation* model classes. Prior to modernizing GSM these dependencies and their usage was unknown, where changes to the *Molnix* information would have propagated through much of the simulation results without the client or user being fully aware of where changes had occurred and what sub-models were affected by the change. Migrating to a modular framework allows more controlled and focused improvements to be made within each of the sub-models. Migrating GSM from a monolithic-type structure to a modular one is necessary, particularly when considering the technical debt reduction in GSM.

Monolithic programs, like GSM prior to its modernization, can have significant technical debt. GSM, prior to its modernization, had much debt in that no test suite existed prior to its modernization, each sub-model was tightly coupled to one another (being highly monolithic), little code documentation existed. The modernization of GSM has increased the code documentation, as a result of added `ReadMe` files for some models and the documentation contained in the Appendices[1], and in addition the interfaced code was refactored to use meaningful variable names within objects, such as `numBaryons` in a `compound` object, as opposed to `ap` among a set of non-coupled variables. GSM currently has a small regression suite that utilizes shell and python scripting to compare output files produced by GSM. The current regression testing is not necessarily platform-independent, however users may write their own scripts, alongside usage of the python script, to compare output files produced by GSM. Lastly, the migration to an object-oriented GSM has led to a fairly loosely-coupled GSM, where each sub-model and data object of GSM can be easily upgrade or replaced with another. Utilizing the example of the *Molnix* data class, clients do not need to incorporate any changes when the entirety of the underlying *Molnix* code could be significantly changed. This addition provides great inherent flexibility.

[1] See Appendix B, Appendix C, Appendix D, and Appendix E for some model documentation.

The modernized GSM, including both preliminary and final modernization, aids in its scalability across high performance computing (HPC) environments. Utilizing data and numerical constants for each sub-model within a module provides a smaller memory footprint when utilizing GSM with many processors or computer cores. Additionally, migration of GSM to modern, object-oriented Fortran allows it to more easily compute results as a result of the robust protection added from the object-oriented approach. The increased scalability of GSM provides for its clients and users the ability to perform simulations simultaneously, both for a single reaction within a usage of GSM and for multiple reactions within a usage of GSM. This increased scalability through parallelization provides great HPC utilization.

III.2.1 A Basic Case Study: the "chabs" Procedure

GSM, prior to modernization, utilized the `chabs` procedure within the standard DCM to determine isospin of nucleons after pion absorption, for example, according to Fig. III.3. Modernization of GSM and its sub-models has led this routine to have the interface shown in Fig. III.4 in addition to being migrated to the Fortran2003 standard as opposed to the Fortran66 and Fortran77 standard. The modernized version of the `chabs` procedure, as well as for most all procedures contained by GSM and its sub-models, now utilize an interface taking a similar format as that shown in Fig. III.4. Developers, and even users, may now obtain significantly more information about the procedure's arguments, their intent, and the entire namespace of the procedure without much investigation. Additionally, the modernized interfaces now take the form where some of the arguments cannot change value within the procedure, and numerical constants are now used as opposed to trusting developers to not change a value of the data variable within the procedure.

Improvements to the statements within the procedure include usage of the message handling procedure pointer for the printing of all messages, such as mathematical error protection messages. Additionally, parameterized constants are utilized within the sub-models for each mathematical error protection check, creating a relative standard within the model that may easily be modified as

```
      subroutine chabs(l, ine, ne1, ne2, a, z, r1)

      implicit real*8 (a-h, o-z), integer (i-n)

      data one /1.d0/

      .
      .
      .

      return
      end
```

FIGURE III.3. The standard DCM “chabs” procedure preceeding GSM modernization.

```
  subroutine chabs(sDCM, l, ine, ne1, ne2, a, z, r1)

     use, intrinsic:: iso_fortran_env, only: int32, real64
     use standardDCMParams, only: one

     implicit none
     class(StandardDCM), intent(inout) :: sDCM
     integer(int32), intent(in   ) :: l
     integer(int32), intent(in   ) :: ine
     integer(int32), intent(inout) :: ne1
     integer(int32), intent(inout) :: ne2
     real(real64),   intent(in   ) :: a
     real(real64),   intent(in   ) :: z
     real(real64),   intent(in   ) :: r1

     real(real64) :: t1, t2, temp, temp1

     .
     .
     .

     return
  end subroutine chabs
```

FIGURE III.4. The standard DCM “chabs” procedure proceeding modernization.

needed. The procedures that may also call to model-specific procedures is also highly limited in that a model's private procedures may only be accessed by the model itself and not by its clients, providing further protection to potential errors that may occur as a result of incorrectly calling the model's internal procedures. Robust interfaces may instead be utilized by clients, where a client's access to a model's procedure is strictly limited by the model itself. Usage of the sub-model class within the class procedure also allows for all necessary information needed by the procedure to be given to the procedure without much effort. Usage of a procedure-specific `results` object, passed in by clients at the model interface, also aids in scalability and robustness of the simulation. Inclusion of a simulation end-state flag in the results object also aids in that it provides clients with the ability to know what warnings, errors, or unexpected approximations may have been flagged during the simulation. Some of these flags are used to prevent simulations from occurring in the event that the object was not constructed, or failed during construction, or when the passed in `results` object was not constructed.

III.3 MIGRATING GSM TO AN OBJECT-ORIENTED ARCHITECTURE

Migrating GSM to an object-oriented approach comprised the majority of modernizing GSM. The monolithic nature of GSM was nearly completely removed as a result of migrating to an object-oriented GSM. Most every data and physics model of GSM, except its output handling and usage of the modified DCM, was migrated to a module containing the various objects needed for its usage, simulation, or both. The migration also provided a fine-tuned directory structure regarding the source code of GSM, therefore creating a clear and concise access and usage system within GSM.

The object-oriented GSM also has clearly and well-defined dependencies within itself and its sub-models as a result of migrating to an object-oriented code. For example, the source code in the `Preequilibrium` model would need to be searched and thoroughly studied to know its dependencies prior to modernization and migration. The dependencies of the `Preequilibrium`

model can now be easily determined by simply looking at the structure of the `Preequilibrium` object, as shown in Fig. III.5. According to Fig. III.5, the `Preequilibrium` model relies on the `Molnix`, `FissionBarrier`, and `PreequilibriumData` objects. It is of importance to note that the `PreequilibriumData` object, not shown here, contains within it the `Molnix`, `FissionBarrier`, and a `GammaJ` object, where the `Preequilibrium` object references those of its data object for consistency.

Dependencies on external libraries within a model can be easily discovered by reading the basic `CMakeLists.txt` file contained in the model's code directory, in particular the libraries the model must be linked to. The `Preequilibrium` model must, for example, link to the `Molnix`, `FissionBarrier`, `PreequilibriumData`, `FermiBreakup`, and `General` libraries contained within the GSM project. Observing the model's source code is required, however, to know the exact procedures and modules `used` by the model from a particular library. Fig. III.5 demonstrates a clear dependence on some of the sub-model data and physics objects. The `Preequilibrium` model utilizes the `General` modules and module procedures on a local basis within itself, thus those dependencies are not seen within Fig. III.5. Migrating the `General` modules' procedures to an object-oriented code would more clearly demonstrate the `Preequilibrium` model's dependency on the various procedures within the `General` physics modules utilized by the GSM and its sub-models.

The object-oriented GSM and its sub-models utilize many small objects to easily contain, store, and utilize information in the object. The majority of GSM and its sub-models, being organized similarly, all utilize some form of an `options`, `io`, `data`, and `results` objects, where appropriate. The `options` objects contain all client and user specifiable simulation options and behaviors. All constructors for GSM and its sub-models optionally allow users and clients to pass in the respective `options` object during construction. Note that, at this time, none of the constructors allow single values to be passed in to the model's constructor in order to simplify client use. Clients and users can easily adjust numerics, such as radius multipliers[2] used to obtain

[2]The optional multiplier, r_0, is used in the equation $r(A) = r_0 A^{1/3}$.

```
type, public :: Preequilibrium
   private

   ! Logic flag for being constructed:
   logical, private :: constructed =  .FALSE.

   ! For handling all messages:
   ! NOTE: Contains a character array and print procedure pointer
   type(preequilibriumIO), private :: io

   ! For simulation options:
   type(preequilibriumOptions), private :: options

   ! Required Data:
   type(Molnix),             private, pointer :: molEnergy => NULL()
   type(FissionBarrier),     private, pointer :: fissBarr  => NULL()
   type(PreequilibriumData), private, pointer :: preeqData => NULL()

   ! Random Number Generator Procedure:
   procedure(RANDOM), private, pointer, nopass :: rng => NULL()

   ! Fermi Break-Up object (for when small residuals)
   type(FermiBreakup), private :: fbuObj

   ! For photon emission (if client defined)
   logical, private :: usePhotonEmission = .FALSE.
   procedure(PHOTOEMISSION), private, pointer, nopass :: photonEmission =>
      NULL()
 contains
   private
   ! All member procedures go here; for example:

   ! Simultion execution names:
   procedure, public  :: simulatePreequilibrium
   procedure, public  :: equilibrate => simulatePreequilibrium
   procedure, public  :: decay       => simulatePreequilibrium
   procedure, public  :: deexcite    => simulatePreequilibrium
   procedure, public  :: execute     => simulatePreequilibrium
   procedure, public  :: start       => simulatePreequilibrium
   procedure, public  :: simulate    => simulatePreequilibrium

   ! For querying the object:
   procedure, public  :: properlyConstructed
   procedure, public  :: queryOptions
   procedure, public  :: queryMolnix
   .
   .
   .

end type Preequilibrium
```

FIGURE III.5. The class structure of the GSM “Preequilibrium” sub-model.

approximate nucleus radii, inclusion of an expanded feature, limiting the number of particles considered for emission, *etc.* Most of the GSM models check for the validity of the client-defined options. GSM utilizes within its `options` object all of its sub-model `options` objects in addition to several other options, such as at what energies GSM should use the modified DCM in place of the standard DCM, how often to print to user the last successful event number, and choice of fission multiplier sets.

GSM and its sub-models all utilize an `io` object to easily contain a character array and procedure pointer for the generation and control of errors, warnings, and comments produced during the simulation. Most models within GSM, including itself, utilize a `data` object for construction prior to simulation to allow for computationally efficient simulations. The respective model data may be dependent on the particular reaction being simulated, therefore GSM constructs all appropriate reaction-specific data upon initial usage or at each event when being used by the input driver or by a client program, respectively. All general data not requiring information about the simulation is constructed by GSM within its own constructor. The `results` objects utilized by GSM and its sub-models all have several embedded objects as well, typically including an array of particle data, residual or compound nuclei information, exciton information, and a simulation end-state flag. Some of the `results` objects also contain other non-object data types, such as private logic flags to indicate construction, flags for the number of errors, *etc.*

Usage of a `results` object in this way allows for significantly better memory utilization. Fortran is a column-major programming language, meaning that elements of the same column are stored contiguously in memory. Many of the results tracked by GSM previously utilized arrays of several dimensions, and thus provided poor memory utilization. Migration to an array of objects increases memory utilization according to Fortran's column-major memory access methodology. The computer can query the cache of memory for a single object, as opposed to previously needing to query for each variable of the object, when the information for an object is needed. The models themselves also act as containers by possessing some of the above objects, and in addition by containing construction flags, other procedure pointers, and object pointers. The inclusion of these

many smaller objects allows for a highly flexible framework within each sub-model and GSM itself. Addition of options to a sub-model in this way, such as allowing for the emission of more particles, is easily propagated through to all clients of the model. Usage of objects to execute simulations also aids in future expansions of the models as new properties to the object are easily incorporated. Usage of many smaller objects in this sense aids in the flexibility of the model, particularly regarding the consistency of client usage of the models after modifications have been made to a model.

III.4 BENEFITS OF THE OBJECT-ORIENTED APPROACH

Migrating GSM to an object-oriented framework has had many intrinsic benefits. GSM is now based upon a modern programmatic framework that is more flexible and portable across platforms. As stated in Sec. III.3, GSM and its sub-models inherently utilize better memory access as a result of utilizing many small objects. In addition to utilizing a modern framework for its code, GSM has many benefits for both its clients or users and its developers.

Client programs and users benefit from the modern GSM framework through a simple object setup, significant client control, reusability of the object among different and simultaneous simulations, being highly scalable, and as a result being highly parallelizable. Setup of GSM, its sub-models, or any combination of GSM's objects, is very simple in that clients and users simply pass the required objects to the object's constructor, such as a random number generator procedure pointer, a data object, or other. Model data is established via calling the model's data initialization procedure or by constructing an object itself. Each model and data object of GSM verifies that its numerical data is established prior to simulation, providing a layer of protection against clients and users who fail to establish model data prior to model usage. Failure of all model data to initialize properly will result in the model not allowing any simulations to occur through it. Each data and model object within GSM also verifies that all passed in arguments are valid. Validating constructor arguments within GSM may take the form of checking the physicality of an argument, ensuring all procedure pointers are associated, and ensuring all model and data objects are constructed prior to

use. Instances where the model cannot utilize a default value, such as in cases of model objects, data objects, or random number generator procedures, will result in the model flagging the error and neglecting the physics that should have been simulated. Client programs can then, if desired, utilize the flagged error to determine the cause of the error and correct it. Exiting the model's simulation in this way allows clients to cleanly exit the program, if desired, without causing a potential crash of the program. The models in this manner provide a robust protection to themselves, potential crashes, and to their clients when utilized by the client.

Clients may `use` and pass the model's respective `options` object to the model's constructor to control its behaviors internally without being required to know how the change is propagated through the model, providing a "black-box" type of use to the model's clients. Utilizing these options provides significant control to clients by allowing to neglect certain physics, or bias them if desired, within the models. For example, clients could prevent the compounding of alpha particles as a result of coalescence by reducing the coalescence radius for alpha particles to zero. Clients can control these models on an individual basis as a result of the object-oriented implementation. Note that clients and users should be aware of the default values and behaviors of the objects prior to modifying the parameterizations in each of the model's respective `options` object[3].

GSM and its sub-models were designed to be highly reusable. The models, to an extent, act as a skeleton for a simulation. Each of the models provides clients and users with the algorithms and behaviors necessary to perform the simulation. Clients and users of the models must simply the required data, any necessary procedure pointers, and simulation inputs to then perform simulations with the object. The objects then, having a fairly minimal state, can be used as a skeleton for any number of simultaneous simulations. Clients of these models can then construct and utilize a single object during parallel calculations using `MPI` and can be copied across processors using `openMP` without introducing dependencies between simultaneous simulations of the model. Note that usage of these models in this way is discouraged as many of GSM's sub-models, including itself, rely

[3]For example, the coalescence model defaults to negative coalescence radii. This flags to the coalescence model to use default values. Clients in this manner simply change the desired particle's coalescence radius to any positive value to use a non-default coalescence radius.

on a single random number generator procedure pointer that does not utilize a skip-ahead feature, possibly introducing inconsistencies between simulations on different computers.

Each object of GSM, its sub-models, or any combination thereof, are highly scalable in this manner. The minimal state of each object, and the lack of global memory usage within each object, allows the object to be scaled up as much as required by the client or user for its simulations. Scaling of the model is typically done through use of horizontal scaling, better known as parallel calculations on performed on computers. Vertical scaling of the models, as done by providing more powerful computers to perform the simulations, will also work well in that GSM and most of its sub-models utilize better memory management via minimal object states, migration to utilize mostly single-arrayed objects[4], and by reduction of the namespace within each procedure of each modernized model. Data utilized by each of the models is also global across all models, thereby reducing the amount of memory necessary to copy among processors. Utilizing compiler optimizations may also provide for an improved vertical scaling of the models as parameterized data may be substituted, as well as other potential optimizations depending on the model and flagged optimization level.

Developers of GSM benefit from the object-oriented framework as issues and bugs found within GSM, or its sub-models, can be easily fixed and tested within the model itself. Resolved issues can be quickly deployed to all of the model's user base. Addition of new features, or improving existing features, of a model may also be completed easier as the edited model may be individually modified, tested, and quickly deployed. The addition or improvement of new or existing features may also be done without modifying client usage of the model itself, providing clients with simple and easy to use interface to the model. Many of these benefits for developers stem from the highly decoupled nature of the models within GSM. The decoupled nature of GSM provides developers with a robust architecture to update existing models, create new features, and even replace, or test, other models in place of existing models.

[4] As opposed to multi-dimensional arrays, arrays of objects provide better memory utilization as the Fortran language caches data using a column-major memory storage scheme. Arrays of objects also provide better memory utilization across other programming languages, however Fortran is focused on here.

III.5 THE GSM OBJECT

The GSM object contains within it several general physics classes and variables that can be used by all simulations of GSM. GSM in this manner is highly scalable and balances computational efficiency with its state, requiring little memory for its existence when not being used for simulations. Note that GSMcannot be used for parallel simulations at the time of writing this document, however the architecture it utilizes can be easily migrated to be fully parallelizable.

It is important to note that the following discussion regards the internal structure of GSM, and this discussion also applies to most of the sub-models and classes utilized by GSM for consistency. Many similarities exist in the sub-models architecture in that all sub-models utilize a logical `constructed` flag, an I/O-type object, an object for containing model behaviors and options, a random number generator procedure pointer where applicable, necessary data objects, and in some cases, interfaces, procedure pointers, or both, relating to other model physics. GSM utilizes all of these and is self-contained in that clients are not required to provide any arguments upon construction, with optional arguments to the GSM constructor being simulation options, a random number generator procedure pointer, and a procedure pointer to control the handling of messages generated during GSM simulation.

The GSM object contains within it a logical `constructed` flag, an I/O object, an options object, and random number generator procedure pointer, a general physics data object, and a general physics model object. The general physics data and model objects contain within them all reaction-general physics that do not require information about the reaction for their construction. Model objects requiring information about the simulation require data objects for the particularly model that the model stores a pointer to for its simulations. All reaction-dependent models are constructed at the start of a GSM simulation.

The GSM object contains within it a logical `constructed` flag, informing itself and users if the object was successfully constructed, meaning all required model data and associated pointers are initialized. GSM fails to construct in the event its reaction-general data and model objects

fail to construct. A GSM object, by default, is flagged as uninitialized, meaning `.FALSE.`, upon declaration. The `constructed` flag is changed to `.TRUE.` when its associated object is successfully constructed without error. GSM objects flagged as not being constructed will not allow users to simulate spallation physics with it. The GSM object will attempt to construct all model data, upon construction, if the model data was not flagged as being constructed. Failure of model data to be flagged as properly initialized will flag the object as not being constructed.

GSM objects all contain an I/O-type of object. The object contains within it a character array of length 512. The length of the array is thought to be sufficiently large that all error messages printed by GSM will not be trimmed. The I/O object also contains within it a procedure pointer associated with a message handling procedure. The procedure has the interface where its arguments all have `intent(in)`, being an integer message verbosity flag, an integer message type flag, and the message itself. The verbosity flag is used to restrict the message printed, where all messages with verbosity below the desired verbosity level are printed. The message type flag indicates the type of message being printed, being a `Fatal Error`, an `Error`, a `Warning`, and a `Comment`, having a value of 1, 2, 3, or 4, respectively. The default GSM printing procedure allows a miscellaneous message type as well, having a value of 5, where no text is prepended to the message being printed.

The default printing procedure of GSM prints all `Fatal Error`s regardless of verbosity. Messages flagged as `Fatal Error`s, `Error`s, having a negative line type flag, or any combination thereof, are printed to the standard error unit, with all other messages printed to the standard output unit. These units are associated, respectively, with the `error_unit` and `output_unit` parameters of the intrinsic `iso_fortran_env` module. Messages flagged with a negative message type simply utilize the message type associated with the positive value of the flag. Provided messages with an invalid message type flag utilize the `Fatal Error` indicator. A default verbosity of one or two is recommend when using GSM in clients, a verbosity of two to three when using GSM in standalone, and a verbosity of four or higher is recommended for developers of GSM. The interface to the GSM procedure pointer, more explicitly, has the form of Fig. III.6. Clients can provide a GSM object with their own message printing procedure, if desired, during the object's construction.

```
! The interface for the I/O procedure pointer:
abstract interface
   subroutine IOHANDLER(verbosity, type, text)
      use, intrinsic:: iso_fortran_env, only: int32
      implicit none
      integer(int32),   intent(in) :: verbosity
      integer(int32),   intent(in) :: type
      character(len=*), intent(in) :: text
   end subroutine IOHANDLER
end interface

! Creation of the procedure pointer:
procedure(IOHANDLER), pointer :: thisClientsIO =>
    someProcedureWithTheIOHANDLERInterface

! Construction of GSM with the new procedure pointer:
type(GSM) :: myGSMObj = newGSM( clientIO = thisClientsIO )
```

FIGURE III.6. The procedure pointer interface and its usage for all I/O handling within GSM and its sub-models.

The GSM object contains within it the `GSMOptions` object. The object contains within it all associated `options` objects from its data objects and model objects in addition to GSM-specific options. All options are publicly available to clients to allow clients to control the behavior of GSM and each of its sub-model objects. Clients may specify the options GSM will use during its construction only. GSM internally validates all construction options for itself and allows its sub-models to validate the respective options utilized and updates its options by querying the respective object.

GSM utilizes the random number generator stored within the GSM data module by default. The default random number generator, being a copy of the MCNP6.2 random number generator, is initialized and pointed to by the GSM data module when its data is initialized. Clients of GSM can set the type of random number generator utilized by GSM by passing in an integer flag, ranging from one to seven, to the GSM data initialization procedure as its first argument. Passing in a flag outside the valid range will result in the default type being used. Fig. III.7 demonstrates the usage of one of the default RNG types in GSM. Clients may choose to pass in to the GSM constructor their own

random number generator procedure. Fig. III.7 demonstrates the interface of said procedure and demonstrates the construction of GSM with a client-defined random number generator procedure.

GSM, when utilizing its default random number generator, will utilize the stride corresponding to the inelastic event number. GSM will move to the next stride in the sequence after an inelastic event is successfully simulated. GSM does not utilize the stride incrementing feature when using a client-defined random number generator to aid in computational efficiency. Note the stride of the default random number generator in GSM is 152,917 and was seen to be exceeded at least one time by approximately 10% of the regression simulations.

```
! The interface for the RNG procedure pointer:
abstract interface
   function RANDOM() result(rang) BIND(C)
      use, intrinsic:: iso_C_binding, only: c_double
      real(c_double) :: rang
   end function RANDOM
end interface

! Creation of RNG procedure pointer:
procedure(RANDOM), pointer :: rngPointer => clientRNG

! Using RNG of type four (one of the seven) in GSM:
call initializeGSMData( clientRndmType = 4 )

! Construction of GSM with RNG options:
type(GSM) :: myGSMObj = newGSM( )    ! GSM using default RNG of type 4 (
   LEcuyer3)
type(GSM) :: myGSMObj = newGSM( clientRNG = rngPointer )  ! Using client RNG
    instead
```

FIGURE III.7. The procedure pointer interface and its usage regarding the GSM random number generator.

GSM utilizes for its data objects a container-object to easily store all general physics data objects, namely general physics data objects as well as model-data objects. This object gets constructed at the end of GSM construction. The general data objects stored here include the `Molnix`, `FissionBarrier`, and `EvaporationData` objects. The same is done for general model physics objects. The general physics models are stored within a container in GSM. These objects include the standard DCM, Fermi break-up, and evaporation models. The coalescence and

preequilibrium models, being reaction-general themselves, point to reaction-specific data objects internally, and thus cannot be stored within these general objects.

GSM lastly contains within it a logical flag stating whether or not photon emission is considered during simulations by that object. By default, GSM neglects all photon emission. Clients of GSM can include photon emission in their simulations by initializing the photon emission procedure pointer within GSM. Initializing this procedure includes photon emission in GSM simulations. Instances of GSM constructed prior to inclusion of a photon emission procedure will not use the photon emission procedure. Clients may include photon emission in GSM by constructing, or re-constructing, the instance of GSM after pointing the GSM module to a valid photon emission procedure. The photon emission procedure pointer utilized by GSM is provided to all instances of GSM as it is an external model to GSM and its sub-models. The photon emission procedure may be initialized by calling the static `setPhotonEmission` subroutine, contained by the `gsmClass` module. The procedure utilized by GSM for photon emission has the interface shown in Fig. III.8, also demonstrating the initialization of a photon emission procedure within GSM. Note that GSM internally interfaces to all models where photon emission is thought to occur, namely being preequilibrium, evaporation, and Fermi break-up. Not all information made available by the `GSMResidual` object may have been initialized as a result of the interfacing, defaulting to a value of zero.

```
! Import necessary objects and procedures from GSM:
use gsmClass, only: GSM, newGSM, setPhotonEmission
type(GSM) :: myGSMObj1, myGSMObj2

! Interface for photon emission procedure:
abstract interface
   subroutine PHOTOEMISSION( progeny, residual )
      use gsmClass, only: GSMProgeny, GSMResidual
      implicit none
      type(GSMProgeny),  intent(in) :: progeny
      type(GSMResidual), intent(in) :: residual
   end subroutine PHOTOEMISSION
end interface

! Creation of photon emission procedure pointer:
procedure(PHOTOEMISSION), pointer :: photoPointer =>
    clientPhotonEmissionProcedure

! Construct GSM object that does NOT utilize photon emission procedure:
myGSMObj1 = newGSM()

! Setting the photon emission procedure to use:
call setPhotonEmission( photoPointer )

! Construct GSM object that DOES utilize photon emission:
myGSMObj2 = newGSM()
myGSMObj1 = newGSM()    ! Re-constructing GSM object to utilize photon
    emission
```

FIGURE III.8. The procedure pointer interface, initialization, and usage within GSM.

Chapter IV. GSM Usage

GSM may be easily adapted for use in a number of ways given its updated and modern code architecture. GSM requires that all of its simulation data is initialized and that the GSM object being utilized is constructed. The simulation data utilized by GSM is required to be properly initialized prior to any GSM object being constructed. GSM may be used to simulate spallation physics by drivers and clients by providing the appropriate GSM procedure with a projectile and target object, and either an output file or simulation results object. The GSM driver, being contained within a module outside of the GSM object itself, acts simply as a client to GSM that end-users may utilize to have GSM generate output files in a standalone setup. Clients wanting to simulate GSM for single-events may instead `use` the `gsmClass` module and the `GSM` object accordingly.

IV.1 GSM DATA INITIALIZATION

GSM utilizes the `generalizedSpallationData` module to contain all of its simulation data. This module contains within it a procedure, `initializeGSMData`, clients may call to initialize all required GSM data. GSM objects themselves verify that model data has been initialized prior to being constructed, requiring that model data is established without error for GSM to perform simulations. Note that GSM, during construction, will attempt to initialize all of its model data if not previously performed by clients using default arguments. The `initializeGSMData` procedure does not require any arguments be passed in, however clients may specify several optional arguments, as discussed below.

The `initializeGSMData` procedure, upon being invoked, checks if the GSM data has already been successfully initialized, sets the data module's message handling procedure, and initializes the required default random number generator (RNG) and all sub-model data using any passed in arguments. Note the procedure does not re-initialize data if it has already been initialized

properly. Fig. IV.1 demonstrates the initialization of the GSM data using both default options and client-specified options. Note the client specified options utilize the default values used within the various modules. The default random number generator utilized by GSM may have an integer type ranging from one to seven, where type one is the standard GSM RNG and the remaining RNG types are named L'Ecuyer one, two , . *etc.*, up to six. Note all of the RNGs utilized by GSM are of the linear congruential type. Clients may specify the type of RNG that all instances of GSM will utilize by passing in an integer, within range one to seven, to the `clientRndmType` argument of the `initializeGSMData` procedure. The GSM data module will utilize the standard GSM RNG type when not, or incorrectly, specified by the client. The GSM data module creates a private procedure pointer to the RNG. All GSM objects then query this procedure pointer via the data modules `queryRNG` procedure to utilize for their simulations.

The `initializeGSMData` procedure initializes all sub-model data after initializing the RNG. This procedure first initializes the standard DCM data using a data file name for photon cross section data. Clients may specify the data file's name using the `gammaFile` character array argument. Clients may additionally specify the default unit number to use for reading data from the file in using the `gammaUnit` integer argument of the procedure. The standard DCM data initialization procedure then attempts to use the client defined unit, or the default, if available. The unit number is incremented up to 50 times in the case that the unit is already open and associated with another file being used by the program. The initializer will attempt to verify that the desired data file name exists according to the file's associated path and name. In the case where the file is not found, the initializer will attempt to utilize the default file name. Failure to find the required data file will result in the failure of data to be initialized. Failure of the standard DCM data to initialize will flag the GSM data module to state that data is not properly initialized, thus preventing the simulation of spallation physics using GSM.

The modified DCM sub-library requires two data files be read and stored in memory for its simulations. The `initializeModifiedDCMData` subroutine, being the primary method clients utilize for initializing the modified DCM's data, allows up to five arguments for its usage.

```
subroutine clientGSMDataInitialization( clientProcedure )

   use generalizedSpallationData, only: &
        & initializeGSMData, &   ! Subroutine to initialize data
        & gsmDataInitialized     ! Logical flag indicating if data was
            initialized

   implicit none
   procedure(IOHANDLER), intent(in   ), pointer :: clientProcedure

   ! Verify data was not already initialized:
   if ( gsmDataInitialized ) then
      write(*,*) "The GSM data was already successfully initialized."
      return
   end if

   ! Initialize data using default arguments:
   call initializeGSMData()

   ! Initialize data using optional arguments (with default values):
   call initializeGSMData( &
        & clientRndmType = 1,           &   ! Random Number Generator
        & gammaFile = "channel1.tab", gammaUnit = 17, &   ! sDCM Defaults
        & photoFile = "channel1.tab", photoUnit = 17, &   ! mDCM Defaults (
            photon file)
        & decayFile = "atab.dat",     decayUnit = 18, &   ! mDCM Defaults (
            decay file)
        & massFile  = "mass.tbl",     massUnit  = 19, &   ! Evaporation
            Defaults (mass file)
        & levelFile = "level.tbl",    levelUnit = 20, &   ! Evaporation
            Defaults (level file)
        & shellFile = "shell.tbl",    shellUnit = 21, &   ! Evaporation
            Defaults (shell file)
        & clientIO = clientProcedure                  &   ! I/O for message
            handling
        & )

   ! Check if initialization was successful:
   if( .not.gsmDataInitialized ) then
      write(*,*) "The GSM data failed to initialize."
   end if

   return
end subroutine clientGSMDataInitialization
```

FIGURE IV.1. Sample GSM data initialization utilizing both no arguments and all available arguments.

These arguments include the respective photon file name and desired unit number, the respective decay file name and desired unit number, and a message handling procedure pointer. The modified DCM initialization procedure does not, unlike the other GSM sub-libraries, attempt to find an available unit number or attempt to determine the existence of the provided file name. Errors generated during the data initialization of the modified DCM do not currently flag its improper initialization. Note the photon file read by the modified DCM is, by default, the same as that of the standard DCM. Clients may in this manner provide a unique data file with different numerical values. The decay file provided in GSM, `atab.dat`, provides the modified DCM's quark-based identity code with numerical data.

The `initializeGSMData` procedure initializes both the Fermi break-up and preequilibrium model data as well. The data required for these models does not utilize data files or outside arguments, thus invoking the data initialization procedures for these models will flag the data as initialized. The evaporation model data is established by providing the evaporation data initialization procedure with mass, level, and shell data file names and units. The optional arguments associated with the mass data file are named `massFile` and `massUnit`, respectively. Arguments associated with the level data file are named `levelFile` and `levelUnit`, and those associated with the shell data file are named `shellFile` and `shellUnit`. In the same regard as the standard DCM data initialization procedure, the evaporation data module will utilize only available and non-associated unit numbers and verify the data files exist prior to opening them, thus preventing unexpected program failures.

Clients may pass in a message handling procedure pointer that the GSM data module may utilize. The module contains within itself its own message handling procedure, however in this respect allows its clients to specify the procedure if desired. Clients of the GSM data module may provide the GSM data module with this procedure pointer by assigning the `clientIO` argument of the `initializeGSMData` procedure. Note the data module will not utilize the procedure pointer when it is not associated.

IV.2 GSM CONSTRUCTION

An instance of the GSM class must first be constructed prior to its use for simulations. GSM will utilize all default arguments if it has not been flagged as being constructed. Prior to being constructed, the GSM object will verify that its model and sub-model data have all been initialized. Failure of the data to be initialized will flag the GSM object as not being constructed. Sec. IV.1 provides more details on how to initialize GSM model and sub-model data. GSM objects that have been flagged as unconstructed will not be able to simulate spallation physics. The GSM object may be constructed via the `newGSM` constructor interface, allowing clients to construct the GSM object and use it for their high-energy spallation simulations. The GSM object only needs to be constructed once by its clients and may be used to simulate any reaction afterwards.

Clients of GSM may construct a GSM object according to that shown in Fig. IV.2 using none and all of the optional arguments given by the GSM constructor. The procedure pointer interfaces shown in Fig. IV.2 are those shown in Fig. III.7 and Fig. III.6 for the RNG and message handling procedures, respectively. It is important to note that the GSM object will not utilize the pointers if they are not associated, thus requiring clients of the GSM object to associate all pointers provided to GSM prior to the object's construction. The GSM object during its construction will verify that model data has been initialized, utilize the message handling procedure optionally provided by the client, establish its RNG procedure, set and validate model options and behaviors, and lastly construct reaction-general data and model objects.

Clients may construct GSM objects to use a client defined RNG function by providing the constructor with the random number generator procedure pointer. GSMwill then utilize the client's procedure, if associated, for its simulations. Note that GSM uses, in conjunction with its default RNG, a single stride for each successfully simulated inelastic event. Warning messages are provided to the message handling procedure when an inelastic event has exceeded the stride, causing pseudo-random numbers to be reused. GSM objects having a client-defined RNG do not utilize this skip-ahead feature to aid in computational efficiency. It is recommended that clients provide GSM

```
subroutine constructGSM( thisRNG, thisGSMOptions, thisIO )

   use gsmClass, only: &
        & GSM,          & ! GSM class/object type
        & newGSM,       & ! GSM constructor interface
        & gsmOptions      ! Options for the GSM simulation

   implicit none
   procedure(RANDOM),     intent(in   ), pointer :: thisRNG
   type(gsmOptions),      intent(inout)          :: thisGSMOptions
   procedure(IOHANDLER), intent(in   ), pointer :: thisIO

   ! Declare a GSM object:
   type(GSM) :: gsmObj


   ! Construction of GSM with default arguments:
   gsmObj = newGSM()

   ! Construction of GSM with all optional arguments:
   gsmObj = newGSM(                          &
        & clientRNG     = thisRNG,           &
        & clientOptions = thisGSMOptions, &
        & clientIO      = thisIO             &
        & )

   ! Update client's options object to reflect those used:
   thisGSMOptions = gsmObj%queryOptions()

   return
end subroutine constructGSM
```

FIGURE IV.2. A sample GSM class construction interface.

with their own RNG procedures, even if the procedure is the same as that used by default in GSM, to aid in event queries by end-users regarding client simulations. Clients utilizing multiple GSM objects are highly recommended to provide their own RNG procedures as each GSM object utilizes the same RNG procedure by default, potentially causing several strides of random numbers to be skipped without merit during client simulations that heavily utilize GSM.

The `gsmOptions` object contains within it several options and behaviors used by GSM, and in addition contains the options objects for its sub-models and their associated data objects. Clients in this manner may alter the default values utilized by the various options as they see fit. Upon construction, GSM will validate the options object when provided by the client as it constructs its sub-models. The sub-model objects constructed by GSM validate their own options objects. GSM will, after constructing the object, update its options object to that being used by its sub-model data or class object to reflect to its clients the options and behaviors being utilized by GSM. Clients of GSM can query the options object it utilizes by calling the `queryOptions` function of GSM, as shown in Fig. IV.2.

Clients are strongly encouraged to provide GSM with their own message handling procedure to maintain uniformity across the messages generated by the client. The GSM message handling procedure, detailed more in Fig. III.6 of Sec. III.5, is thought to provide sufficient information to its clients regarding the message type, its importance to the simulation, and the message itself. Clients may easily create their own interfaces to their message handling procedure(s), if any, using the information provided by GSM for a given message. Clients may also desire to print the information, depending on its flagged verbosity or importance and its type, to multiple units. In this sense, clients may important messages to the end-user on the terminal, to any axillary or output files currently running, *etc.*, to provide end-users with important information about the simulation. GSM, when using its default message handling procedure, utilizes a module-global verbosity limit, where all messages with flagged verbosity greater than the specified limit are not written to the end-user or to any files. Clients may alter the value of the `gsmVerbose` integer flag as desired by `use`ing and directly modifying it within the `gsmClass` module. A verbosity limit of one to

two is recommended when GSM is being used in clients, two to three for end-user simulations, or three or higher for developer simulations. The default verbosity limit at the time of this thesis is four, where all messages produced with verbosity less than or equal to the limit are printed. This level of verbosity prints most information generated by GSM during its simulations, except for IntraNuclear Cascade simulation restarts, typically occurring as a result of unphysical residual nuclei being created during the IntraNuclear Cascade simulation.

IV.3 SIMULATION ARGUMENTS

GSM requires that the user, being its driver or a client program, pass in a `GSMProjectile` object, a `GSMTarget` object, and either a `GSMOutput` or `GSMResults` object, depending on whether the driver or a client is utilizing GSM, respectively, for spallation physics simulations. Clients of GSM will need to utilize the `GSMResults` object directly to specify the size of the progeny nucleus array, constructing the `GSMResults` object with a `GSMProgeny` array. The GSM driver utilizes the `GSMOutput` object, containing most of the input file specifications, to generate an output file based on the end-user's input file. GSM internally creates and utilizes a `GSMResults` object to simulate as many events as specified by the end-user, tallying results at the end of inelastic events and using the `GSMOutput` object to determine what information is written to the desired output file.

