CLASSE
Cornell Laboratory for Accelerator-based ScienceS &Education

The Center for BRIGHT BEAMS
A National Science Foundation Science & Technology Center

Brookhaven National Laboratory

U.S. DEPARTMENT OF ENERGY

Programming with the Bmad Toolkit

David Sagan

Cornell ERL / EIC group

Advisor: Georg Hoffstaetter de Torquat

@BrookhavenLab

Introduction to Bmad

What is Bmad?

Bmad is an ecosystem of:

- Open-source toolkits (software libraries) and
- Programs constructed with the toolkits.

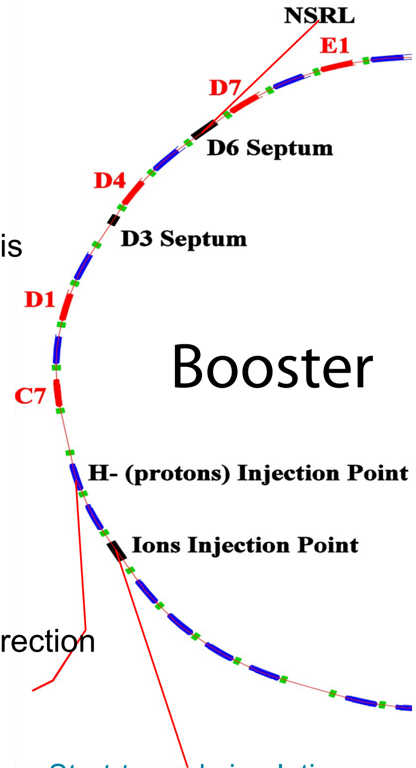
Bmad Simulations

Bmad has been used to study:

- Lattice design
- Space charge simulations including cathode effects.
- Beam breakup (BBU) simulations
- Coherent Synchrotron Radiation (CSR)
- Halo studies
- Microbunching evaluation
- Machine online modeling
- Spin tracking
- Intra Beam Scattering (IBS)
- Touschek scattering
- Wakefields

- Weak-strong beam-beam studies
- Phase noise on Crabbing dynamics
- Feedback systems
- Energy ramping
- Bunch merging
- Electron cooling
- Resonant extraction
- Spin matching
- Spin resonance studies
- Invariant spin field calculations
- Dynamic aperture
- Tune scans plots

- Frequency map analysis
- Long term tracking
- Stripper foils
- Positron converters
- Injection studies
- Cathode laser shaping
- Orbit correction
- Twiss and coupling correction
- X-ray simulations
- Resonance strengths
- Normal form analysis
- Etc., Etc.



Start-to-end simulations: Bmad can simulate an entire accelerator complex including injection lines, extraction lines, dual colliding beam rings, etc.

Bmad Community

Bmad is open source (hosted on GitHub) and has a thriving community with a SLACK workspace for communication and regular schools and training workshops.

Next workshop at BNL July 29th – August 2.

This has enabled people at numerous labs to be able to use Bmad to simulate many machines:

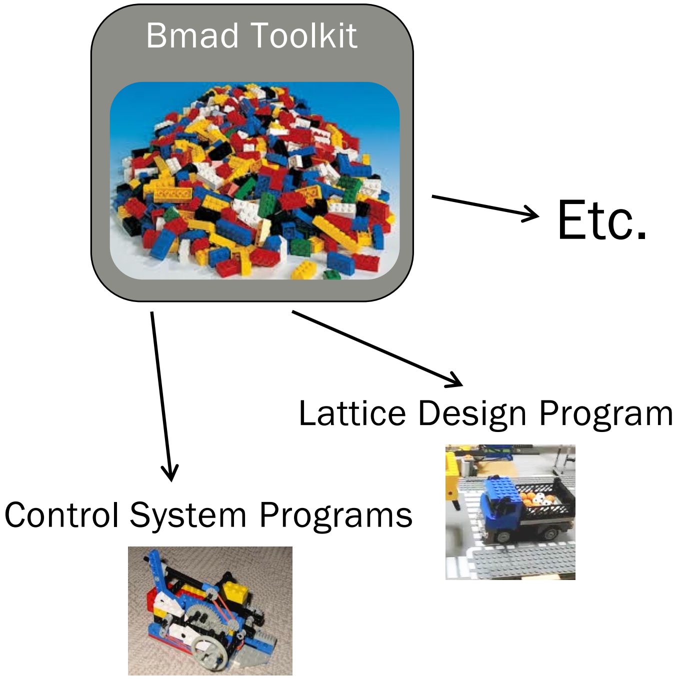
- ✓ Cornell CESR ring
- ✓ CORNELL CESR injection chain.
- ✓ CBETA - Cornell/BNL ERL
- ✓ CERN FCC
- ✓ CERN LHC
- ✓ Julich COSY ring
- ✓ International Linear Collider (ILC)
- ✓ BNL EIC
- ✓ BNL SSRL
- ✓ BNL RHIC
- ✓ Fermilab G-2
- ✓ Fermilab Main Injector
- ✓ KEK SuperKEK-B
- ✓ SLAC LCLS-II
- ✓ Budker VEPP-4M
- ✓ China CEPC
- ✓ Beijing High Energy Photon Source (HEPS)
- ✓ TRIUMF
- ✓ Spallation Neutron Source (SNS)
- ✓ JLab CEBAF
- ✓ JLab FEL
- ✓ Frascati linear accelerator
- ✓ Paris Synchrotron Soleil
- ✓ ... etc ...

Bmad Toolkits

How can Bmad simulate so many different things?

Compared to developing from scratch, the Bmad toolkits allow for the development of simulation programs

- ✓ In less time
- ✓ With fewer bugs (due to module reuse).
- ✓ Enable inter-program data communication (via common lattice and beam format, and other standardizations).
- ✓ Since programs are modular it is easier to adapt them to meet changing simulation needs



Bmad Toolkit

Etc.

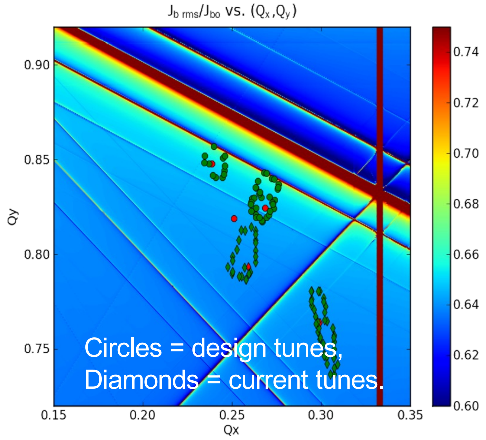
Lattice Design Program

Control System Programs

Bmad Programs

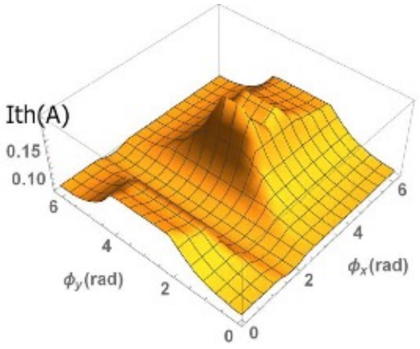
As a result of Bmad's modular structure, a number of simulation programs that use Bmad have been developed:

✓ Tao	-- General purpose simulation program
✓ long_term_tracking	-- Long term tracking program
✓ dynamic_aperture	-- Dynamic aperture program
✓ CesrV	-- Digital Twin for the Cornell CESR storage ring.
✓ CBETA-V	-- Digital Twin for the Cornell/BNL CBETA ERL
✓ bbu	-- RF cavity induced beam breakup instability
✓ synrad3d	-- Synch X-rays tracking within a vac chamber.
✓ ibs_ring	-- Intra beam scattering
✓ tune_scan	-- Tune plane scan
✓ And many more...	



Circles = design tunes,
Diamonds = current tunes.

Tune Scan for CESR Ring Upgrade

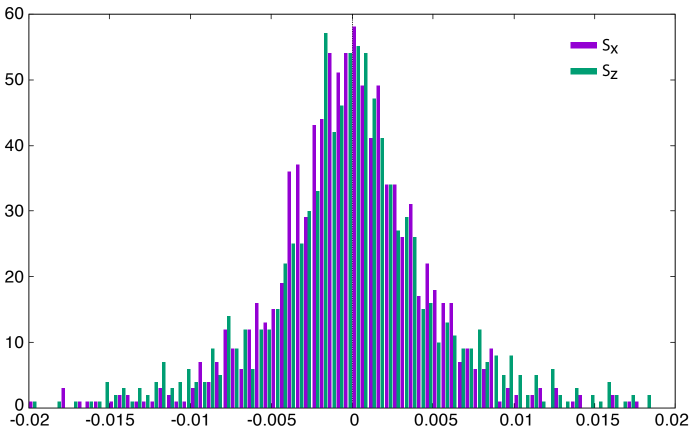


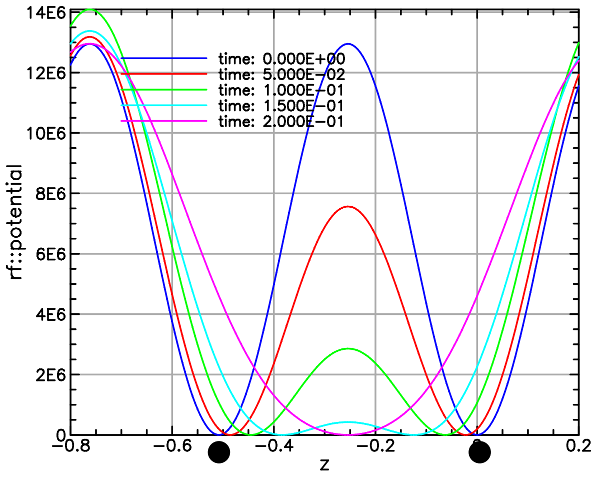
BBU threshold current for CBETA as a function of the phase advance between cavities.

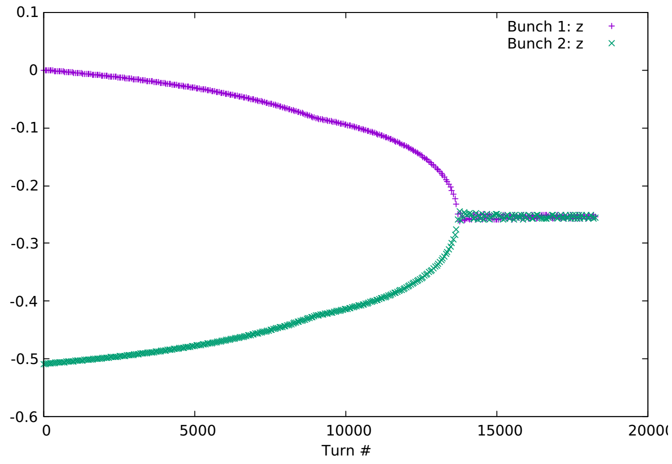
Bmad @ BNL

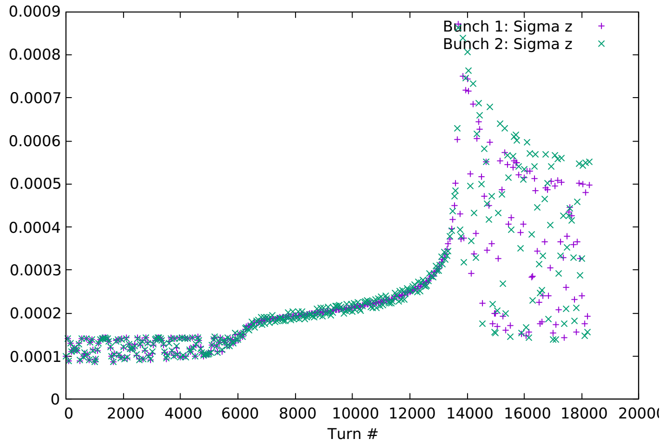
RCS Bunch Merging

Bunch merging while ramping in the RCS.
Simulation includes spin tracking.







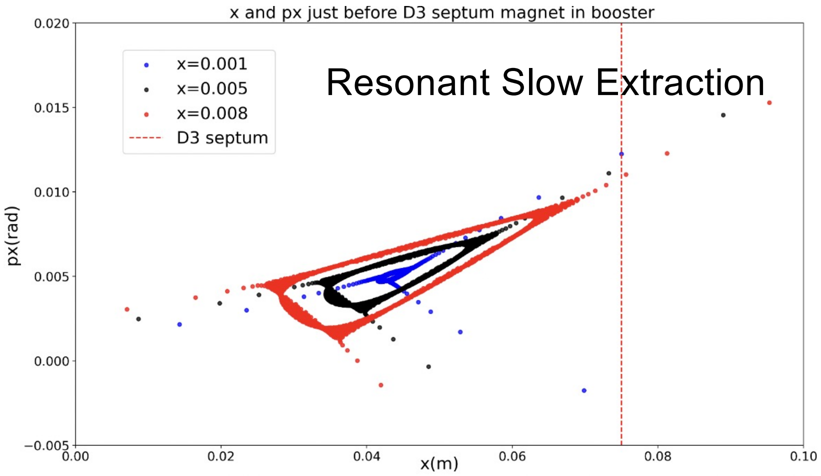


Slow Extraction and AGS Polarization

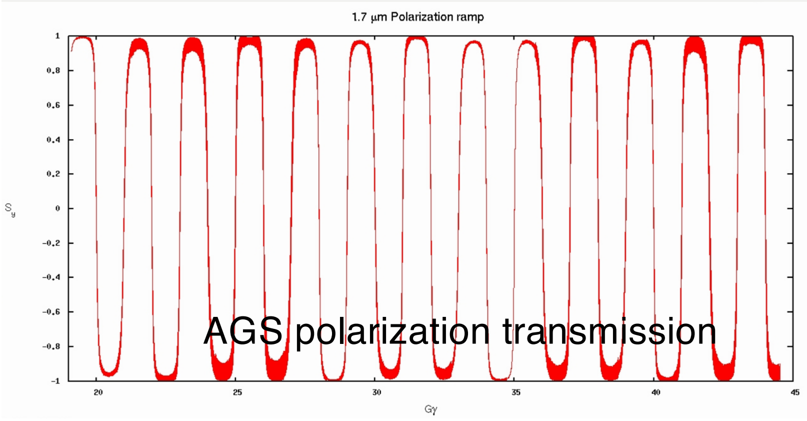
Bmad used for:

- Booster -> NSRL slow extraction
- AGS polarization transmission

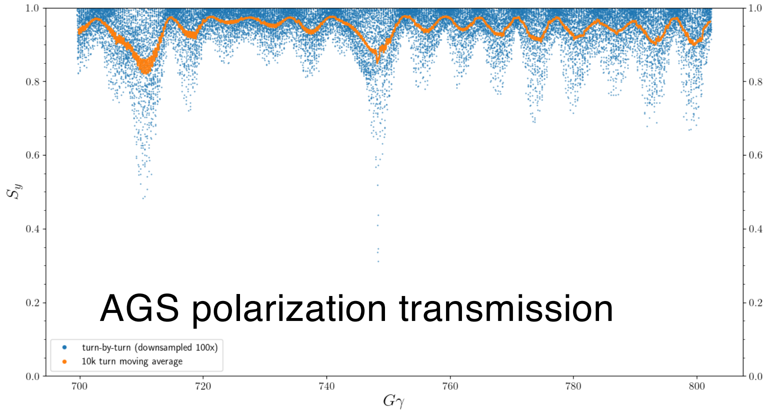
  -- Eiad Hamwi, Cornell



Resonant Slow Extraction



AGS polarization transmission



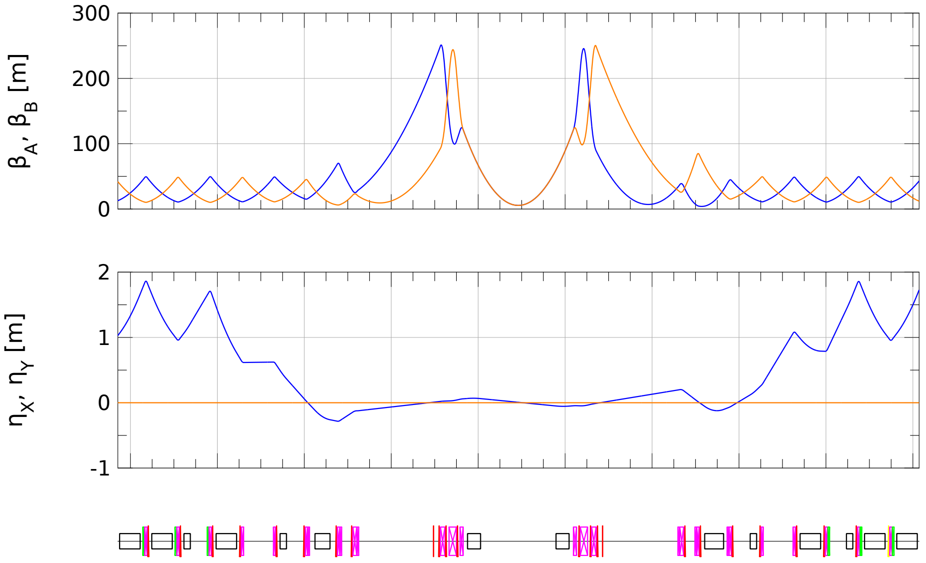
AGS polarization transmission

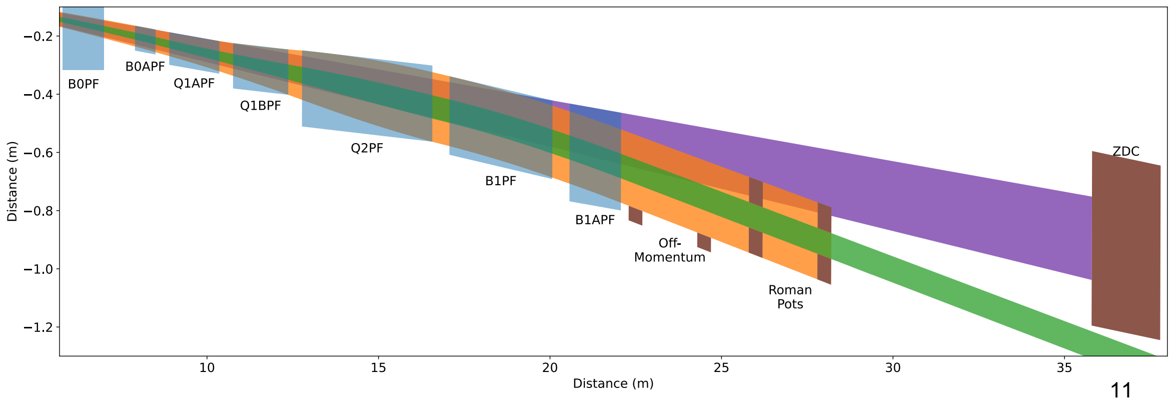
Lattice Design

Bmad used for:

- Interaction region design (ESR and HSR, layout, matching)
- HSR ring design
- Superbend calculations in the ESR (emittance and excursion vs. lengths of dipoles)

-- Scott Berg, BNL

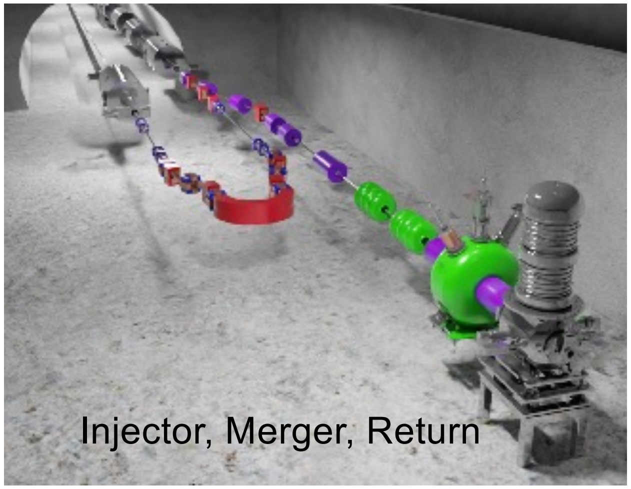




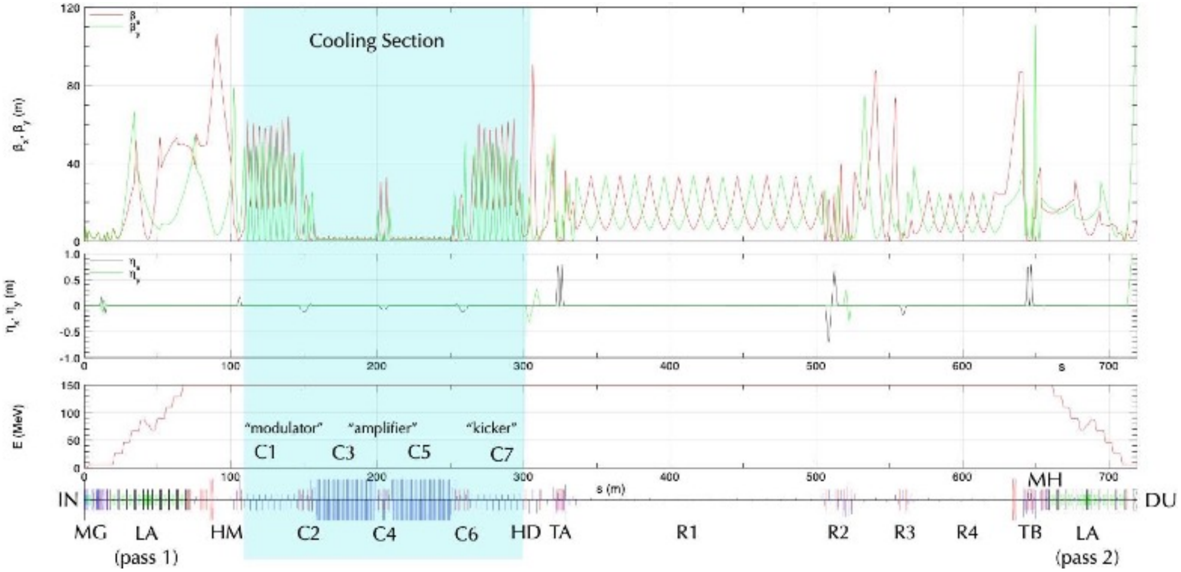
ERL Cooler

- Bmad used extensively for Xelera's SBIR project (through Phase II) to design the EIC ERL cooler, including the precooler.
- Bmad was used for the injector as well as the main lattice, including the ERL multipass optics and start-to-end simulations.

-- Chris Mayes, Xelera Research LLC.



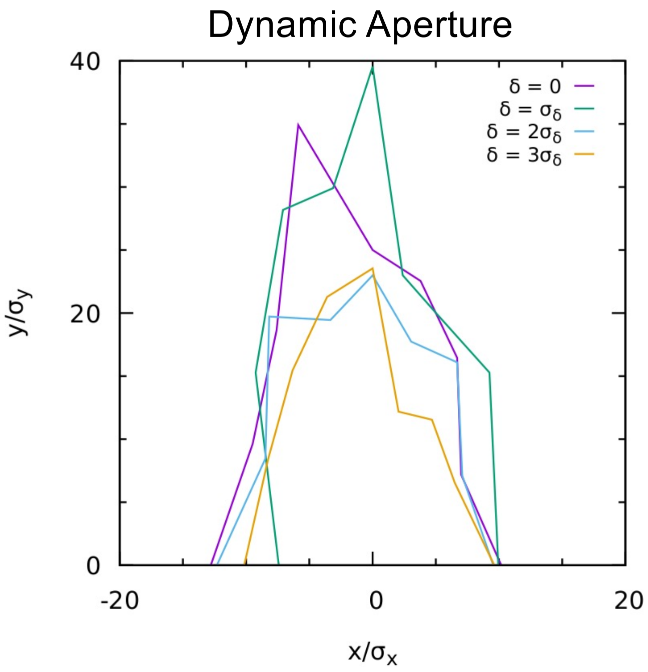
Injector, Merger, Return



Detector Solenoid Integration in the ESR and HSR

“Detector solenoid integration faces unique requirements at the EIC due to a large beam crossing angle and a tilt of the ESR plane. One has to simultaneously account for orbit excursion, transverse and longitudinal coupling, optics and polarization. Unlike most other codes, Bmad has the tools to consider and correct all of these aspects at the same time. These tools greatly simplify integration of correction elements, design optimization and visualization of the results.”

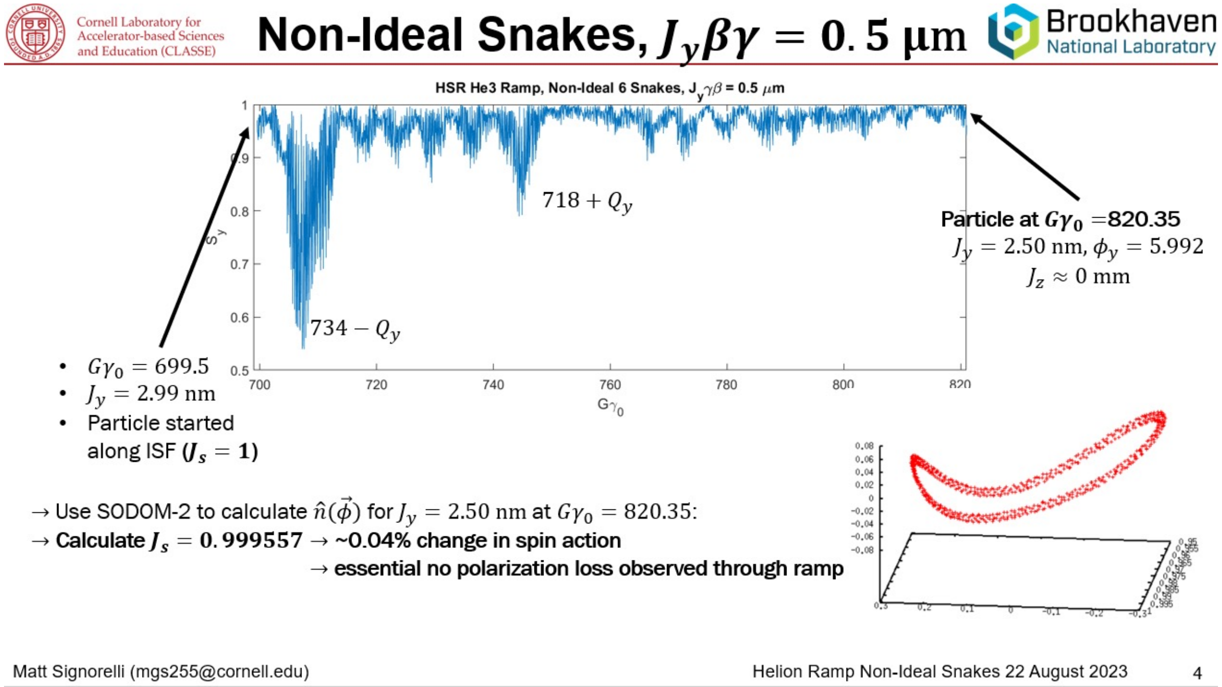
-- Vasiliy Morozov, ORNAL



SODOM-2

- “Last summer, major questions about hadron polarization loss during HSR ramp.
- In 1 week, used Bmad’s routines to implement an Invariant Spin Field (ISF) calculating program SODOM-2.
- Program easily interfaced with long_term_tracking to do ramping and observe polarization loss.
- **Program still used today for polarization calculations/tracking.”**