The `GSMProjectile` object is detailed in Table IV.I. GSM utilizes the `GSMProjectile` object for its spallation simulations, utilizing the information provided by the clients. Clients are recommended to the `GSMProjectile` object with as much information as desired. The default incident particle simulated by GSM is a proton with kinetic energy of 1.0 GeV, where GSM determines the rest mass of the particle and the A_f and C_Z fission parameter multipliers to be used with the particle.

The projectile's name may be specified solely at the client's discretion, except for instances of incident pions and photons, where the projectile name is either `pipl`, `pimi`, `pize`, `gamm`,

TABLE IV.I. Variable Members of the "GSMProjectile" Object

Member Name	Scope	Data Type	Default	Description
particleName	Public	character(6)	*prot*	Name of the particle
numBaryons	Public	integer(int32)	1	Number of nucleons in the nucleus
numProtons	Public	integer(int32)	1	Number of protons in the nucleus
decayNumber	Public	integer(int32)	0	Quantum decay number
kinEnergy	Public	real(real64)	1.0	Kinetic energy of the incident nucleus [GeV]
kinEnergyMax	Public	real(real64)	$1E9$	Maximum kinetic energy of the incident nucleus [GeV]
dKinEnergy	Public	real(real64)	-50.0	Step size of incident energy [MeV] (≤ 0 unused)
restMass	Public	real(real64)	-1.0	Rest mass of the nucleus [GeV/c^2]
afMultiplier	Public	real(real64)	-1.0	A_f multiplier within the fission model
czMultiplier	Public	real(real64)	-1.0	C_Z multiplier within the fission model
particleFlag	Public	integer(int32)	1	Flags the incident particle type as a nucleus (0), mono energetic photon (1), bremsstrahlung photon (2), or a pion (3)
system	Private	integer(int32)	1	Flags the system used during the INC calculation

or `gamb`, representing incident π^+, π^-, π^0, mono-energetic photons, or bremsstrahlung photons, respectively. GSM does allow clients to utilize its default names for incident protons and neutrons, being `prot` and `neut` respectively, however usage of these names is not required by GSM. Note that GSM will override the incident particle's provided baryon and proton numbers, its rest mass, and particle flag when these default projectile names are utilized. Note GSM does not utilize the `kinEnergyMax` or `dKinEnergy` variables for single-event simulations, except when the incident particle is a bremsstrahlung photon. Clients may specify the minimum energy of an incident bremsstrahlung photon as the `kinEnergy` variable, and the maximum energy of the bremsstrahlung photon as the `kinEnergyMax` variable. GSM will attempt to set the incident particle's rest mass, in GeV/c^2, when a negative value is present in the `restMass` field. Clients may specify the rest mass desired by setting the `restMass` field to any positive value. GSM uses Eqn. IV.1 to determine the rest mass for its incident particles.

$$E_0 = \begin{cases} 0.9382723, & \text{for “particleName”=“prot”,} \\ 0.9395656, & \text{for “particleName”=“neut”,} \\ 0.139568, & \text{for “particleName”=“pipl”,} \\ 0.139568, & \text{for “particleName”=“pimi”,} \\ 0.134973, & \text{for “particleName”=“pize”,} \\ 0.00, & \text{for “particleName”=“gamm”,} \\ 0.00, & \text{for “particleName”=“gamb”,} \\ 0.9382723Z_p + 0.9395656(A_p - Z_p), & \text{otherwise} \end{cases} \tag{IV.1}$$

Clients may specify the A_f multiplier, the C_Z multiplier, or both, to utilize in the simulation. GSM utilizes its own A_f and C_Z multipliers, being based on experimental fission data, when the respective multiplier's field is negative. Clients may then specify the A_f multiplier and allow GSM to specify the C_Z multiplier, or any combination thereof. Note that invalid entries for the variable-

members of the `GSMProjectile` data type, being thought of as unphysical or not considered, will result in the default value for that variable-member being utilized.

GSM utilizes the `GSMTarget` object as the target nucleus of the incident particle for its simulations. GSM needs only information about the target nucleus's nuclear composition for simulation. Table IV.II depicts the variable-members of the `GSMTarget` object. The default particle considered as the target nucleus for spallation simulations in GSM is ^{208}Pb. It is recommended that clients name the target according to its ZAID, being specified as $ZAID = 1000Z + A$. GSM will check for invalid entries in the `GSMTarget` data type, where all negative values utilize the default target considered and all out-of-bounds values, such as nuclei with more than 300 nucleons, utilize the closest valid value. In this manner, GSM will approximate a collision of an incident particle on one with 140 protons and 165 neutrons as a collision of the incident particle on one with 118 protons and no more than 300 nucleons, being a ^{300}Og nucleus. Note the rest mass of the target nucleus is, at the time of writing this document, not utilized by GSM for any purpose other than to allow clients to specify the rest mass of the particle in the anti-laboratory system, where the target is treated as the projectile and the projectile is treated as the target. The rest mass is altered in GSM according to Eqn. IV.1 when not correctly specified by the client. GSM will determine the A_f and C_Z multipliers, by default, according to its multiplier interpolation procedures when not specified by the client.

TABLE IV.II. Variable Members of the "GSMTarget" Object

Member Name	Scope	Data Type	Default	Description
particleName	Public	character(6)	*Pb-208*	Name of the target nucleus
numBaryons	Public	real(real64)	208.0	Number of nucleons in the nucleus
numProtons	Public	real(real64)	82.0	Number of protons in the nucleus
restMass	Public	real(real64)	−1.0	Rest mass of the nucleus [GeV/c^2]
afMultiplier	Public	real(real64)	−1.0	A_f multiplier within the fission model
czMultiplier	Public	real(real64)	−1.0	C_Z multiplier within the fission model

GSM will determine the IntraNuclear Cascade model best suited for the simulation given the collision of the incident and target particles, being represented by the `GSMProjectile` and

GSMTarget objects respectively. GSM will utilize the standard DCM for its IntraNuclear Cascade simulations regarding incident protons, neutrons, pions, and photons below some threshold energy, being set by the GSMOptions object, and will utilize the modified DCM for its IntraNuclear Cascade simulation otherwise. Note that the modified DCM within GSM has not been fully migrated to an object-oriented model at the time of writing this document, thus performing multiple simultaneous GSM simulations that utilize the modified DCM is highly discouraged as results may become mixed during the simulation due to global memory within the modified DCM code. Collisions where the incident particle is larger is nuclear composition than the target particle will utilize the anti-laboratory system. GSM will swap the particles' name, nuclear composition fields, and rest mass accordingly in the GSMProjectile and GSMTarget objects when utilizing the anti-laboratory system. Currently usage of the anti-laboratory system is restricted within GSM to simulations utilizing the modified DCM for IntraNuclear Cascade simulations, thus GSM will force usage of the modified DCM when simulating collisions in the anti-laboratory system.

Clients may simulate a single event in GSM by providing the associated procedure, detailed in Sec. IV.5, with a GSMProjectile, GSMTarget, and GSMResults object. Results of the simulation are tracked within the GSMResults object during the simulation, with some information regarding the sub-model simulations being omitted from results tracked. Table IV.III details the member-variables of the GSMResults object. Note that clients need to construct the GSMResults object prior to utilizing the object for simulation to ensure the simulation can track and accumulate progeny created during the reaction. The object may be constructed via the newGSMResults constructor interface. Fig. IV.3 details the required construction of the GSMResults object, where the only argument utilized is an array of GSMProgeny that will be filled with progeny as a result of a spallation simulation. Failure to construct the GSMResults object prior to its usage in a GSM simulation will trigger GSM to exit the simulation at the start of the simulation, prior to any progeny creation. Note that GSM internally resets the GSMResults object at the start of an event, therefore clients should not partially set any of the members of the GSMResults object as this information will be lost during the GSM simulation.

TABLE IV.III. Variable Members of the "GSMResults" Object

Member Name	Scope	Data Type	Description
initialProj	Public	GSMProjectile	Pointer to the initial projectile nucleus
initialTarg	Public	GSMTarget	Pointer to the initial target nucleus
projRes	Public	GSMResidual	Residual projectile nucleus at the end of the simulation
targRes	Public	GSMResidual	Residual target nucleus at the end of the simulation
projExc	Public	excitonData	Projectile exciton information at the end of the simulation
targExc	Public	excitonData	Target exciton information at the end of the simulation
progenyBnk	Public	GSMProgeny	Array pointer to the client-defined progeny array
numProgeny	Public	integer(int32)	Number of progeny existing the in the progeny bank
maxProgeny	Private	integer(int32)	Size of the progeny array
maxProgenyM1	Private	integer(int32)	Size of the progeny array minus one
numElasticEvents	Public	integer(int32)	Number of elastic events that occurred prior to the occurence of an inelastic event
modelUsage	Public	GSMModelUsage	Provides information on the sub-model usage within GSM
restarts	Public	eventRestarts	Specifies the number of times the event was restarted due to various detected errors
simState	Public	integer(int32)	Flags the end-state of the simulation
constructed	Private	logical	Flags whether or not the GSM object was constructed

```
subroutine constructGSMResults( bankSize )

   use gsmClass, only:  &
        & GSMProgeny,   &   ! Progeny tracked in GSM simulation
        & GSMResults,   &   ! GSM class/object type
    & newGSMResults     ! GSM constructor interface

   implicit none
   integer(int32), intent(in   ) :: bankSize

   ! Declare progeny array:
   type(GSMProgeny), dimension( bankSize ) :: progenyBank


   ! Declare results object and construct it:
   type(GSMResults) :: results
   results = newGSMResults( progenyBank )

   return
end subroutine constructGSMResults
```

FIGURE IV.3. Construction of the "GSMResults" object.

The GSM driver instead utilizes the `GSMOutput` object for its simulations. The driver for GSM provides a `GSMProjectile`, `GSMTarget`, and `GSMOutput` object as arguments to GSM to generate output files for its users. Table IV.IV details the members of the `GSMOutput` object. GSM uses the `GSMOutput` object to determine the type of information contained within the generated output file. GSM internally creates a `GSMResults` object to simulate each event, tallying the simulation data at the occurrence of inelastic events. GSM utilizes the `GSMOutput` object primarily during the generation of the output file, with minor utilization during the event simulation phase of the GSM output file generating procedure.

TABLE IV.IV: Variable Members of the "GSMOutput" Object

Member Name	Scope	Data Type	Default	Description
outputFile	Public	character(128)	gsm.out	Output file containing simulation results
auxillaryFile	Public	character(128)	gsm.aux	Auxiliary file containing simulation warnings and errors
numInelasticEvents	Public	integer(int64)	100,000	Number of inelastic events to simulate
dkinEnergy	Public	real(real64)	−1.0	Step size in projectile kinetic

Table IV.IV, continued

Member Name	Scope	Data Type	Default	Description
				energy, in MeV, for calculating several incident energies within a single simulation
maxKinEnergy	Public	real(real64)	−1.0	Maximum projectile kinetic, energy in MeV, for calculating several incident energies within a single simulation
angularSpectra	Public	integer(int32)	0	Flags to calculate and print energy-integrated angular spectra of ejectiles, $d\sigma/d\Omega$ [mb/sr]
deltaTheta	Public	real(real64)	5.0	Step size, in degrees, in the ejectiles' angular spectra ($d\sigma/d\Omega$ [mb/sr])
energySpectra	Public	integer(int32)	0	Flags to calculate and print angle-integrated energy spectra of ejectiles, $d\sigma/dE$ [mb/MeV]
doubleDiffSpectra	Public	integer(int32)	0	Flags to calculate and print double differential spectra, $d\sigma/(d\Omega \cdot dE)$ [mb/(sr · MeV)], of ejectiles
spectraEjectiles	Public	integer(int32)	1	Flags to include the ejectile type in calculations of ejectile spectra (array of nine)
angleBins	Public	Bin	*N/A*	Angle bin for calculating double differential spectra of ejectiles (array of ten)
energyBins	Public	Bin	*N/A*	Energy bin for calculating double differential and angle integrated energy spectra of ejectiles (array of four)
energyBinSubStep	Public	real(real64)	1.0	Sub-step within energy bin (see "energyBins") i (array of four)
multiplicities	Public	integer(int32)	0	Flags to calculate and print secondary particle multiplicities, yields [mb], and mean kinetic energies [MeV]
channelCrossSection	Public	integer(int32)	0	Flags to calculate and print cross sections of 192 possible reaction channels contributing to final isotope production
nuclideCrossSection	Public	integer(int32)	0	Flags to calculate and print nuclide cross sections [mb], or yields [mb]

Table IV.IV, continued

Member Name	Scope	Data Type	Default	Description
printPISA	Public	integer(int32)	0	Flags to calculate and print double differential [mb/sr/MeV], angular [mb/sr], and energy [mb/MeV] spectra of ejectiles ranging from neutrons, *etc.*, up to ^{28}Mg
pisaBins	Public	Bin	*N/A*	Angle bins for calculating double differential, angular, and energy spectra of ejectiles (array of ten)
pisaDTheta	Public	real(real64)	5.0	$\Delta\theta$ values for the associated angular bins (see "pisaBins")
printHist	Public	integer(int32)	0	Flags to calculate and print particle histograms
comments	Public	character(128)	*N/A*	Simulation comments to be printed on the output file (array of ten)

Table IV.IV illustrates the default values utilized by GSM for its `GSMOutput` object. The default values are chosen such that no, or minimal, information is printed to the output file. GSM will validate all options held within the `GSMOutput` object to ensure no errors occur. GSM will approximate out-of-bounds values with the nearest valid value or by utilizing the default option for the argument.

IV.4 THE GSM DRIVER

The GSM driver is provided within the GSM package as an example interface to end-users. Clients do not need to compile GSM with the provided driver, however doing so may provide clients with an end-user interface that they may utilize to simulate high-energy spallation physics from GSM directly. GSM was migrated to a mostly object-oriented architecture using `gfortran`, version 6.3.0. GSM compiles well using most versions of `gfortran`, however has not been tested when utilizing the `Intel` Fortran (`ifort`) compilers. Fig. IV.4 details how one can build GSM from source using a `Unix` or `OSX`-type operating system. GSM utilizes a very basic `CMake` build system to allow users to build specific sub-libraries of GSM, such as its standard DCM or

evaporation models. GSM requires an out-of-source build, meaning that those compiling GSM must compile it outside of all source directories of GSM, helping to maintain a clean source code directory. Instead, those compiling GSM are suggested to create a `build` sub-directory, or other named, from the top-level directory of GSM. Those compiling GSM can then utilize `CMake` from within the `build` directory to compile GSM. The compiled GSM executable is located, from inside the `build` directory, within the `bin` directory. Libraries generated during the compilation are stored in the `lib` directory, and all module files are contained within the `modules` directory. The location of all files may be modified from within the `CMakeLists.txt` file in the top-level GSM directory. The generated executable, at the end of program compilation, is copied to the top-level GSM source directory within the `my_GSM` directory. All required data files are copied to the `my_GSM` directory for end-user ease to run simulations in.

```
mkdir ./build
cd ./build

# On Linux/macOS:
cmake ../
make

# On Windows, for example, with GNU and Unix Makefile generators:
cmake -D CMAKE_Fortran_COMPILER=gfortran -G "Unix Makefiles" ../
```

FIGURE IV.4. A simple build of GSM from the command prompt located in the top-level GSM directory.

A minimal regression suite is provided to help aid end-users in determining simulation differences that may occur across different builds of GSM. The provided regression suite utilizes a simple `shell` script to perform all simulations and utilizes a `Python3` script to determine differences in the resulting output files. The provided `Python` script also averages computational efficiency differences. Fig. IV.5 provides a sample of how to run the provided regression tests. The regression suite is structured with a `bin` directory, containing the various shell scripts and `Python` scripts utilized, a `resultsPrevious` directory containing the output file and a file containing messages produced by GSM, a `resultsCurrent` directory containing the same information as in the

`resultsPrevious` directory in addition to all `*.aux` files associated with the regression simulations, and a `differences` directory, containing text files, `*.txt`, that detail the differences from the provided and end-user generated regression simulations. End-users and developers may update the provided regression suite's results by utilizing another simple shell script that moves all of the currently saved results to those being compared against. Updating the regression suite is demonstrated in Fig. IV.5.

```
# Perform regression tests:
sh -e ./test/regression/bin/regression.sh

# Update regression files:
sh -e ./test/regression/bin/updateOutputs.sh
```

FIGURE IV.5. A simple example of regression testing from a Unix command prompt.

GSM requires an input file for its simulations that simulation information is to be read from. The GSM input file is similar to that specified within the CEM manual[1], with four additions to the input file. Within the GSM input file, end-users may utilize a ZAID[2] for incident light- and heavy-ions, where a zero-valued quantum decay number is utilized. End-users may also specify a non-zero quantum decay number with the incident particle by utilizing the projectile card, line three of the input file, as `Z, A, S`, where the `Z` and `A` are provided as the atomic and mass numbers of the nucleus, and the `S` represents the desired quantum decay number. A `npreqtyp` card is added proceeding the `nevtype` card described in the CEM manual. The `npreqtyp` card specifies the number of particles to be considered for emission within the preequilibrium sub-model of GSM. Proceeding the added `npreqtyp` card, a `pisa` card is added to allow end-users to have double differential [mb/sr/MeV], energy [mb/MeV], and angle [mb/sr] spectra of up to 66 ejectiles, ranging from neutrons, protons, *etc.* up to ^{28}Mg. End-users supply an extra card immediately proceeding the `pisa` card containing ten angle bins, and a single angle bin width at the card's end, to utilize for each angle bin when the `pisa` card specifies to calculate the ejectile spectra. Proceeding the `pisa`

[1]The CEM manual is provided by the Los Alamos National Laboratory as LA-UR-12-01364.
[2]The ZAID has the form $ZAID = 1000 \times Z + A$.

card and any associated angle bins is a `hist` card, flagging GSM to calculate and print histogram information of residual nuclei created during the simulation.

The provided GSM driver is contained within the `standaloneGSM` module provided in the `src/DriverGSM` directory of GSM. The `gsmDriver` procedure contained within the driver is utilized by the main program to determine command line arguments, read the input file, and run the GSM simulation to generate an output file. The driver establishes the `GSMOutput` object, as specified by Table IV.IV, by reading the user-provided input file in the `readInput` procedure. Users and developers may provide their own input file reading procedure[3] for the GSM driver to specify sub-model and GSM physics options and behaviors, allowing great flexibility in simulations by the user. It is recommended that the input file be updated to include these options prior to licensing GSM for public use. The default procedure utilized to read the input file, `readInput`, first initializes the necessary random number generator and all GSM data. The procedure then reads in the input file provided by the user, if it exists, and then constructs the GSM object, fills in the appropriate fields for the `GSMOutput` object, and then performs the GSM simulation and output file generation.

Users may determine the available command line options by providing the `-help` flag on the command line, as demonstrated in Fig. IV.6. The driver will, by default, attempt to read an input file named `gsm1.inp`. Users may supply GSM with the input file name by utilizing the command line arguments as needed. Fig. IV.6 details a GSM simulation from a command prompt when utilizing all command line arguments. End-users may determine the available command line arguments as shown in Fig. IV.6. End-users specify the input file to be read by providing the `-i=<input file>` flag on the command line. Note that a path name to the file may be provided, as `-i=path/to/file`, and spaces may be present in the path and input file name. The path and input file must be enclosed in quotations when spaces are present. Users may also set the simulation verbosity by utilizing the `-v=X` command line flag, where `X` is replaced by any integer or real number. GSM will attempt to convert the provided verbose flag to an integer value. GSM

[3]The provided procedure may be utilized by *include*ing the file in the *standaloneGSM* module and by replacing the call to the default *readInput* procedure with a call to their own input reading procedure, as desired.

will utilize for its arguments only the last valid argument specified. In other words, users may specify the input file to use multiple times, however only the last specified input file will be used. Regarding the simulation verbosity, users may specify the verbosity many times, however only the last valid verbosity detected will be used for the simulation. Note also that the CEM and LAQGSM event generators were also modified to include usage of input file specification on the command line. Mildly different to that of GSM, the modified versions of CEM and LAQGSM utilize the last provided command line argument as that simulation's input file.

```
# Build GSM
mkdir ./build && cd ./build
cmake ../ && make

# Move to the simulation directory:
cd ../my_GSM

# See what arguments are available:
./xgsm1 -help

# Perform a simulation:
./xgsm1 -v=2 -i=myInputFile.inp
```

FIGURE IV.6. A sample simulation of GSM utilizing all available command line arguments.

IV.5 THE GSM API

The GSM application programming interface (API) is necessary for clients to utilize the features provided by GSM. GSM was designed such that its API is easy and clear to use. Much of the GSM API provides robust protection, allowing clients to utilize GSM in a minimal way without failure. The GSM API consists of a data initialization procedure, an object constructor, and various procedures to allow clients to simulate high-energy spallation physics and to determine the state of the GSM object. The current architecture of GSM allows clients to easily query results, providing all information tracked by GSM, allowing clients to "pick and choose" the desired results from a single simulation.

Clients initialize the GSM model data as described in Sec. IV.1, where Fig. IV.1 demonstrates the model data's initialization using default and all optional arguments. Clients may, at their discretion, omit various arguments if no control over a variable is desired. The file unit numbers under which the sub-libraries will open the files is used simply as a starting point for the unit in the event the file unit is already in use. The GSM sub-libraries, except for the modified DCM library, will increment the unit number, up to 50 times, to find an available unit to ensure that data read in for its simulation is not skewed. The limits provided by the GSM sub-libraries when attempting to find available file units is utilized only to prevent potential infinite loops.

Clients may construct the GSM object upon its declaration within the program. The GSM construct, described in more details in Sec. IV.2, was designed to be versatile and easy for clients to use. The GSM constructor utilizes only optional arguments, allowing clients of GSM to simply use the high-energy spallation physics object without modification and without needing to develop a deep understanding of the various options it utilizes. Clients may also provide strict control over GSM by providing it with their own RNG function pointer, physics options and behaviors, and they may also control how all messages generated by the GSM object are handled. Fig. IV.2 demonstrates a simple construction of GSM without usage of any arguments as well as a GSM construction utilizing strict control over the object.

Clients may in this manner determine many different aspects of the various physics contained within GSM and its sub-libraries. For example, clients may choose to omit the coalescence of alpha particles by setting the coalescence radius for alpha particles to zero. Clients may also easily determine the number of fragments to be considered for emission for both the preequilibrium and evaporation sub-libraries explicitly, increase the emission width utilized when determining preequilibrium particle emission, *etc.*, as shown in Fig. IV.7. Fig. IV.7 demonstrates also the modification of Fermi break-up nucleus size, where residual and compound nuclei containing less than or equal to 16 nucleons, controlled by `numRecNucleons`, are utilized for Fermi break-up physics within the GSM, preequilibrium, and evaporation libraries. The `gsmOptions` object provides clients with great control of the physics within GSM with each GSM object being utilized.

```
subroutine constructGSM( thisRNG, thisIO )

   use gsmClass, only: &
        & GSM,          & ! GSM class/object type
        & newGSM,       & ! GSM constructor interface
        & gsmOptions      ! Options for the GSM simulation

   implicit none
   procedure(RANDOM),     intent(in   ), pointer :: thisRNG
   procedure(IOHANDLER), intent(in   ), pointer :: thisIO

   ! Declare physics options object:
   type(gsmOptions) :: theseOptions

   ! Modify some physics options from library defaults:
   theseOptions%coalesOpts%coalesRadiiAlpha = 0.0   ! Omit alpha clustering
   theseOptions%preeqOpts%numPreeqType  = 45     ! Consider emission of 45
      particles in the preeq. sub-library
   theseOptions%evapOpts%numEvapType    = 36     ! Consider emission of 36
      particle in the evap. sub-library
   theseOptions%preeqOpts%emissionWidth = 0.60   ! Use larger emission width
       for emission probability calculation
   theseOptions%fbuOpts%recNumNucleons  = 16     ! Increase max nucleus size
       considered for Fermi Break-up physics

   ! Declare a GSM object:
   type(GSM) :: gsmObj

   ! Construction of GSM with all optional arguments:
   gsmObj = newGSM(                          &
        & clientRNG     = thisRNG,           &
        & clientOptions = theseGSMOptions, &
        & clientIO      = thisIO             &
        & )

   ! Update client's options object to reflect those used:
   theseGSMOptions = gsmObj%queryOptions()

   return
end subroutine constructGSM
```

FIGURE IV.7. A sample GSM class construction interface.

GSM contains within itself many procedures clients may utilize to query the state of the GSM object as well as to simulate the desired physics by the client. Table IV.V lists and describes the procedures available for client use. GSM provides, outside its constructor, methods to allow clients to set and modify the state of the GSM object as desired. Additionally, GSM allows its clients to simulate its sub-model physics, such as only the standard DCM, Fermi break-up, evaporation, *etc.*, for individual testing or for utilizing the sub-models of GSM for its own physics simulations. The available procedures allow clients to simulate a single event as desired. Clients may also generate an output file using the same GSM object. GSM does utilize some of the same procedure names for simulating a single event or generating an output file. GSM utilizes a generic procedure for this, where the computations performed by GSM are dictated by the arguments provided by the clients, namely the `GSMProjectile`, `GSMTarget`, and either the `GSMResults` or the `GSMOutput` objects for simulating a single event or generating an output file, respectively. GSM clients may also utilize some of the other available procedures in an attempt to establish the projectile and target objects utilized and to know their states prior to the simulation, and also to determine what INC model GSM would utilize for a single event's simulation. Sec. D.4 details the arguments for each of the unique procedures and some of the procedure's use cases.

TABLE IV.V: A List of GSM Procedures Available to Clients

Procedure Name	Type	Description
For Establishing the GSM State (without the Constructor):		
updateOptions	Subroutine	Sets the *gsmOptions* type used by GSM for its simulations
updateRNG	Subroutine	Sets the RNG procedure pointer utilized by GSM
updateMessageHandler	Subroutine	Sets the message handling procedure utilized by GSM when messages are generated
For Querying the GSM State:		
properlyConstructed	Function	Returns logical flag for GSM construction state
queryOptions	Function	Returns the `GSMOptions` object utilized by GSM for its simulations
queryRNG	Function	Returns the RNG procedure pointer used by GSM
queryPhotoEmissionUse	Function	Returns a logical flag for whether or not photon emission is used in GSM
For Simulating A Single Event:		
simulateEvent	Subroutine	Simulates a single spallation event
performSimulation	Subroutine	See the *simulateEvent* procedure for details
collide	Subroutine	See the *simulateEvent* procedure for details
interact	Subroutine	See the *simulateEvent* procedure for details
simulate	Subroutine	See the *simulateEvent* procedure for details
execute	Subroutine	See the *simulateEvent* procedure for details
start	Subroutine	See the *simulateEvent* procedure for details
For Generating an Output File:		
generateOutput	Subroutine	Generates an output file
output	Subroutine	See the *generateOutput* procedure for details
simulate	Subroutine	See the *generateOutput* procedure for details
execute	Subroutine	See the *generateOutput* procedure for details
start	Subroutine	See the *generateOutput* procedure for details
Physics Interfaces:		
standardDCMInterface	Subroutine	Interface to the standard DCM sub-library
setupMDCM	Subroutine	Establishes the modified DCM nuclei data for a given simulation
modifiedDCMInterface	Subroutine	Interface to the modified DCM sub-library
coalescenceInterface	Subroutine	Interface to the coalescence sub-library
fermiBreakUpInterface	Subroutine	Interface to the Fermi break-up sub-library
preequilibriumInterface	Subroutine	Interface to the preequilibrium sub-library
evaporationInterface	Subroutine	Interface to the evaporation sub-library
simulateDecay	Subroutine	Interfaces to a residual's preequilibrium and evaporation decay scheme
Other Available Procedures:		
formNuclei	Subroutine	Establishes all default fields for a given

Table IV.V, continued

Procedure Name	Type	Description
		projectile and target nucleus
buildNuclei	Subroutine	See the *formNuclei* procedure for details
inquireINCModel	Function	Determines the INC model to be used given a projectile and target
sampleEnergy	Function	Returns an energy directly sampled from a Gaussian distribution around a provided mean and standard deviation, when GSM uses continuous transitions around the INC transition energy
determineRestMass	Function	Returns the GSM approximation for a nucleus's rest mass [GeV/c^2]

Chapter V. Verification of GSM

Verification of the GSM concerns primarily code metric analyses, standards compliance verification, and ensuring the models' code is consistent and complete compared to its predecessors, the CEM and LAQGSM. Note that verification of GSM, as defined by the ASME, is not explicitly explored here. GSM is a coupling of the CEM and LAQGSM event generators in that the INC model of LAQGSM, the modified DCM, is made available to the CEM physics modules and incorporated to expand the capabilities of the CEM event generator, now GSM. GSM "resets" the `GSMResults` object utilized for a simulated spallation event, performs the INC simulation for the fast stage of the reaction according to the projectile size and energy, verifies momentum is relatively conserved, then performs the intermediate and slow simulation of the residual nuclei according to those created during the INC process. Some modification of the coded physics models has been made to accommodate the additional de-excitation and decay simulation of the projectile residual when the modified DCM INC model is used.

The sub-models utilized by GSM have been mostly migrated to an object-oriented Fortran code, where integral regression testing has been conducted to note numerical differences in simulation results during their modernization. The sub-model physics has remained consistent with its usage in the CEM event generator, being previously validated and verified, where some modifications have been made to resolve coding errors ("bugs") and being implemented in a modern architecture. Each sub-model of GSM, including GSM itself, has been developed with a robust API. The modified DCM sub-model, being the only non-object oriented model within GSM, now utilizes a minimal API using a modified DCM-specific common block[1] and the global `MDCMResults` object within the `modifiedDCMClass` module, where modifications have been made internal to the modified DCM for computational efficiency, protection against invalid progeny and nuclei, and some minor

[1]The modified DCM common block utilized by GSM in its modified DCM interface is the "resultLAQ" block, containing information about the residual nuclei remaining after the modified DCM simulation.

migration to an object-oriented architecture. Both of GSM's predecessors, CEM and LAQGSM, have been extensively verified throughout the years based on fundamental physics by several scientists, namely S. Mashnik, K. Gudima, and A. Sierk, being developed and utilized in many particle transport codes[2] since their creation in the early 1990s.

The inclusion of the modified DCM sub-model to CEM to form GSM, alongside the object-oriented migration, suggests that the GSM physics sub-models simulate their respective physics in a similar method to those of the CEM and LAQGSM event generators, in addition to providing their clients with similar numerical results for a given input. GSM has been found to provide equivalent results for its simulations when utilizing the standard DCM prior to the GSM object-oriented code migration [98]. GSM and it sub-models now provide different numerical results for their simulations as a result of utilizing the object-oriented codes, where coding errors have been resolved and improvements to the underlying physics models have been made. GSM incorporates the object-oriented models within itself well, correctly and sufficiently, according to its previous usage and implementation in the CEM and LAQGSM event generators. Results have been verified for consistency when utilizing various GNU Fortran compiler versions, namely *GCC 4.7.3* up to *GCC 6.3.0*, when utilizing no compiler optimization. Several compiler flags have also been utilized for the object-oriented migration to detect where potential errors may occur, flagging most all warnings, enforcing no implicit declarations except for those in the modified DCM code, and ensuring array bounds are not exceeded during simulations.

The object-oriented GSM is more maintainable than its predecessors, the CEM and LAQGSM event generators. The architecture of the GSM event generator is such that GSM further reduces the logical complexity of using two separate, but related, models for spallation simulations, and in addition reduces the complexity within each of the sub-models, where clear and robust interfaces exist between models. Inclusion of a verbose filter additionally allows users and developers to determine when and where errors may be originating from. Developers may isolate and improve these discovered errors more easily in this sense. The highly modular nature of the object-oriented

[2]Some particle transport codes utilizing CEM, LAQGSM, or both include several releases of MCNP, MARS, MCATK, and others.

GSM code additionally requires less time for developers when providing improvements or patches to the code than previously, when dependencies within the code were unknown and may have gone undetected. The overall effective length of the code is less as a result of utilizing one, instead of two, event generators as well, thus reducing the time required for its further development.

The reduction in the number of sub-models used in the GSM allows it to be $\sim 5\%$ smaller in size than the CEM and LAQGSM event generators when combined. The reduction allowed by the use of the GSM equates to roughly $\sim 3\,000$ lines of legacy Fortran code, where much of the modernized code comprises its new and advanced architecture[3]. Note that, due to the GSM modernization, the code now utilizes several structural layers for each of its sub-models, requiring some length of code that is used simply as declaring a data structure within the sub-model, providing ease and clarity during development. Although the overall code has not been significantly reduced, it is more maintainable and flexible. The individual routines used within GSM, particularly for the code transferred from the LAQGSM event generator, have also been split into multiple files with significant addition of comments and argument declaration. The process of splitting files into smaller, more manageable pieces is accounted for in the module by utilizing the `include` statement, where files are included in the sub-models' module, aiding future developers by describing the physics of the simulation more clearly, logically, and coherently. Developers more quickly understand the process modeled within a particular procedure as a result of shorter files, and in addition provides more modularity within each sub-model. The modified DCM code, for example, was and still is primarily contained within two files, being near 12 000 lines of code or more, containing minimal comments. All primary procedures utilized by the modified DCM were parsed from these two files, among others of importance, to aid in simplifying the overall code and its potential modernization. The reduction in length of a given file boosts the maintainability of the GSM code.

The complexity of the physics within the GSM matches that of the pre-existing CEM and LAQGSM codes. The modernization of the GSM code has reduced the complexity of the underlying logic within GSM and its comprehensibility. GSM now has clearly defined dependencies and

[3]Prior to modernization, the GSM code was $\sim 10\,000$ lines less. The new architecture of the GSM code thus uses nearly 7 000 lines of code.

interfaces, where developers may quickly understand the sub-models' code and where clients may easily, simply, and robustly `use` GSM or one of its sub-models for their simulations. The logic internally utilized by GSM and its sub-models is now more clear between each of the sub-models, and where tracking of important interim and standing results is clear within each sub-model. Many comments were added to the routines of GSM to help improve the readability of the code when modified by future developers regarding this issue. Heavily used variable names were refactored for improved readability. GSM utilizes a directory structure now where each sub-model procedure is clearly contained within that sub-model's directory, highly simplifying the further development of each of the sub-models.

GSM in addition is more flexible than either CEM or LAQGSM. This is due to the nature of the physics coupling and additional modifications made. The physics simulated within both CEM and LAQGSM are tightly coupled, however are now loosely coupled within GSM. Additional sub-models may be inserted, improved, inserted, or any combination thereof, within GSM to provide improvements to its existing simulations.

For example, a well developed Coulomb barrier module exists and is utilized by the preequilibrium sub-model, however is not utilized by the other GSM sub-models. Many of the GSM sub-models utilize a unique form of a Coulomb barrier calculation within themselves. Incorporation of the improved Coulomb barrier module to all of the GSM sub-models will aid in flexibility, providing GSM with a robust Coulomb barrier calculational mechanism. GSM and each of its sub-models now utilize a data type to control all model options, allowing clients to easily modify GSM and its sub-models behavior and usage, whereby clients may effectively use GSM without requiring the client to know how the models' behavior is propagated through the model. GSM offers additional information to developers and users via its terminal output and verbosity filter, informing them of assumptions or misgivings of a simulation, in addition to both mathematical and programmatic errors, aiding in the use and further development of GSM.

Logical decoupling of the code in the modified DCM physics model was also explored during GSM's development, however has not been verified at this time. Preliminary development of an

object-oriented modified DCM took place, where a skeleton parameter, data, and class module exist for the model. The logical decoupling and modernization of this code has the ability to make the modified DCM, and thus GSM, highly flexible, and in addition will make the modified DCM code significantly more human readable and less error prone. Error resolution methods were placed into GSM and its sub-models such that, prior to any simulation, GSM will verify the physicality of the provided arguments as well as ensuring the GSM object is sufficiently established to perform the task being requested by the client.

GSM and its sub-models warn clients when provided arguments are invalid. In some instances, GSM and its sub-models will provided approximations to desired information when the provided arguments exist outside of the models' scope. For example, the standard DCM target data object provides approximations for the target when the target is deemed invalid within the standard DCM, having no baryons or protons in the target. Default values are utilized in the event where approximations cannot be made, such as when specifying the incident projectile and target objects in GSM.

GSM and its sub-models may also be built as needed by the client. Each of the GSM libraries[4] may all be build individually as a result of utilizing a basic `CMake` build system. Clients may desire to only utilize the coalescence sub-library, and thus only need to `make gsm_coalescence`, for example. Previously, clients of GSM, its sub-libraries, or both, could not individually build the desired library and utilize only the desired library. Clients were instead required to utilize all of the GSM, loading significantly more data than required. Well developed interfaces to each of the GSM sub-models exist as well, providing clients of GSM, its sub-libraries, or both, great flexibility in their usage. GSM also allows its clients to access the sub-library interfaces it utilizes for added flexibility and testing.

Coding standards were employed during the development of GSM. Where possible, no new *common blocks* were created in GSM when coupling the non-standard DCM model with CEM, and

[4]The possible build targets include: *gsmMain*, *gsmOther*, *gsm_standardDCM*, *gsm_modifiedDCM*, *gsm_coalescence*, *gsm_fermiBreakUp*, *gsm_preequilibrium*, and *gsm_evaporation*. Running *make help* will show the all available build targets.

instead modules were used according to the Fortran2003 and Fortran2008 standards. General, and very basic, guidelines for "good" programming have also been employed in most all additions of code to GSM. The modified DCM, and parts of GSM, should have all global memory removed in their future modifications. Most of the global memory utilized by the modified DCM and GSM is contained by common blocks, being a heavily used Fortran66 and Fortran77 feature in most scientific codes. Several embedded objects are recommended for use within these models, such as a `mDCMResults` and `mDCMCalculation` object, a `gsmOutputCalc` object, *etc.*, to aid in the scalability of the model and increase the robustness of its API.

GSM appears to well satisfy the complexity for spallation physics calculations. The functionalities of GSM are the same as both CEM and LAQGSM, but are held within a more robust and comprehensive programmatic framework. The physics models within GSM check for conservation of several tracked particle characteristics, and correct or eliminate data as necessary if the various applicable conservation laws are not met. GSM, as will be seen in the proceeding sections, also well reproduces simulated data, agreeable to that of experiment. This is anticipated considering the extensive validation efforts that have been made on the CEM and LAQGSM event generators that GSM is based on. Internal code reviews informally occurred throughout the GSM development process.

GSM provides slight approximations where available, as briefly mentioned earlier. GSM and its sub-models provide for their clients safety in these assumptions by not attempting to access memory that does not exist. Note that GSM does not perform these checks with each instance of data access, however does so in libraries where the data is more likely to be exceeded as a result of client or end-user error, such as in the Fermi break-up library. Providing approximate values in this way allows the specific simulation to continue and provides clients with an approximate result, where isotopic invariance is inherently assumed.

GSM utilizes the robust API of its sub-models to obtain results tracked by the sub-model. Note that GSM and its sub-models are all developed in a similar manner, containing architectures and structures that closely resemble each other while providing clients and end-users with different,

complex algorithms for their simulations. The modified DCM, being only modernized to a minimal level, utilizes data structures for its progeny array and contains the infrastructure for residual tracking. All of GSM's sub-models, excluding the modified DCM sub-model, also utilize a `simState` flag in their results objects, where clients may easily determine the end-state of the simulation for that models' physics procedures. The provided end-state flags aid clients in determining the occurrence of errors, warnings, approximations, and when the simulation ended with no complications or unexpected statements.

A list of discovered and known coding errors, or bugs, is listed in Appendix A. Appendix A lists both coding errors that have been resolved and unresolved, the estimated difficulty of the resolution, and other addition related to the coding error. Note that, due to the complex physics of the simulations, the code contained by GSM is prone to containing coding errors. The list of coding errors provided in Appendix A is thought to be an incomplete list of all coding errors in GSM, where other coding errors are likely to exist.

External code reviews of GSM are recommended for its further verification, in addition to further simplification and decoupling of the modified DCM code. Note many procedures exist within the modified DCM that are paralleled by the standard DCM with the possibility of slight modification. It is recommended to further study these procedures and simplify the overall code architecture of the modified DCM, in addition to GSM, where possible. Note also minor discrepancies within GSM were detected, when using the modified DCM, with that of the LAQGSM event generator from a code profiling perspective. Many procedures utilized equal timings, however some large computation time differences were noted. It is recommended that, during the future modernization of the modified DCM these procedures be observed more closely after the model's modernization. Few conclusions regarding the discrepancies can be made at this time. Continued verification of GSM is recommended. It is clear that GSM is well verified at this time apart from some very specific areas of interest within the GSM code.

Chapter VI. Validation of GSM

The validation of the GSM presented here is defined by unit testing of the GSM via a black-box methodology, *i.e.* without modification of any internal GSM variables or consideration of the internal workings of GSM, and via regression testing to compare the GSM against its predecessors, the CEM and LAQGSM event generators. Much of the analysis presented here considers both regression and unit testing of the GSM simultaneously. The amount and diversity of simulated interactions is assumed to reach all relevant physics models available to GSM. Note that GSM's predecessors, CEM and LAQGSM, have been extensively validated, most recently in Ref. [58, 99–101]. GSM, as stated in previous sections, uses the same underlying physics models and codes that are used in both CEM and LAQGSM, using that of CEM except where the modified DCM is better suited for the INC simulation. GSM is tested against results produced by both CEM and LAQGSM when available for this reason. GSM is primarily validated against experimental data here.

Known deficiencies of GSM, and potential unstable areas of GSM, are listed in Appendix A. Software crash protection has been added to many parts of the modified DCM code, as was done to a large portion of GSM's underlying CEM code base [58]. Said crash protection involves preventing two types of mathematical errors, namely divide-by-zero and square-root protection, where a warning is printed to the user and corrected for in-simulation. It is recommended to streamline the invoked protection within the modified DCM code further to reduce the computation time of the simulation. Data obtained from profiling both GSM and LAQGSM has also shown that GSM replicates the physics used in LAQGSM, a thoroughly validated code [99–101], for most physics models. Observed differences in these simulations are noted primarily in the use of the *repij* procedure of the modified DCM.

GSM has been shown to produce consistent results across a few machines. It is recommended to improve the `CMake` build system utilized by GSM to improve its portability and robustness. GSM

currently has been seen to operate well on Linux systems, with interoperability on macOS assumed. GSM uses the GNU Fortran compiler, *gfortran*, at this time, with no interdependencies on the machine's operating system. Syntactical errors regarding some portions of the GSM sub-libraries have been noted when utilizing the Intel Fortran compiler that are not yet resolved.

VI.1 INTEGRAL REGRESSION TESTING OF GSM

Integral regression testing of the GSM was completed prior to modernization of the GSM. Validation of GSM, after modernization and various bug-fixes, has not been thoroughly completed due to recent issues relating to accessing the software. Validation of the modernized GSM is briefly explored in Sec. VI.2. The validation of GSM, prior to its modernization, is included here for completeness. Note the unit and regression testing presented here primarily focuses on results produced by GSM regarding differential fragment production cross sections. It is of importance to note that estimation of experimental results within an order of magnitude is commonly accepted to accurately represent reality within the high-energy physics community. This is partially due to the little experimental data available and in addition due to the complexity of estimating the various physics over the entire energy regime encountered.

VI.1.1 Proton Induced Reactions

Fig. VI.1 demonstrates that the predicted production cross sections made by GSM are in good agreement with that of experimental data, and duplicate the predictions made by CEM. LAQGSM predictions were produced for the simulation shown in Fig. VI.1, and are shown to be an order of magnitude smaller than experiment. LAQGSM data was observed to have better agreement with that of experiment and the other event generators for ^{6}Li near angles representative of backscatter, however this figure is not shown here. It was observed that all of the event generators were within an order of magnitude at all angles, being the least representative of experiment near glancing angles. The GSM and CEM event generators appear to over estimate experiment for low energy ^{6}Li

fragments, however have an overall good agreement with the experiment. Fig. VI.1 demonstrates that neither GSM or CEM well model the affect the coulomb barrier has on the emittance probability of various fragments.

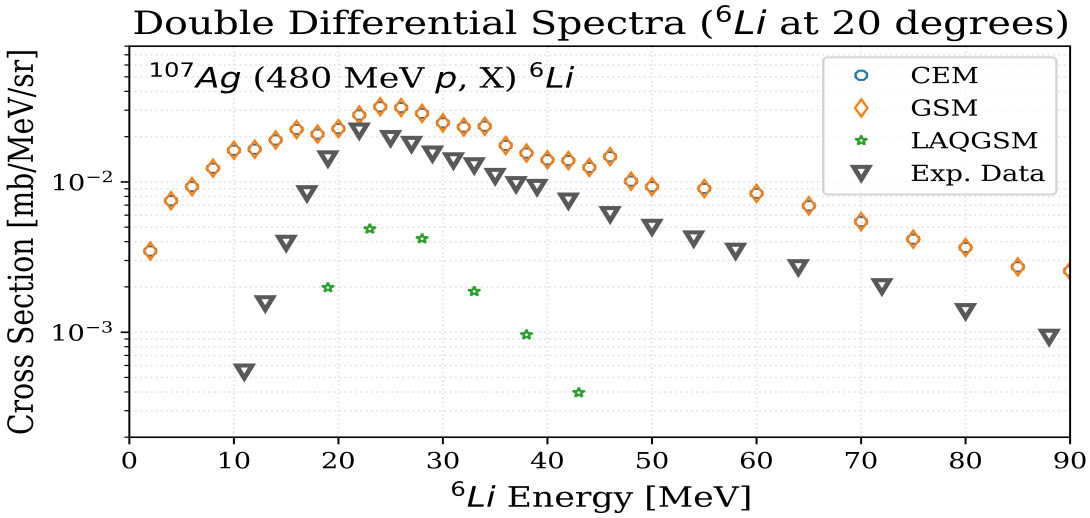

FIGURE VI.1. GSM, CEM, and LAQGSM predictions for the double differential production cross section of ^{6}Li fragments emitted at 20° to a 480 MeV proton beam incident on a ^{107}Ag target. Experimental data was obtained from Ref. [102].

GSM, in perfect agreement with the CEM predictions, appears to well predict experiment for the simulation shown in Fig. VI.2. GSM and CEM well predict the experiment of Fig. VI.2 at other angles near perfectly, however have the worst predictions at glancing angles, as is shown in Fig. VI.2. Note also that Fig. VI.2 suggests both GSM and CEM over estimate transport of the incident proton through the nucleus, meaning the event generators do not correctly model physics within this energy regime, or that the experiment was not approximately representative of a proton incident upon a single layer of ^{232}Th. The thickness of the target used for the experiment represented in Fig. VI.2 was 3 cm thick, violating this assumption and suppressing the transport of the incident proton through the 232Th medium. The event generators do agree well with experiment for most other fragment energies. LAQGSM data was not produced for neutron fragments, however was within an order of magnitude for other fragments observed (not shown here).

Fig. VI.3 demonstrates predictions from both GSM, CEM, and LAQGSM for the ^{232}Th *(*1.2 *GeV*

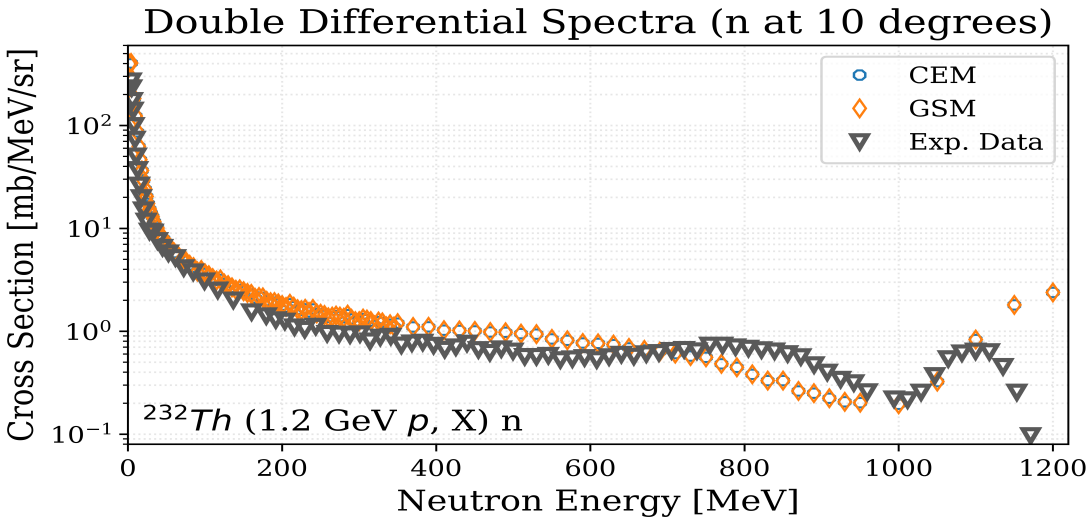

FIGURE VI.2. GSM and CEM predictions for the double differential production cross section of neutron fragments emitted at 10° to a 1.2 GeV proton beam incident on a ^{232}Th target. Experimental data was obtained from Ref. [103].

p, X) reaction. Both GSM and CEM simulations over-predict experimental data for fragments having kinetic energies between ~ 20 to 60 MeV, while LAQGSM under-predicts the data everywhere but within the same energy region. It is concluded then that improvements made to the CEM coalescence and equilibration models, from Ref. [49, 50, 58, 104, 105], have improved the predictions of the GSM and CEM event generators for most all fragment kinetic energies, particularly for the high energy fragments. It is suggested that more events be simulated using the LAQGSM event generator to better determine its accuracy and precision relative to the experimental data shown in Fig. VI.3.

Fig. VI.4 demonstrates the general agreement of data produced by the event generators and LAQGSM. Note that the predictions made by LAQGSM do not predict the data of the experiment in Fig. VI.4 as well as the other event generators, that it tends to diverge from experimental results for higher energy ^{7}Li fragments. This affect was not as prevalent for ejected ^{10}B fragments, not shown here, at varying angles. Fig. VI.4 also demonstrates that GSM and CEM well predict the maximum production cross sections found during experiment.

Fig. VI.5 demonstrates that GSM, like in other figures, produces exactly the results of CEM for reactions involving incident nucleons below ~ 4.5 *GeV*. It is apparent in Fig. VI.5 that both event

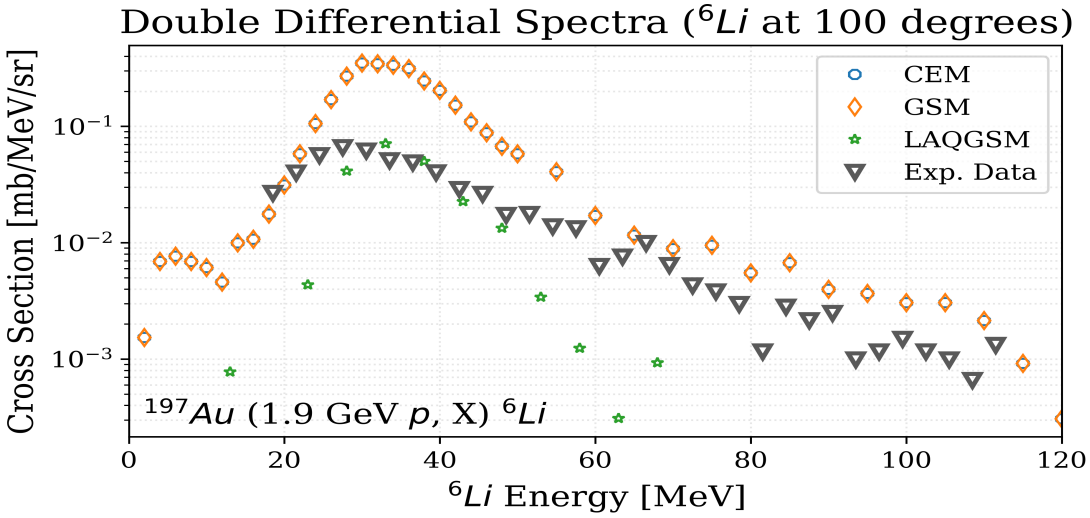

FIGURE VI.3. GSM, CEM, and LAQGSM predictions for the double differential production cross section of ^{6}Li fragments emitted at 100° to a 1.9 GeV proton beam incident on a ^{197}Au target. Experimental data was obtained from Ref. [106].

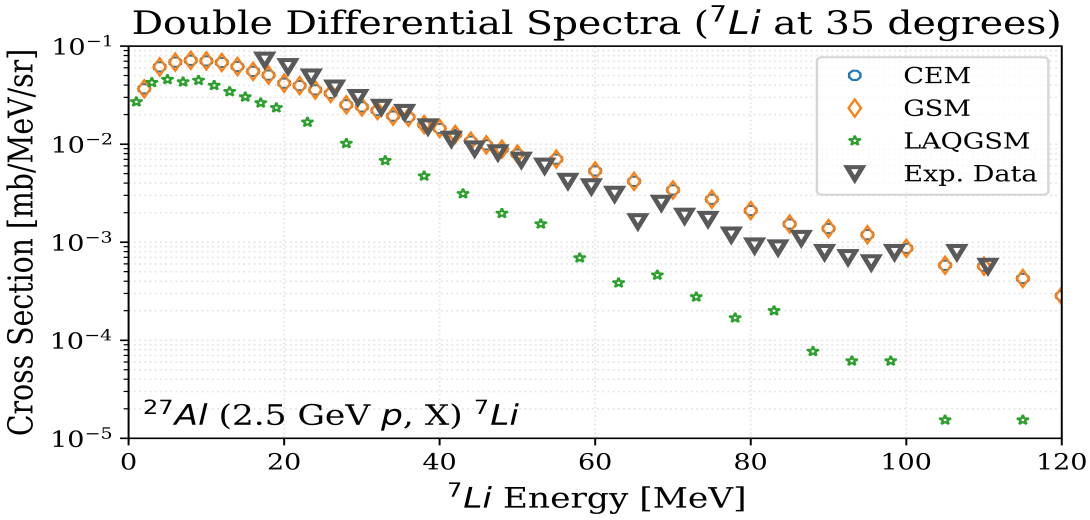

FIGURE VI.4. GSM, CEM, and LAQGSM predictions for the double differential production cross section of ^{7}Li fragments emitted at 35° to a 2.5 GeV proton beam incident on a ^{27}Al target. Experimental data was obtained from Ref. [107].

generators underestimate the experimental data observed by roughly half for most fragment energies. Observance of other fragments yielded that LAQGSM was in agreement with the predictions by both GSM and CEM, giving the conclusion that the LAQGSM data would have similar results to those shown in Fig. VI.5. The underestimation observed questions if the INC and Fermi break-up models used within GSM need further examination, since all fragments and residual nuclei simulated are sufficiently small that the physics modeled in simulation precludes the preequilibrium and evaporation/fission of the residual compound nucleus. More comparisons of this reaction type are recommended to fully determine the necessity of such an investigation considering the fair agreement of the event generator predictions to the data.

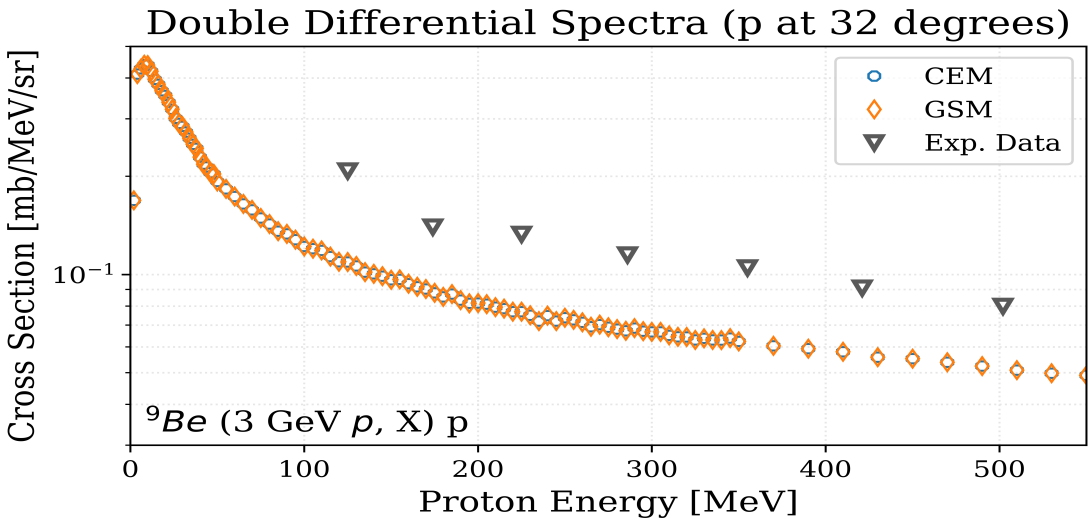

FIGURE VI.5. GSM and CEM predictions for the double differential production cross section of proton fragments emitted at 32° to a 3 GeV proton beam incident on a ^{9}Be target. Experimental data was obtained from Ref. [108].

Fig. VI.6 demonstrates the predictions produced by GSM and CEM compared to experimental results. It is seen in Fig. VI.6 that GSM and CEM do both produce results in good agreement with the experimental data. The results of CEM appear to better predict data at this incident energy than GSM does, mainly for high energy ^{4}He fragments. Note that GSM is using the non-standard DCM for its INC simulation. GSM does so for incident nucleons, photons, and pions with kinetic energies above $\sim 4.5\ GeV$, and also for all light- and heavy-ion induced reactions. It is recommended to

perform a best-fit analysis to determine the event generator that best predicts experimental data for the reaction shown in Fig. VI.6. Note however that GSM can use the standard DCM for its INC simulation with a different cutoff energy.

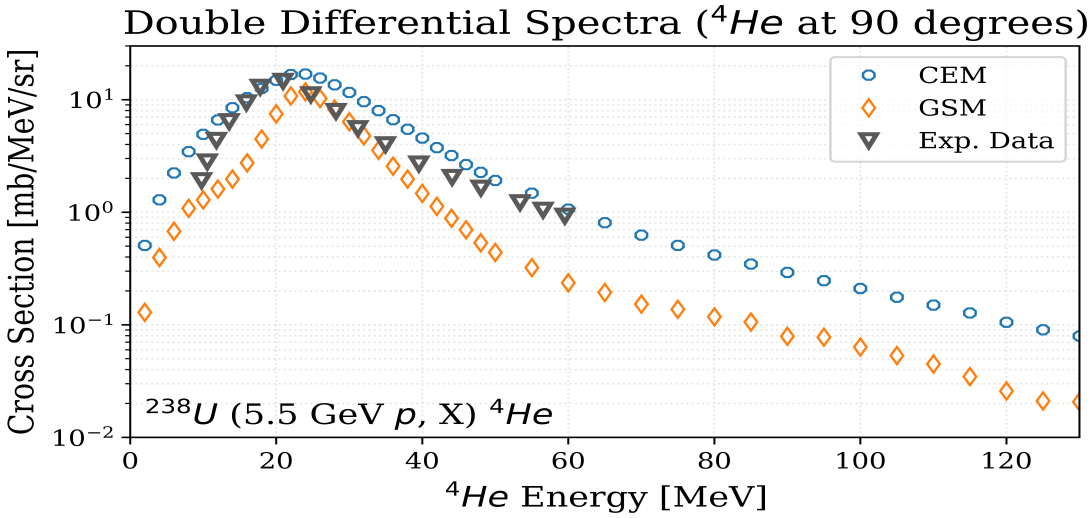

FIGURE VI.6. GSM and CEM predictions for the double differential production cross section of ^{4}He fragments emitted at 90° to a 5.5 GeV proton beam incident on a ^{238}U target. Experimental data was obtained from Ref. [109].

Fig. VI.7 shows the angular dependence of the cross section, where 0 degrees represents a forward peaked scatter, and 180 degrees backscatter of the fragment. The predictions shown in Fig. VI.7 seem to yield the conclusion that GSM better predicts experiment for this reaction, contrary to the conclusion that can be drawn from Fig. VI.6. Note also that LAQGSM seriously underestimates this experiment for this, and later observed, energy integrated spectrum.

Table VI.I shows the required computation times for each proton-induced reaction. GSM required, on average[1], 101.9% of the time required by CEM when using CEM physics. It is suspected the slight increase in computation times is due to the modified handling of errors in GSM, for which refactoring is recommended. GSM required 50.0%, on average[2], and 94.3% of the time required by LAQGSM when using CEM and LAQGSM physics, respectively. A larger sample of

[1]The standard deviation in GSM-CEM computation time ratios was observed to be 1.2% for these reactions when using CEM physics.

[2]The standard deviation in GSM-LAQGSM computation time ratios was observed to be 16.7% (16.6% for CEM-LAQGSM ratios) for these reactions when using CEM physics.

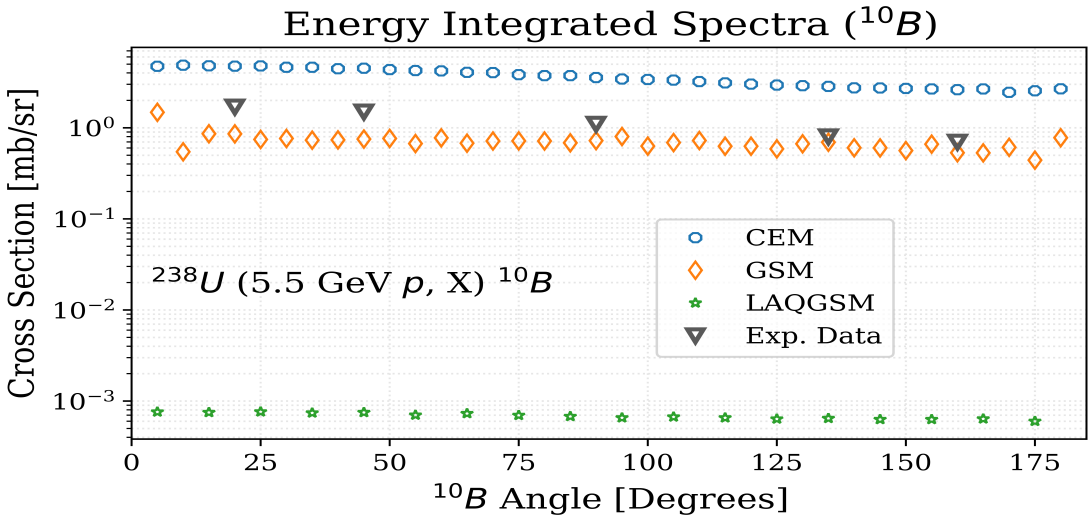

FIGURE VI.7. GSM, CEM, and LAQGSM predictions for the energy integrated production cross section of ^{10}B fragments from a 5.5 GeV proton beam incident on a ^{238}U target. Experimental data was obtained from Ref. [109].

experimental data is sought after for proton-induced reactions above ~ 4.5 *GeV* to provide statistical information on GSM computation times when LAQGSM physics models are used. Note each simulation using CEM, GSM, and LAQGSM simulated the same number of events, 10 million, to provide comparable computation times.

VI.1.2 Neutron Induced Reactions

Fig. VI.8 shows the predictions made by GSM and CEM for the reaction shown. LAQGSM was used to simulate the reaction, however did not produce any predictions for ^{12}C fragments. It seems that both GSM, in exact agreement with CEM, predicts the reaction well compared to approximate experimental data. The experimental data shown in Fig. VI.8 is for ^{12}C fragments in an excited state of 4.44 MeV. The comparison from this is at best approximate, however it seems that GSM data is in fair agreement with this experiment. The excited state of the nucleus likely leads to larger cross sections, combined with the low incident energy of the particle, seem to negate the anticipated overestimation of ^{12}C fragment production, as was seen in other reactions with incident particles having kinetic energies below ~ 4.5 *GeV*.

TABLE VI.I. Computation Times for Proton Induced Reactions (in hours)

Reaction	CEM	GSM	LAQGSM
^{15}N (35.05 *MeV p, X)*	0.05	0.05	0.08
^{19}F (96 *MeV p, X)*	0.57	0.58	1.03
^{107}Ag (480 *MeV p, X)*	1.65	1.67	3.09
^{197}Ni (1.2 *GeV p, X)*	0.57	0.58	0.98
^{232}Th (1.2 *GeV p, X)*	0.63	0.64	1.17
^{63}Cu (1.732 *GeV p, X)*	0.28	0.28	0.46
^{197}Au (1.9 *GeV p, X)*	0.86	0.87	1.41
^{27}Al (2.5 *GeV p, X)*	0.97	0.99	2.80
^{9}Be (3 *GeV p, X)*	0.31	0.32	1.71
^{208}Pb (3.17 *GeV p, X)*	1.28	1.30	2.16
^{12}C (3.17 *GeV p, X)*	0.04	0.04	0.20
^{238}U (5.5 *GeV p, X)*	1.88	3.56	3.78

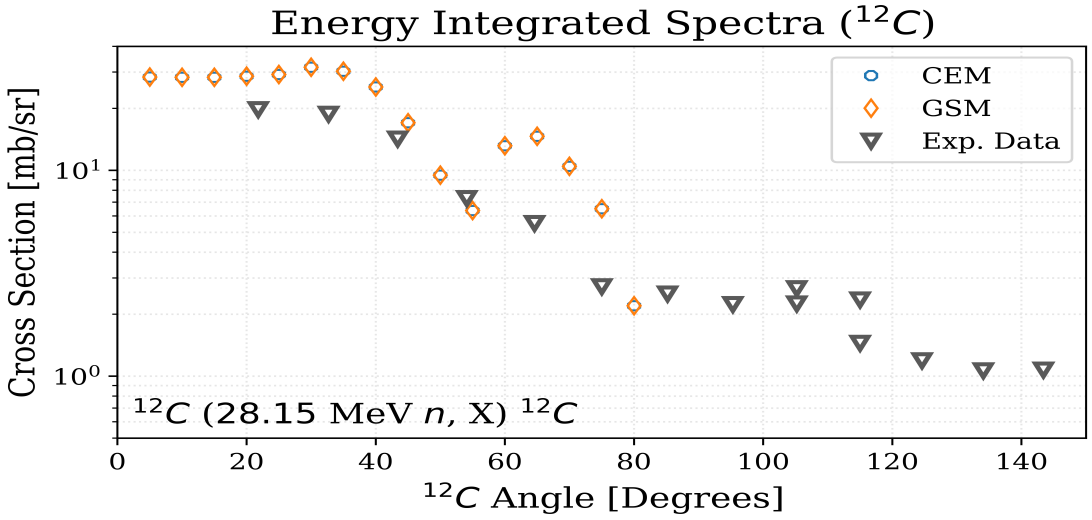

FIGURE VI.8. GSM, CEM, and LAQGSM predictions for the energy integrated production cross section of ^{12}C fragments from a 28.15 MeV neutron beam incident on a ^{12}C target. Experimental data was obtained from Ref. [110].

Fig. VI.9 demonstrates a good agreement between experimental data and that of the GSM and CEM event generators. Inference of the data shown in Fig. VI.9 provides this conclusion, however the event generator data produced does not as well match the slight variations in the data. Not shown in Fig. VI.9 are large relative uncertainties within the experimental data[3]. The general agreement by the experimental data and that of the event generators is brought into question given the large uncertainties within the experimentally available results. A simulation by LAQGSM for the same number of events produced no recognizable results for any of its fragments, and the results produced were well below those of GSM and CEM.

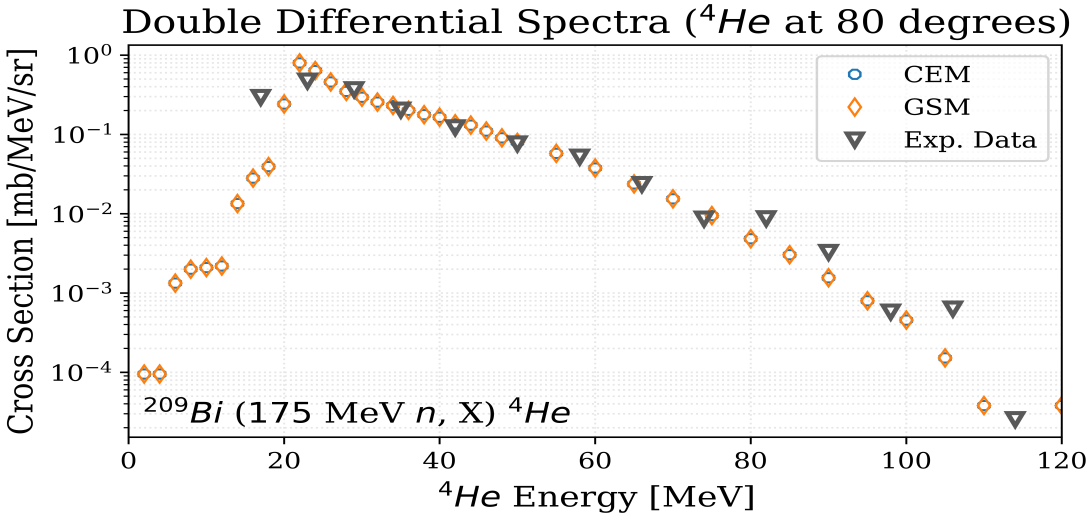

FIGURE VI.9. GSM and CEM predictions for the double differential production cross section of ^{4}He fragments emitted at 80° to a 175 MeV neutron beam incident on a ^{209}Bi target. Experimental data was obtained from Ref. [111–116].

Predictions for ^{3}He fragment production cross sections are compared to experimental data in Fig. VI.10. Good general agreement is found between GSM and experimental data at low ^{3}He energies, however GSM tends to underestimate the data for higher fragment kinetic energies. LAQGSM predictions for other fragments appear to be in general agreement with that of GSM, and CEM is in perfect agreement with GSM. Error within the experimental data is roughly represented by the size of the markers, being around ~ 20% for most of the data, relative to the respective

[3]Examples of the uncertainties quoted by Ref. [111–116] include 0.49366 mb/sr/MeV at 17 MeV to $7.52 \cdot 10^{-4}$ mb/sr/MeV at 114 MeV, equating to relative uncertainties up to ~ 1 000%.

datum point. The small error within the experimental data infers that the event generators may need further improvement, however they do provide a fairly accurate first-order estimate of the data of Fig. VI.10. Percent differences and relative errors are additionally provided by Table VI.II to tabulate, for example, the points where GSM is in good agreement with experimental data.

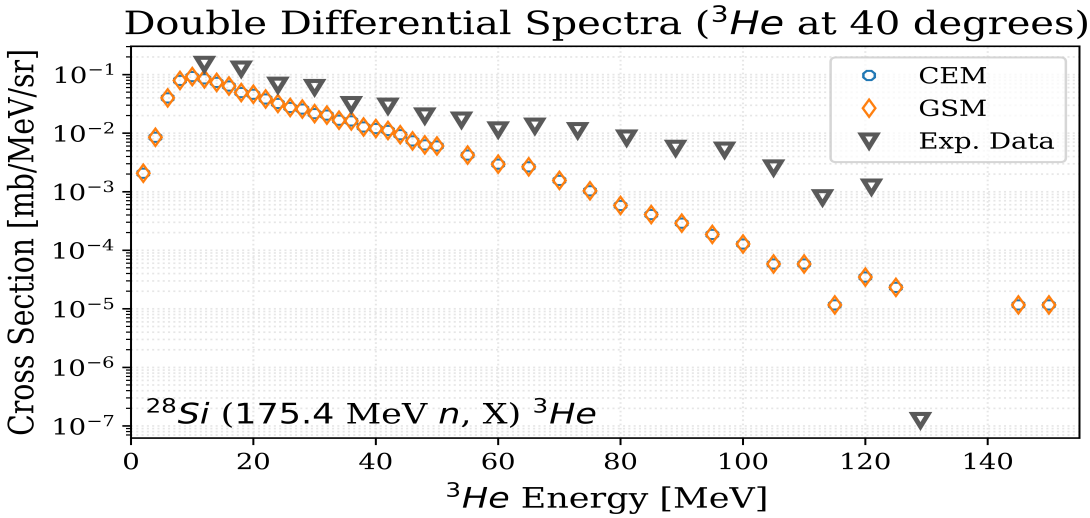

FIGURE VI.10. GSM and CEM predictions for the double differential production cross section of ^{3}He fragments emitted at 40° to a 175.4 MeV neutron beam incident on a ^{28}Si target. Experimental data was obtained from Ref. [116–118].

Predictions by GSM and CEM for the fragment production of tritium are shown in Fig. VI.11. Fig. VI.11 shows general overall agreement between data of the event generators and of experiment. LAQGSM predictions for heavier fragments, such as ^{7}Li, were observed to be in fair agreement with the GSM and CEM predictions. Fair agreement of data was observed at all angles observed for the given reaction.

Backscatter predictions made by GSM and CEM are compared to experimentally obtained data in Fig. VI.12. Fig. VI.12 demonstrates general agreement with experimentally obtained data, however with a slight overestimation of the measured results. Uncertainties in the data are on the order of $\sim 6\%$ for the lower energy fragments, and up to $\sim 40\%$, relative to the respective data, for the experimental data shown. Also note the equivalence of the GSM and CEM predictions in Fig. VI.12.

TABLE VI.II. Percent Difference and Relative Error of Modernized GSM Results to Experimental Data (mb/sr/MeV) for the ^{28}Si *(175.4 MeV n, X)* ^{3}He Reaction

Energy (MeV)	Ref. [116–118]	Modern GSM	Percent Difference	Relative Error
12	$1.51E-01$	$7.16E-02$	71.4%	52.6%
18	$1.26E-01$	$4.52E-02$	94.3%	64.1%
24	$6.68E-02$	$3.00E-02$	75.9%	55.0%
30	$6.08E-02$	$2.07E-02$	98.5%	66.0%
36	$3.11E-02$	$1.44E-02$	73.1%	53.6%
42	$2.92E-02$	$9.37E-03$	102.8%	67.9%
48	$1.99E-02$	$6.40E-03$	102.5%	67.8%
54	$1.69E-02$	$3.86E-03$	125.7%	77.2%
60	$1.15E-02$	$2.63E-03$	125.6%	77.1%
66	$1.33E-02$	$1.83E-03$	151.5%	86.2%
73	$1.12E-02$	$1.09E-03$	164.5%	90.3%
81	$8.36E-03$	$6.15E-04$	172.6%	92.6%
89	$5.56E-03$	$2.07E-04$	185.7%	96.3%
97	$5.18E-03$	$1.37E-04$	189.7%	97.4%
105	$2.58E-03$	$3.50E-05$	194.6%	98.6%
113	$7.97E-03$	$2.72E-05$	186.8%	96.6%
121	$1.22E-03$	$1.75E-05$	194.3%	98.6%
129	$1.25E-03$	$2.91E-06$	183.5%	2230.2%

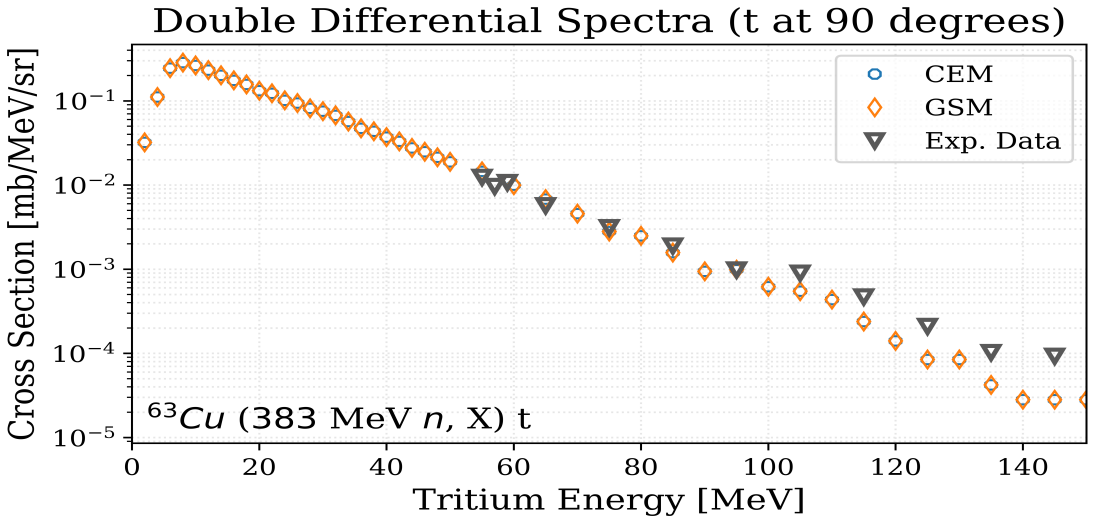

FIGURE VI.11. GSM and CEM predictions for the double differential production cross section of tritium fragments emitted at 90° to a 383 MeV neutron beam incident on a ^{63}Cu target. Experimental data was obtained from Ref. [119].