-- Matt Signorelli, Cornell


Cornell Laboratory for Accelerator-based Sciences and Education (CLASSE)

Non-Ideal Snakes, $J_y\beta\gamma = 0.5\ \mu\text{m}$

Brookhaven National Laboratory

HSR He3 Ramp, Non-Ideal 6 Snakes, $J_y\gamma\beta = 0.5\ \mu\text{m}$

- $G\gamma_0 = 699.5$
- $J_y = 2.99$ nm
- Particle started along ISF ($\boldsymbol{J_s = 1}$)

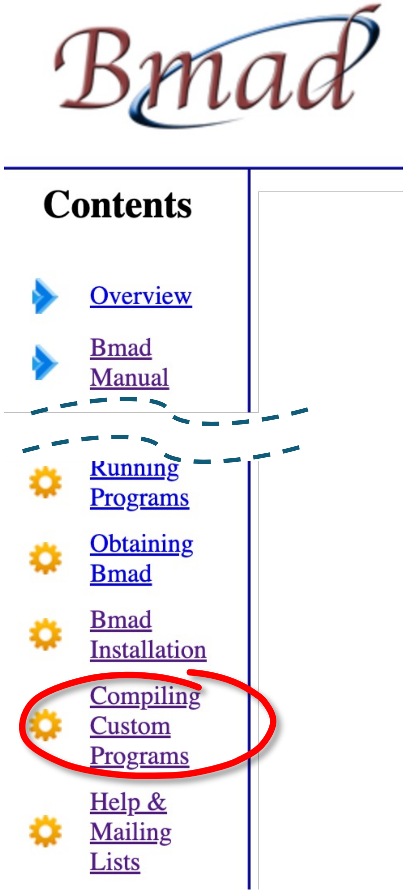
Particle at $G\gamma_0$ =820.35
$J_y = 2.50$ nm, $\phi_y = 5.992$
$J_z \approx 0$ mm

→ Use SODOM-2 to calculate $\hat{n}(\vec{\phi})$ for $J_y = 2.50$ nm at $G\gamma_0 = 820.35$:
→ **Calculate $\boldsymbol{J_s = 0.999557}$ → ~0.04% change in spin action**
→ essential no polarization loss observed through ramp

Matt Signorelli (mgs255@cornell.edu) Helion Ramp Non-Ideal Snakes 22 August 2023 4

Simple Bmad Program

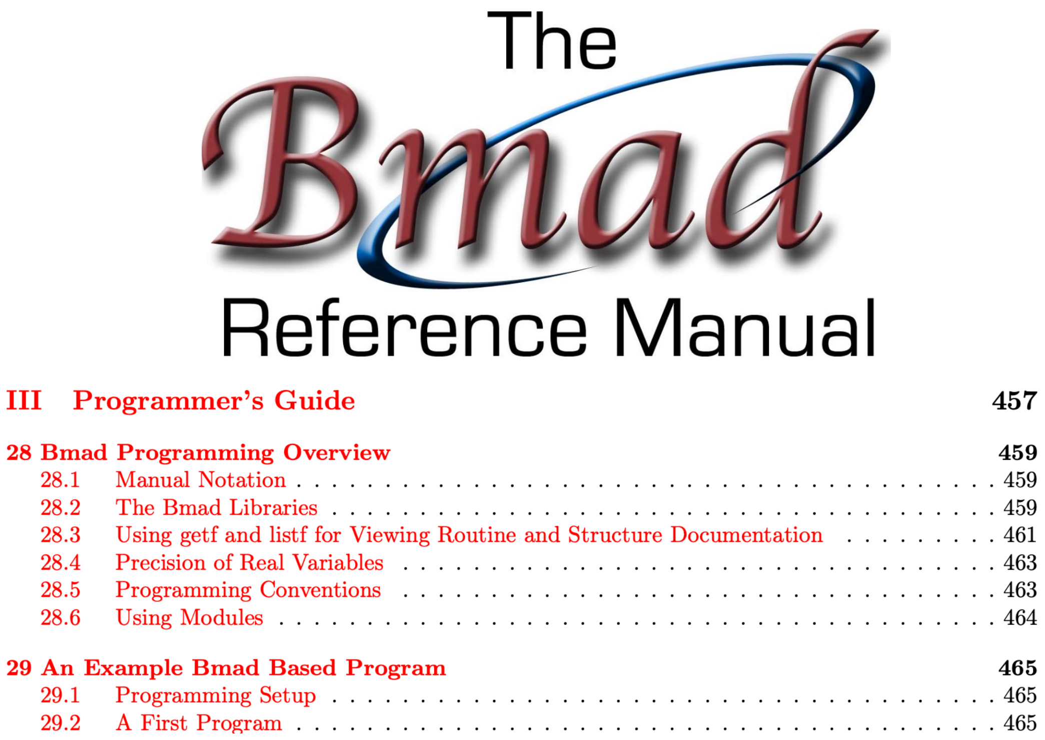
Programming Documentation



Bmad

Contents

- Overview
- Bmad Manual
- Running Programs
- Obtaining Bmad
- Bmad Installation
- Compiling Custom Programs
- Help & Mailing Lists



The Bmad Reference Manual

Code Examples

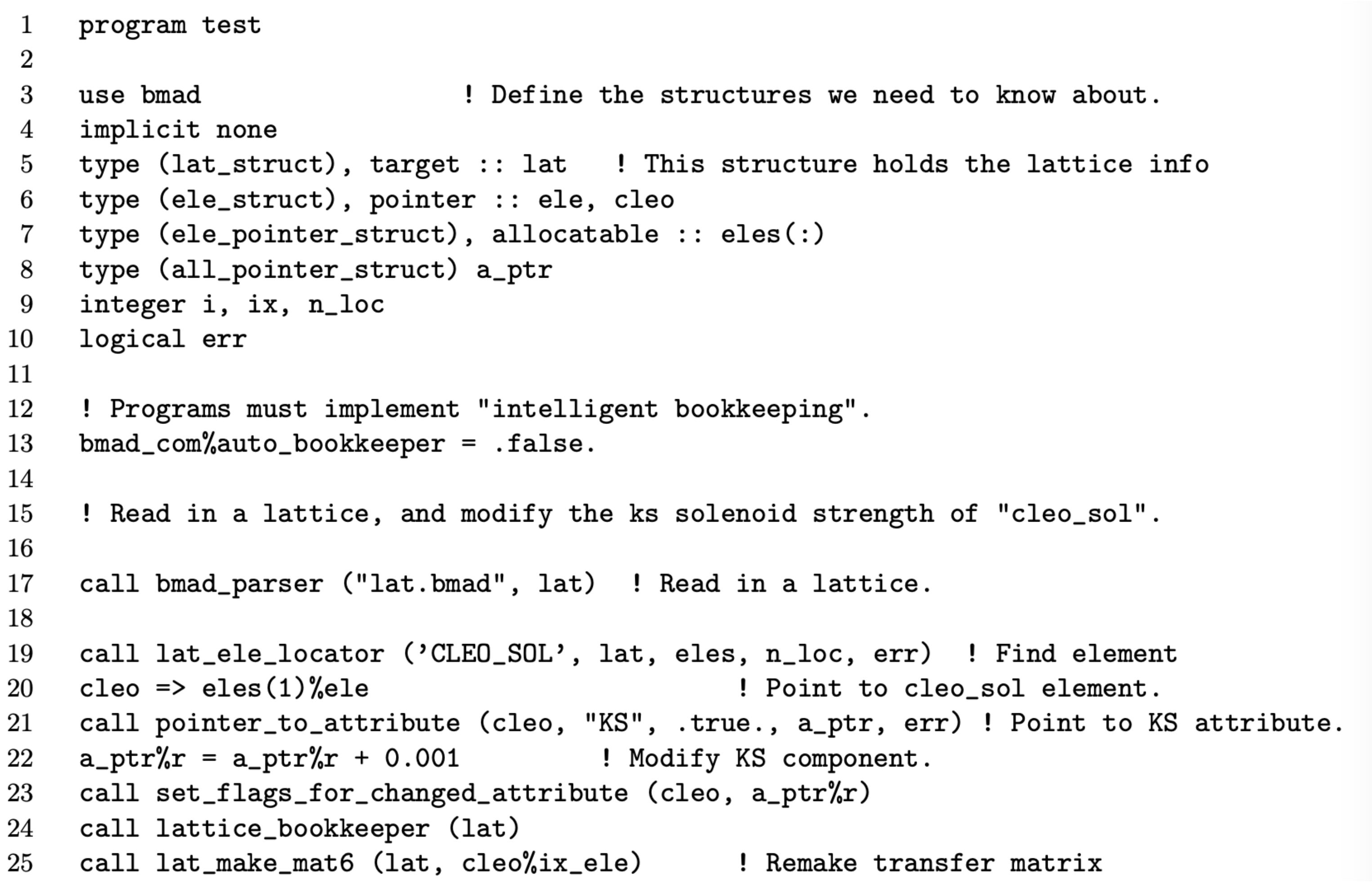
```
~/Bmad/bmad_dist> ls code_examples/

CMakeLists.txt                      mpi_mp
beam_track_example                  multi_turn_tracking_example
bmad_to_opal_example                parallel_track_example
cmake_files                         particle_track_example
cmake_template_scripts              plot_example
coarray_example                     production
construct_taylor_map                ptc_layout_example
csr_example                         ptc_spin_orbital_normal_form
dispersion_simulation               searchf.namelist
em_field_query_example              simple_bmad_program
lapack_examples                     spin_amplitude_dependent_tune
lattice_geometry_example            spin_matching
```