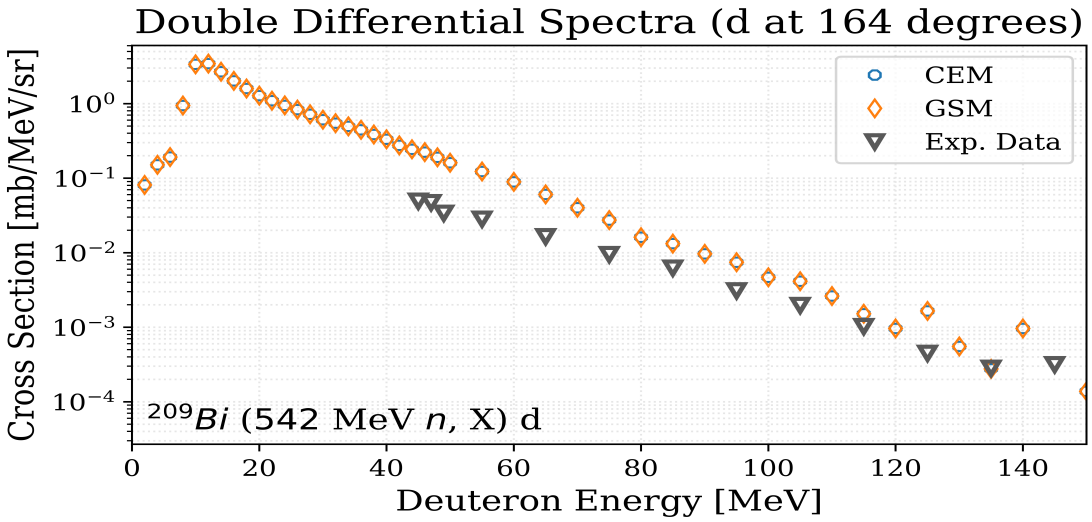

FIGURE VI.12. GSM and CEM predictions for the double differential production cross section of deuterium fragments emitted at 164° to a 542 MeV neutron beam incident on a ^{209}Bi target. Experimental data was obtained from Ref. [119].

Table VI.III shows the required computation times for each neutron-induced reaction. GSM required, on average[4], 102.3% of the time required by CEM when using CEM physics. It is suspected the slight increase in computation times is due to the modified handling of errors in GSM. GSM required 44.8%, on average[5], of the time required by LAQGSM when using CEM physics, respectively. Experimental data is sought after for neutron-induced reactions above ~ 4.5 *GeV* to provide statistical information on GSM computation times when LAQGSM physics models are used. Note each simulation using CEM, GSM, and LAQGSM simulated the same number of events to provide comparable computation times. Each reaction shown in Table VI.III simulated 10 million events.

[4]The standard deviation in GSM-CEM computation time ratios was observed to be 2.3% for these reactions when using CEM physics.

[5]The standard deviation in GSM-LAQGSM computation time ratios was observed to be 8.5% (8.7% for CEM-LAQGSM ratios) for these reactions when using CEM physics.

TABLE VI.III. Computation Times for Neutron Induced Reactions (in hours)

Reaction	CEM	GSM	LAQGSM
^{12}C *(28.15 MeV n, X)*	0.56	0.58	0.94
^{9}Be *(60 MeV n, X)*	0.35	0.36	0.81
^{6}Li *(118 MeV n, X)*	0.27	0.28	0.76
^{209}Bi *(175 MeV n, X)*	1.41	1.42	3.07
^{28}Si *(175.4 MeV n, X)*	0.56	0.56	1.17
^{12}C *(225 MeV n, X)*	0.31	0.33	1.03
^{63}Cu *(383 MeV n, X)*	1.03	1.02	2.02
^{209}Bi *(542 MeV n, X)*	2.47	2.50	5.67
^{27}Al *(800.0 MeV n, X)*	0.73	0.74	1.89

VI.1.3 Light-Ion Induced Reactions

GSM was observed to have slight discrepancies in the physics models used from that of LAQGSM when profiling various light-ion induced reactions, namely with the use of the *repij* routine. The *repij* routine requires near 4% of the simulation time for most interactions with the non-standard DCM of LAQGSM, and is nearly never used within the non-standard DCM of GSM for the observed light-ion interactions. The profiling was done using *gprof*, revealing that GSM may deviate from the appropriate physics, or that the produced fragments have different properties than those of GSM normally. It is suspected this is occurring due to missing or modified physics parameters used within the QGSM portion of GSM's INC, and in addition is thought to skew the results produced by GSM. Minimal results are shown regarding light-ion data due to the known deficiencies within GSM, and it is highly recommended to further develop GSM for light- and heavy-ion induced reactions.

Scattering differential cross sections are plotted in Fig. VI.13. GSM is seen to better predict the angular dependence of the production cross sections than LAQGSM in Fig. VI.13. The results produced by the event generators however are in poor agreement with the available experimental data. Fig. VI.13 also shows that LAQGSM was able to produce more predictions for the interaction than GSM, the opposite of that seen in previous figures. The event generators do however produce a spectrum that appears to match that of experimental results, but at two orders of magnitude smaller.

The poor data agreement between the event generators and experimental data could also be due to the fact that the incident ^{10}B particles are well below the desired threshold energy of ~ 4.5 *GeV* per nucleon. Additionally, the scattering models used in GSM and LAQGSM during the INC simulation may need to be further evaluated and modified to better fit the produced data.

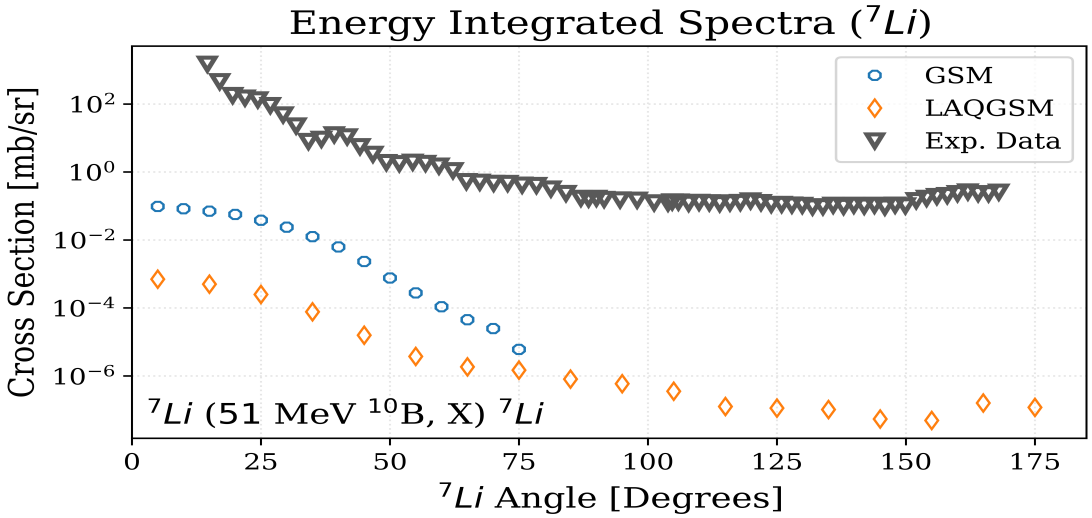

FIGURE VI.13. GSM and LAQGSM predictions for the energy integrated production cross section of ^{7}Li fragments from a 51 MeV ^{10}B beam incident on a ^{7}Li target. Experimental data was obtained from Ref. [120].

Fig. VI.14 demonstrates that GSM well predicts experimental data for the shown reaction. GSM is observed to underestimate the experimentally obtained data marginally for low-energy fragments, however underestimates experiment by a factor of ~ 3 for fragments with kinetic energies ~ 40 MeV. Note however that relative uncertainties were as high as ~ 70% for fragments near ~ 40 MeV, and thus the GSM data lies within these deviations. LAQGSM data was compared to predictions made by GSM for ^{6}Li fragments, and was in near-perfect agreement with GSM for fragment energies up to ~ 80 MeV. Production cross sections for higher fragment energies were predicted to be less than that of GSM. Observed angular dependence of the experimentally determined cross sections, not shown here, did not agree as well with GSM. GSM had over predicted this data by roughly a factor of 2 near backscattering angles, but was in general agreement with experiment for forward peaked angles.

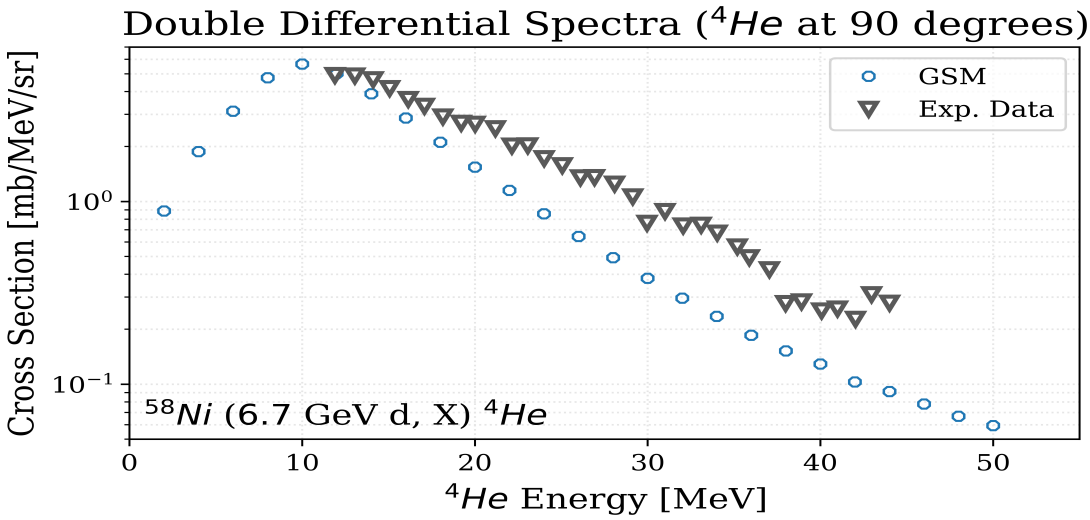

FIGURE VI.14. GSM predictions for the double differential production cross section of ^{4}He fragments emitted at 90° to a 6.7 GeV deterium beam incident on a ^{58}Ni target. Experimental data was obtained from Ref. [121].

Table VI.IV shows the required computation times for each of the observed light-ion induced reactions. GSM required 76.7%, on average[6], of the time required by LAQGSM when using the non-standard DCM for its INC physics. The increased speed by which GSM computes these reactions is thought to be considerable given how often the models will be used when calculating high-energy spallation physics within a particle transport code, such as MCNP. Note each reaction shown in Table VI.IV simulated 1 million events.

VI.2 VALIDATION OF THE MODERNIZED GSM

Validation of the GSM, after its modernization and coding-error resolutions, has not been thoroughly completed due to recent issues relating to accessing the software. Validation of the modernized GSM has been briefly explored and is presented here through the usage of unit and regression tests.

The predicted energy spectrum of emitted protons from the ^{197}Au *(*319.00 *MeV* ^7Li*, X)* reaction

[6]The standard deviation in GSM-LAQGSM computation time ratios was observed to be 27.6% for these reactions when using the non-standard DCM for its INC physics.

TABLE VI.IV. Computation Times for Light-Ion Induced Reactions (in hours)

Reaction	CEM	GSM	LAQGSM
^{7}Li *(51 MeV* ^{10}B*, X)*	*N/A*	6.64	6.69
^{232}Th *(220 MeV* ^{11}B*, X)*	*N/A*	23.08	20.71
^{197}Au *(319 MeV* ^{11}Li*, X)*	*N/A*	68.86	67.26
^{1}H *(490.4 MeV* ^{8}He*, X)*	*N/A*	0.21	0.90
^{27}Al *(1.14 GeV* ^{12}C*, X)*	*N/A*	3.01	4.14
^{16}O *(1.14 GeV* ^{12}C*, X)*	*N/A*	1.71	2.89
^{48}Ti *(1.14 GeV* ^{12}C*, X)*	*N/A*	5.66	6.75
^{238}U *(4.2 GeV d, X)*	*N/A*	33.45	35.39
^{12}C *(4.64 GeV* ^{16}O*, X)*	*N/A*	1.41	3.04
^{58}Ni *(6.7 GeV d, X)*	*N/A*	5.69	7.67

is depicted in Fig. VI.15, plotted according to the protons' production mechanism. Protons are seen to most often be emitted as a result of the slow evaporation phase of the reaction, making up the majority of the energy spectrum, whereas the fast INC phase dominates at intermediate energies. Note also that the energy spectrum for high-energy protons produced via the fast INC stage declines due to particle coalescence, where the emitted protons compound to create light-ions. The evaporation process dominates at larger emittance energies as the compound nucleus de-excites, imparting their excitation energy into the emitted particles. Fig. VI.15 additionally demonstrates how the preequilibrium sub-model of the GSM models the Coulomb barrier well, while the cascade and evaporation versions of the Coulomb barrier could be improved. The sub-models of GSM each utilize their own Coulomb barrier calculation, a phenomena limiting the emittance of charged particles below a threshold energy. Recent updates to the Coulomb barrier of the preequilibrium model indicates the Coulomb barrier model better replicates the limiting threshold value, validating its future implementation into the other sub-models of the GSM. The implementation of the preequilibrium model's Coulomb barrier could easily be implemented given the modernized architecture of the GSM and its sub-models.

The emittance of ^{6}He fragments from the ^{107}Ag *(480.00 MeV p, X)* reaction is demonstrated by Fig. VI.16. The predictions made by each of the event generators depicted is shown to under-predict the experimental data at each emittance angle, however closely matching the shape of the

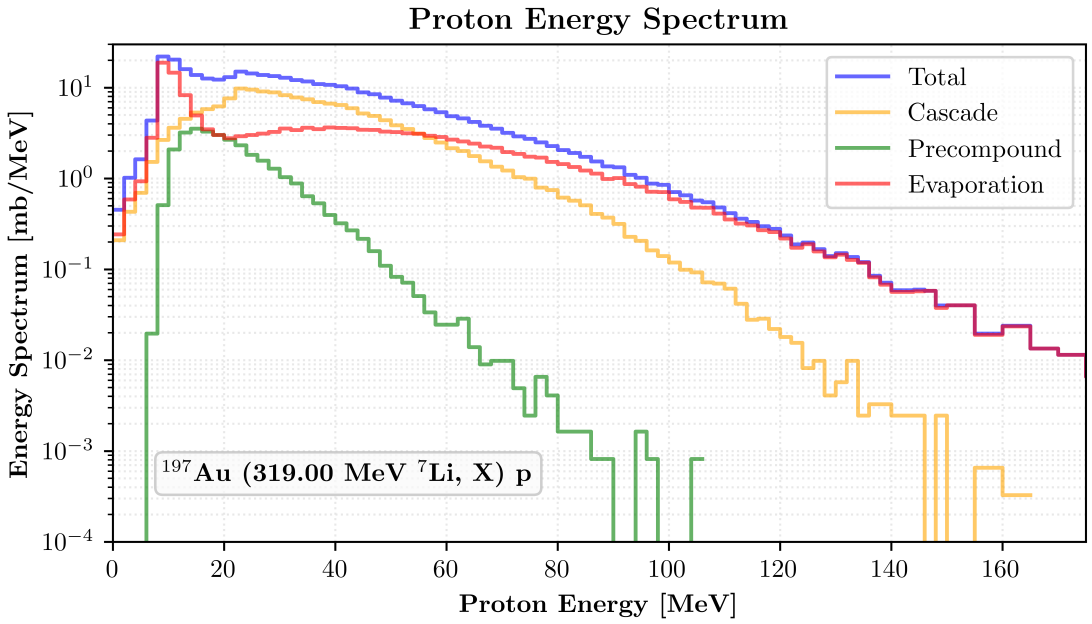

FIGURE VI.15. The modernized GSM prediction for the energy spectrum of emitted proton fragments from a 319.00 MeV ^{7}Li beam incident on a ^{197}Au target. Experimental data is unavailable for this reaction.

distribution at all intermediate and high emittance energies. Fragment estimation at low emittance energies for the charged ^{6}He particles ought to be negligible, yielding a sharp cutoff as a result of the nucleus's Coulomb barrier. The effect of the Coulomb barrier is most easily observed in the experimental data for ^{6}He emitted at 90° to the incident proton beam, where each of the observed event generators does not correctly predict the distribution of fragments at low emittance energies.

The GSM does not include with its double differential production cross sections variances regarding the predicted results via Monte-Carlo techniques. The GSM was ran using several different random number generator types available to it, each using various seeds, to estimate the uncertainty in the predictions for the ^{107}Ag *(480 MeV p, X)* ^{6}He reaction. Six of the seven available GSM random number generators utilized six different seeds each, yielding 36 statistically independent GSM simulations for the reaction. Fig. VI.17 plots the results of each of these reactions, and Fig. VI.18 plots the average of the GSM predictions along with the 67% and 99% confidence intervals.

It is seen in Fig. VI.17 that GSM fairly consistently estimates low- and mid-energy particle

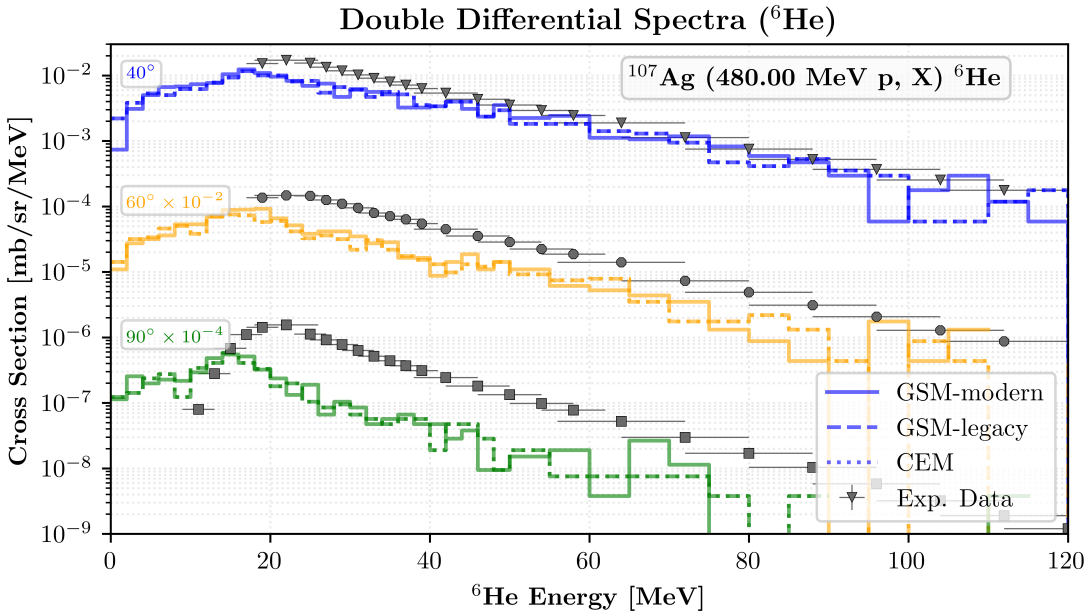

FIGURE VI.16. GSM and CEM predictions for the double differential production cross section of ^{6}He fragments from a 480.00 MeV proton beam incident on a ^{107}Ag target. Experimental data was obtained from Ref. [102].

emission probabilities. The plotted simulations shown do not sample high-energy particle emission as frequently, being a result of the physics simulated, thus showing larger variances at these energies. The mean, μ, alongside the 67% and 99% confidence intervals, are shown in Fig. VI.18. It is seen in Fig. VI.18 that experimental data is under-estimated by GSM in the 99% confidence interval, particularly for particles emitted with the maximum measured probabilities within the mid-energy range for larger emission angles. The GSM estimates correctly particle production at high-emission energies within the 99% confidence interval.

Fig. VI.19 depicts the emittance of neutrons from the ^{232}*Th (1.2 GeV p, X)* reaction at several angles. It is seen that the incident proton effectively "knocks out" one or more neutrons from the ^{232}Th nucleus, being caused as a result of direct collisions during the fast INC reaction phase and as a result of nucleus de-excitation and decay. There is a relatively good agreement with experimental data. Neutrons emitted at large energies are likely those due to direct collisions with the incident proton or those due to compound evaporation, as shown later in Fig. VI.15. Low energy neutrons are likely a result of the de-excitation of the residual nucleus during the intermediate preequilibrium

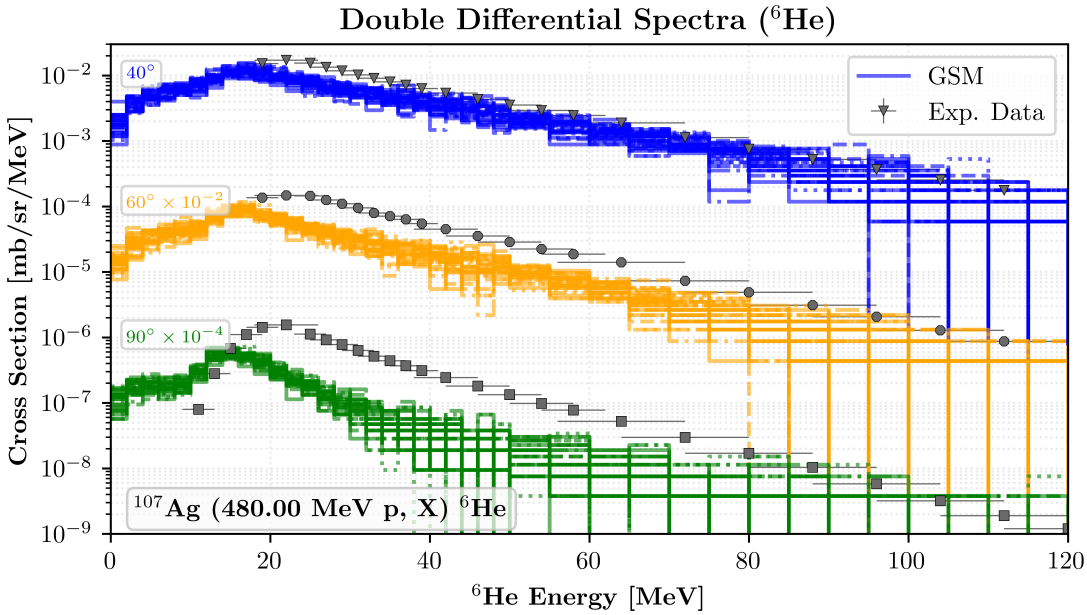

FIGURE VI.17. Multiple GSM predictions for the double differential production cross section of ^{6}He fragments from a 480.00 MeV proton beam incident on a ^{107}Ag target. Each simulation of GSM utilized one of the seven available random number generators, using one of six initial random number generator seeds, yielding a total of 36 unique simulations and associated predictions. Experimental data was obtained from Ref. [102].

phase of the reaction. It is important to note the legacy GSM predictions are equivalent to those of the CEM event generator, whereas the modernized GSM numerically differs slightly due to the implemented bug fixes and code improvements to the sub-models of the modernized GSM.

Predictions by both the legacy and modernized GSM, in addition to those of the CEM, are shown in Fig. VI.20 for the ^{63}*Cu (*383.00 *MeV n, X)* reaction. Both the legacy and modernized GSM results appear to be similar, where results appear to fall within statistical deviations. The predictions of the CEM and the modernized GSM appear to well replicate experimental data as shown in Fig. VI.20. It is important to note however the predictions of fragments with little to no emittance energy, as these particles would not be created due to the effect of the Coulomb barrier. It is suggested that the improved Coulomb barrier model, currently utilized by the preequilibrium model, be adapted and implemented into the other sub-models of the GSM to better predict particle emission at emittance energies below the Coulomb barrier.

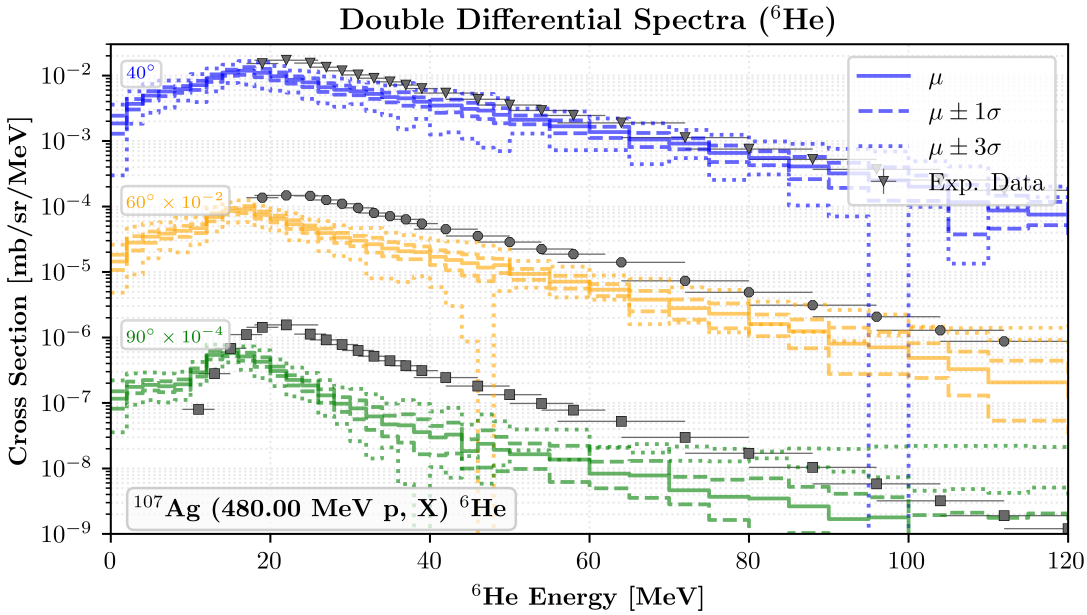

FIGURE VI.18. The mean, μ, and the 67% and 99% confidence intervals of GSM predictions for the double differential production cross section of ^{6}He fragments from a 480.00 MeV proton beam incident on a ^{107}Ag target. The 67% confidence interval is labeled as $\mu \pm \sigma$, and the 99% confidence interval is labeled as $\mu \pm 3\sigma$. Experimental data was obtained from Ref. [102].

Fig. VI.21 depicts results of the modernized GSM similar to those of its legacy counterpart, as also shown in Fig. VI.14. The modernized GSM appears to predict double differential cross sections of emitted particles similarly to its legacy counterpart, as seen in Fig. VI.21. The under-predictions produced by both the legacy and modernized GSM indicate that the modified DCM or evaporation sub-models of the GSM ought to be further validated to aid the GSM in providing its clients and end-users with more accurate predictions. Fig. VI.21 indicates the evaporation sub-model may be the cause of the under-predications, as the peak of the curve well replicates that of experimental data, suggesting the modified DCM well models INC physics for the depicted collision. According to Fig. VI.15, emittance of ^{4}He fragments may occur primarily as a result of particle evaporation.

Fig. VI.22 depicts predictions by both the modern and legacy GSM event generators. Although not shown here, the predictions by the LAQGSM event generator were observed to be similar to those of both the legacy and modernized GSM. Fig. VI.22 clearly depicts the need for improvement in the GSM event generator regarding collisions of large particles, whereby the GSM event generator

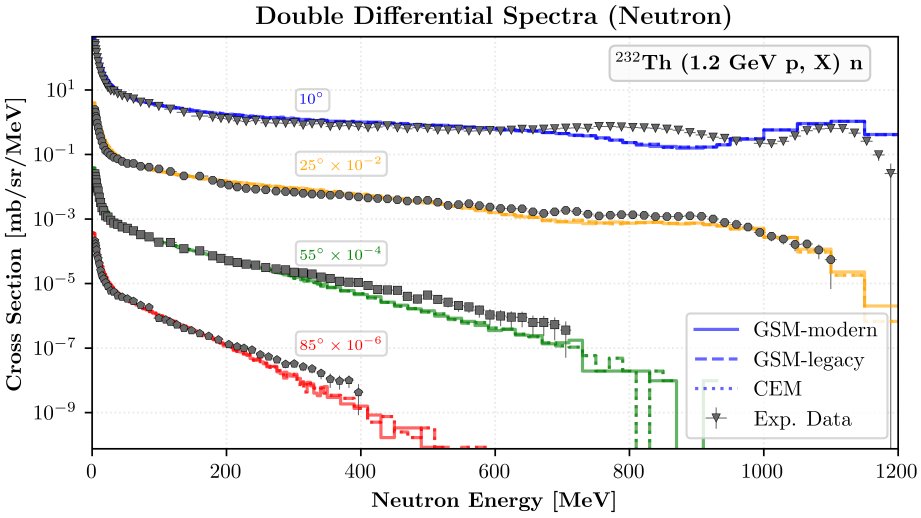

FIGURE VI.19. GSM and CEM predictions for the double differential production cross section of neutron fragments from a 1.2 GeV proton beam incident on a ^{232}Th target. Experimental data was obtained from Ref. [103].

under-predicts those collisions by an order of magnitude at low to intermiediate emmitance energies. It is thought that improvement in the modified DCM or evaporation sub-model of the GSM will prove the most beneficial, as most particles emitted near these energies are produced via INC and evaporation physics. Additionally, further improvement of the modified DCM to better predict experimental data would produce more neutrons and protons with high emittance energies, allowing for their coalescence and thus better predicting formation of these fragments at higher emittance energies. It is also of importance to note the LAQGSM event generator, although not shown here, produces similar results for this interaction [122], indicating the under-estimation of experimental results pre-existed within the modified DCM model, the evaporation model, or both, prior to the creation and modernization of the GSM.

The computation times required by the GSM for some simulations, being those in the regression testing suite, were observed throughout its continued development. Fig. VI.23 details some computation times utilized by the GSM for various simulations, as compared to its predecessors, the CEM and LAQGSM event generators, and its legacy version, being GSM prior to its modernization. Note the computation times for the simulations presented in Fig. VI.23 are not those of the regression

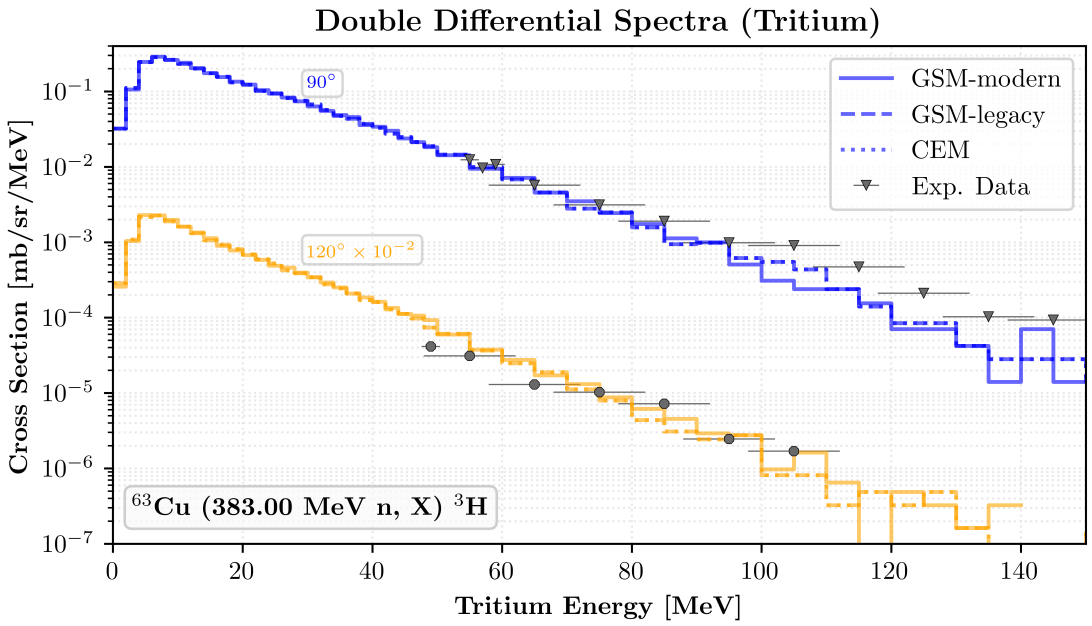

FIGURE VI.20. Modern and legacy GSM and CEM predictions for the double differential production cross section of tritium from a 383.00 MeV neutron beam incident on a ^{63}Cu target. Experimental data was obtained from Ref. [119].

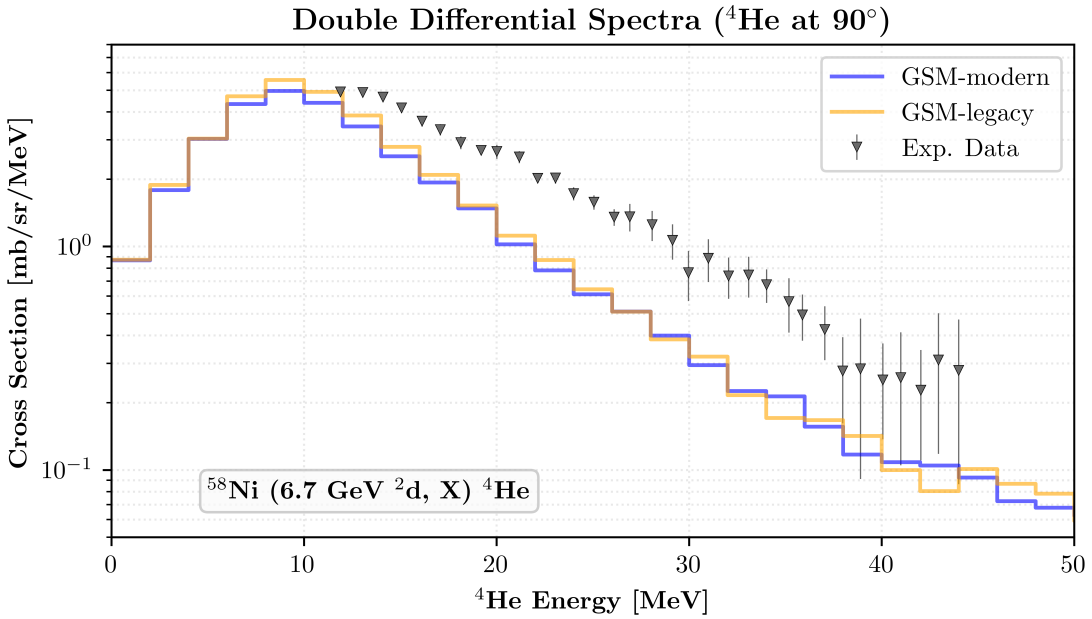

FIGURE VI.21. Modernized and legacy GSM predictions for the double differential production cross section of ^{4}He fragments from a 6.70 GeV ^{2}d beam incident on a ^{58}Ni target. Experimental data was obtained from Ref. [121].

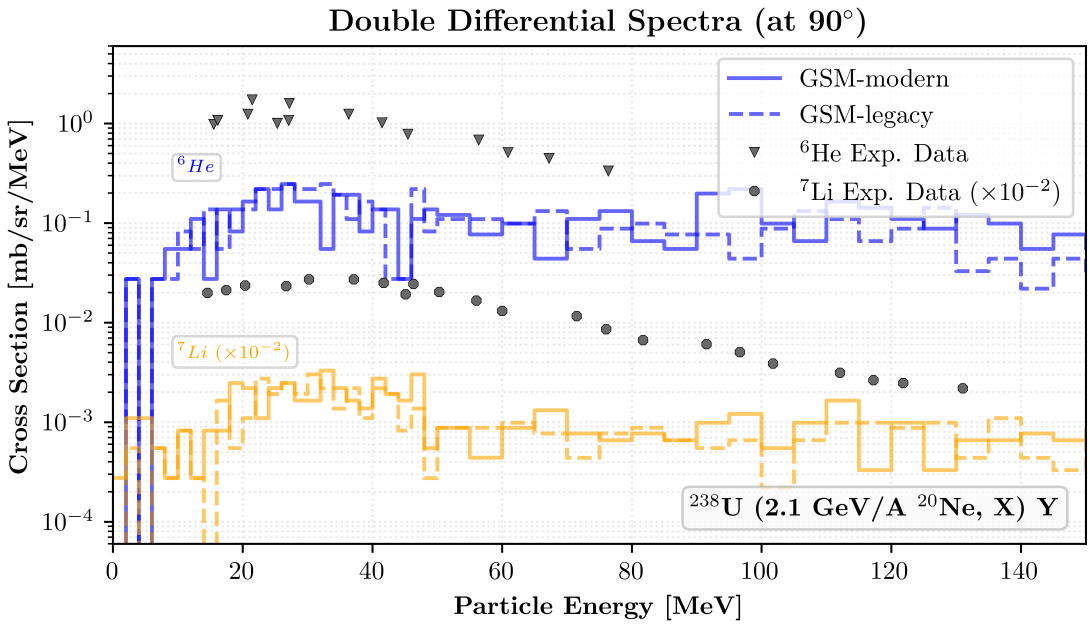

FIGURE VI.22. The legacy and modernized GSM predictions for the double differential production cross section of both ^{6}He and ^{7}Li fragments from a 42.00 GeV ^{20}Ne beam incident on a ^{238}U target. Experimental data was obtained from Ref. [123].

suite, but instead are those observed. Fig. VI.23 depicts the required computation times of these event generators, normalized per event, from a variety of simulations. It is seen that the modernized GSM event generator usually requires less time than the LAQGSM event generator, while incurring little penalty compared to the CEM for the benefits of a modern object-oriented structure. In comparison to the CEM, the modernized GSM event generator consumes slightly more computation time due to the coupling and selection of the standard and modified Dubna Cascade Model (DCM), the method utilized to generate verbose terminal output - allowing clients of the GSM to set verbosity limits - and also due to additional sampling that occurs within itself throughout the duration of a given simulation.

The coupling of the INC models in the GSM, the standard DCM and modified DCM, currently utilizes a switch-case statement for its sampling, where the usage of the standard DCM and modified DCM are sampled with each event for incident photons, pions, neutrons, and protons, and are sampled only once for heavier particles. Note usage of the anti-lab system presently requires usage

of the modified DCM for the INC simulation. Coupling of the standard DCM and modified DCM within the GSM additionally includes various `if` statements in an attempt to improve the sampling efficiency of the object.

It is important to note the GSM always generates its verbose output. Clients and end-users, when selecting the simulation verbosity, simply filter out those messages. The GSM generates terminal messages, consisting of comments, warnings, and errors, via a `write` statement to populate the internal I/O object's message character array. Upon generation, the GSM then utilizes its I/O object's `print` function to write the generated message to the terminal. Note clients may provide their own `print` function for the internal GSM I/O object to utilize, as stated in Sec. III.5, particularly in Fig. III.6.

The modernized GSM utilizes the provided random number generator (RNG) more frequently as well. The GSM, when using its default RNG, utilizes a skip-ahead feature to aid in maintaining consistency and reproducibility in simulations. This addition allows future development of the GSM to allow end-users and clients to "skip-ahead" to a particular inelastic event for testing the progeny produced. This feature is, at present, not utilized.

The computation times depicted in Fig. VI.23 are those for the modernized GSM when a high verbosity is selected, where a low verbosity reduces computation times nearly to those of the legacy GSM. The impact on computation time when utilizing a low verbosity limit indicates that the modernized GSM may require more computation time for simulations where warnings and errors may more frequently occur. This is observed due to the inefficient methodology GSM utilizes for generating and printing messages. Such simulations will primarily involve small target nuclei as the standard DCM does not well handle light targets[7]. Modification of the GSM to maintain the simulation state in the event of generated warnings will improve the computation times required by the GSM as it would not restart an event.

[7]The standard DCM suggests event restarts when residuals with $A_T \leq 4$ are generated during the simulation. The GSM presently adheres to this recommendation.

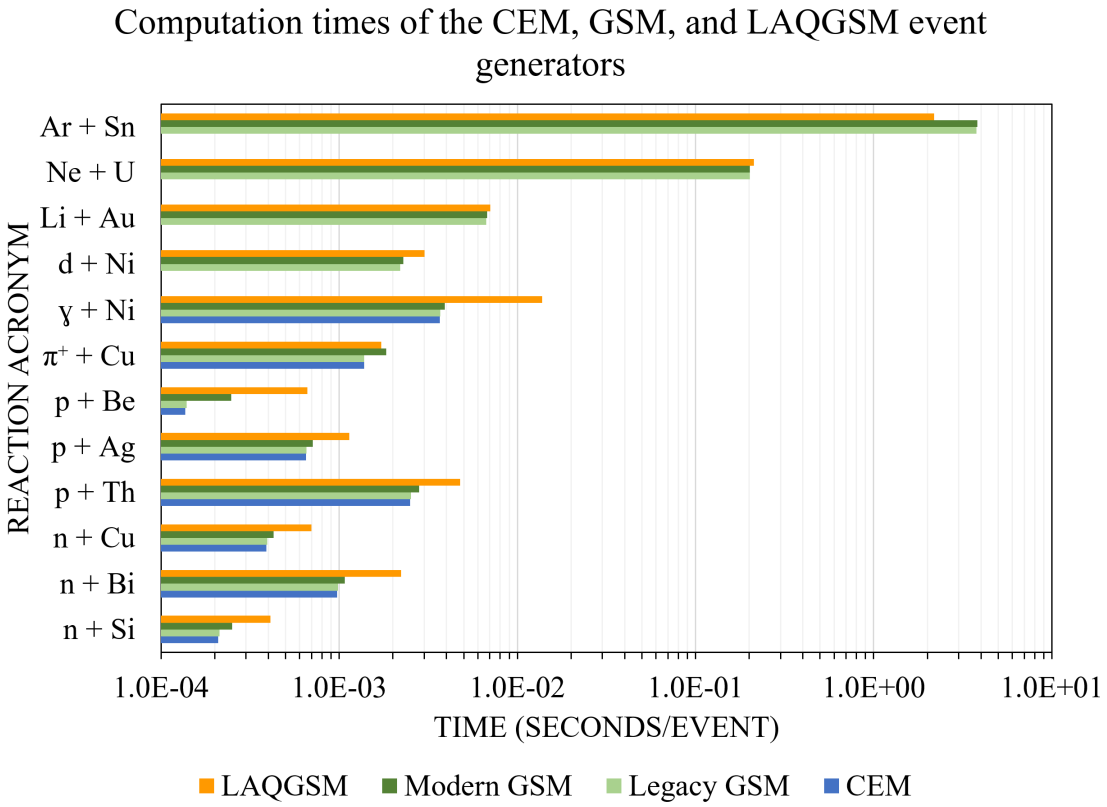

FIGURE VI.23. Computation times of the modern and legacy GSM event generator compared against those of the CEM and LAQGSM event generators, normalized to seconds per event, for all observed reactions.

VI.3 CONTINUED VALIDATION EFFORTS OF GSM

GSM appears to replicate experimental data well for most of the reactions simulated. It is highly recommended that further validation of GSM occur with the inclusion of more tests for its modern architecture. The tests ought to require, in total, the use of all calculation options available to GSM and its various sub-models. Analysis tools should be created in addition to aid in this endeavor. This process includes creating tools to analyze particle multiplicities, yields, and other output calculations GSM can perform. Details on the various calculation options may be found in Ref. [1]. It is also recommended to perform a best fit analysis for all simulations performed to quantitatively state the effectiveness of GSM as an event generator, to be used in place of CEM and LAQGSM, determining if the event generator simulates energetic spallation reactions and their properties during experimentation well, in addition to providing a numerical basis for the

deprecation of LAQGSM.

Continued validation efforts ought to focus primarily on expanding the number of reactions GSM is compared against, particularly within its suggested energy regime of $\sim 100\ MeV$ to $\sim 1\ TeV$ per nucleon. The tests should collectively require the use of all GSM features. Daily regression testing on many interactions is also suggested, such as is possible by inclusion of `CTest` within GSM's `CMake` build platform. An extensive regression testing suite is recommended to utilize all of GSM's calculation options for all reaction types within several energy regimes each. The regression testing suite is recommended to contain around 100 simulations, at minimum, to ensure most reactions are considered for its further validation and development. Testing procedures also ought to be written for each of GSM's sub-libraries given the modern architecture utilized by GSM and its sub-models. Clients may individually test each sub-library for a variety of reactions, applications, or both. GSM provides clients access to its sub-library interfaces for client ease, however it is recommended that clients create their own interface to each of the appropriate sub-libraries to ensure full coverage of the library's features and behaviors.

Continued validation efforts can be further improved and expanded to include the use of white box methods, meaning tested simulations focus on using specific areas of the physics models internal to GSM, taking into account the code architecture of GSM. Recommendations stated above are also strongly encouraged for the continuing validation efforts of GSM. In short, the inclusion of more simulations of various types should be done to further validate the physics simulated by GSM.

Further validation of GSM is recommended, however it has been seen that GSM simulates high-energy events well for incident neutrons and protons. A more extensive validation of GSM, with use of its other available features, such as fragment multiplicity, is highly recommended in addition to the previously stated recommendations. Further development of GSM, in particular for light- and heavy-ion reactions, is recommended prior to placement of GSM into MCNP.

Chapter VII. Summary

GSM is a highly robust event generator with a physics-rich history, whose sub-models and code basis dates back to the 1970s. The GSM is recommended for simulations involving high-energy spallation physics for particles colliding at energies nearing $\sim 100\ MeV$, up to nearly $\sim 1\ TeV$ per nucleon.

GSM simulates its spallation physics in three primary phases: the fast INC state, the intermediate equilibration or preequilibrium stage, and lastly the slow, compound stage of the reaction. The fast stage consists primarily of the IntraNuclear Cascade process, where IntraNuclear Cascade progeny may coalesce to form compound nuclei. Equilibration of the residual nucleus, or nuclei when the modified DCM is utilized, occurs as the residual nucleus decays to a more stable form via the emission of single, pairs of nucleons, or both. Evaporation and fission of the resulting compound nucleus is then considered as the nucleus begins to de-excite, emitting various particles during its deexcitation. Fermi breakup of an excited nucleus is considered when the statistical assumption utilized in both the preequilibrium and evaporation models is not satisfied, where the excited nucleus is instead disintegrated as a result of its excitation.

The modernization of GSM to utilize advanced modern Fortran, in place of legacy Fortran, has led to numerous improvements within the GSM code itself. GSM is now highly robust in its data protection, ensuring parameterizations are held constant and restricting the usage of various components of each GSM sub-model within and between the various sub-models utilized. The modernization of GSM has greatly reduced the technical debt inherent in its code, whereby GSM has become a highly sustainable code. Developers of GSM may additionally modify the existing GSM code base easily, where external applications exist to perform regression tests and verify the results of GSM simulations. The internal structure of each modernized model was encapsulated into an object with the general structure shown in Fig. VII.1.

The object-oriented architecture of GSM additionally offers great flexibility to GSM itself and

```
! Import all object-dependencies (data objects, other models, etc.):
!    For example: Molnix, FissionBarrier, PreequilibriumData, FermiBreakUp
use someModelClass, only: ModelData

type, public:: GeneralObject
   ! ================================================
   ! Object member-variables
   ! ================================================
   private

   ! Flags if the object was constructued
   logical, private :: constructed = .FALSE.

   ! I/O object for the model:
   type(generalIO), private :: io

   ! Controller for model behavior and numerics:
   type(generalOptions), private :: options

   ! Data objects:
   type(ModelData), private :: data
   procedure(RANDOM), private, pointer :: rang => NULL()

   ! For Preequilibrium, and Evaporation (GSM uses global to all instances):
   logical, private :: usePhotonEmission = .FALSE.
   procedure(PHOTOEMISSION), private, pointer :: photonEmission => NULL()

 contains
   ! ================================================
   ! Object member-procedures
   ! ================================================
   private

   ! Simulate procedures, for example:
   procedure, public  :: simulate   ! Interfaces to "startSimulation"
   procedure, public  :: start => simulate

   ! Setter/getter-type methods, for example:
   procedure, public  :: properlyConstructed
   procedure, public  :: setOptions
   procedure, public  :: queryOptions

   ! Internal members for the simulation (highly model-specific):
   procedure, private :: startSimulation
   procedure, private :: someOtherProcedure
   .
   .
   .

end type GeneralObject
```

FIGURE VII.1. The general structure utilized for each modernized object within the GSM.

its clients, increased modularization, limited scopes, reduced dependencies, increased scalability, and platform-independence. Initialization of GSM and its sub-models' data has also been enhanced to allow clients complete control over its data initialization. Construction of a GSM object, in addition to each of its sub-model objects, is highly flexible where a robust API is utilized to aid in flexibility. Modification of the state of the GSM or sub-model object is additionally further protected by the API as a result of included limits and physicality protection. The level of encapsulation utilized within GSM and each of its sub-models provides the software library with longevity and ease of further development, where usage of each data object is clearly present in a given piece of the code. The addition of other IntraNuclear Cascade models and physics options to GSM is suggested to aid in the generality and scope of GSM, whereby GSM may be used at potentially lower energies for its simulations and provide its clients and end-users with greater physics processes to be considered, aiding in simplification of the client and its usage by clients, end-users, or both. Fig. VII.2 demonstrates a minimal API for clients when using the GSM.

Verification of GSM has occurred continuously during its migration to a modern, advanced object-oriented Fortran code. Various coding errors have been resolved throughout this process, documented nearly completely in Appendix A, to ensure GSM is performing the calculations as is claimed in this document. All of GSM and its sub-models, excluding the modified DCM, utilize a similar framework to aid in minimal model object states and are consistently validated, all using a fairly similar API structure, containing various methods to modify and query the state of the model object. Fig. VII.1 demonstrates the structure utilized by the GSM and each of its sub-models, where applicable. The modernization of GSM and its sub-models additionally allows for a logical simplification of its various physics constructs, again reducing the technical debt existing within GSM and improving its maintainability.

The validation of the GSM, both in the legacy and modernized versions, was explored briefly. It was seen that the GSM event generator well predicts interactions involving light incident particles, as shown in Fig. VII.3 and in Fig. VII.4. It is seen in Fig. VII.3 that the GSM well predicts experimental data for emitted neutrons at various angles for most emittance energies. It is important

```
subroutine sample_GSM_API(projectilePbj, targetObj)
! ========================================================================
! A composite sample API for software clients of the GSM
! ========================================================================
   use, intrinsic:: iso_fortran_env, only: int32, real64
   use gsmClass, only:   &
        & GSM,           &   ! GSM Object
        & newGSM,        &   ! GSM Constructor
        & gsmOptions,    &   ! Options controller
        & gsmVerbose,    &   ! Controls all object's verbosity
        & gsmProjectile, &   ! Projectile object
        & gsmTarget,     &   ! Target object
        & gsmProgeny     &   ! Progeny tracked by GSM
        & gsmResults,    &   ! Results object
        & newGSMResults      ! Results object constructor

   type(gsmProjectile), intent(inout) :: projectileObj
   type(gsmTarget),     intent(inout) :: targetObj

   ! ========================================================================
   ! Set options:
   type(gsmOptions) :: options
   options%nucleonTransitionE = 1000.0_real64 ! Reduce INC transition to 1 GeV

   ! Construct the  object:
   type(GSM) :: gsmObj
   gsmObj = newGSM(clientOptions = options)

   ! ========================================================================
   ! Create and construct the results object
   type(gsmProgeny), dimension(150):: progenyBnk
   type(gsmResults):: results
   results = newGSMResults(progenyBnk)   ! Object construction

   ! Simulate a collision:
   gsmObj%collide(projectileObj, targetObj)

   ! Interface to the results, for example
   write(*, 1000) results%numProgeny
   if(results%simState /= 0_int32 ) then
      write(*, 1100) results%simState
   end if
   return
! ========================================================================
1000 format("There were ", i3, " progeny produced during the simulation.")
1100 format("Warning: the GSM simulation ended with a warning or error (", i3,
    ").")
! ========================================================================
end subroutine sample_GSM_API
```

FIGURE VII.2. A sample GSM API.

to note the legacy GSM predictions are equivalent to those of the CEM event generator, whereas the modernized GSM numerically differs slightly due to the implemented bug fixes and code improvements to the sub-models of the modernized GSM.

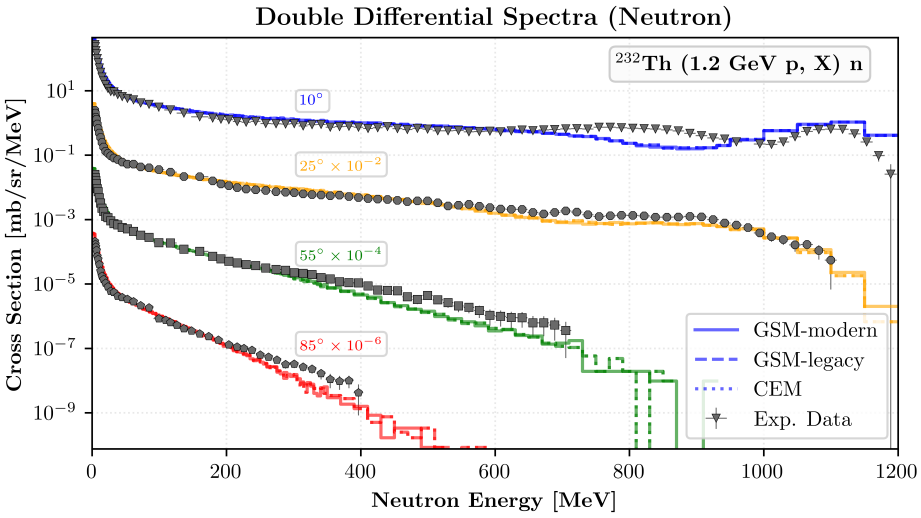

FIGURE VII.3. GSM and CEM predictions for the double differential production cross section of neutron fragments from a 1.2 GeV proton beam incident on a ^{232}Th target. Experimental data was obtained from Ref. [103].

The GSM, however, could be further improved upon as seen in Fig. VII.4, and from others observed in Chapter VI. It is indicated from Fig. VII.4 that the Coulomb barrier, restricting fragment production at low emittance energies, ought to be improved. It was observed that the Coulomb barrier of the preequilibrium model well estimates this phenomena, and thus it is recommended to incorporate it into the other sub-models of the GSM for better fragment estimation at these energies. It was also seen that the evaporation model utilized under-estimates fragment production at intermediate to high emittance energies as the evaporation model's predictions estimate fragment production to rapidly decrease at these larger energies, as seen in Fig. VI.15. It was also seen that the GSM event generator under-estimates fragment production by an order of magnitude for interactions involving larger incident particles, such as ^{20}Ne, as seen in Fig. VII.5. Note the LAQGSM event generator, utilizing the modified DCM in the same manner as the GSM, produces similar predictions for such events [122].

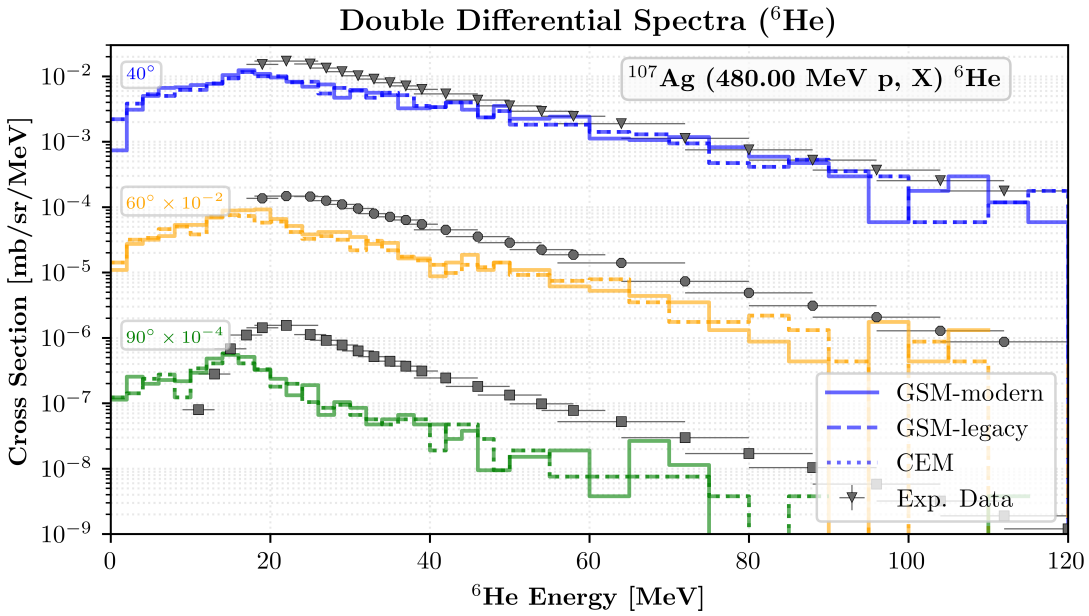

FIGURE VII.4. GSM and CEM predictions for the double differential production cross section of ^{6}He fragments from a 480.00 MeV proton beam incident on a ^{107}Ag target. Experimental data was obtained from Ref. [102].

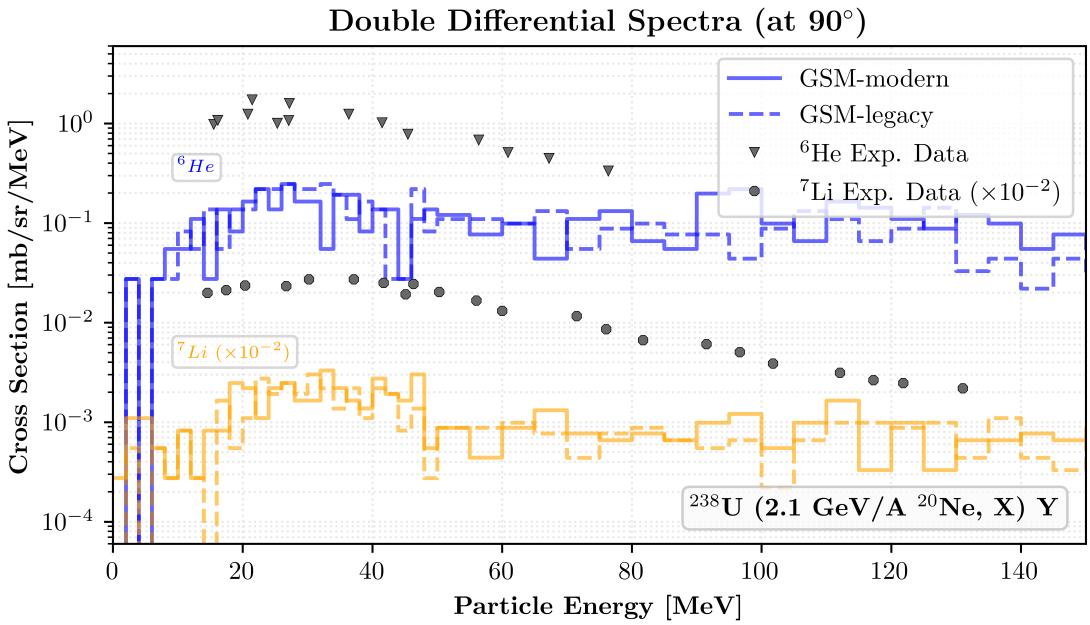

FIGURE VII.5. The legacy and modernized GSM predictions for the double differential production cross section of both ^{6}He and ^{7}Li fragments from a 42.00 GeV ^{20}Ne beam incident on a ^{238}U target. Experimental data was obtained from Ref. [123].

Several computation times for the CEM, legacy and modernized GSM, and LAQGSM event generators were obtained and analyzed. It is seen in Fig. VII.6 that the modernized GSM event generator usually requires less time than the LAQGSM event generator, while incurring little penalty compared to the CEM for the benefits of a modern object-oriented structure. It is noted that the modernized GSM may require more computation time for collisions that may generate more warnings, errors, or both, during the simulation. Such collisions, for example, may involve small target nuclei.

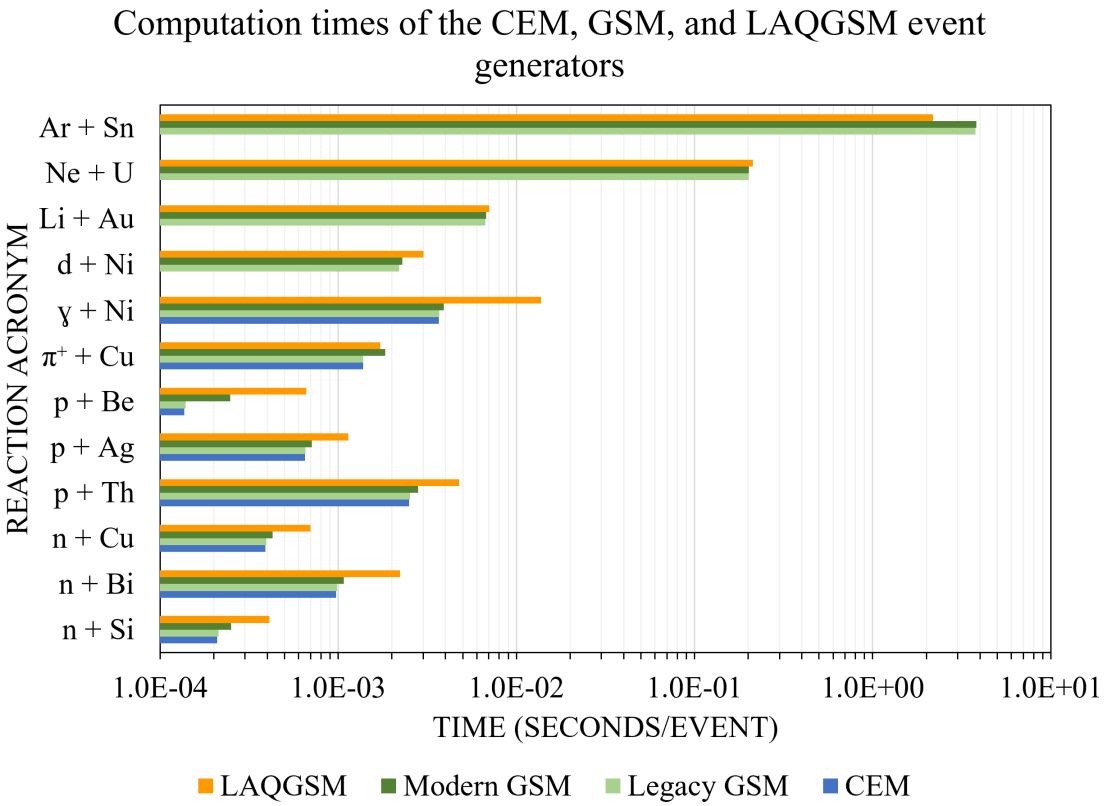

FIGURE VII.6. Computation times of the modern and legacy GSM event generator compared against those of the CEM and LAQGSM event generators, normalized to seconds per event, for all observed reactions.

GSM is in addition highly modular due to the work presented here. The sub-models of GSM now utilize clear and succinct dependencies. Models may be simplified, enhanced, replaced, *etc.*, very simply now as a result of the increased modularity. Further modification of GSM and its sub-libraries is additionally highly simplified as a result of the codes modularity. Appendix A

contains a list of recommended improvements, all of which may be easily implemented as a result of the code's modularity. Each of the recommended improvements listed in Appendix A will require minimal effort and yield even greater flexibility as a result of the modern and advanced architecture utilized by GSM.

The individually available modern components of GSM and its sub-models allows for enhanced portability between clients and platforms, increasing the usefulness and reach of each of the GSM sub-libraries. Inclusion of a minimal `CMake` build and packaging system has increased the portability of each GSM sub-library, whereby clients may link their own libraries to those of GSM to incorporate additional physics processes into their simulations. The scalability of GSM has also been highly improved, whereby allowing a single GSM object to perform several simultaneous simulations, when utilizing the standard DCM for its IntraNuclear Cascade simulation, when called to do so by its clients. The highly scalable architecture of GSM, alongside its minimal object state, allows clients to utilize any number of GSM objects with minimal memory footprint until the object is called to perform a simulation of some sort according to those laid out in its API. Presently end-users of GSM cannot scale GSM as much as event tallying and output calculations are performed utilizing legacy Fortran common blocks. Note that there are many benefits that have been acquired as a result of the work presented here, however only those of great importance are listed and discussed for brevity.

Continued development of GSM is highly suggested, with simple improvements being outlined in Appendix A. A full and complete modernization and object-oriented migration of the modified DCM is highly encouraged to allow GSM to simulate single spallation events concurrently, as this feature would be highly desirable for clients of GSM. Continued migration of GSM to an object-oriented architecture is highly recommended for its event tallying and output file generation as well, in an effort to provide end-users with a maintainable and quickly generated output file. Allowing concurrent event tallying within GSM for end-users will aid in the precision and timeliness of GSM results for its end-users as well, aiding them greatly in their respective fields of expertise.

The development, modernization, object-oriented migration, verification, and validation of

GSM has proven fruitful and highly beneficial. The migration of GSM, previously being a legacy Fortran code, to utilize an advanced and modern Fortran architecture has provided many benefits as outlined above, providing developers with great flexibility, portability, and maintainability, and providing its clients and end-users with flexibility, robustness, simplification, scalability, and much more. Continued development of GSM is highly recommended to allow GSM to be fully scalable in all facets of its usage and scope. The implementation of GSM into particle transport codes for their simulations is recommended according to the many benefits outlined previously and due to the significant verification and validation that has taken place regarding GSM throughout its development. GSM is a robust and well-developed software library that clients and end-users may confidently use, within the scope outlined here, as a result of this work.

REFERENCES

[1] S. Mashnik and A. Sierk. *CEM03.03 User Manual*. Tech. rep. LA-UR-12-01364. Los Alamos National Laboratory, 2012.

[2] *Development of the Generalized Spallation Model*. Vol. 117. Computational Tools for Radiation Protection and Shielding. ANS, 2017, pp. 1182–1185.

[3] K. Gudima, S. Mashnik, and A. Sierk. *User Manual for the Code LAQGSM*. Tech. rep. LA-UR-01-6804. Los Alamos National Laboratory, 2001.

[4] S. Mashnik. *MCNP6 High-Energy Event Generators*. Tech. rep. unpublished. Los Alamos National Laboratory, 2017.

[5] S. Mashnik et al. "CEM03.03 and LAQGSM03.03 Event Generators for the MCNP6, MCNPX, and MARS15 Transport Codes". In: *arXiv preprint arXiv:0805.0751* (2008).

[6] D. Filges and F. Goldenbaum. *Handbook of Spallation Research: Theory, Experiments and Applications*. Wiley, 2009.

[7] V. Toneev and K. Gudima. "Particle Emission in Light and Heavy Ion Reactions". In: *Nuclear Physics A* 400 (1983), pp. 173–189.

[8] K. Gudima et al. "The Coalescence Model and Pauli Quenching in High-Energy Heavy-Ion Collisions". In: *Joint Institute for Nuclear Research Report JINR-E2-83-101, Dubna* (1983), pp. 458–460.

[9] K. Gudima, G. Ososkov, and V. Toneev. "Model for Pre-Equilibrium Decay of Excited Nuclei". In: *Yad. Fiz* 21 (1975), pp. 260–272.

[10] V. Toneev S. Mashnik. "MODEX - the Program for Calculation of the Energy Spectra of Particles Emitted in the Reactions of Pre-Equilibrium and Equilibrium Statistical Decays". In: *JINR Communication* P4-8417 (1974).

[11] I. Dostrovsky, Z. Fraenkel, and G. Friedlander. "Monte Carlo Calculations of Nuclear Evaporation Processes. III. Applications to Low-Energy Reactions". In: *Physical Review* 116.3 (1959), p. 683.

[12] F. Atchison. *Spallation and Fission in Heavy Metal Nuclei Under Medium Energy Proton Bombardement*. Tech. rep. Juel-Conf–34. Kernforschungsanlage Juelich, 1980, pp. 17–46.

[13] F. Atchison. "A Treatment of Fission for HETC". In: *Intermediate Energy Nuclear Data: Models and Codes* (1994), pp. 199–218.

[14] E. Fermi. “High Energy Nuclear Events”. In: *Progress of theoretical physics* 5.4 (1950), pp. 570–583.

[15] V. Weisskopf and D. Ewing. “On the Yield of Nuclear Reactions with Heavy Elements”. In: *Physical Review* 57.1 (1940), pp. 472–483.

[16] V. Weisskopf. “Statistics and Nuclear Reactions”. In: *Physical Review* 52.4 (1937), p. 295.

[17] V. Barashenkov and V. Toneev. “Interaction of High Energy Particle and Nuclei with Atomic Nuclei”. In: *Sov. Phys. Usp* 16 (1973), p. 31.

[18] V. Barashenkov et al. “Interaction of Particles and Nuclei of High and Ultrahigh Energy with Nuclei”. In: *Physics-Uspekhi* 16.1 (1973), pp. 31–52.

[19] A. Boudard et al. “Intranuclear Cascade Model for a Comprehensive Description of Spallation Reaction Data”. In: *Physical Review C* 66.4 (2002), p. 044615.

[20] V. Barashenkov, K. Gudima, and V. Toneev. “Intranuclear Cascade Calculation Scheme”. In: *JINR Communication P2-4065, Dubna* (1968).

[21] V. Barashenkov, K. Gudima, and V. Toneev. *Statistical Calculation of Inelastic Collision of Fast Particles with the Intranuclear Nucleons*. Tech. rep. Joint Inst. for Nuclear Research, Dubna (USSR). Lab. of Theoretical Physics, 1968.

[22] M. Baznat, A. Sierk, and R. Preal. “CEM03 and LAQGSM03: Extension of the CEM2k+GEM2 and LAQGSM Codes to Describe Photo-nNclear Reactions at Intermediate Energies (30 MeV to 1.5 GeV)”. In: *Journal of nuclear and radiochemical sciences* 6.2 (2005), A1–A19.

[23] K. Gudima, A. Iljinov, and V. Toneev. “A Cascade Model for Photonuclear Reactions”. In: *JINR Communication P2-4661* (1969).

[24] V. Barashenkov et al. “A Cascade-Evaporation Model for Photonuclear Reactions”. In: *Nuclear Physics A* 231.3 (1974), pp. 462–476.

[25] J. Levinger. “The High Energy Nuclear Photoeffect”. In: *Physical Review* 84.1 (1951), p. 43.

[26] W. Lock and D. Measday. “Intermediate Energy Nuclear Physics”. In: (1970).

[27] S. Mashnik. “User Manual for the Code CEM95”. In: *JINR, Dubna* (1995).

[28] S. Mashnik and A. Sierk. “Improved Cascade-Exciton Model of Nuclear Reactions”. In: *arXiv preprint nucl-th/9812069* (1998).

[29] H. Duarte. “An Intranuclear Cascade Model for High Energy Transport Codes”. In: (1999).

[30] A. Iljinov et al. “Extension of the Intranuclear Cascade Model for Photonuclear Reactions at Energies up to 10 GeV”. In: *Nuclear Physics A* 616.3-4 (1997), pp. 575–605.

[31] S. Lindenbaum and R. Sternheimer. “Isobaric Nucleon Model for Pion Production in Nucleon-Nucleon Collisions”. In: *Physical Review* 105.6 (1957), p. 1874.

[32] S. Mashnik et al. “Improved Intranuclear Cascade Models for the Codes CEM2k and LAQGSM”. In: *AIP Conference Proceedings*. Vol. 769. 1. AIP. 2005, pp. 1188–1192.

[33] H. Bertini. “Low-Energy Intranuclear Cascade Calculation”. In: *Physical Review* 131.4 (1963), p. 1801.

[34] H. Bertini. “Intranuclear-Cascade Calculation of the Secondary Nucleon Spectra from Nucleon-Nucleus Interactions in the Energy Range 340 to 2900 MeV and Comparisons with Experiment”. In: *Physical Review* 188.4 (1969), p. 1711.

[35] K. Gudima, S. Mashnik, and V. Toneev. “Cascade-Exciton Model of Nuclear Reactions”. In: *Nuclear Physics A* 401.2 (1983), pp. 329–361.

[36] S. Mashnik et al. *LAQGSM03.03 Upgrade and its Validation*. Tech. rep. LA-UR-07-6198. Los Alamos National Laboratory, 2007.

[37] S. Mashnik and K. Gudima. “Extension of the LAQGSM03 Code to Describe Photo-Nuclear Reactions up to Tens of GeV”. In: *arXiv preprint nucl-th/0607007* (2006).

[38] N. Amelin and V. Barashenkov. “Intranuclear Cascade with Quark-Gluon Strings”. In: *JINR Preprint P2-83-770, Dubna* (1983).

[39] N. Amelin, K. Gudima, and V. Toneev. “The Quark-Gluon String Model and Ultrarelativistic Heavy-Ion Collisions”. In: *Soviet Journal on Nucl. Phys.* 51 (1990), pp. 327–333.

[40] N. Amelin, K. Gudima, and V. Toneev. “Further Development of the Model of Quark-Gluon Strings for the Description of High-Energy Collisions with a Target Nucleus”. In: *Soviet Journal on Nucl. Phys.* 52 (1990), pp. 172–178.

[41] N. Amelin. “Simulation of Nuclear Collisions at High Energy in the Framework of the Quark-Gluon String Model”. In: *JINR Communication JINR-86-802, Dubna* (1986).

[42] A. Kaidalov. “Quark and Diquark Fragmentation Functions in the Model of Quark-Gluon Strings”. In: *Soviet Journal on Nucl. Phys.* 45 (1987), pp. 902–907.

[43] N. Amelin, V. Barashenkov, and N. Slavin. “Monte-Carlo Simulation of Multiparticle Production in High-Energy Collisions of Hadrons”. In: *Soviet Journal on Nucl. Phys.* 40 (1984), pp. 991–996.

[44] V. Toneev et al. “Dynamics of Relativistic Heavy-Ion Collisions”. In: *Nuclear Physics A* 519.1-2 (1990), pp. 463–478.

[45] N. Amelin, K. Gudima, and V. Toneev. "Ultrarelativistic Nucleus-Nucleus Collisions in a Dynamical Model of Independent Quark-Gluon Strings". In: *Yadernaya Fizika* 51.6 (1990), pp. 1730–1743.

[46] V. Toneev, N. Amelin, and K. Gudima. *The Independent Quark-Gluon String Model for Heavy-Ion Collisions at Ultrarelativistic Energies*. Tech. rep. Gesellschaft fuer Schwerionenforschung mbH, 1989.

[47] NS Amelin, VD Toneev, and KK Gudima. *Quark-Gluon String Model and Ultrarelativistic Heavy-Ion Collisions*. Tech. rep. Joint Inst. for Nuclear Research, 1989.

[48] N. Amelin et al. "Further Development of the Model of Quark-Gluon Strings for the Descripton of High-Energy Collisions with a Target Nuclues". In: *Soviet Journal of Nuclear Physics, USSR* 52.1 (1990), pp. 172–178.

[49] L. Kerby and S. Mashnik. "An Expanded Coalescence Model within the Intranuclear Cascade of CEM". In: *LANL Report, LA-UR-15-20322* (2015).

[50] L. Kerby and S. Mashnik. "Production of Heavy Clusters with an Expanded Coalescence Model in CEM". In: *arXiv preprint arXiv:1501.06488* (2015).

[51] S. Furihata. "The GEM Code Version 2 Users Manual". In: *Mitsubishi Research Institute, Inc., Tokyo, Japan* (2001).