Sanity Check: accinfo

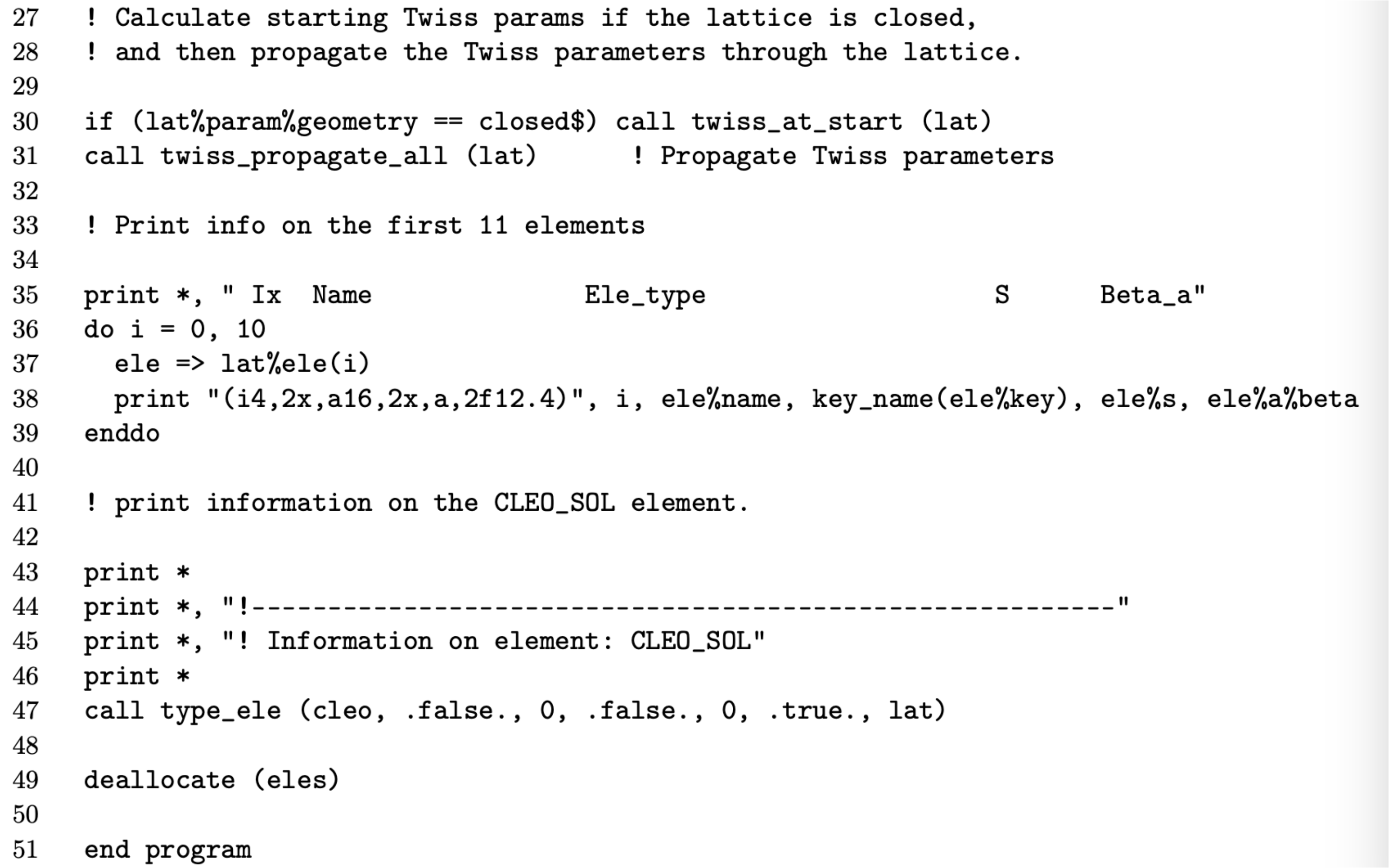
```
MacBook-Pro-3:~/Downloads/bta_lattice_2> accinfo
DIST_DEBUG=/Users/dcs16/Bmad/bmad-ecosystem/debug/bin
DIST_OS=Darwin
DIST_BUILD=TRUE
DIST_F90=gfortran
DIST_UTIL=/Users/dcs16/Bmad/bmad-ecosystem/util
DIST_BASE_DIR=/Users/dcs16/Bmad/bmad-ecosystem
DIST_ARCH=arm64
DIST_EXE=/Users/dcs16/Bmad/bmad-ecosystem/production/bin
DIST_F90_REQUEST=gfortran
DIST_OS_ARCH=Darwin_arm64
ACC_EXE=/Users/dcs16/Bmad/bmad-ecosystem/production/bin
ACC_CMAKE_VERSION=3.13.2
ACC_ENABLE_SHARED_ONLY=Y
… etc …
```

Bmad manual

29.2 A First Program

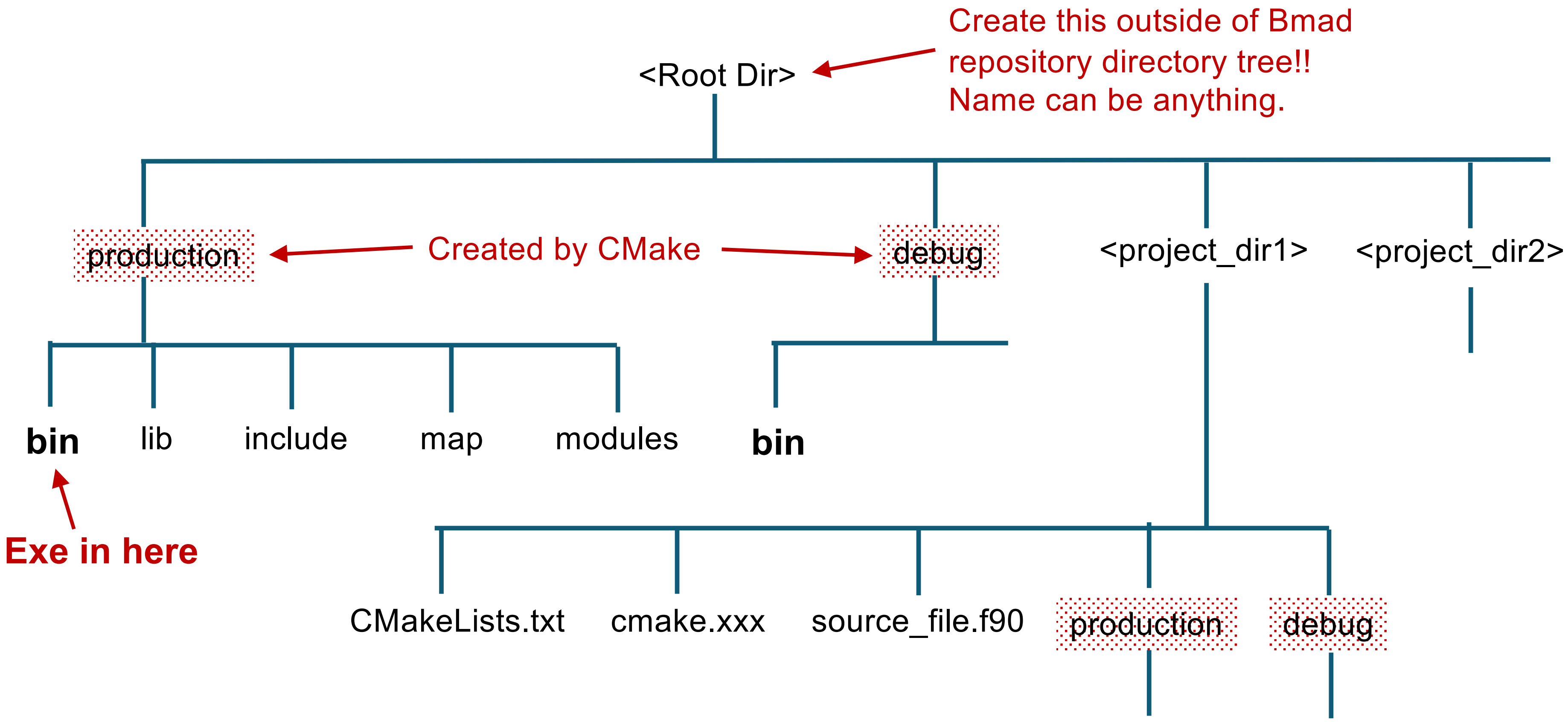


```
program test

use bmad                 ! Define the structures we need to know about.
implicit none
type (lat_struct), target :: lat   ! This structure holds the lattice info
type (ele_struct), pointer :: ele, cleo
type (ele_pointer_struct), allocatable :: eles(:)
type (all_pointer_struct) a_ptr
integer i, ix, n_loc
logical err

! Programs must implement "intelligent bookkeeping".
bmad_com%auto_bookkeeper = .false.

! Read in a lattice, and modify the ks solenoid strength of "cleo_sol".

call bmad_parser ("lat.bmad", lat)  ! Read in a lattice.

call lat_ele_locator ('CLEO_SOL', lat, eles, n_loc, err)  ! Find element
cleo => eles(1)%ele                        ! Point to cleo_sol element.
call pointer_to_attribute (cleo, "KS", .true., a_ptr, err) ! Point to KS attribute.
a_ptr%r = a_ptr%r + 0.001         ! Modify KS component.
call set_flags_for_changed_attribute (cleo, a_ptr%r)
call lattice_bookkeeper (lat)
call lat_make_mat6 (lat, cleo%ix_ele)      ! Remake transfer matrix
```



```
! Calculate starting Twiss params if the lattice is closed,
! and then propagate the Twiss parameters through the lattice.

if (lat%param%geometry == closed$) call twiss_at_start (lat)
call twiss_propagate_all (lat)      ! Propagate Twiss parameters

! Print info on the first 11 elements

print *, " Ix  Name              Ele_type                   S      Beta_a"
do i = 0, 10
  ele => lat%ele(i)
  print "(i4,2x,a16,2x,a,2f12.4)", i, ele%name, key_name(ele%key), ele%s, ele%a%beta
enddo

! print information on the CLEO_SOL element.

print *
print *, "!---------------------------------------------------------"
print *, "! Information on element: CLEO_SOL"
print *
call type_ele (cleo, .false., 0, .false., 0, .true., lat)

deallocate (eles)

end program
```