[52] N. Amelin. *Physics and Algorithms of the Hadronic Monte-Carlo Event Generators: Notes for a Developer*. Tech. rep. CERN, 1999.

[53] "Fermi Break-up Simulation for Light Nuclei". In: *GEANT4 Collaboration* (). URL: http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/PhysicsReferenceManual/html/hadronic/FermiBreakup/FermiBreakSim.html.

[54] E Gradsztajn et al. "Intranuclear Cascade and Fermi-Model Breakup Calculations on the Production of Li, Be, and B isotopes in C^{12} by 156-MeV Protons". In: *Physical Review Letters* 14.12 (1965), p. 436.

[55] V. Barašenkov, B. Barbašev, and E. Bubelev. "Statistical Theory of Particle Multiple Production in High Energy Nucleon Collisions". In: *Il Nuovo Cimento (1955-1965)* 7.1 (1958), pp. 117–128.

[56] GI Kopylov. *Principles of Resonance Kinematics*. 1970.

[57] K. Gudima, S. Mashnik, and V. Toneev. "Cascade-Exciton Model of Nuclear Reactions". In: *Nuclear Physics A* 401.2 (1983), pp. 329–361.

[58] L. Kerby. "Precompound Emission of Energetic Light Fragments in Spallation Reactions". PhD thesis. University of Idaho, 2015.

[59] T. Ericson. “The Statistical Model and Nuclear Level Densities”. In: *Advances in Physics* 9.36 (1960), pp. 425–511.

[60] F. Williams. “Intermediate State Transition Rates in the Griffin Model”. In: *Physics Letters B* 31.4 (1970), pp. 184–186.

[61] F. Williams Jr. “Particle-Hole State Density in the Uniform Spacing Model”. In: *Nuclear Physics A* 166.2 (1971), pp. 231–240.

[62] I. Ribanskỳ, P. Obložinskỳ, and E. Běták. “Pre-Equilibrium Decay and the Exciton Model”. In: *Nuclear Physics A* 205.3 (1973), pp. 545–560.

[63] N. Metropolis et al. “Monte Carlo Calculations on IntraNuclear Cascades. I. Low-Energy Studies”. In: *Physical Review* 110.1 (1958), p. 185.

[64] J. Bohr N. Wheeler. “The Mechanism of Nuclear Fission”. In: *Physical Review* 56.1 (1939), pp. 426–450.

[65] Mashnik S. and A. Sierk. *Modeling Fission in the Cascade-Exciton Model*. Tech. rep. LA-UR-98-5998. Los Alamos National Laboratory, 1998.

[66] A. Ignatyuk, G. Smirenkin, and A. Tishin. “Phenomenological Description of the Energy Dependence of the Level Density Parameter”. In: *Yad. Fiz* 21 (1975), pp. 485–490.

[67] G. Smirenkin and A. Tishin. “Fission of Pre-Actinide Nuclei. Excitation Functions for the (α, f) Reactions”. In: *Yad. Fiz* 21 (1975), pp. 1185–1205.

[68] A. Iljinov et al. “Phenomenological Statistical Analysis of Level Densities, Decay Width and Lifetimes of Excited Nuclei”. In: *Nuclear Physics A* 543 (1992), pp. 517–557.

[69] P. Möller et al. “Nuclear Ground-State Masses and Deformations”. In: *arXiv preprint nucl-th/9308022* (1993).

[70] P. Möller, J. Nix, and K. Kratz. “Nuclear Properties for Astrophysical and Radioactive-Ion-Beam Application”. In: *Atomic data and nuclear data tables* 66.2 (1997), pp. 131–343.

[71] Veselsk’y. “Production Mechanism of Hot Nuclei in Violent Collisions in the Fermi Energy Domain”. In: *Nuclear Physics A* 705 (2002), pp. 193–222.

[72] G. Mantzouranis, H. Weidenmüller, and D. Agassi. “Generalized Exciton Model for the Description of Preequilibrium Angular Distributions”. In: *Zeitschrift für Physik A Atoms and Nuclei* 276.2 (1976), pp. 145–154.

[73] C. Kalbach. “Systematics of Continuum Angular Distributions: Extensions to Higher Energies”. In: *Physical Review C* 37.6 (1988), p. 2350.

[74] R. Prael and H. Lichtenstein. “User Guide to LCS: the LAHET Code System”. In: *Group* 10 (1989), p. 6.

[75] S. Furihata. “Statistical Analysis of Light Fragment Production from Medium Energy Proton-Induced Reactions”. In: *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 171.3 (2000), pp. 251–258.

[76] S. Furihata et al. “The GEM Code-a Simulation Program for the Evaporation and the Fission Process of an Excited Nucleus”. In: *Data/Code/Japan atomic energy research institute (Tokyo)* 2001 (2001), p. 015.

[77] S. Furihata. “Development of a Generalized Evaporation Model and Study of Residual Nuclei Production”. PhD thesis. Tohoku University, 2003.

[78] T. Matsuse, A. Arima, and S. Lee. “Critical Distance in Fusion Reactions”. In: *Physical Review C* 26.5 (1982), p. 2338.

[79] A. Botvina et al. “Statistical Simulation of the Break-up of Highly Excited Nuclei”. In: *Nuclear Physics A* 475.4 (1987), pp. 663–686.

[80] G. Audi and A. Wapstra. “The 1995 Update to the Atomic Mass Evaluation”. In: *Nuclear Physics A* 595.4 (1995), pp. 409–480.

[81] P. Haustein. “An Overview of the 1986–1987 Atomic Mass Predictions”. In: *Atomic data and nuclear data tables* 39.2 (1988), pp. 185–200.

[82] A. Cameron. “A Revised Semiempirical Atomic Mass Formula”. In: *Canadian Journal of Physics* 35.9 (1957), pp. 1021–1032.

[83] A. Gilbert and A. Cameron. “A Composite Nuclear-Level Density Formula with Shell Corrections”. In: *Canadian Journal of Physics* 43.8 (1965), pp. 1446–1496.

[84] J. Cook, H. Ferguson, and A. Musgrove. “Nuclear Level Densities in Intermediate and Heavy Nuclei”. In: *Australian Journal of Physics* 20.5 (1967), pp. 477–488.

[85] W. Friedman and W. Lynch. “Statistical Formalism for Particle Emission”. In: *Physical Review C* 28.1 (1983), p. 16.

[86] H. Behrens et al. “The Evaluated Nuclear Structure Data File (ENSDF)”. In: *Verhandlungen der Deutschen Physikalischen Gesellschaft* (1980), pp. 1118–1119.

[87] R. Vandenbosch and J. Huizenga. “Nuclear Fission”. In: *Academic Press, New York* (1973), p. 233.

[88] E. Neuzil and A. Fairhall. “Fission Product Yields in Helium Ion-Induced Fission of ^{197}Au, ^{204}Pb, and ^{206}Pb Targets”. In: *Physical Review* 129.6 (1963), p. 2705.

[89] A. Rusanov, M. Itkis, and V. Okolovich. “Features of Mass Distributions of Hot Rotating Nuclei”. In: *Physics of Atomic Nuclei* 60.5 (1997), pp. 683–712.

[90] M. Itkis, Y. Muzychka, Y. Oganesyan, et al. “Fission of Excited Nuclei with $Z^2/A = 20 - 33$: Mass-Energy Distributions of Fragments, Angular Momentum and Liquid-Drop Model”. In: *Yadernaya Fizika* 58.12 (1995), pp. 2140–2165.

[91] W. Myers and W. Światecki. “Thomas-Fermi Fission Barriers”. In: *Physical Review C* 60.1 (1999), p. 014606.

[92] M. Itkis et al. “Experimental Study of the Mass and Energy Distributions of Fragments from Fission”. In: *Ya. Fiz* 52 (1990), pp. 23–35.

[93] F. Atchison. *A Revised Calculational Model for Fission*. Tech. rep. Paul Scherrer Inst., 1998.

[94] S. Mashnik, K. Gudima, and A. Sierk. “Merging the CEM2k and LAQGSM Codes with GEM2 to Describe Fission and Light-Fragment Production”. In: *arXiv preprint nucl-th/0304012* (2003).

[95] M. Baznat, K. Gudima, and S. Mashnik. “Proton-Induced Fission Cross Section Calculation with the LANL Codes CEM2k+GEM2 and LAQGSM+GEM2”. In: *arXiv preprint nucl-th/0307014* (2003).

[96] S. Mashnik, A. Sierk, and K. Gudima. “Complex Particle and Light Fragment Emission in the Cascade-Exciton Model of Nuclear Reactions”. In: *arXiv preprint nucl-th/0208048* (2002).

[97] L. Kerby, S. Mashnik, and J. Mulvaney. “MCNP6 Updated Proton-Induced Fission Cross Section Calculations at Intermediate Energies”. In: *EPJ Web of Conferences*. Vol. 146. EDP Sciences. 2017, p. 04061.

[98] C. Juneau and L. Kerby. *The Preliminary Validation and Verification of GSM*. Tech. rep. LA-UR-18-25048. Los Alamos National Laboratory, 2018.

[99] S. Mashnik, K. Gudima, and R. Prael. “LAQGSM03.03 Upgrade and Validation”. In: *E-print: arXiv* 709.LA-UR-07-6198 (2007), p. v1.

[100] S. Mashnik. “Validation and Verification of MCNP6 Against Intermediate and High-Energy Experimental Data and Results by Other Codes”. In: *The European Physical Journal Plus* 126.5 (2011), p. 49.

[101] S. Mashnik. *Validation and Verification of MCNP6 Against High-Energy Experimental Data and Calculation by Other Codes. III. The MPI Testing Primer*. Tech. rep. Los Alamos National Laboratory (LANL), 2013.

[102] R. Green, R. Korteling, and K. Jackson. “Inclusive Production of Isotopically Resolved Li through Mg Fragments by 480 MeV p + Ag Reactions”. In: *Physical Review, Part C, Nuclear Physics* 29 (1984), p. 1806. DOI: 10.1103/PhysRevC.29.1806. URL: http://dx.doi.org/10.1103/PhysRevC.29.1806.

[103] S. Leray et al. “Spallation Neutron Production by 0.8, 1.2, and 1.6 GeV Protons on Various Targets”. In: *Physical Review, Part C, Nuclear Physics* 65 (2002), p. 044621. DOI: 10.1103/PhysRevC.65.044621. URL: http://dx.doi.org/10.1103/PhysRevC.65.044621.

[104] L Kerby and G Mashnik. “An Energy-Dependent Numerical Model for the Condensation Probability, γ_j”. In: *Computer Physics Communications* 213 (2017), pp. 29–39.

[105] L. Kerby et al. *Production of Energetic Heavy Clusters in CEM and MCNP6*. Tech. rep. LA-UR-15-29524. Los Alamos National Laboratory, 2015.

[106] A. Budzanowski et al. “Competition of Coalescence and “fireball” Processes in Non-Equilibrium Emission of Light Charged Particles from p + Au Collisions”. In: *Physical Review, Part C, Nuclear Physics* 78 (2008), p. 024603. DOI: 10.1103/PhysRevC.78.024603. URL: http://dx.doi.org/10.1103/PhysRevC.78.024603.

[107] M. Fidelus et al. “Sequential and Simultaneous Emission of Particles from p + Al Collisions at GeV Energies”. In: *Physical Review, Part C, Nuclear Physics* 89 (2014), p. 054617. DOI: 10.1103/PhysRevC.89.054617. URL: http://dx.doi.org/10.1103/PhysRevC.89.054617.

[108] R. Edge, D. Tompkins, and J. Glenn. “Proton Production from Nuclei Bombarded by Protons of 1, 2, and 3 BeV”. In: *Physical Review* 183 (1969), p. 849. DOI: 10.1103/PhysRev.183.849. URL: http://dx.doi.org/10.1103/PhysRev.183.849.

[109] A. Poskanzer, G. Butler, and E. Hyde. “Fragment Production in the Interaction of 5.5-GeV Protons with Uranium”. In: *Physical Review, Part C, Nuclear Physics* 3 (1971), p. 882. DOI: 10.1103/PhysRevC.3.882. URL: http://dx.doi.org/10.1103/PhysRevC.3.882.

[110] Y. Yamanouti et al. *Scattering of* 28.15 *MeV Neutrons from* ^{12}C. Tech. rep. 89. JAERI-M Reports. 1989, p. 177.

[111] S. Pomp et al. “Light-Ion Production in 175 MeV Quasi-Monoenergetic Neutron-Induced Reactions on Iron and Bismuth and Comparison with INCL4 Calculations”. In: *The 12th International Conference on Radiation Shielding (ICRS-12) and the 17th Topical Meeting of the Radiation Protection and Shielding Division of ANS (RPSD-2012)*. Vol. 4. 2012, p. 601.

[112] R. Bevilacqua et al. “Medley Spectrometer for Light Ions in Neutron-Induced Reactions at 175 MeV”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 646.1 (2011), pp. 100–107. URL: http://dx.doi.org/10.1016/j.nima.2011.04.036.

[113] Riccardo Bevilacqua et al. “Study of Pre-Equilibrium Emission of Light Complex Particles from Fe and Bi Induced by Intermediate Energy Neutrons”. In: 312.8 (2011), p. 082013. URL: http://dx.doi.org/10.1088/1742-6596/312/8/082013.

[114] R. Bevilacqua et al. “Neutron Induced Light-Ion Production from Iron and Bismuth at 175 MeV”. In: *Radiation Measurements* 45 (2010), p. 1145. DOI: 10.1016/j.radmeas.2010.06.034. URL: http://dx.doi.org/10.1016/j.radmeas.2010.06.034.

[115] R. Bevilacqua et al. “Light-Ion Production in The Interaction of 175 MeV Neutrons with Iron and Bismuth”. In: *Journal of the Korean Physical Society* 59 (2010), p. 1701. DOI: 10.3938/jkps.59.1701. URL: http://dx.doi.org/10.3938/jkps.59.1701.

[116] R. Bevilacqua et al. “Light-Ion Production from O, Si, Fe and Bi Induced by 175 MeV Quasi-Monoenergetic Neutrons”. In: *Nuclear Data Sheets* 119 (2014), p. 190. DOI: 10.1016/j.nds.2014.08.053. URL: http://dx.doi.org/10.1016/j.nds.2014.08.053.

[117] Y. Watanabe et al. “Light Ion Production in 175 MeV Quasi Mono-Energetic Neutron Induced Reactions on Carbon, Oxygen, and Silicon”. In: *Progress in Nuclear Science and Technology* 4 (2014), p. 569. DOI: 10.15669/pnst.4.569. URL: http://dx.doi.org/10.15669/pnst.4.569.

[118] S. Hirayama et al. “Light-Ion Production from a Thin Silicon Target Bombarded by 175 MeV Quasi Monoenergetic Neutrons”. In: *Journal of the Korean Physical Society* 59.2 (2011), p. 1447. DOI: 10.3938/jkps.59.1447. URL: http://dx.doi.org/10.3938/jkps.59.1447.

[119] J. Franz et al. “Neutron-Induced Production of Protons, Deuterons and Tritons on Copper and Bismuth”. In: *Nuclear Physics, Section A* 510 (1990), p. 774. DOI: 10.1016/0375-9474(90)90360-X. URL: http://dx.doi.org/10.1016/0375-9474(90)90360-X.

[120] A. Rudchik et al. “Isotopic effects in the ^{7}Li + ^{10,11}B elastic and inelastic scattering”. In: *European Physical Journal A: Hadrons and Nuclei* 33 (2007), p. 317. DOI: 10.1140/epja/i2007-10483-5. URL: http://dx.doi.org/10.1140/epja/i2007-10483-5.

[121] V. Kondratev et al. “Investigation of Energetic Particle Spectra with Z=1,2, Resulting from the Interaction of Deuterons with ^{58}Ni and ^{64}Ni Nuclei”. In: *Izv. Rossiiskoi Akademii Nauk, Ser.Fiz.* 49.1 (1985), p. 147.

[122] C. Juneau and L. Kerby. *The Development of the Generalized Spallation Model*. Tech. rep. LA-UR-17-29889. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2017.

[123] J. Gosset et al. “Central Collisions of Relativistic Heavy Ions”. In: *Physical Review, Part C, Nuclear Physics* 16 (1977), p. 629. DOI: 10.1103/PhysRevC.16.629. URL: http://dx.doi.org/10.1103/PhysRevC.16.629.

[124] V. Strutinskii. “The Form of the Fissioning Nucleus at the Saddle Point and the Liquid Drop Nuclear Model”. In: *Yadern. Fiz.* 1 (1965).

[125] H. Krappe, J. Nix, and A. Sierk. “Unified Nuclear Potential for Heavy-Ion Elastic Scattering, Fusion, Fission, and Ground-State Masses and Deformations”. In: *Physical Review C* 20.3 (1979), p. 992.

[126] A. Gavron et al. “New Evaluation of Fission-Fragment Angular Distributions in Heavy-Ion Reactions”. In: *Physical Review Letters* 52.8 (1984), p. 589.

[127] A. Sierk. “Macroscopic Model of Rotating Nuclei”. In: *Physical Review C* 33.6 (1986), p. 2039.

[128] M. Shapiro. In: *Phys. Rev.* 90 (1953), p. 171.

[129] J. Blatt and V. Weisskopf. *Theoretical Nuclear Physics*. Courier Corporation, 1991.

Appendix A. Known Bugs in GSM and its Sub-Models

A list of known and discovered coding errors contained by GSM and its sub-libraries is listed here. A short list of recommended improvements are also included here to aid developers in the further improving GSM and its sub-libraries. Note that numerics were parameterized and made more precise, *e.g.* `half=0.5_real64`, and nuclear rest masses were improved where possible. No known coding errors were discovered or are known to exist in the standard DCM, modified DCM, coalescence, and the Fermi break-up sub-libraries. Note however that the modified DCM allows residual nuclei to take the form of small neutron clusters ($A \leq 12$) post-IntraNuclear Cascade simulation. The presence of neutron clusters was found to yield out-of-bounds array access in the preequilibrium data object. The current resolution is to provide an approximation where the residual nucleus is assumed to contain one proton.

GSM relies on several general modules for its simulations, being mainly fundamental data, a Coulomb barrier module, and inverse cross sections. The following bug(s) exist, or did exist, within these modules:

1. Inconsistent radius data type approximation:

   (a) The radius parameter utilized in the equation where the nuclear radius, R_A, is estimated as $R_A = r_0 A^{1/3}$, utilized an inconsistent data type for the r_0 parameter. The parameter was originally equal to one, however it was intended to equal 1.25. This was resolved by changing the data type to *real64*.

The preequilibrium model contains within it several known coding errors, being outlined as follows.

1. Allowed unphysical decay channel:

   (a) Quantities for all considered particle emission channels is performed utilizing an effectively parameterized reduced mass calculation. When the reduced mass is that of an

unphysical reaction, where baryon and charge numbers are not conserved, the reduced mass is approximated instead of flagging the reaction as forbidden. No solution has been implemented presently.

2. Inconsistent preequilibrium probability equation:

   (a) The preequilibrium model is quoted to estimate preequilibrium particle emission using Eqn. II.61. The preequilibrium model currently estimates particle emission according to the following equation, where the exponential term differs from that quoted slightly.

$$P_{pre} = 1 - e^{-1/2\left(\frac{n/n_{eq}-1}{\sigma}\right)^2}$$

   The inconsistency may provide significantly different results. The proper equation has been provided in the code, however it is not utilized by the model as significant validation could not take place.

The evaporation model was discovered to contain several coding errors as well. The discovered coding errors are enumerated below:

1. Incorrect recalculation of data:

   (a) The data utilized by the evaporation model was calculated for each reaction twice, the second time being calculated incorrectly. It was verified in the evaporation model of LAQGSM to be incorrectly calculated the second time. This coding error was resolved with the implementation of the evaporation data class.

2. False flagging of fission:

   (a) The evaporation model would flag fission as having occurred when valid fission fragment sampling failed. This was corrected for, and the allowed sampling attempts was increased to reduce the frequency of sampling failure. Additionally, neutron emission is resampled upon fission sampling failure instead of flagging fission as having occurred.

3. Fission fragments are now allowed to undergo Fermi break-up:

    (a) Fission fragments previously were not allowed to undergo Fermi break-up. Fission fragments that occur as a result of very asymmetric fission may now undergo Fermi break-up.

Several coding errors were discovered in GSM regarding how statistics are counted for its simulations. The discovered coding errors are as follows:

1. Unsampled near-stable residual nuclei:

    (a) GSM previously did not sample statistics on residual nuclei with low excitation energy. This was resolved.

2. Inconsistent compound sampling:

    (a) GSM obtained statistics on compound nuclei that underwent Fermi break-up during the evaporation process. The compound nuclei, in undergion Fermi break-up, have been disintegrated and thus valid statistics cannot be obtained on these nuclei. Statistics are no longer obtained on these nuclei.

3. Invalid cross section sums:

    (a) It was seen during rare simulations of GSM that the sum of inelastic and elastic cross sections in GSM did not equal to quoted total. This phenonema may have been due to error protection or other protections, however it is noted that the total cross section is not verified to match that of the summed individual components.

4. Inconsistent residual nuclei statistics:

    (a) Statistics on post-IntraNuclear Cascade nuclei, or residual nuclei, are obtained prior to preequilibrium emission only. Instead, statistics ought to be obtained on all post-IntraNuclear Cascade residual nuclei, despite the applicability of preequilibrium physics.

Below are several enumerated suggested improvements within GSM and its sub-models. Note many of these improvements provide greater flexibility to the models' clients and to the models themselves, without modifying the internal algorithms utilized for the models' simulation themselves.

1. Incorporate a light target modification:

   (a) A version of CEM, the base event generator upon which GSM is constructed, exists that includes a light target modification. This version of CEM exists at present in MCNP6, allowing targets of same size or lighter than ^{4}He. Exclusion of this feature at present restricts GSM from performing simulations of incident particles on light nuclei.

2. Create an explicit data API:

   (a) The majority of the GSM sub-libraries directly access their data module's variables. The models should access this data via function calls to query the data, regardless of being parameterized, where the models' respective data module returns a numerical value based on the validity of the query to ensure array limits are not exceeded.

3. Parse coalescence radii:

   (a) The coalescence model currently utilizes a single coalescence radius for all light fragments to utilize. The compounding of specific light fragments may be better tailored by utilizing unique coalescence radii for ^{6}He, ^{6}Li, ^{7}Li, and ^{7}Be nuclei. The increased flexibility additionally allows clients to tailor light fragment formation to their simulations based on IntraNuclear Cascade distributions.

4. Create and utilize a random number generator (RNG) class:

   (a) Presently, each GSM sub-model utilizing a random number generator (RNG) requires its client to pass in a pointer to the client's RNG procedure. It is recommended to upgrade the RNG presently in GSM to provide a RNG class. The RNG class ought

to primarily utilize a procedure that will either use its own, internal RNG procedure or use a client-provided procedure. Implementation will provide all GSM sub-models, including itself, with a more robust RNG, simplifies client usage, and improves the overall scalability of the model itself. In this manner, the sub-models call to the RNG object's generator, pointing to the procedure that will either call a client specified RNG or call to its own RNG procedure. Computational efficiency of the RNG is anticipated to not be affected by this modification.

5. Create and utilize an *I/O* class:

   (a) GSM utilizes an *I/O* object, containing within it a message handling procedure and a character array to contain the message. The *I/O* object ought to be specified external to all of GSM and its sub-models to allow the models to use the object. This improvement does not modify the generation and usage of the message handling procedure, however it does provide flexibility to clients by allowing them to specify unique message handling procedures and unique verbose limitations on a model by model basis. Currently, all of GSM and its sub-models utilize a global verbose limit. The proposed improvement aids in the flexibility of each model's message generation, usage, and potential modifications.

6. Generate messages for potential simulation completion failures:

   (a) GSM and its sub-models does not presently generate messages when client-provided arguments may not be sufficient, such as for small arrays, when performing a simulations. To aid clients in their simulations, it is recommended to incoporate warnings to users in instances where the provided arrays, namely progeny arrays, may be exceeded during the simulation.

7. Update and utilize the Coulomb barrier module:

   (a) It is recommended to improve and propagate the Coulomb barrier module throughout all of GSM's sub-libraries, namely for the Fermi break-up, preequilibrium, and evaporation

sub-libraries. A Coulomb barrier object ought to be created for added flexibility in each of GSM's sub-libraries.

8. Allowing asymmetric fission products to fission:

   (a) Fission products currently flagged when they are simulated as undergoing fission, as the evaporation model is uncapable to doing so at present. The evaporation model should instead provide a more robust evaporation procedure, whereby a provided nucleus is evaporated and all progeny added to a progeny array. In this manner, fission products may also utilize the same procedure for the fission product's further evaporation. Note that fissioning fission products, although extremely rare, may occur during highly asymmetric fission. This improvement simplifies client usage and provides greater flexibility and robustness to its clients.

9. Expand the Fermi break-up data sets:

   (a) The Fermi break-up data, at present, contains data for only $A_{frag} \leq 12$. The Fermi break-up model allows clients to simulate Fermi break-up physics on larger nuclei, whereby their disintegration may rarely require data outside of these limits. Expanding the size of the data sets utilized will allow the Fermi break-up object to provide its clients with more precise and robust simulation results.

10. Sample fragment emission angle smoothly:

    (a) The preequilibrium model samples the fragment emission angle differently for particles with incident energy greater than 210 MeV. A sampling ought to be done such that a smooth transition between the two sampling methods occurs for reactions with particles having incident energies within some threshold of 210 MeV.

11. Utilize post-IntraNuclear Cascade fission parameter scaling:

(a) The GSM currently provides scaling of the a_f and C_Z fission parameters by utilizing the original nuclei considered for evaporation. It is thought that more appropriate results may be obtained when providing these scaling parameters on post-IntraNuclear Cascade residual nuclei, allowing the evaporation model to utilize more precise scalings for a given particle's evaporation. Testing of this has not been done, as the fission parameter scaling would need to be re-parameterized and compared to previous scaling parameters.

12. Improve kinetic energy sampling of evaporation fragments:

(a) The evaporation model utilizes a poor sampling procedure to simulate the kinetic energy distribution of emitted fragments. Current sampling efficiency requires a larger memory footprint than necessary and has a very poor sampling efficiency, whereas providing an improvement could produce sampling efficiency near 90% or better.

Appendix B. Documentation for the Molnix Class

The Molnix object is utilized as a data class interface for clients to simply query various energy values of nuclei from. The Molnix object allows clients to determine ground-state microscopic shell corrections [69], experimental, or theoretical if not measured, mass excesses [69], and energy shifts from the Moller, Nix, and Kratz pairing gaps [70]. The Molnix object utilizes an object-oriented design, being written in modern Fortran (Fortran 2008). The details of the Molnix code are laid out here for the purposes of future development and maintenance, and in addition for ease of client use and API consumption. The documentation presented here is written primarily as an aid to users in developing their own interfaces for consumptions of the Molnix API.

B.1 BUILD REQUIREMENTS

The Molnix object can be built by compiling and linking the modules shown in Table II.I. Table II.I states the purpose of each of the modules within the context of the Molnix object, and in addition a recommended build order. Clients utilizing the Molnix sub-library ought to only need access to the object's module itself for full usage.

TABLE II.I. Molnix Modules

Module Name	Description	Build Order
molnixParams	Parameter module for the Molnix class	1
molnixClass	Contains the Molnix class itself	2

The Molnix object does not have any known dependencies on external libraries, being self-contained except for the usage of the intrinsic `iso_fortran_env` module.

Table II.II details the minimum required Fortran compiler version that the Molnix class is supported by in its current form. Note the Molnix object was originally developed using the GNU compiler. Building the Molnix object requires clients and user to utilize the minimum version

shown in Table II.II. Numerical differences in simulation results may be observed between different builds, in compiler and version, of the Molnix object. Potential differences in numerical results were not tested.

TABLE II.II. Compiler Support for the Molnix Class

Compiler	Compiler Version
GNU	*N/A*
Intel	Not tested
IBM XL	Not tested
g95	*N/A*
NAG	Not tested
PGI	Not tested

B.2 PRIMER

The Molnix code was designed to be simple to use. Clients of the Molnix code are not required to construct the object, however construction is recommended. Clients may simply construct the Molnix object and query its data as needed. The Molnix object has several interfaces that clients may utilize, namely `massExcess`, `shellEnergy`, `pairingGap`, and `defineEnergy`, being a simple combination of the three previously mentioned interfaces. Clients may also query the state of the Molnix object and some of the data contained within the Molnix module via the Molnix object. The Molnix object does not require proper construction to query its data. The Molnix object may optionally be constructed by passing in a `molnixOptions` object, a subroutine pointer that handles the printing of all messages, or both.

Fig. B.1 details a sample API for clients, such as GSM, using the Molnix object for their simulations. It is recommended that the Molnix object is constructed one time only for computational efficiency within the respective clients. Note that Fig. B.1 demonstrates default construction as well as construction when providing the Molnix constructor with all optional arguments. Details on the Molnix API are further discussed in the following sections.

```
subroutine molnixAPIExample(nucleusA, nucleusZ, myIOPtr)

   use, intrinsic:: iso_fortran_env, only: int32, real64
   use molnixClass, only: &
        & Molnix,     &   ! Molnix object
        & newMolnix,  &   ! Molnix constructor
        & molnixOptions   ! Molnix behaviors controller

   implicit none
   real(real64),          intent(in   ) :: nucleusA
   real(real64),          intent(in   ) :: nucleusZ
   procedure(IOHANDLER), intent(in   ), pointer :: myIOPtr

   ! Initialize Molnix (default):
   type(Molnix) :: molObj = Molnix()

   ! Re-initialize Molnix (with optional arguments):
   type(molnixOptions) :: molOps
   type(Molnix) :: molObj = Molnix( &
        & clientOptions = molOpts,  &   ! Behaviors controller
        & clientIO      = myIOPtr,  &   ! Message handling procedure
        & )

   ! Query from Molnix object (using interfaces):
   real(real64) :: shellGap   = molObj%shellEnergy(nucleusA, nucleusZ)
   real(real64) :: massExcess = molObj%massExcess(nucleusA, nucleusZ)
   real(real64) :: pairingGap = molObj%pairingGap(nucleusA, nucleusZ)

   ! Query from Molnix object (using "defineEnergy"):
   !    NOTE: These are the same as above!
   integer(int32) :: A = nint(nucleusA)
   integer(int32) :: N = A - nint(nucleusZ)
   real(real64) :: shellGap   = molObj%defineEnergy(A, N, 1)
   real(real64) :: massExcess = molObj%defineEnergy(A, N, 2)
   real(real64) :: pairingGap = molObj%defineEnergy(A, N, 3)

   ! Query the state of the Molnix object:
   if ( .not.molObj%properlyConstructed() ) then
      write(*,*) "The Molnix object was not constructed properly"
   end if
   ! Update options they represent those used by the object
   !    NOTE: This is useful when invalid options are provided to the object
   molOpts = molObj%queryOptions()

   return
end subroutine molnixAPIExample
```

FIGURE B.1. Sample API for the Molnix object.

B.3 DATA STRUCTURES

The Molnix object utilizes several data structures for its interfacing and architecture to aid in modulation and future upgrades to the model. Data types used by the Molnix object are shown in Table II.III, describing the purpose and use of the data object.

TABLE II.III. Data Types Used by the Molnix Object

Name	Client Access	Purpose
molnixOptions	Public	Controls behavior of Molnix object
molnixIO	Private	Controls message output from the Molnix object

The *molnixOptions* options data type is used by the Molnix object to control its interface behaviors. The *molnixOptions* data type, detailed by Table II.IV, is used to control the pairing energy gap scaling factor for theoretical values where measurements have not been made.

TABLE II.IV. The *molnixOptions* Data Type

Member Name	Default	Controlling Behavior
cevap	12.0,*eal*64	Controls the theoretical pairing gap scaling factor

The *molnixIO* data type, detailed in Table II.V, is used internally to the Molnix object to control the writing and printing of messages. The Molnix object currently uses a *write* statement to populate the desired message, and then calls a message printing message procedure to write the message depending on the message's flagged importance and the client's desired verbosity flag. Most generated messages by the Molnix object consist of approximation statements, where it returns the nearest valid value queried to ensure the bounds of the data arrays are not exceeded.

TABLE II.V. The *molnixIO* Data Type

Member Name	Type	Description
message	character(LEN=512)	Contains the message to be printed
print	procedure(IOHANDLER)	Points to the message printing procedure

B.4 MODEL CONSTRUCTION

The Molnix object does not require that it be properly constructed prior to simulation. The Molnix object constructor, *newMolnix*, accommodates two optional arguments. Clients may optionally pass in a *molnixOptions* data type to set the behavior of the Molnix object, particularly for the scaling factor on pairing gap energies. The Molnix object will utilize the default *molnixOptions* data type if not specified by the client. The second optional argument that client programs may utilize is a procedure pointer that points to a message handling subroutine. Fig. B.1 demonstrates construction of a Molnix object when using the *molnixOptions* optional argument as well as a message handling procedure pointer. Clients may construct the Molnix object with or without use of one or both of these arguments. The message printing procedure has the interface shown in Fig. III.6 for reference. Note that the object's construction state can be verified by calling the *properlyConstructed* function of the Molnix object, returning a logical flag indicating if the Molnix object was successfully constructed or not. There are no conditions at present that prevent the Molnix object from being properly constructed, when invoked by clients, as the constructor handles all errors it encounters.

B.5 USAGE AND RESULTS

Clients of the Molnix object may utilize it to obtain shell correction energies [MeV], mass excesses [MeV], and pairing gap energies [MeV]. Clients may additionally query the state of the Molnix object as desired. Fig. B.1 demonstrates the usage of these various procedures available to the Molnix object's client(s).

The Molnix object, including the module it is housed in, highly limits its clients' access to the object's procedures. Clients have access to a number of query procedures as well as the various intefacing procedures. The `shellEnergy` function, requiring two `real64` data types representing the number of baryons and the number of protons within a nucleus, provides clients with ground-state microscopic shell correction energy according to Ref. [69], with units of [MeV].

The `massExcess` function, requiring two `real64` data types representing the number of baryons and the number of protons within a nucleus, provides clients with the measured, or theoretical when not measured, mass excess of a nucleus according to Ref. [69], with units of [MeV]. The `pairingGap` function, requiring two `real64` data types representing the number of baryons and the number of protons within a nucleus, provides clients with the energy shift due to the pairing gap of a nucleus according to Ref. [70], with units of [MeV]. Clients may also query the state of the Molnix object if desired. The available query functions provided by the Molnix object include `properlyConstructed` and `queryOptions`. Clients may also minimally query the scope of the Molnix data contained within the module via the Molnix class. The appropriate query functions include `nmina`, `nmaxa`, `nmin`, and `nmax`, where in `int32` data type representing the number of protons in a nucleus is provided to determine the limits on the minimum and maximum scope of the Molnix data.

B.6 TIPS, WARNINGS, AND LIMITATIONS

It is recommended to clients of the Molnix object that they be as strict as possible regarding the use of the model. Clients are recommended to `save` the object, or contain it within a module, when performing long simulations that may utilize the object more than once to save on computational efficiency. The object was designed such that it would contain as minimal a state as possible, for the purpose of both memory utilization and parallelization. Clients are additionally recommended to utilize all optional arguments in the model constructor to provide clients with full control of the model. Clients are recommended to use and provide the Molnix constructor with the *molnixOptions* type, in addition to querying the resulting object, to verify and validate all options utilized by the model. Clients, given the information about the message handling procedure pointer in Fig. III.6, may easily create an interface to their message handling procedure, if present. Clients can utilize this feature of the model to prepend information to the message, such as flagging the message as one from the Molnix object, during simulations and to control where the resulting messages are

sent, for example. The verbosity utilized within the Molnix module, `molnixVerbose`, may also be modified by clients to increase or limit the messages printed by all Molnix objects utilizing the default message handling procedure.

Appendix C. Documentation for the Fission Barrier Class

The Fission Barrier object was originally developed in the early to mid-1980s, being based on Ref. [124–127], and later implemented and expanded for use in the CEM2k event generator, the first version of the CEM event generator. The Fission Barrier object provides its clients with ground state and fission barrier energies [MeV]. Clients of the fission barrier object may obtain a nucleus's fission barrier [MeV], rotating ground state energies [MeV], and saddle-point fission barrier energies [MeV]. The Fission Barrier object utilizes an object-oriented design, being written in modern Fortran (Fortran 2008).

The details of the Fission Barrier code are laid out here for the purposes of future development and maintenance, and in addition for ease of client use and API consumption. The documentation presented here is written primarily as an aid to users in developing their own interfaces for consumptions of the Fission Barrier API.

C.1 BUILD REQUIREMENTS

The Fission Barrier object can be built by compiling and linking the modules shown in Table III.I. Table III.I states the purpose of each of the modules within the context of the Fission Barrier object, and in addition a recommended build order. Clients utilizing the Fission Barrier sub-library ought to only need access to the object's module itself for full usage.

TABLE III.I. Fission BarrierModules

Module Name	Description	Build Order
fissionBarrierParams	Parameter module for the Fission Barrier class	1
fissionBarrierClass	Contains the Fission Barrier class itself	2

The Fission Barrier object depends on the intrinsic `iso_fortran_env` module as well as the Molnix class. Documentation for the Molnix class may be found in Appendix B.

Table III.II details the minimum required Fortran compiler version that the Fission Barrier class is supported by in its current form. Note the Fission Barrier object was originally migrated to an object-oriented approach using the GNU compiler. Building the Fission Barrier object requires clients and end-users to utilize the minimum version shown in Table III.II, if shown. Numerical differences in simulation results may be observed between different builds, in compiler and version, of the Fission Barrier object. Potential differences in numerical results were not tested.

TABLE III.II. Compiler Support for the Fission Barrier Class

Compiler	Compiler Version
GNU	*N/A*
Intel	Not tested
IBM XL	Not tested
g95	*N/A*
NAG	Not tested
PGI	Not tested

C.2 PRIMER

The Fission Barrier code was designed to be simple to use. Clients of the Fission Barrier code are required to construct the object for it to properly function as its internal Molnix object points to that specified by the client. Upon being constructed, clients simply utilize the procedures provided by the generated Fission Barrier object to obtain the requested data or queired state of the object. The Fission Barrier class has several procedures that clients may utilize, namely `bf` to obtain fission barriers [MeV], `barfit` to obtain the fission barrier [MeV] and rotating ground state energy [MeV], and `bsfit` to obtain an approximation to the value of the surface area of the macroscopic saddle-point shape, with associated units to that of the spherical nucleus. The Fission Barrier object also allows clients to query the state of the objectas if desired. The Fission Barrier object requires construction to query utilize its `bf` and `barfit` procedures. The Fission Barrier object may be constructed by passing in a `Molnix` object and, optionally, a `fissionBarrierOptions` object, a subroutine pointer that handles the printing of all messages, or both. Failure to construct

the object will result in fission barrier, ground state energy, or both, to fail at retrieval.

Fig. C.1 details a sample API for clients, such as GSM, using the Fission Barrier object for their simulations. It is recommended that the Fission Barrier object is constructed one time only for computational efficiency within the respective clients. Note that Fig. C.1 demonstrates default construction as well as construction when providing the Fission Barrier constructor with all optional arguments. Details on the Fission Barrier API are further discussed in the following sections.

C.3 DATA STRUCTURES

The Fission Barrier object utilizes several data structures for its procedures and architecture to aid in modulation and future upgrades to the model. Data types used by the Fission Barrier object are shown in Table III.III, describing the purpose and use of the data object.

TABLE III.III. Data Types Used by the Fission Barrier Object

Name	Client Access	Purpose
fissionBarrierOptions	Public	Controls behavior of Fission Barrier object
fissionBarrierIO	Private	Controls message output from the Fission Barrier object

The *fissionBarrierOptions* options data type is used by the Fission Barrier object to control its interface behaviors. The *fissionBarrierOptions* data type, detailed by Table III.IV, is used to control the approximate nuclear radius scaling factor.

TABLE III.IV. The *fissionBarrierOptions* Data Type

Member Name	Default	Controlling Behavior
r0m	1.2,*eal*64	Controls the scaling for approximate nuclear radii, where $r_A \approx r0mA^{1/3}$

The *fissionBarrierIO* data type, detailed in Table III.V, is used internally to the Fission Barrier object to control the writing and printing of messages. The Fission Barrier object currently uses a *write* statement to populate the desired message, and then calls a message printing message proce-

```
subroutine fissionBarrierAPIExample(nucleusA, nucleusZ, angularMom, molObj,
    myIOPtr)

    use, intrinsic:: iso_fortran_env, only: int32, real64
    use molnixClass, only: Molnix
    use fissionBarrierClass, only: &
         & FissionBarrier,     &   ! Object
         & newFissionBarrier,  &   ! Constructor
         & fissionBarrierOptions   ! Behavior Handler

    implicit none
    real(real64),          intent(in   ) :: nucleusA
    real(real64),          intent(in   ) :: nucleusZ
    integer(int32),        intent(in   ) :: angularMom
    class(Molnix),         intent(in   ) :: molObj
    procedure(IOHANDLER), intent(in   ), pointer :: myIOPtr

    ! Construct Fission Barrier object (default):
    type(FissionBarrier) :: fissBarrObj = newFissionBarrier( molObj )

    ! Reconstruct Fission Barrier object (all arguments):
    type(FissionBarrier) :: fissBarrObj = newFissionBarrier( molObj, &
         & clientOptions = fbOps,     &
         & clientIO      = myIOPtr,   )

    ! Query the state of the created object:
    if( .not.fissBarrObj%properlyConstructed() ) then
       write(*,*) "The Fission Barrier object failed to construct."
       return
    end if
    fbOpts = fissBarrObj%queryOptions()

    ! Declare useful variables:
    real(real64) :: fissBarrier = 0.0_real64, groundState = 0.0_real64,
       saddlePoint = 0.0_real64

    ! NOTE: The "groundState" variable is updated with this function call
    fissBarrier = fissBarrObj%bf(nucleusA, nucleusZ, angularMom, groundState)

    ! Updates both "groundState" and "fissBarrier" variables
    fissBarrObj%barfit(nucleusA, nucleusZ, angularMom, fissBarrier,
       groundState)

    ! Obtain saddle point energy:
    fissBarrObj%bsfit(nucleusA, nucleusZ, angularMom, saddlePoint)

    return
end subroutine fissionBarrierAPIExample
```

FIGURE C.1. Sample API for the Fission Barrier object.

dure to write the message depending on the message's flagged importance and the client's desired verbosity flag. Most generated messages by the Fission Barrier object consist of approximation statements, where it returns the nearest valid value queried to ensure the bounds of the data arrays are not exceeded.

TABLE III.V. The *fissionBarrierIO* Data Type

Member Name	Type	Description
message	character(LEN=512)	Contains the message to be printed
print	procedure(IOHANDLER)	Points to the message printing procedure

C.4 MODEL CONSTRUCTION

The Fission Barrier object requires that it be properly constructed prior to usage in simulations. The Fission Barrier object constructor, *newFissionBarrier*, requires one argument and accommodates up to two optional arguments. The Fission Barrier object requires clients to provide its constructor with a `Molnix` object, regardless of its construction state, as the Fission Barrier object stores a pointer to the provided Molnix object. Clients may, at their discretion, pass in a *fissionBarrierOptions* data type to set the behavior of the Fission Barrier object, particularly for the scaling factor of approximate nuclear radii The Fission Barrier object will utilize the default *fissionBarrierOptions* data type if not specified by the client. The second optional argument that client programs may utilize is a procedure pointer that points to a message handling subroutine. Fig. C.1 demonstrates construction of a Fission Barrier object when using the *fissionBarrierOptions* optional argument as well as a message handling procedure pointer. Clients may construct the Fission Barrier object with or without use of one or both of these arguments. The message printing procedure has the interface shown in Fig. III.6 for reference. Note that the object's construction state can be verified by calling the *properlyConstructed* function of the Fission Barrier object, returning a logical flag indicating if the Fission Barrier object was successfully constructed or not. There are no conditions at present that prevent the Fission Barrier object from being properly constructed, when

invoked by clients, as the constructor handles all errors it may encounter.

C.5 USAGE AND RESULTS

Clients of the Fission Barrier object may utilize it to obtain rotating ground state energies [MeV], fission barrier heights [MeV], and saddle-point fission energies [MeV] of nuclei. Clients may additionally query the state of the Fission Barrier object as desired. Fig. C.1 demonstrates the usage of the various procedures available to the Fission Barrier object's client(s). The Fission Barrier object, including the module it is housed in, highly limits its clients' access to the object's procedures. Clients have access to a number of query procedures as well as the various defined procedures. The function `bf` function returns to its function call the fission barrier [MeV] of a nucleus. The function requires clients to provide it with `real64` data types of the specific nucleus's baryon and proton numbers, an `int32` data type of the nucleis's angular momentum [units of $\hbar$], and a `real64` data type representing the ground state energy of the nucleus, being modified by the Fission Barrier object in the function call. The subroutine `barfit` requires clients to provide it with `real64` data types of the specific nucleus's baryon and proton numbers, an `int32` data type of the nucleis's angular momentum [units of $\hbar$], and `real64` data types representing the fission barrier height and ground state energies [MeV], respectively, of the nucleus that are established in the subroutine call. The subroutine `bsfit` returns to its calling procedure the approximate surface area of the macroscopic saddle-point shape, in units of the area of the spherical nucleus. The `bsfit` subroutine requires clients to provide it with `real64` data types of the specific nucleus's baryon and proton numbers, an `int32` data type of the nucleis's angular momentum [units of $\hbar$], and a `real64` data type in which to store the result. Clients may also query the state of the Fission Barrier object if desired. The available query functions provided by the Fission Barrier object include `properlyConstructed`, returning a logical flag stating the construction state of the object, `queryOptions`, returning to the client the `fissionBarrierOptions` data type utilized by the object, and `queryMolnix`, returning a pointer to the object's Molnix object.

C.6 TIPS, WARNINGS, AND LIMITATIONS

It is recommended to clients of the Fission Barrier object that they be as strict as possible regarding the use of the model. Clients are recommended to `save` the object, or contain it within a module, when performing long simulations that may utilize the object more than once to save on computational efficiency in addition to saving locally or globally all dependencies of the Fission Barrier object provided to it. The object was designed such that it would contain as minimal a state as possible, for the purpose of both memory utilization and parallelization, and at the same time being highly flexible for client usage. Clients are additionally recommended to utilize all optional arguments in the model constructor to provide clients with full control of the model. Clients are recommended to use and provide the Fission Barrier constructor with the *fissionBarrierOptions* type, in addition to querying the resulting object, to verify and validate all options utilized by the model. Clients, given the information about the message handling procedure pointer in Fig. III.6, may easily create an interface to their message handling procedure, if present. Clients can utilize this feature of the model to prepend information to the message, such as flagging the message as one from the Fission Barrier object, during simulations and to control where the resulting messages are sent, for example. The verbosity utilized within the Fission Barrier module, `fissBarrVerbose`, may also be modified by clients to increase or limit the messages printed by all Fission Barrier objects utilizing the default message handling procedure.

Appendix D. Documentation for the GSM

GSM simulates high-energy spallation physics, as discussed in Chapter II. Information on the physics simulated may be found there. The documentation here is very basic and minimal and ought to be expanded upon, however the essential components of GSM are documented. The GSM utilizes an object-oriented design, being written in modern Fortran (Fortran 2008).

The details of the GSM code are laid out here for the purposes of future development and maintenance, and in addition for ease of client use and API consumption. The documentation presented here is written primarily as an aid to users in developing their own interfaces for consumptions of the GSM API.

D.1 BUILD REQUIREMENTS

The GSM can be built by compiling and linking the modules shown in Table IV.I. Table IV.I states the purpose of each of the modules within the context of the GSM, and in addition a recommended build order. Clients utilizing the GSM ought to only access the target characterization data class and the GSM class itself.

TABLE IV.I. GSM Modules

Module Name	Description	Build Order
gsm_params	Parameter module for the GSM data class	1
gsm_derived_types	Contains the *gsmNucleus* data type	2
gsm_options	To be deprecated	3
defaults_mod	To be deprecated	4
generalizedSpallationData	Contains data and its initialization for GSM	5
generalizedSpallationClass	Contains the GSM class	6

The GSM object depends on many sub-libraries, all being developed in an object-oriented, modern Fortran architecture for the purpose of being consumed by GSM and potential other clients.

Table IV.II details the minimum required Fortran compiler version that the GSM is supported by in its current form. Note the GSM object was originally developed using the GNU compiler. Building the GSM object requires clients and user to utilize the minimum version shown in Table IV.II. Numerical differences in simulation results may be observed between different builds, in compiler and version, of the GSM object. Potential differences in numerical results were not tested.

TABLE IV.II. Compiler Support for the Standard Dubna Cascade Model

Compiler	Compiler Version
GNU	*N/A*
Intel	Not tested
IBM XL	Not tested
g95	*N/A*
NAG	Not tested
PGI	Not tested

D.2 DATA INITIALIZATION

Sec. IV.1 contains the information on GSM data intialization. See there for details.

D.3 DATA STRUCTURES

The GSM utilizes several data structures for its simulations and architecture to aid in modulation and future upgrades to the model, in addition to using those of its sub-models to store simulation behaviors. Data types used by the GSM object are shown in Table IV.III, describing the purpose and use of the data object. The *gsmProjectile*, *gsmTarget*, *gsmResults*, and *gsmOutput* are details in Chapter IV, particularly in Table IV.I, Table IV.II, Table IV.III, and Table IV.IV. Note also that the data types not accessible to clients or end-user of GSM are not details here for brevity, as they are utilized only internal to any simulation performed by GSM.

The *gsmOptions* options data type is used by the GSM and its sub-model objects to control all

TABLE IV.III. Data Types Used by the GSM Object

Name	Client Access	Purpose
gsmOptions	Public	Controls behavior of GSM object
gsmProjectile	Public	Contains all information available for an incident particle
gsmTarget	Public	Contains all information available for a target nucleus
gsmProgeny	Public	Contains information of a single tracked progeny
gsmResults	Public	Container for all results of a simulation
gsmOutput	Public	Controls the information contained in the output file
eventRestarts	Public	Tracks the number of times an event restarted for a condition (not used currently)
gsmResidual	Public	Contains all information of tracked residuals during the simulation
gsmModelUsage	Public	Counts the number of uses for each sub-model and an event's last utilized model (not used currently)
generalPhysicsData	Private	Contains all reaction-general data objects
rxnSpecificData	Private	Contains all reaction-specific data objects
generalPhysicsModels	Private	Contains all reaction-general sub-model objects
rxnSpecificModels	Private	Contains all reaction-specific sub-model objects
gsmReaction	Private	Container for all information needed during a simulation
excitonData	Private	Contains exciton information for standard DCM and preequilibrium use
gsmIO	Private	Message controller

simulation behaviors, numerics, and utilized parameterizations. The *gsmOptions* data type, detailed by Table IV.IV, specifies within GSM primarily when to utilize each of its INC models as well as various characterizations to use. The various options data types utilized by each of the GSM sub-models and data objects is detailed there.

The *gsmProgeny* data type is detailed by Table IV.V. This data type is utilized by the GSM to track all progeny created during its sub-models' simulations. The *gsmResidual* data type characterizes the residual nucleus tracked by GSM during its simulations. Table IV.VI details this data type. Note however that, at this time, the *gsmResidual* is not updated during the simulation beyond the INC stage as it may fission or disintegrate via Fermi break-up physics.

The *gsmOptions* type utilizes within it several sub-model *options* data types as well to allow clients to control the numerics and behaviors contained within the GSM and its sub-models. The associated tables for each of the sub-models are shown here as they are not documented elsewhere at this time. Note the source code for the GSM event generator may contain helpful information in the form of *ReadMe.md* files. The various *options* data types contained by each model to provide controlling behavior of the model are detailed below where not previously documented.

Table IV.VIII and Table IV.IX control options for the coalescence model. Note that negative values for the members of Table IV.VIII indicate that the coalescence model will determine the appropriate coalescence radii according to their default parameterizations. Any positive value given to any member of the data type of Table IV.VIII will result in its usage within the coalescence model. Table IV.IX details the data type utilized by the coalescence model to control the production of light fragments, being ^{6}He, ^{6}Li, ^{7}Li, and ^{7}Be nuclei.

Table IV.X controls options for the Fermi break-up model. The *recNumNucleons* member of the *fermiBreakUpOptions* data type flags the nuclei that the Fermi break-up model will be utilized for, where nuclei with *fermiBreakUpOptions* nucleons or less will be modeled with the Fermi break-up model. It is important to note that the Fermi break-up model ought to be queried to determine whether or not the Fermi break-up model is recommended to be utilized for a nuclei's disintegration. The Fermi break-up model will allow for the modeling of a nucleus's disintegration, containing no

TABLE IV.IV. The *gsmOptions* Data Type

Member Name	Data Type	Default	Controlling Behavior
converseTotalMomentum	logical	false	Conserve momentum on-total (true) or on-average post-INC process (false)
tallyResidualProjectile	logical	true	Include projectile tallying in residual (for modified DCM usage)
useSDCMAfCzMultipliers	logical	true	Use the A_f and C_Z fission multipliers for the standard DCM (true) or the modified DCM (false)
printIncrement	int32	-1	For output file generation, how often to print the event number to the terminal
smoothTransition	logical	true	For providing smooth transitions between usage of the standard DCM and modified DCM INC models
transitionWidth	real64	100.0	Transition width around the central transition energy to utilize for smooth transitions
nucleonTransitionE	real64	$4,490.0$ MeV	INC transition energy for incident nucleons
pionTransitionE	real64	$2,490.0$ MeV	INC transition energy for incident pions
monoPhotonTransitionE	real64	$1,200.0$ MeV	INC transition energy for incident mono-energetic photons
bremPhotonTransitionE	real64	$5,000.0$ MeV	INC transition energy for incident bremsstrahlung photons
molnixOps	See Table II.IV data object	*N/A*	Controls the Molnix
fissBarOpts	See Table III.IV data object	*N/A*	Controls the Fission Barrier
sDCMDataOpts	See Table V.XIV data object	*N/A*	Controls the standard DCM
sDCMOpts	See Table V.V model object	*N/A*	Controls the standard DCM
coalesDataOpts	See Table IV.VIII data object	*N/A*	Controls the coalescence
coalesOpts	See Table IV.IX model object	*N/A*	Controls the coalescence
fbuOpts	See Table IV.X model object	*N/A*	Controls the Fermi break-up
preeqOpts	See Table IV.XI model object	*N/A*	Controls the preequilibrium
evapDataOpts	See Table IV.XII data object	*N/A*	Controls the evaporation
evapOpts	See Table IV.XIII model object	*N/A*	Controls the evaporation

TABLE IV.V. The *gsmProgeny* Data Type

Member Name	Type	Description
numBaryons	real64	Total number of nucleons (i.e. baryons)
numProtons	real64	Total number of protons (i.e. baryons)
kinEnergy	real64	Kinetic energy of particle [GeV]
restMass	real64	Rest mass of particle [GeV/c^2]
phi	real64	ϕ component of particle's motion
theta	real64	θ component of particle's motion
sinTheta	real64	$\sin\theta$
cosTheta	real64	$\cos\theta$
typeID	real64	ID of the particle, being one up to nine for n, p, d, t, ^{3}He, ^{4}He, π^-, π^0, π^+, or = $1000 * Z + N$
prodMech	real64	Mechanism by which progeny was produced

TABLE IV.VI. The *gsmResidual* Data Type

Member Name	Type	Description
numBaryons	real64	Total number of nucleons (i.e. baryons)
numProtons	real64	Total number of protons (i.e. baryons)
kinEnergy	real64	Kinetic energy of particle [GeV]
linearMom	real64	3-dimensional (x, y, z) linear momentum [GeV/c]
angularMom	real64	3-dimensional (x, y, z) angular momentum [GeV (unit length)/c]

TABLE IV.VII. The *gsmIO* Data Type

Member Name	Type	Description
message	character(LEN=512)	Contains the message to be printed
print	procedure(IOHANDLER)	Points to the message printing procedure

more than 16 nucleons, and no less than 1 nucleon as the nucleus, being a neutron or proton, cannot further disintegrate within the Fermi break-up model's scope. It is important to note that the data parameterized by the Fermi break-up model contains only data for a nucleus containing up to 12 nucleons, thus it may be possible for nuclei with more than 12 nucleons to cause the Fermi break-up model to fail and crash. Protection against this was embedded into the Fermi break-up model where necessary to prevent potential model failures.

Table IV.XI controls options for the preequilibrium model. Simulation of preequilibrium particle emission may be excluded utilizing the member of the data type depicted in Table IV.XI. Table IV.XII and Table IV.XIII control options for the evaporation model. These data types control the various parameterizations that are utilized by the evaporation and fission model for their simulations.

TABLE IV.VIII. The *coalescenceDataOptions* Data Type

Member Name	Data Type	Default	Controlling Behavior
coalesRadiiDeut	real64	−1.0	Coalescence radius for deuterons
coalesRadiiTrit	real64	−1.0	Coalescence radius for tritons and ^{3}He
coalesRadiiAlpha	real64	−1.0	Coalescence radius for ^{4}He
coalesRadiiLFrag	real64	−1.0	Coalescence radius for light fragments

TABLE IV.IX. The *coalescenceOptions* Data Type

Member Name	Data Type	Default	Controlling Behavior
expandedCoalescence	int32	2	Use expanded coalescence model (> 0) or not (≤ 1)

TABLE IV.X. The *fermiBreakUpOptions* Data Type

Member Name	Data Type	Default	Controlling Behavior
recNumNucleons	int32	12	Flags nuclei that the Fermi break-up model ought to be utilized on
akpScalingFlag	real64	0.0	Controls scaling of the a_{kp} variable

TABLE IV.XI. The *preequilibriumOptions* Data Type

Member Name	Data Type	Default	Controlling Behavior
r0Mult	real64	1.5	Radius multiplier, r_0, for $r_A = r_0 A^{1/3}$
numPreeqType	int32	66	Number of fragments considered for emission (6 – 66)
levelDenParam	int32	12	Flag for which parameters to use for the level density parameter calculation (11, 12)
emissionWidth	real64	0.4	Standard deviation for the preequilibrium emission probability
excludePreeq	int32	0	Flags to not simulate preequilibrium physics

TABLE IV.XII. The *evaporationDataOptions* Data Type

Member Name	Data Type	Default	Controlling Behavior
aLev	real64	0.0	Flag for the level density parameterization, being GCCI (0)= $A/8$ (1), or = $A/aLeV$ (> 1)

TABLE IV.XIII. The *evaporationOptions* Data Type

Member Name	Data Type	Default	Controlling Behavior
numEvapType	int32	66	Number of fragments considered for emission
inverseParameter	real64	0.0	Parameterization for inverse reaction cross sections, flagging Dostrovsky and Matsuse (= 0), a simple one with r_0 = 1.5 (= 10), or a simple one with $r_0 = inverseParameter$ ($0 < inverseParameter < 10$)
fissParameter	int32	0	Use the reevaluation of parameters (= 0) or the original (≠ 0) parameterization
redCompTime	int32	0	Flags use (= 1) of reduced computation time flag or not (≠ 1)

D.4 GSM PROCEDURES

The GSM procedures available to clients are described here, stating the purpose, usage, and arguments to the procedures, being functions or subroutines. A total list of procedures available to software clients of the GSM are listed in Table IV.V. Note also that many of the arguments for each of the procedures utilize primarily the GSM data types described in Sec. D.3.

Subroutine *updateOptions(options)*:

Modifies the options, behaviors, and numerics utilized by the GSM object, being contained in the *options* of the GSM model. The argument *options* is of the *gsmOptions* data type.

Function *queryOptions()*:

Returns the *gsmOptions* data type utilized by the GSM object.

Subroutine *updateRNG(rngPointer)*:

Modifies the current random number generator utilized by the GSM object for its simulations and disables calls to the next-stride procedure provided with the default random number generator of GSM. The *rngPointer* is a procedure pointer with the interface shown in Fig. III.7. Note the RNG pointer utilized by GSM will not be modified if the provided pointer is not associated.

Function *queryRNG()*:

Returns the RNG procedure pointer, with the interface of Fig. III.7, to the caller.

Function *updateMessageHandler(newIO)*:

Modified the procedure pointer utilized by the GSM object for handling all messages. The argument *newIO*, is a procedure pointer with the interface shown in Fig. III.6. Note that the message handler utilized by the GSM object will not be modified if the provided argument is not associated.

Function *queryPhotoEmissionUse()*:

Returns to the caller a logical flag indicating if photon emission is being utilized by the sub-models of the GSM object.

Function *properlyConstructed()*:

Returns a logical flag to the caller stating the construction state of the GSM object, where the returned value is *false* if the GSM object failed to properly construct.

Subroutine *generateOutput(projectile, target, outputDetails[, progenyArraySize])*:

Simulates all desired events according to the projectile and target provided, generating an output file according to the provided outputDetails argument. Optionally, software clients may provide an integer progeny array size, whereby setting the maximum number of progeny to be allowed throughout every event's simulation. The data type of each argument is *gsmProjectile*, *gsmTarget*, *gsmOutput*, and *int32*, respectively.

Subroutine *simulateEvent(projectile, target, results)*:

Simulates a single spallation event according to the state of the GSM object at runtime. The arguments data types are *gsmProjectile*, *gsmTarget*, and *gsmResults*, respectively. The *gsmResults* object is populated throughout the event's simulation.

Subroutine *standardDCMInterface(rxnData, sDCMProj, result)*:

The GSM interface to utilize the standard DCM for INC simulations. The argument data types are *StandardDCMData*, *sDCMProjectile*, and *gsmResults*, respectively.

Subroutine *setupMDMC()*:

Currently this procedure has not statements for its implementation and is not utilized.

Subroutine *modifiedDCMInterface(gsmRxn)*:

The GSM interface to utilize the modified DCM for INC simulations. The argument data type is

gsmReaction, containing various pointers to the necessary information. Note this procedure is, at this time, of no use to software clients as the *gsmReaction* data object is private within the GSM implementation.

Subroutine *coalescenceInterface(rxnPhysics, results)*:

The GSM interface to utilize the coalescence model for the compounding of INC progeny. The argument data types are *Coalescence* and *gsmResults*, respectively.

Subroutine *fermiBreakUpInterface(ap, zp, up, pnx, pny, pnz, results)*:

The GSM interface to the Fermi break-up model for the simulation of a nucleus's disintegration. The arguments provide the interface with information on the number of nucleons (ap), number of protons (zp), the kinetic energy of the nucleus (up), and the respective momentum components (pnx, pny, and pnz), in Cartesian coordinates. The argument data types are *real64*, *real64*, *real64*, *real64*, *real64*, *real64*, and *gsmResults*, respectively.

Subroutine *preequilibriumInterface(preeqObj, gsmNucleus, exciton, results)*:

The GSM interface to the preequilibrium model for its simulations. The argument data types are *Preequilbrium*, *nucleus* (from the *gsm_derived_types* module), *excitonData*, and *gsmResults*, respectively.

Subroutine *evaporationInterface(a, z, ue, trec, ln, bf0, fitaf, fitaf1, results)*:

The GSM interface to the evaporation and fission model's simulation. Input arguments are the number of nucleons (a) and protons (z), the thermal (ue) and recoil energies (trec), in MeV, angular momentum, with units of $\hbar$, fission barrier height (bf0), the a_f and C_Z scaling factors, and the results object to be populated during the evaporation and fission simulation. The argument data types are all *real64*, except for the *results* argument, being of data type gsmResults.

Subroutine *smiulateDecay(preeqObj, residual, exciton, afMult, czMult, results, wam)*:

The GSM interface to simulate both the preequilibrium and evaporation physics processes. The residual and exciton arguments provide the procedure with information about the nucleus to be decayed, where the other arguments provide additional information for the simulation. The *wam* argument is a flag to indicate events where the progeny array is overfilled prior to the end of the simulation. The argument data types are *Preequilibrium*, *gsmResidual*, *excitonData*, *real64*, *real64*, *gsmResults*, and *real64*, respectively.

Subroutine *formNuclei(projectile, target)*:

Populates all unspecified states of the provided projectile and target nuclei. The target and projectile nuclei, being the number of nucleons and number of protons, are swapped when the projectile contains more nucleons than the target, flagging use of the anti-laboratory system. The arguments each have the data type of *gsmProjectile* and gsmTarget, respectively.

Function *inquireINCModel(projectile, target)*:

Determines the INC model to be utilized based on the provided projectile and target nuclei information. It is important to note the procedure assumes the nuclei have already been formed. The function verifies the projectile is smaller than the target, and calls the *formNuclei* procedure if not. Each argument has intent *inout*, with the data type being *gsmProjectile* and *gsmTarget*, respectively.

Function *sampleEnergy(centralValue, stdDev)*:

Returns a value sampled randomly around the central value with a standard deviation of *stdDev*, according to a Gaussian distribution. It is important to note that the procedure will always return the *centralValue* when the *smoothTransition* flag of the *gsmOptions* is flagged as *false*. This procedure is utilized within GSM to sample the INC model to be utilized.

Function *determineRestMass(nuclA, nuclZ)*:

Returns the rest mass, according to the formula utilized by the GSM, of a nucleus provided the nucleus's number of nucleons (nuclA) and number of protons (nuclZ). GSM utilizes this procedure when a rest mass is not specified by the software client.

D.5 MODEL CONSTRUCTION

Construction of the GSM object is detailed in Sec. IV.2. Details on the construction of a GSM object may be found there as well as in some portions of Sec. IV.5, particularly Fig. IV.7.

D.6 USAGE AND RESULTS

Clients may start spallation simulations by calling the GSM object's *simulate* subroutine with a projectile object, the target data object, and a results object passed in as arguments. Note that Table IV.V details these procedures, as well as other available naming conventions for all simulation processes, and all other available procedures in GSM. Simulations in GSM may be started by constructing a *gsmResults* object and passing that object in to the appropriate GSM procedure. Upon completion of the simulation, GSM will have populated the provided *gsmResults* object according to the client provided projectile and target objects. Clients may then directly access members of the *gsmResults* object as desired. A complete list of client and end-user available procedures is detailed by Table IV.V.

Appendix E. Documentation for the Standard Dubna Cascade Model

The standard Dubna Cascade Model (standard DCM) is utilized to simulate, using Monte Carlo techniques, the intranuclear cascade of primary and participant particles occurring during the fast stage of high energy spallation reactions. The standard DCM simulates intranuclear physics according to Fig. II.2. Sec. II.2 details the physics approximations, models, and details that are utilized by the standard DCM. The standard DCM utilizes an object-oriented design, being written in modern Fortran (Fortran 2008).

The details of the standard DCM code are laid out here for the purposes of future development and maintenance, and in addition for ease of client use and API consumption. The documentation presented here is written primarily as an aid to users in developing their own interfaces for consumptions of the standard DCM API.

E.1 BUILD REQUIREMENTS

The standard DCM can be built by compiling and linking the modules shown in Table V.I. Table V.I states the purpose of each of the modules within the context of the standard DCM, and in addition a recommended build order. Clients utilizing the standard DCM ought to only access the target characterization data class and the standard DCM class itself.
The standard DCM object relies on the existence of the Molnix class, detailed by Appendix B, and the Photon Event Generator classNote that a single photon event generator object is utilized by each instance of the standard DCM object, being treated as a data object by the standard DCMobject.

Table V.II details the minimum required Fortran compiler version that the standard DCM is supported by in its current form. Note the standard DCM object was originally developed using the

TABLE V.I. Standard DCM Modules

Module Name	Description	Build Order
standardDCMDataParams	Parameter module for the standard DCM data class	1
standardDCMDataClass	Contains target characterization data	2
standardDCMParams	Parameter module for the standard DCM class	3
standardDCMData	Various required constants and data	4
standardDCMClass	Standard DCM class module	5

GNU compiler. Building the standard DCM object requires clients and user to utilize the minimum version shown in Table V.II. Numerical differences in simulation results may be observed between different builds, in compiler and version, of the standard DCM object. Potential differences in numerical results were not tested.

TABLE V.II. Compiler Support for the Standard Dubna Cascade Model

Compiler	Compiler Version
GNU	*N/A*
Intel	Not tested
IBM XL	Not tested
g95	Not tested
NAG	Not tested
PGI	Not tested

E.2 PRIMER

The standard DCM code was designed to be simple to use. Clients of the standard DCM code simply need to initialize data[1], construct the standard DCM, target data, and results objects, and properly query the results object. The standard DCM requires a projectile, target data, and results object for its arguments. The standard DCM models the interaction of the projectile object incident on the target object. The projectile and target objects remain unmodified and the results object is populated during the simulation.

[1]Data initialization is optional as the standard DCM object ensures it is initialized during the object's construction. Initialization by the client is highly recommended if using the standard DCM in calculations utilizing parallel processing.

The standard DCM data module contains all numerical data utilized by the standard DCM object, including a photon event generator object. The standard DCM will verify its numerical data was properly initialized upon the object's construction, and fail to simulate IntraNuclear Cascade physics if numerical data is not initialized. It is recommended that clients initialize the required data, particularly when parallel processing, to ensure the data is not doubly initialized as a result of the data initialization check embedded in the standard DCM object construction, therefore reducing computational performance.

The standard DCM requires proper construction to perform its simulations. The standard DCM simulation object is constructed by passing in a function pointer to a random number generator, a molnix object, and, optionally, a simulation behaviors object and a subroutine pointer that handles the printing of all messages. The standard DCM target characterization object is constructed using information of the total number of nucleons and the number of protons in the target nucleus, and, optionally, an object containing numerics utilized by the object for its construction, and a subroutine pointer that handles the printing of all messages. The results object is constructed by passing in an array of standard DCM progeny data structures, allowing the results object to contain within it all information regarding the progeny, residual nucleus, and exciton information that resulted from the specified interaction.

A simulation by the standard DCM is triggered by calling the standard DCM's *simulate* subroutine, passing in a projectile object, the constructed target characterization object, and the constructed results object for its arguments. The passed in results object becomes populated with information during the simulation, namely the progeny, residual nucleus, exciton information, number of elastic events, end-state of the simulation, and the number of times interactions were restarted due to some detected unphysical result. The implementation of the results object, and utilization of an interal interim results object, allows the state of the standard DCM object to be as minimal as possible, where the object retains no information about the simulation other than the message received to be printed.

Fig. E.1 details a sample API for clients, such as GSM, using the standard DCM object for a

single event. It is recommended that the standard DCM object is constructed one time only for computational efficiency within the respective clients. It is also recommended that clients construct a single instance of the target characterization object when simulating IntraNuclear Cascade physics in a Monte Carlo fashion to obtain average values. Note that Fig. E.1 does not demonstrate the usage of the optional arguments during the construction of the standard DCM and standard DCM target objects. Details on the standard DCM API are further discussed in the following sections.

E.3 DATA INITIALIZATION

The standard DCM object interfaces to the standard DCM data module for its numerical constants and the empirical and experimental data it utilizes. It is recommended that client programs interface directly to the standard DCM data module for the data initialization as clients will have more information regarding the data initialization. The data initialization function provided by the standard DCM acts as an interface to the data module's function, where arguments are optional and, when not client specified, utilze the default values from the standard DCM data module.

The standard DCM data initialization function, *initializeStandardDCMData*, may have three arguments passed in to it, being a file name containing the appropriate data file name for which data is to be read from, a file unit number, and a message handling subroutine pointer. Usage of the message handling subroutine pointer in the data initialization function will allow clients to have control over all messages produced by the data module during data initialization and by the photon event generator object that all instances of the standard DCM will use. The data initialization function will attempt to read the photon cross section data file by first validating the file unit to be used, ensuring it is not currently open as another file elsewhere in the active program, and then it will verify that the file exists as specified. The function will attempt to use a valid file unit up to 50 times, after which file data will not be read. The function will also attempt to look for the default data file in the instance where the specified file could not be found and does not match the default file name, upon which the file's data will not be read and data initialization will fail. Data exists

```fortran
subroutine sDCMAPIExample(projectileObj, targetA, targetZ)

   use, intrinsic:: iso_fortran_env, only: int32, real64
   use otherModules, only: molnixObj, rngPtr
   use standardDCMDataClass, only: &
        & StandardDCMData, &    ! sDCM data object
        & newStandardDCMData    ! sDCM data object constructor
   use standardDCMClass, only: &
        & StandardDCM, newStandardDCM, &    ! sDCM object and its constructor
        & standardDCMResults, &      ! sDCM results object
        & newStandardDCMResults, &   ! sDCM results constructor
        & sDCMProgeny, sDCMProjectile, &
        & initializeSDCMData    ! Initializes all sDCM data

   implicit none
   type(sDCMProjectile), intent(in   ) :: projectileObj
   real(real64),         intent(in   ) :: targetA
   real(real64),         intent(in   ) :: targetZ

   ! Initialize sDCM data:
   integer(int32) :: errorFlag = initializeSDCMData()
   if ( errorFlag /= 0_int32) write(*,*) "sDCM data initialization failed."

   ! Construct sDCM simulation and data objects (with default behaviors)
   type(StandardDCM) :: mySDCM = newStandardDCM(rngPtr, molnixObj)
   type(StandardDCMData) :: mySDCMTarget = newStandardDCMData(targetA,
      targetZ)

   ! Construct results object:
   type(sDCMProgeny), dimension( nint(targetA) ) :: progenyBnk
   type(standardDCMResults) :: results = newStandardDCMResults(progenyBnk)

   ! Simulate sDCM:
   call mySDCM%simulate(projectileObj, mySDCMTarget, results)

   ! Query from results:
   .
   .
   .

   return
end subroutine sDCMAPIExample
```

FIGURE E.1. Sample API for the standard DCM object.

that is used in conjunction with the *qints* function of the standard DCM. This data currently has no known failure modes, however flags exist for when such occurrences are known.

The standard DCM data module contains several flags that can be used to verify the initialization state of its data members. An overarching logical flag exists, *sDCMDataEstablished*, to indicate whether or not the standard DCM data was successfully initialized without error. The value of this logical flag is protected within the data initialization module and can only be flagged as *true* after the success of data initialization. The data initialization function may return one of several values to indicate the occurrence of errors. The errors flags are parameterized integer values that can be used by client programs to indicate the errors that occurred. These parameterized error flags are described in Table V.III.

TABLE V.III. Parameterized Data Initialization Flags for the Standard DCM

Flag Name	Type	Description
sDCMDataEstablished	logical	Flags proper initialization of the standard DCM data
sDCMDataSetup	integer	Indicates no error occurred during data initialization
sDCMQintError	integer	Indicates failure of *qints* data initialization
sDCMGammaError	integer	Indicates failure of photon cross section data initialization
sDCMDataError	integer	Indicates a miscellaneous error during data initialization

Usage of the data initialization interface within the standard DCM code itself does not provide a loss of information, however does result in the printing of additional messages in the event an error occurs. Note the value returned by the data initialization function is a cumulative sum of all the errors encountered during data initialization. The returned error flag is increased by one in the event of an error during construction of the photon event generator. Note that data initialization is attempted with every construction of the standard DCM object when the *sDCMDataEstablished* indicates that the data utilized by the standard DCM has not beed established. It is important to note that the standard DCM object can still simulate IntraNuclear Cascade physics in the event of data initialization failure.

E.4 DATA STRUCTURES

The standard DCM utilizes several data structures for its simulations and architecture to aid in modulation and future upgrades to the model. Data types used by the standard DCM object are shown in Table V.IV, describing the purpose and use of the data object.