Directory Setup for Custom Programs



Setup

1. Create <root_dir> directory and <project_dir> subdirectories.
2. Copy files from:

   $DIST_BASE_DIR/code_examples/simple_bmad_program/

   to:

   <project_dir>

```
Files:
   CMakeLists.txt
   README
   cmake.simple_bmad_program
   lat.bmad
   layout.bmad
   simple_bmad_program.f90
```

Cmake Build System

CMake: A Powerful Software Build System

Cmake.org

CMakeLists.txt File

```
cmake_minimum_required(VERSION $ENV{ACC_CMAKE_VERSION})
project(ACC)

set(EXE_SPECS
  cmake.simple_bmad_program
)

include($ENV{ACC_BUILD_SYSTEM}/Master.cmake)
```

Can add to this list to create multiple exes

cmake.simple_bmad_program File

```
set(EXENAME simple_bmad_program)
set (SRC_FILES
  simple_bmad_program.f90
)

set (LINK_LIBS
  bmad
  sim_utils
  ${ACC_BMAD_LINK_LIBS}
)
```

Can glob

Compile Program

```
MacBook-Pro-3:~/Bmad/test/simple> mk

-- The C compiler identification is GNU 12.3.0
-- The CXX compiler identification is GNU 12.3.0
-- Checking whether C compiler has -isysroot
-- Checking whether C compiler has -isysroot - yes
-- Checking whether C compiler supports OSX deployment target flag
-- Checking whether C compiler supports OSX deployment target flag - yes
            … etc., etc. …
-- Build files have been written to: /Users/dcs16/Bmad/test/simple/production
[ 50%] Building Fortran object CMakeFiles/simple_bmad_program-exe.dir/simple_bmad_program.f90.o
[100%] Linking Fortran executable /Users/dcs16/Bmad/test/production/bin/simple_bmad_program
-macosx_version_min has been renamed to -macos_version_min
[100%] Built target simple_bmad_program-exe

/Users/dcs16/Bmad/test/simple/production Compile/Link time: 9.00sec
```

Run Program

```
MacBook-Pro-3:~/Bmad/test/simple> ../production/bin/simple_bmad_program
[INFO] bmad_parser:
    Parsing lattice file(s). This might take a minute or so...
[INFO] bmad_parser:
    Created new digested file
[MESSAGE | 2024-JUL-30 23:07:19] bmad_parser:
    Lattice parse time(min): 0.00
  Ix  Name                 Ele_type                    S      Beta_a
   0  BEGINNING            Beginning_Ele          0.0000      0.9379
   1  IP_L0                Marker                 0.0000      0.9379
   2  CLEO_SOL#3           Solenoid               0.6223      1.3472
               … etc., etc. …
   8  DET_01W              Marker                 2.4934     28.5769
   9  D004                 Drift                  2.9240     48.4524
  10  Q01W                 Quadrupole             3.8740     66.8800
```

Program Output Continued

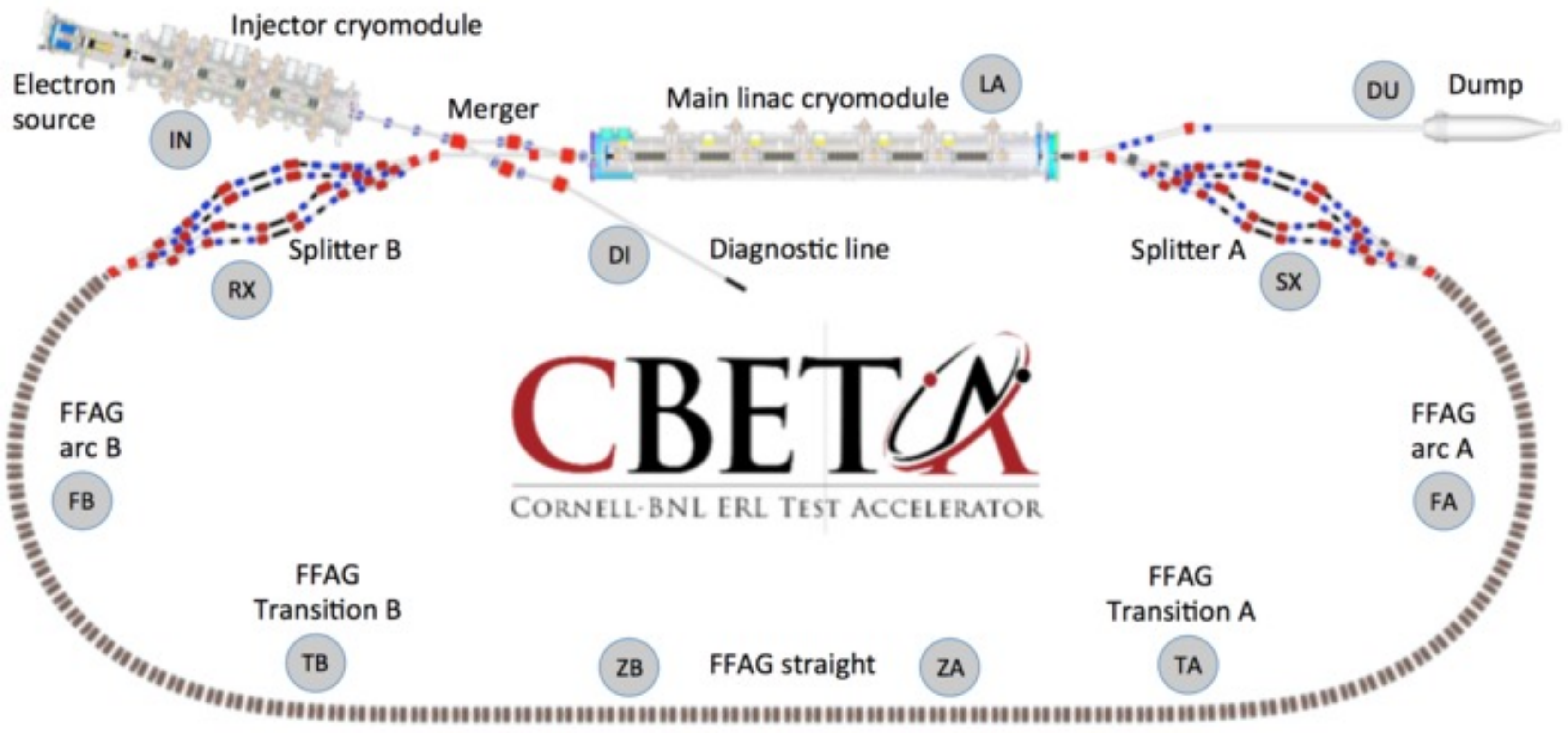
```
!---------------------------------------------------------
! Information on element: CLEO_SOL

Element # 872
Element Name: CLEO_SOL
Key: Solenoid
S_start, S:    766.671421,      1.755000
Ref_time_start, Ref_time:  2.557341E-06,  5.854050E-09

Attribute values [Only non-zero values shown]:
    1  L                       =  3.5100000E+00 m       31  L_SOFT_EDGE        =  0.0000000E+00 m
    3  R_SOLENOID              =  0.0000000E+00 m
    5  KS                      = -8.4023386E-02 1/m     49  BS_FIELD           = -1.4823578E+00 T
   10  FRINGE_TYPE             =  None (1)              11  FRINGE_AT          =  Both_Ends (3)
   13  SPIN_FRINGE_ON          =  T (1)
   17  STATIC_LINEAR_MAP       =  F (0)
   47  PTC_CANONICAL_COORDS    =  T (1)
   53  P0C                     =  5.2890000E+09 eV          BETA               =  1.0000000E+00
                … etc., etc. …
```

Other compile commands

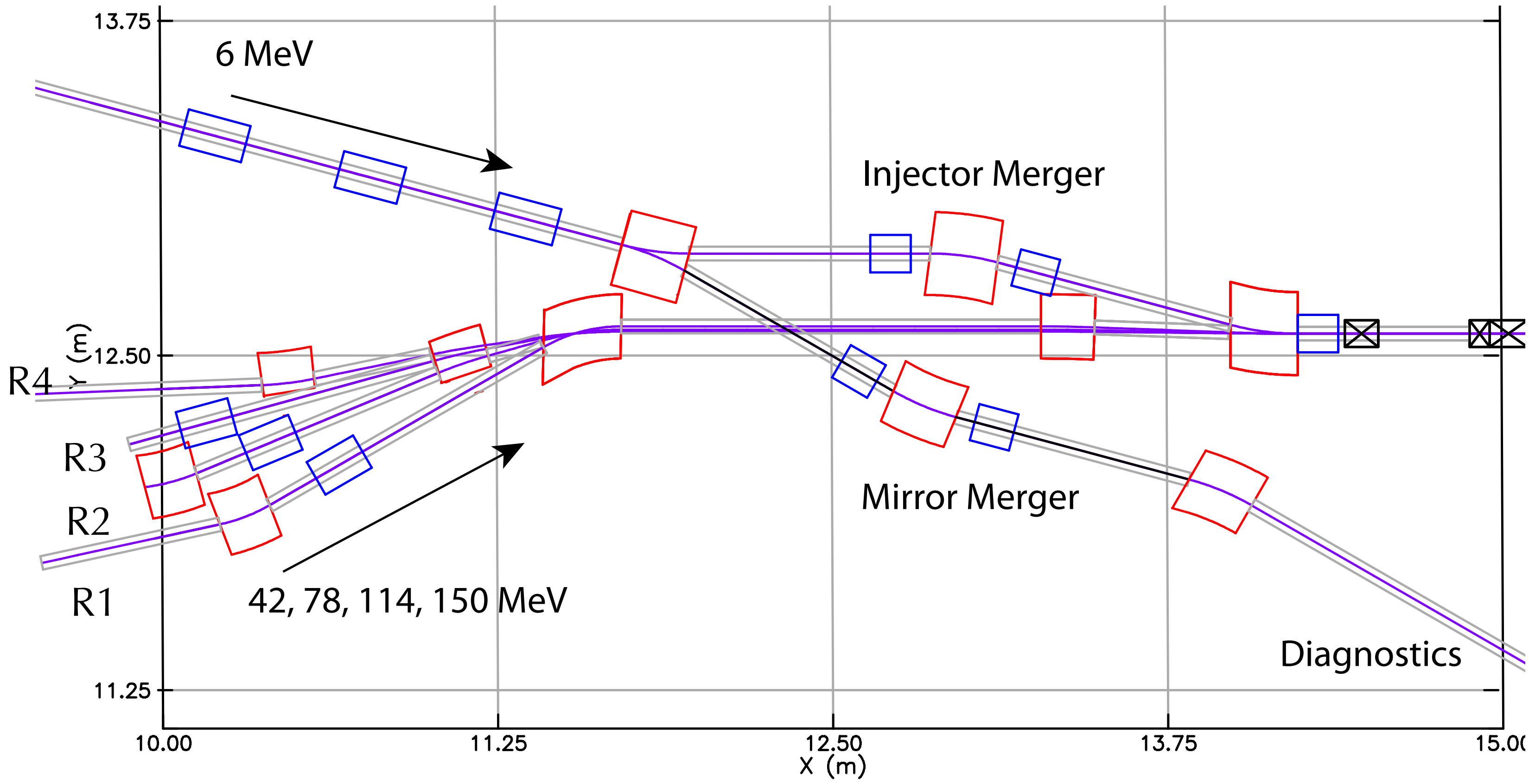
mk cleaner	-- Remove intermediate production files
mkd	-- Compile debug version
mkd cleaner	-- Remove intermediate debug files

Customizing Tao to be a Digital Twin

Cornell-BNL CBETA Machine



Cornell/BNL CBETA Machine Injection Region



Brookhaven National Laboratory

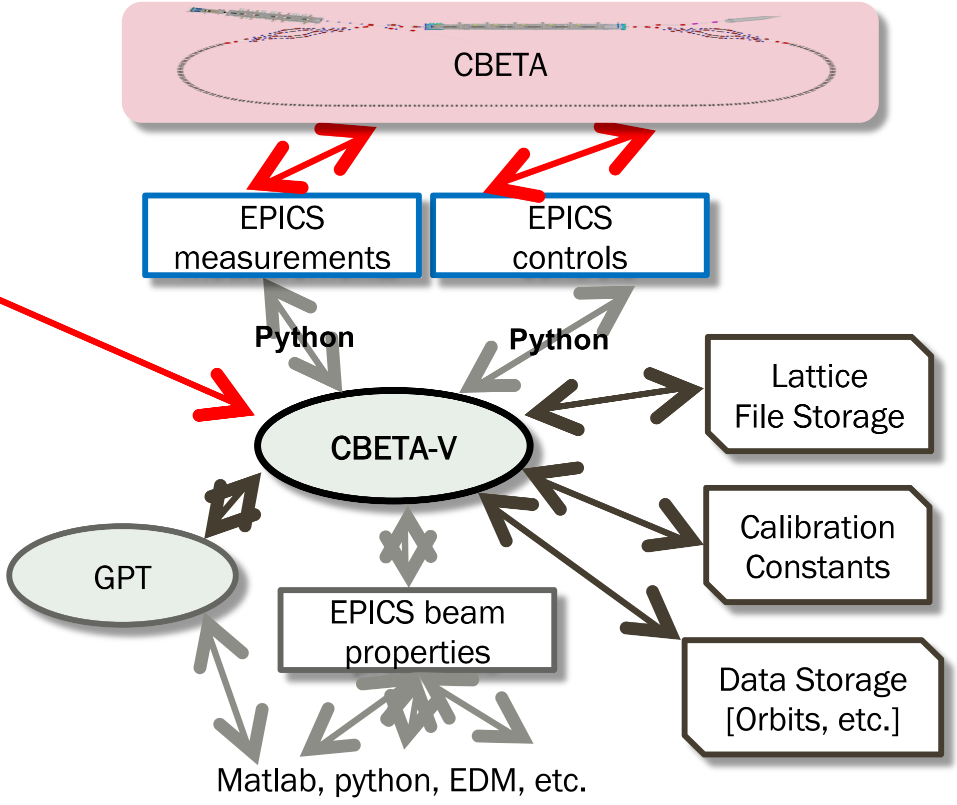
CLASSE Cornell Laboratory for Accelerator-based ScienceS & Education

CBETA Control System Flow Chart

- **CBETA:** Cornell – Brookhaven ERL under development at Cornell.
- **CBETA-V**: Customized version of Tao for online modeling of CBETA.
- **CBETA-V/Tao** is used for:
  - Online modeling.
  - Offline modeling of online system.
  - Lattice design.

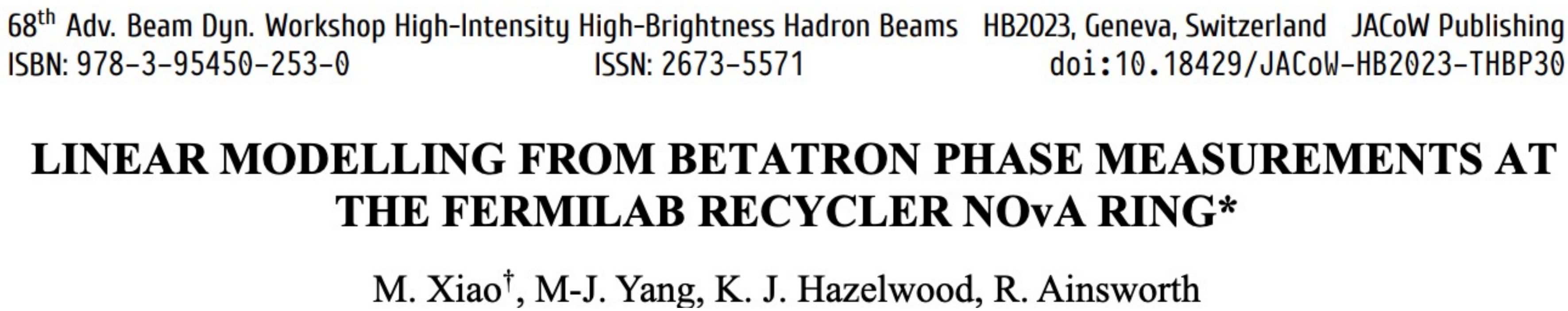
Bottom Line: Using Tao as a starting point for CBETA-V enabled the development of a flexible digital twin in less time and with fewer bugs.