TABLE V.IV. Data Types Used by the Standard DCM Object

Name	Client Access	Purpose
sDCMOptions	Public	Controls behavior of standard DCM object
sDCMProjectile	Public	Projectile information for initial collision
StandardDCMData	Public	Target information and reaction-specific data
sDCMProgeny	Public	Contains information for collision progeny
sDCMCompound	Public	Contains information for the residual nucleus
sDCMExcitons	Public	Contains tracked exciton information
StandardDCMResults	Public	Contains all results tracked in simulation
sDCMPhotonCrossSections	Private	Contains relevant photon cross section data
sDCMPauliInfo	Private	Contains relevant generation information
sDCMIO	Private	Message controller

The *sDCMOptions* options data type is used by the standard DCM object to control its simulation behaviors. The *sDCMOptions* data type, detailed by Table V.V, is used to control the inclusion or exclusion of reflection and refraction of particles at the various nuclear zones and to control the inclusion of the updated angular distributions. Inclusion of reflection and refraction at the nuclear boundaries is not recommended as worse overall agreement with experimental data has been observed. The use of the updated angular distributions removes an unphysical pole-like behavior observed near emittance angles of 0° and 180°.

TABLE V.V. The *sDCMOptions* Data Type

Member Name	Default	Controlling Behavior
boundaryEffects	Off	Inclusion (> 0) or exclusion (≤ 0) of reflection and refraction at nuclear zone boundaries
newAngularDistributions	On	Use (> 0) or not use (≤ 0) updated angular distribution approximations

The *sDCMProjectile* data type is a required argument when simulating IntraNuclear Cascade physics using the standard DCM. Table V.VI describes the various member variables of the data type. Client programs are required to initialize the member-variables of the *sDCMProjectile* object prior using the standard DCM for simulation. Failure to initialize the *sDCMProjectile* object will yield in simulating IntraNuclear Cascade physics where the projectile has the characteristics of a photon with no incident energy, resuling in an unphysical IntraNuclear Cascade simulation.

TABLE V.VI. The *sDCMProjectile* Data Type

Member Name	Type	Description
numBaryons	integer(int32)	The number of nucleons in the projectile
numProtons	integer(int32)	The number of protons in the projectile
kinEnergy	real(real64)	The kinetic energy [MeV] of the projectile
restMass	real(real64)	The rest mass [MeV/c^2] of the projectile
decayNumber	integer(int32)	The quantum decay number of the projectile, typically = 0
gammaFlag	integer(int32)	Flags if projectile is a photon (> 0) or not (= 0)

The *StandardDCMData* object is utilized by the standard DCM object during its IntraNuclear Cascade simulations to easily and quickly obtain data about the target nucleus, such as the number of nucleons and protons within the target, the neutron and proton densities of the nucleus, *etc.* The *StandardDCMData* object is structured similarly to the standard DCM object, containing a construction flag, message controller, an options type, and an internal data type used for target nucleus characterization. The *StandardDCMData* object is detailed further in Sec. E.5. Note that the standard DCM obbject will not allow clients to simulate IntraNuclear Cascade physics unless the *StandardDCMData* object has been properly constructed.

The *sDCMProgeny* data type is utilized by the standard DCM to track information regarding the progeny created during its IntraNuclear Cascade simulation. Note that the particle's strangeness is rarely flagged as being strange, and the index flag of the *sDCMProgeny* type indicates the cascade generation the particle was created in. Clients are required to utilize the *sDCMProgeny* data type for simulations within the standard DCM object. The standard DCM object requires an array of *sDCMProgeny* data types to be initialized by the client of unknown size. The array of *sDCMProgeny*

must then be passed in to the *standardDCMResults* object, where the *standardDCMResults* object is required to be passed in to the standard DCM object during its IntraNuclear Cascade simulation.

TABLE V.VII. The *sDCMProgeny* Data Type

Member Name	Type	Description
xCoord	real(real64)	X-coordinate of the particle [*fm*]
yCoord	real(real64)	Y-coordinate of the particle [*fm*]
zCoord	real(real64)	Z-coordinate of the particle [*fm*]
sinTheta	real(real64)	Sine of the particle's theta angle of movement
cosTheta	real(real64)	Cosine of the particle's theta angle of movement
sinPhi	real(real64)	Sine of the particle's phi angle of movement
cosPhi	real(real64)	Cosine of the particle's phi angle of movement
kinEnergy	real(real64)	Kinetic energy of the particle [*GeV*]
restMass	real(real64)	Rest mass of the particle [GeV/c^2]
numProtons	integer(int32)	Number of protons in the particle
numBaryons	integer(int32)	Number of nucleons in the particle
strangeness	integer(int32)	Quantum decay number of the particle
photonFlag	integer(int32)	Flags if the particle is a photon (> 0) or not (= 0)
nuclearZone	integer(int32)	Last nuclear zone where the particle was located
index	integer(int32)	Index of the particle's creation (generation created)

The *sDCMCompound* data type is used to track information on the resulting residual nucleus that remains after the cascade stage of the reaction. Information about this nucleus is required to simulate preequilibrium emission of particles during the nucleus's deexcitation as well as to simulate the evaporation or disintegration of the resulting nucleus. Table V.VIII details the information tracked by the standard DCM object in the *sDCMCompound* data type. The *sDCMCompound* data type is contained within the *standardDCMResults* object to maintain a general and minimal state of the standard DCM object. Note the *linearMom* and *angularMom* members of the *sDCMCompound* data type are arrays of three elements, where the X, Y, and Z components of the linear and angular momentum are represented in the arrays by elements one, two, and three, respectively.

The standard DCM simulation tracks exciton information during its simulations. The *sDCMExcitons* type, details in Table V.IX, is used by the standard DCM to track the simulation's exciton-related information. The *sDCMExcitons* type contains information regarding the number of created charged and neutral excitons, as well as the number of exciton holes. The *sDCMExci-*

TABLE V.VIII. The *sDCMCompound* Data Type

Member Name	Type	Description
numBaryons	real(real64)	Number of nucleons contained in the nucleus
numProtons	real(real64)	Number of protons contained in the nucleus
kinEnergy	real(real64)	Kinetic energy of the nucleus [*GeV*]
linearMom	real(real64)	Three-dimensional linear momentum [*GeV/c*]
angularMom	real(real64)	Three-dimensional angular momentum [*GeV*×*m/c*]

tons type is embedded in the *standardDCMResults* type to enable client programs to easily query exciton-related information as a result of a simulation.

TABLE V.IX. The *sDCMExcitons* Data Type

Member Name	Type	Description
numExcProt	integer(int32)	Number of charged excitons
numExcNeut	integer(int32)	Number of neutral excitons
numExcHoles	integer(int32)	Number of exciton holes

The *standardDCMResults* data object is used by the standard DCM object to store all simulation related results and information. Table V.X details the public information contained by the *standardDCMResults* object. Note that information private to the standard DCM object lies within the *standardDCMResults* object, namely information about the progeny bank size, pointers to secondary progeny arrays used internal to the standard DCM simulations, and an initialization flag. The *standardDCMResults* object must be constructed prior to simulating IntraNuclear Cascade physics, where its progeny bank array points to a client specified array. The standard DCM object will fail to simulate IntraNuclear Cascade physics in the event the *standardDCMResults* object is not properly constructed and will exit with an error message and flag. The *standardDCMResults* object also contains a simulation state flag, indicating if the simulation ended with or without error. The simulation state flag will equal 0 when the simulation ended cleanly, and will be greater than 0 when errors are encountered in the simulation.

The *sDCMPhotonCrossSections* data type, detailed in Table V.XI, is a private data type to the standard DCM object that is used to store the angular distribution and the cumulative distribution of

TABLE V.X. The *standardDCMResults* Data Type

Member Name	Type	Description
numProgeny	integer(int32)	Number of progeny created during simulation
progenyBnk	*sDCMProgeny*	Pointer to an array of simulated progeny
residual	*sDCMCompound*	The predicted residual nucleus at simulation end
excitons	*sDCMExcitons*	Exciton information at the end of the simulation
numElastic	integer(int32)	Number of elastic events that occurred
numRestarts	integer(int32)	Indicates how many times the simulation restarted due to an issue in the *geom8* function (element 1) and due to failure of the *typint* function to determine the number of produced secondaries (element 2)
simState	integer(int32)	Flags the end-state of the simulation

photon cross sections at 1° increments for two photon cross section channels. The two photon cross section channels considered include, for channel one, $\gamma + p \rightarrow \pi^+ + n$ reactions and, for channel two, $\gamma + p \rightarrow \pi^0 + p$ reactions. Isotopic invariance is assumed between $\gamma + p$ and $\gamma + n$ reactions. Note that the *sDCMPhotonCrossSections* data type is used only when the considered primary particle[2] is a photon.

TABLE V.XI. The *sDCMPhotonCrossSections* Data Type

Member Name	Type	Description
si	real(real64)	Contains the channel's angular cross section
ri	real(real64)	Contains the channel's cumulative angular cross section

The *sDCMPauliInfo* data type, detailed by Table V.XII, is used between the main and pauli blocking procedures to track generation information in the information. The *sDCMPauliInfo* also contains a flag that indicates when three particle production has occurred in an event. The *sDCMIO* data type, detailed in Table V.XIII, is used internally to the standard DCM object to control the writing and printing of messages. The standard DCM object currently uses a *write* statement to populate the desired message, and then calls a message printing message procedure to write the message depending on the message's flagged importance and the client's desired verbosity flag.

[2]Recall that the primary particle includes in initial incident particle as well as all further interacting particles.

TABLE V.XII. The *sDCMPauliInfo* Data Type

Member Name	Type	Description
indi	integer(int32)	Flags $N + N \rightarrow \pi + N + N$ reactions
ing	integer(int32)	Used as generation information
meso	integer(int32)	Used as generation information
ngen	integer(int32)	Used as generation information
igs	integer(int32)	Used as generation information

TABLE V.XIII. The *sDCMIO* Data Type

Member Name	Type	Description
message	character(LEN=512)	Contains the message to be printed
print	procedure(IOHANDLER)	Points to the message printing procedure

E.5 THE TARGET DATA OBJECT

The *StandardDCMData* class is used to store information regarding the target nucleus for a spallation reaction. The resulting object must be passed in to the main *StandardDCM* object's *simulate* procedure to simulate IntraNuclear Cascade physics. The *StandardDCMData* class contains private data that client programs, such as the *StandardDCM* object, may query to obtain target characteristics. The *StandardDCMData* object is structured in such a way that client programs may not change any values within the data object. The *StandardDCMData* object contains within it an integer construction flag, a message handler with the same structure as Table V.XIII, a *sDCMDataOptions* data type to control the numerical values used within the object, and a target object to contain all information about the target nucleus.

Client programs can utilize the standard DCM data object by using the *StandardDCMData* object, *newStandardDCMData* constructor interface, and the *sDCMDataOptions* data type from the *standardDCMDataClass* module. The *sDCMDataOptions* data type is used by the *Standard-DCMData* object to control various constants used by the model, such as the number of nuclear zones to model. The *StandardDCMData* object is constructed by passing in the number of nucleons and protons of the target nucleus into the constructor. Clients may also optionally pass in the

sDCMDataOptions type to control the various numerics used within the data class and may also pass in a procedure pointer to a message printing subroutine. Fig. E.2 provides a recommended client interface to the *StandardDCMData* object.

The *StandardDCMData* object utilizes three different data types. The three data types include *sDCMDataIO*, being the same as described by Table V.V, *sDCMDataOptions*, and *sDCMDataTarget*. The *sDCMDataOptions* data type, described in Table V.XIV, is used to control the numerics and behaviors used by the standard DCM data object. Note that the data object copies and validates the options passed in by the client, thus creating its own copy of the options. The *sDCMDataTarget* data type, detailed in Table V.XV, is used to store the various target nucleus characterizations needed by the standard DCM object during simulation. Note that the standard DCM object, during simulation, will attempt to determine an accurate value for the separation energy, or binding energy, of a single nucleon within the nucleus. All values stored within the *sDCMDataTarget* object can be queried by clients of the *StandardDCMData* object by referring directly to the *sDCMDataTarget* member variables name, *i.e.*

```
zoneOneBound = myDataTarget%zoneBoundR( 1 )
```

Note that an integer value must be passed in to the member variables whose values refer to a specific zone, namely *zoneBoundR, zoneBRFrac, protonDensity, neutronDensity, coulombPote, protFermiMom,* and *neutFermiMom*. The data object will return the value in that zone, if valid, and return the value closest to the requested zone when the requested zone is not valid or does not exist within the arrays.

Fig. E.2 details how the standard DCM data object is constructed. The constructor, *newStandardDCMData*, has two required and two optional arguments. The two required arguments for the constructor include the total number of nucleons and the number of protons within the target nucleus. The two optional arguments include a *sDCMDataOptions* data type and a procedure pointer, with the interface shown in Fig. E.2. Clients may pass in an object of type *sDCMDataOptions* to control some of the characteristics of the data object, to control such things as the number of nuclear zones, the numerics in Eqn. II.1, the maximum nuclear radius allowed, and an average density fraction for

```
  subroutine sDCMDataAPIExample(targetA, targetZ)

    use, intrinsic:: iso_fortran_env, only: int32, real64
    use clientModules, only: clientPrintFunction
    use standardDCMDataClass, only: &
        & StandardDCMData,    &   ! sDCM data object
        & newStandardDCMData, &   ! sDCM data object constructor
        & sDCMDataOptions,    &   ! sDCM data object numerics
        & properlyConstructed     ! Import flag to verify construction state

    implicit none
    real(real64),   intent(in   ) :: targetA
    real(real64),   intent(in   ) :: targetZ

    ! Create message handling procedure pointer:
    abstract interface
       subroutine IOHANDLER(verbosity, type, text)
          use, intrinsic:: iso_fortran_env, only: int32
          implicit none
          integer(int32),   intent(in) :: verbosity
          integer(int32),   intent(in) :: type
          character(len=*), intent(in) :: text
       end subroutine IOHANDLER
    end interface
    procedure(IOHANDLER), pointer :: clientIO => clientPrintFunction

    ! Create instance of data object and its options:
    type(StandardDCMData) :: mySDCMTarget
    type(sDCMOptions) :: dataOptions

    ! If desired, modify numerics:
    dataOptions%numZones = 8    ! Use 8 nuclear zones instead of 7
    dataOptions%maxRad = 12     ! Increase maximum nuclear radius to 12 fm

    ! Construct data object:
    mySDCMTarget = newStandardDCMData(targetA, targetZ, dataOptions, clientIO
       )

    ! Verify construction:
    if ( mySDCMTarget.constructionState() /= properlyConstructed ) then
       write(*,*) "The sDCM data object failed to properly construct."
    end if

    return
  end subroutine sDCMDataAPIExample
```

FIGURE E.2. Sample API for the *StandardDCMData* object.

TABLE V.XIV. The *sDCMDataOptions* Data Type

Member Name	Default Value	Description
numZones	7	Number of nuclear zones to use
expDenom	0.545	a term for Eqn. II.1
r0	1.07	c term for Eqn. II.1
maxRad	10.0	Maximum nuclear radius [fm] to be used
aveDen	0.95, 0.80, 0.50, 0.20, 0.10, 0.05, 0.01, 0.00, 0.00, 0.00	Average density fractions for each allowable zone (up to 10 allowed)

TABLE V.XV. The *sDCMDataTarget* Data Type

Member Name	Type	Description
numBaryons	real(real64)	Number of nucleons
numProtons	real(real64)	Number of protons
aTargThrd	real(real64)	Cube root of the number of nucleons ($A^{1/3}$)
pionPote	real(real64)	Constant pion potential well [GeV]
sepEnergy	real(real64)	Seperation energy [GeV]
geomCrossSection	real(real64)	Geometric cross section [fm^2]
zoneBoundR	real(real64)	Zone bounds [fm]
zoneBRFrac	real(real64)	Zone bounds as fraction of maximum allowed
protonDensity	real(real64)	Proton density within each zone
neutronDensity	real(real64)	Neutron density within each zone
coulombPote	real(real64)	Coulomb potential within each zone
protFermiMom	real(real64)	Fermi momentum of protons within each zone
neutFermiMom	real(real64)	Fermi momentum of neutrons within each zone

TABLE V.XVI. Construction Flag Values

Flag Name	Value	Description
objectNotConstructed	−1	Flags object as not constructed
properlyConstructed	0	Flags object as being constructed
invalidTargetFlag	1	Flags an invalid target nucleus is used
targetInitFailed	10	Flags when target data initialization failed

the nucleus. Table V.XIV details the members of the *sDCMDataOptions* type. Upon declaring a *sDCMDataOptions* object clients may alter one, or several, of these values to alter the characteristics of the target nucleus. Parameterized default values are given to each member upon declaration of the *sDCMDataOptions* data type. A procedure pointer, with the abstract interface shown in Fig. E.2, may also be passed in to the data object's constructor. The data object utilizes the procedure it points to as a method of specifying a message and its importance and type. Message types used by the data object include fatal errors, errors, warnings, and comments, respectively. The default verbosity of the data object is four, where all messages with lower verbosity, or importance, are printed, be default. Clients may pass in a procedure pointer with the appropriate interface to directly control usage of these messages, such as printing every message, writing them to files, *etc.*

The private construction flag internal to the *StandardDCMData* object can be queried by client programs to ensure that the data object was properly setup without error prior to simulation. The standard DCM object performs this check prior to simulation to ensure the correct client specified reaction is being simulated. The construction flag may take on one of several values according to Table V.XVI, being incremented with each error that occurred. Client programs may use the parameterized values of Table V.XVI to determine the state of the *StandardDCMData* object. Note that the standard DCM object checks for proper construction using the *properlyConstructed* flag. All member variables and child member variables of the data object may be queried by simply calling the member variables' name as a function call. Table V.XVII clarifies this by stating explicitly all member procedures of the *StandardDCMData* object. The data object also utilizes two private procedures, namely *checkIndex* and *setupTarget*. The *checkIndex* function simply verifies the zone of the client's desired value exists and is within limits, and if not it provides the nearest valid index

TABLE V.XVII. Member Procedures of the *StandardDCMData* Object

Procedure Name	Type	Description
constructionState	Function	Returns the construction state of the object
zoneBoundR	Function	Returns a zone's boundary radius [*fm*]
zoneBRFrac	Function	Returns a zone's boundary radius fraction
protonDensity	Function	Returns a zone's proton density
neutronDensity	Function	Returns a zone's neutron density
coulombPote	Function	Returns a zone's coulomb potential
protFermiMom	Function	Returns a zone's proton Fermi momentum
neutFermiMom	Function	Returns a zone's neutron Fermi momentum
numBaryons	Function	Returns the number of nucleons in the target
numProtons	Function	Returns the number of protons in the target
numZones	Function	Returns the number of zones
aTargThrd	Function	Returns the cube root of the *numBaryons* member
geomCrossSection	Function	Returns the geometrical cross section [fm^2]
pionPote	Function	Returns the pion potential
setSepEnergy	Subroutine	Sets the separation energy of the target nucleons'
getSepEnergy	Function	Returns the target's nucleon separation energy [*GeV*]

available in the object. The *setupTarget* subroutine is used by the data object during its construction to establish the member variables of the data object's *sDCMTarget* object.

Clients to the standard DCM data object should be very cautious to verify the construction state of the object, refusing to use the object in the event of any errors, and should also be vigilant to ensure physicality of the numbers to be utilized by the data objects' clients. The data object itself verifies that the client-specified numerics in the *sDCMDataOptions* data type are within a valid and physical range. Note that the standard DCM data object may utilize as few as one zone, and up to no more than ten zones. Clients to the data object may attempt to specify otherwise, however the data object will utilize the default number of zones in these instances. Default values for all members of the *sDCMDataOptions* object are utilized in the case of unphysical or invalid values.

E.6 MODEL CONSTRUCTION

The standard DCM object requires that it be properly constructed prior to simulation. The standard DCM object will fail to perform any INC physics simulations until it has been properly

constructed. The standard DCM object constructor, *newStandardDCM*, requires two arguments and accommodates two optional arguments. The required arguments include a procedure pointer that points to a random number generator function and a Molnix data object, respectively. Clients may optionally pass in a *sDCMOptions* data type to set the behavior of the standard DCM object, regarding whether or not it uses reflection and refraction at nuclear boundaries and whether or not to use updated angular distribution interpolations. The standard DCM object, by default, utilizes the *sDCMOptions* data type with its default values to determine these behaviors. The second optional argument that client programs may utilize is a procedure pointer that points to a message handling subroutine. Fig. E.3 demonstrates construction of a standard DCM object when using the *sDCMOptions* optional argument. Clients may construct the standard DCM object with or without use of one or both of these arguments. The message printing procedure has the interface shown in Fig. E.2 for reference. Note that the object's construction state can be verified by calling the *properlyConstructed* function of the standard DCM object, returning a logical flag indicating if the standard DCM object was successfully constructed or not.

E.7 USAGE AND RESULTS

Clients may start INC simulations by calling the standard DCM object's *simulate* subroutine with a projectile object, the target data object, and a results object passed in as arguments. The standard DCM object will simulate its INC physics accordingly and populate the passed in results object during the simulation. The standard DCM retains no information about a simulation except for the last printed message. The results object, being of the *StandardDCMResults* data type, passed in to the *simulate* subroutine must be constructed prior to using. Clients may construct the object by passing in an array of *sDCMProgeny*, and the results object will point to the passed in progeny bank array and determine its size. Fig. E.4 demonstrates the construction of the *standardDCMResults* object as well as its usage and querying of simulation results. Note the standard DCM object may simulate INC physics through its *simulate, execute, start, interact, collide, simulateCascade*, and

```
subroutine sDCMConstructor(rngPtr, molnixObj)

   use, intrinsic:: iso_fortran_env, only: int32, real64
   use molnixClass, only: Molnix
   use standardDCMClass, only: &
        & StandardDCM, &   ! sDCM Sim. object
        & newStandardDCM   ! Object constructor

   abstract interface
      ! Interface for the random number generator:
      function RANDOM() result(rndmNum)
         use, intrinsic:: iso_fortran_env, only: real64
         implicit none
         real(real64) :: rndmNum
      end function RANDOM
   end interface

   implicit none
   procedure(RANDOM), intent(in   ), pointer :: rngPtr
   class(Molnix),     intent(in   )          :: molnixObj

   ! Declare sDCM object and the options type:
   type(StandardDCM) :: mySDCM
   type(sDCMOptions) :: mySDCMOptions

   ! Change sDCM behavior, if desired:
   mySDCMOptions%boundaryEffects = 1   ! Turn on reflection/refraction at
       boundaries

   ! Construct sDCM object:
   mySDCM = newStandardDCM(rngPtr, molnixObj, mySDCMOptions)

   ! Verify object was constructed properly:
   if ( .not. mySDCM%properlyConstructed() ) then
      write(*,*) "The sDCM object failed to properly construct."
   end if

   return
end subroutine sDCMConstructor
```

FIGURE E.3. A sample of the standard DCM object construction.

simulateINC procedure names. Each of these procedures point to the primary simulation procedure, aiding in a client's ease of use and simplicity.

Table V.X notes that various data types contained by the *standardDCMResults* object. The object stores within it information about how many progeny were created as well as their simulated characteristics from the simulation, the resulting residual nucleus, exciton information, and the number of elastic events that may have occurred. The results object also contains a two-element integer array stating the number of times the IntraNuclear Cascade simulation was restarted due to an issue in the *geom8* procedure and due to a failure of the *typint* procedure to determine the number of produced secondary particles, respectively, and an integer *simState* flag that indicates the last warning or potential error that occurred before simulation end. Sec. E.4 describes some of these members well.

The *numRestarts* member provides little helpful information, but instead provides diagnostic information without affecting the state of the class. The *simState* flag indicates whether the simulation ended in an error or was able to be successfully ran. The *simState* may equal zero, one, two, of anything larger than five to indicate no warnings or errors, an unphysical residual was about to be created, an error obtaining angular distributions during the IntraNuclear Cascade occurred, or the error was such that IntraNuclear Cascade physics could not be simulated by the standard DCM object, respectively. An flag indicating an unphysical residual nucleus indicates that the progeny about to be created would have resulted in a residual that is too small for the current version of the standard DCM to handle or to indicate an unphysical residual. Instances where this error is flagged results in the standard DCM object not creating the progeny and returning program control to the client at that time. Errors that result in the end of the simulation prematurely have values larger than five, indicating an error regarding the standard DCM data object, results object, or the standard DCM object not having been constructed.

The standard DCM object limits its clients' access to the model's procedures highly. Clients have access to a number of query procedures as well as the main `simulate` procedure. The `simulate` procedure, also being invoked by calling `execute`, `start`, `interact`, `collide`,

```
subroutine standardDCMInterface(sDCMModelObj, projectileObj, targetObj)

   use standardDCMDataClass, only: StandardDCMData
   use standardDCMClass,     only: &
        & StandardDCM, &
        & sDCMProgeny, &
        & StandardDCMResults, newStandardDCMResults, &
        & sDCMProjectile

   implicit none
   class(StandardDCM),    intent(inout) :: sDCMModelObj
   type(sDCMProjectile),  intent(inout) :: projectileObj
   type(StandardDCMData), intent(inout) :: targetObj

   ! Create progeny array and results objects
   type(sDCMProgeny), dimension( nint(targetObj%numBaryons()) ) ::
      progenyBnk
   type(StandardDCMResults) :: results

   ! Construct results object:
   results = newStandardDCMResults( progenyBnk )

   ! Simulate:
   call mySDCMObj%simulateINC(projectileObj, targetObj, results)

   ! Query from results:
   if ( results%simState /= 0 ) then
      write(*,*) "An error occurred!"
   end if

   ! Store progeny data in client:
   do i=1, results%numProgeny
      .
      .
      .
   end do

   ! Store residual information for creating spectra later:
   clientObj%residual(thisgeneration) = results%residual

   .
   .
   .
   return
end subroutine standardDCMInterface
```

FIGURE E.4. A sample construction and usage of the *standardDCMResults* object.

`simulateCascade`, and `simulateINC`, requires clients to pass in a `sDCMProjectile`, a `StandardDCMData`, and a `standardDCMResults` object. Sec. E.5 details the setup and usage of the `StandardDCMData` object. Sec. E.4, specifically Table V.VI and Table V.X, detail the members and usage of these objects. Invoking the `simulate` procedure by the client will have the standard DCM object perform a simulation of IntraNuclear Cascade physics. Clients may also query the state of the standard DCM object if desired. The available query functions provided by standard DCM object include `properlyConstructed`, `queryOptions`, `queryMolnix`, and `queryRNG`, where the function returns to the client the object's constructed flag, its `sDCMOptions` object, its `Molnix` object pointer, and its `RNG` procedure pointer, respectively.

E.8 TIPS, WARNINGS, AND LIMITATIONS

It is recommended to clients of the model that they be as strict as possible regarding the use of the model. Clients are recommended to `save` the object, or contain it within a module, when performing long simulations that may utilize the model more than once to save on computational efficiency. The model was designed such that it would contain as minimal a state as possible, for the purpose of both memory utilization and parallelization. Clients are additionally recommended to utilize all optional arguments in the model constructor to provide clients with full control of the model. Clients are recommended to use and provide the standard DCM constructor with the *sDCMOptions* type, in addition to querying the resulting object, to verify and validate all options utilized by the model. Clients, given the information about the message handling procedure pointer in Fig. E.2, or Fig. III.6, can easily create an interface to their message handling procedure, if present. Clients can utilize this feature of the model to prepend information to the message, such as flagging the message as one from the standard DCM, during simulations and to control where the resulting messages are sent, for example. The verbosity utilized within the standard DCM module, `sDCMVerbose`, may also be modified by clients to increase or limit the messages printed by all

standard DCM objects utilizing the default message handling procedure. Usage of all optional arguments is highly recommended for both the physics model as well as its associated data object, including the construction for all required data objects, such as the *Molnix* object.

It is highly beneficial for clients to verify the construction state and end-state of all objects and their associated simulations as well. Unconstructed objects will fail to allow simulations to occur within themselves and will exit their simulations cleanly. Clients, with this architecture, may verify prior to simulation that objects are constructed prior to use. Clients may do so by calling the standard DCM's `properlyConstructed()` function, where a logic flag is returned informing clients of the object's construction state. The object will be flagged as constructed if its model data is flagged as initialized, required data objects are constructed[3], and all other necessary construction arguments are valid, such as the random number generator procedure pointer being `associated`. Clients may also determine the end-state of simulations completed by the standard DCM using the returned results object. The end-state of the simulation, contained in the results object as `simState`, flags if any warnings or errors had occurred during the simulation. More information on the `simState` flag may be found in Sec. E.7.

[3]The standard DCM will attempt to construct objects passed in by clients that have not been constructed using default arguments in cases the object does not rely on other model objects.

Appendix F. Miscellaneous Model Documentation

F.1 EVAPORATION AND FISSION MODEL OPTIONS

F.1.1 Evaporation Parameters

F.1.1.1 Inverse Reaction Cross Section Parameters

The evaporation model has available to it four parameter sets for the estimation of inverse cross sections. The simple and precise parameter sets contain the variables b, c_j, c_n, k_j, R_b, and R_c for each particle type regarding Eqn. II.66 and Eqn. II.67. The simple parameter set utilizes the values $c_j = c_n = k_j = 1$, $b = 0$, and $R_b = R_c = r_0\left(A_j^{1/3} + A_d^{1/3}\right)$ *[fm]*, where r_0 is a required input.

The precise parameter set is used by default in the evaporation model, where parameters defined in Ref. [76] are used for light ejectiles up to ^4He and the parameters defined in Ref. [78] are used for fragments heavier than ^4He. Values for the precise parameter set were chosen by fitting the expression to the theoretical calculation performed in Ref. [128] and Ref. [129] to account for the overlapping wave functions. Light ejectiles utilize the following values:

$$
\begin{aligned}
c_n &= 0.76 + 1.93A_d^{-1/3} \\
b &= \begin{cases} \left(\frac{1.66A_d^{-2/3} - 0.05}{0.76 + 1.93A_d^{-1/3}}\right), & \text{for } A_d < 192 \\ 0, & \text{for } A_d \geq 192 \end{cases} \\
c_p &= 1 + c \\
c_d &= 1 + c/2 \\
c_t &= 1 + c/3 \\
c_{^3He} &= 0 \\
c_\alpha &= 0 \\
k_p &= k \\
k_d &= k + 0.06 \\
k_t &= k + 0.12 \\
k_{^3He} &= k_\alpha - 0.06
\end{aligned}
$$

The values of c, k, and k_α are given in Table VI.I for ranges of daughter nuclei proton numbers, Z_d. Nuclear distances for the Coulomb barrier are expressed as $R_c = R_d + R_j$, where $R_d = 1.7A^{1/3}$ and $R_j = 0$ for neutrons and protons, and $R_j = 1.2$ for *d, t,* ^3He, and ^4He particles.

TABLE VI.I. The Parameters *c, k*, and k_α for the Precise Inverse Cross Section Parameter Set Used in the Evaporation Model

Z_d	k	k_α	c
≤ 20	0.51	0.81	0.00
30	0.60	0.85	−0.06
40	0.66	0.89	−0.10
≥ 50	0.68	0.93	−0.10

Ejectiles heavier than ^4He use parameters established by Ref. [78] within the precise parameter set. The parameters utilized for these heavier ejectiles are as follows:

$$
\begin{aligned}
c_j &= 1 \\
k &= 1 \\
R_0(A) &= 1.12A^{1/3} - 0.86A^{-1/3} \\
R_b &= R_0\left(A_j\right) + R_0(A_d) + 2.85\ [fm] \\
R_c &= R_0\left(A_j\right) + R_0(A_d) + 3.75\ [fm]
\end{aligned}
$$

An additional set of parameters, developed by Furihata in Ref. [77], can also be used for light ejectiles, and the parameters from the precise set for heavier ejectiles. This parameter set, a hybrid of Furihata and Atchison parameterizations, utilizes k_j as the following:

$$k_j = c_1 log(Z_d) + c_2 log(A_d) + c_3$$

The values for c_1, c_2, and c_3 are given in Table VI.II for fragments up to ^4He.

TABLE VI.II. The Parameters c_1, c_2, and c_3 for the Inverse Cross Section Parameter Set Established in Ref. [77]

Ejectile	c_1	c_2	c_3
p	0.0615	0.0167	0.3227
d	0.0556	0.0135	0.4067
t	0.0530	0.0134	0.4374
^3He	0.0484	0.0122	0.4938
^4He	0.0468	0.0122	0.5120

The nuclear radii used in the calculation of the Coulomb barrier, V, and the geometric cross section, σ_g, with this parameter set is also defined separately below.

$$R = \begin{cases} 0.00, & \text{for } A = 1 \\ 1.20, & \text{for } 2 \leq A \leq 4 \\ 2.02, & \text{for } 5 \leq A \leq 6 \\ 2.42, & \text{for } A = 7 \\ 2.83, & \text{for } A = 8 \\ 3.25, & \text{for } A = 9 \\ 1.414A_d^{1/3} + 1, & \text{for } A \geq 10 \end{cases}$$

Lastly, inverse cross sections may use a hybrid combination of the parameters by Ref. [77] for light fragments, up to ^4He, and by Botniva, in Ref. [79], for all heavier fragments to be calculated. It is important to note that Ref. [79] defines the inverse reaction cross section as shown in Eqn. F.1.

$$\sigma_{inv} = \sigma_g \times \begin{cases} 1 - \frac{V}{\epsilon}, & \text{for } \epsilon \geq V + 1 \ [MeV] \\ \frac{1}{V+1} e^{\alpha(\epsilon - V - 1)}, & \text{for } \epsilon < V + 1 \ [MeV] \end{cases} \tag{F.1}$$

Where the values of α and the Coulomb barrier V, with nuclear radius r_0^b, are defined by the following:

$$\begin{aligned} \alpha &= 0.869 + \frac{9.91}{Z_j} \\ V &= \frac{Z_j Z_d}{r_0^b \left(A_j^{1/3} + A_d^{1/3} \right)} \\ r_0^b &= 2.173 \frac{1 + 0.006103 \times Z_j Z_d}{1 + 0.009443 \times Z_j Z_d} \ [fm] \end{aligned}$$

It is important to note that during the fusion reaction in the sub-barrier region, $\epsilon < V + 1$, the total decay width for a fragment can not be calculated analytically and requires more CPU time for

its computation. The use of inverse cross section parameters are controlled by the *rcal* variable. Table VI.III depicts the value that *rcal* assumes for each of the parameterizations listed.

TABLE VI.III. *rcal* Value for Inverse Cross Section Parameters

rcal Value	Parameterization Used
$= 0.0$	Precise Parameter Set (default)
$1.0 \leq rcal < 10.0$	Simple Set ($r_0 = rcal$)
$= 10.0$	Simple Set ($r_0 = 1.5$)
$= -1.0$	Furihata-Botniva Parameter Set
$< 0.0, \neq -1.0$	Furihata-Atchison Parameter Set

F.1.1.2 Level Density Parameters

The level density parameter a, shown in Eqn. II.69, has available a few options for its determination. The Gilbert-Cameron-Cook-Ignatyuk (GCCI) parameterization from LAHET [74] is utilized by default. The GCCI parameterization utilizes the level density a as:

$$
\begin{aligned}
a &= \tilde{a}\frac{1-e^{-u}}{u} + a_I\left(1 - \frac{1-e^{-u}}{u}\right) \\
u &= 0.05\,(E-\delta) \\
a_I &= \left(0.1375 - 8.36\times 10^{-5} A_d\right) A_d \\
\tilde{a} &= \begin{cases} A_d/8, & \text{for } Z_d < 0 \text{ or } N_d < 9 \\ A_d\,(a' - 0.00917S)\,, & \text{otherwise} \end{cases} \\
a' &= \begin{cases} 0.12, & \text{for deformed nuclei; } \textit{i.e.} \\ & \quad Z_d \,\epsilon\, [54,78] \text{ or } Z_d \,\epsilon\, [86,98] \\ & \quad \text{or } N_d \,\epsilon\, [86,122] \text{ or } N_d \,\epsilon\, [130,150] \\ 0.142, & \text{otherwise} \end{cases} \\
S &= S\,(Z_d) + S\,(N_d)
\end{aligned}
$$

Where S represents shell corrections from neutrons and protons [51, 76, 83, 84]. Additionally, users may specify that the level density is parameterized by a simple ratio, $a = A_d/a_0$, where users supply the value of a_0. By default, $a_0 = 8$. The variable *alev* is used to control the selection of the level density parameter set. Table VI.IV lists the value that *alev* takes for each parameterization.

TABLE VI.IV. *alev* Value for Level Density Parameters

alev Value	Parameterization Used
= 0.0	GCCI Parameterization (default)
= 1.0	$a = A_d/8$
> 1.0	$a = A_d/alev$

F.1.1.3 CPU Time Saving Simulation

The evaporation model discussed here requires more CPU time than other evaporation physics models due to being able to evaporate fragments heavier than ^4He. To help reduce the evaporation model's required computation time, emission of only *n, p, d, t,* ^3He, and ^4He will be considered if specific criteria are met. This simplification can be made because of the low emission probability for heavier nuclei. A random number is selected and emission of ejectiles heavier than ^4He is neglected according to the criteria shown in Table VI.V. Use of the CPU time saving calculation is, by default, turned off. The *ired* variable can be used to control this behavior. Table VI.VI shows what values of *ired* determine use of the CPU time saving option.

TABLE VI.V. Criteria for Heavy Ejectile Emission Rejection from Compound Nuclei

Random Number, x	A_c Criteria
$x < 0.95$	$A_c > 40$
$x < 0.93$	$30 < A_c \leq 40$
$x < 0.70$	$20 < A_c \leq 30$

TABLE VI.VI. *ired* Value for Use of Reduced Computation Time

ired Value	Evaporation Model Execution
≤ 0.0	Use standard simulation (default)
> 0.0	Use reduced computation option

F.1.2 Fission Parameters

The fission model has available to it parameterizations by both Atchison and Furihata to characterize fission. The *ifis* variable is utilized to allow users to specify use of Atchison's original parameterization or the updated, GEM2 parameterization. Table VI.VII shows what values of *ifis* are used to characterize the fission process.

TABLE VI.VII. *ifis* Value for Determination of Fission Parameterization

ifis Value	Fission Model Parameterization
≤ 0.0	Furihata's Updated parameterization (default)
> 0.0	Original Atchison parameterization