Implementors: Colwyn Gulliford, Adam Bartnik, Scott Berg, David Sagan



[*For illustrative purposes only. Don’t take this too seriously!]

Extending Tao to Model the NOvA Ring



68th Adv. Beam Dyn. Workshop High-Intensity High-Brightness Hadron Beams HB2023, Geneva, Switzerland JACoW Publishing
ISBN: 978-3-95450-253-0 ISSN: 2673-5571 doi:10.18429/JACoW-HB2023-THBP30

LINEAR MODELLING FROM BETATRON PHASE MEASUREMENTS AT THE FERMILAB RECYCLER NOvA RING*

M. Xiao[†], M-J. Yang, K. J. Hazelwood, R. Ainsworth

Setup and Run

1. Copy the files from:

   $DIST_BASE_DIR/bmad-doc/tao_examples/custom_tao_with_measured_data

   to:

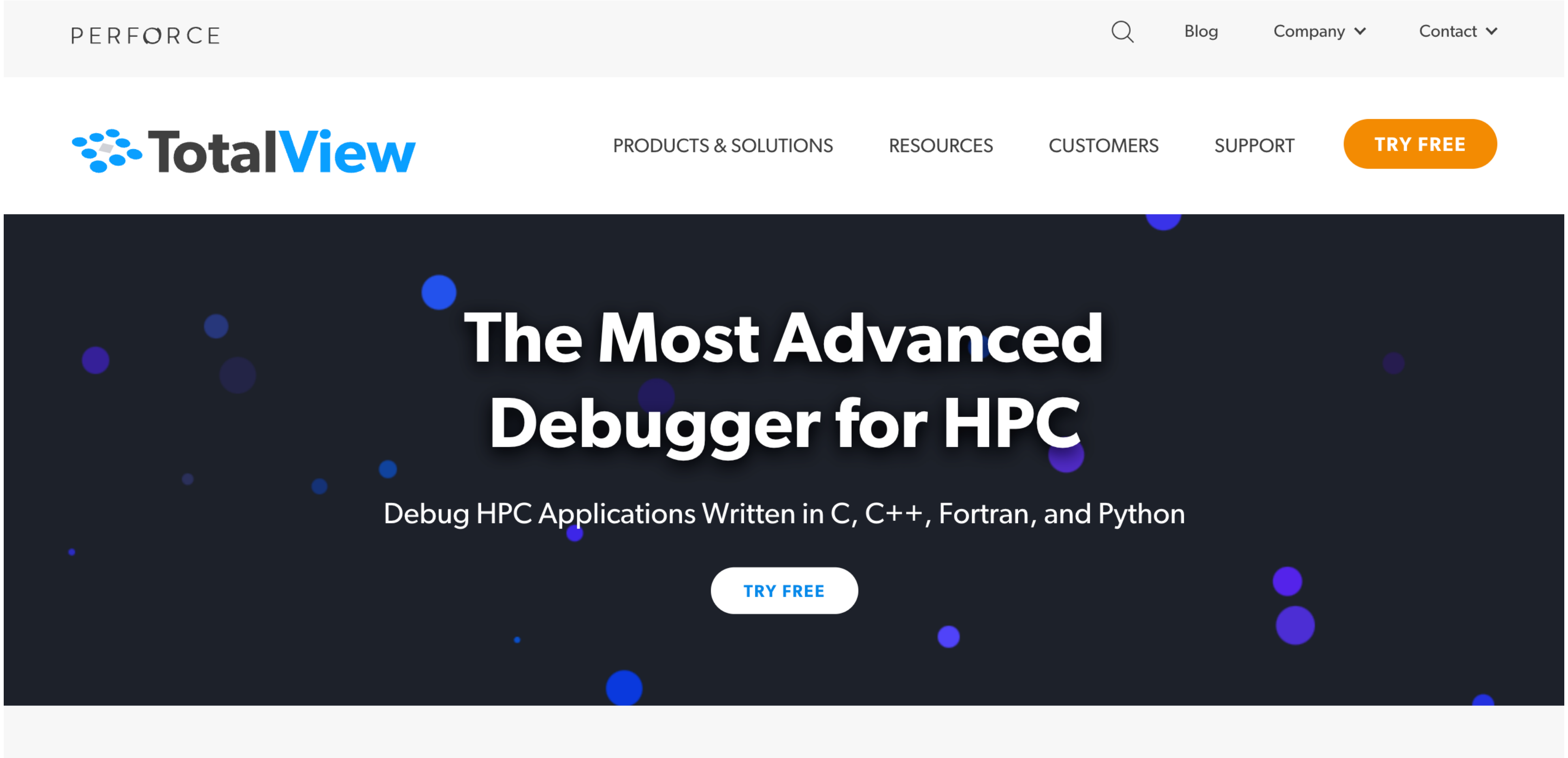
   <project_dir2>

2. mk

3. ../production/bin/ping_tao

Miscellaneous

Totalview Debugger



PERFORCE

Blog Company Contact

TotalView

PRODUCTS & SOLUTIONS RESOURCES CUSTOMERS SUPPORT TRY FREE

The Most Advanced Debugger for HPC

Debug HPC Applications Written in C, C++, Fortran, and Python

TRY FREE

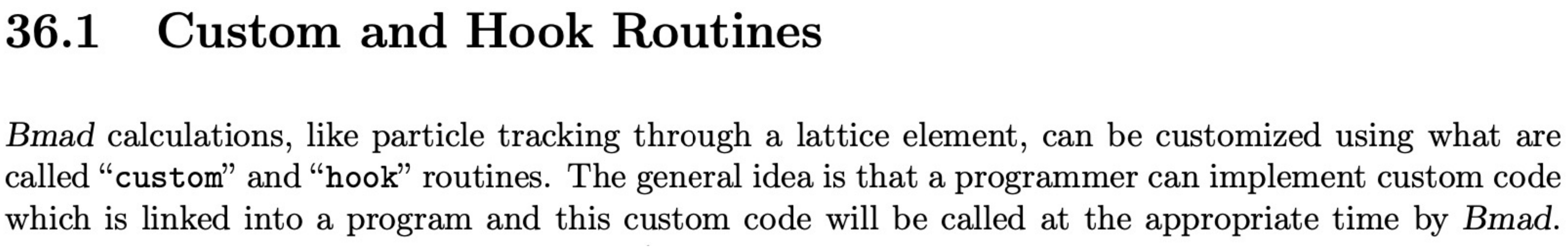
TotalView & NERSC

TotalView by Perforce

TotalView Training - NERSC

MAY 13, 2024

Custom Routines



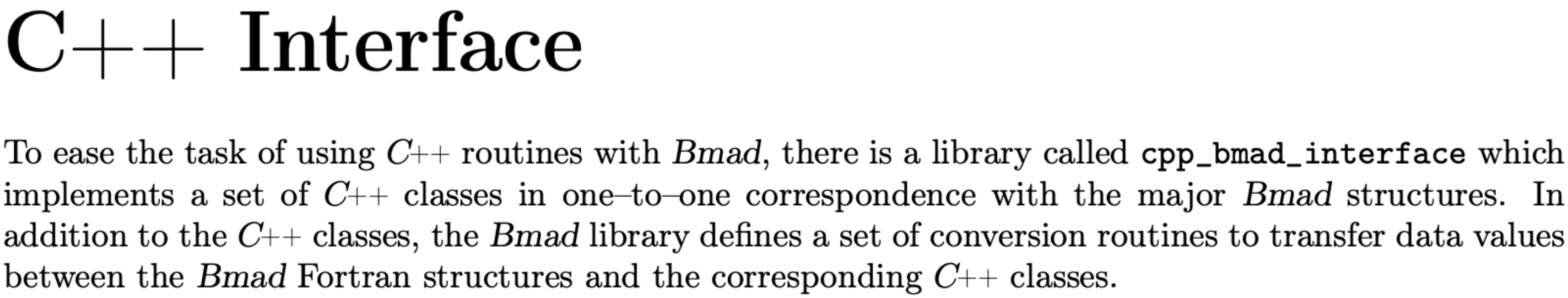
36.1 Custom and Hook Routines

Bmad calculations, like particle tracking through a lattice element, can be customized using what are called “`custom`” and “`hook`” routines. The general idea is that a programmer can implement custom code which is linked into a program and this custom code will be called at the appropriate time by *Bmad*.

```
!+
! Subroutine track1_custom (orbit, ele, param, err_flag, finished, track)
!
! Prototype routine for custom tracking.
!
```

Interface to C++ and Python

Bmad Manual: Chapter 39



C++ Interface

To ease the task of using *C++* routines with *Bmad*, there is a library called **`cpp_bmad_interface`** which implements a set of *C++* classes in one–to–one correspondence with the major *Bmad* structures. In addition to the *C++* classes, the *Bmad* library defines a set of conversion routines to transfer data values between the *Bmad* Fortran structures and the corresponding *C++* classes.

Python structure translation: In development with PyTao (Ken Lauer – XLight)

Thank You