An Optics Package for G4MICE

Chris Rogers

MICE Optics Session

Introduction

- Advertisement for G4MICE Optics code
  - New users/developers welcomed!!
- Motivation
- Requirements
- Design status
- Advanced warning - you will be exposed to technical details
  - But I will assume you don’t know any programming (apologies if I patronise)

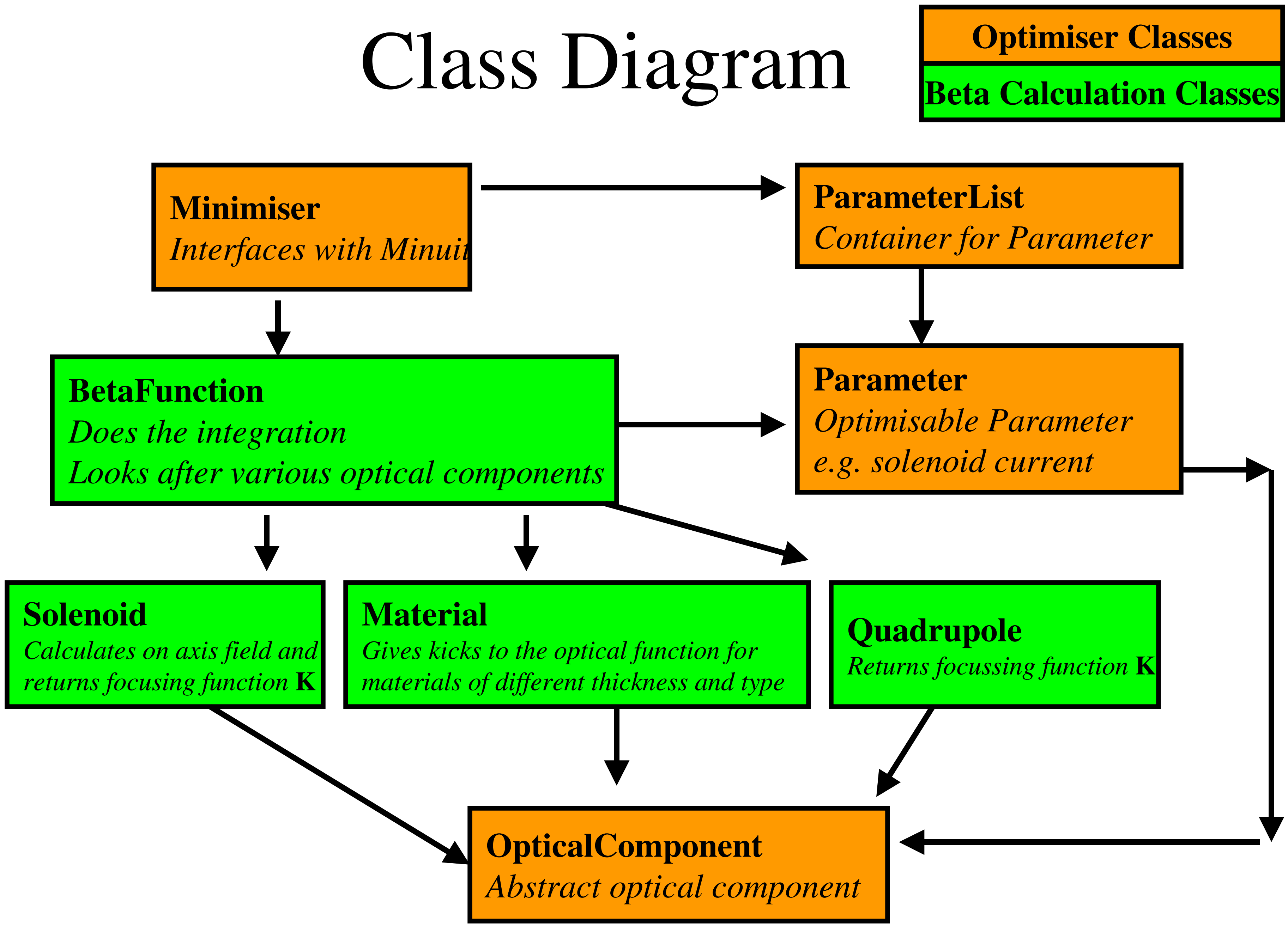
Motivation

- We will need to manipulate beam optics in the MICE analysis
  - E.g. non-linearities/emittance growth
- Integrating with G4MICE provides some nice functions
  - Use G4MICE field maps
- Writing our own package means we can customise functionality
  - RF + Materials + B-Fields in ~ 20 cm
  - Specify sensitivity to emittance changes
- Current design has been used for position diffuser
- Also will be developing longitudinal dynamics

User (My) Requirements

- Add your own to the bottom:
  - Calculate optical functions in solenoids, quads, material, RF
  - Optimise optical function in Beamline
    - Quad positions, currents/field
  - Optimise optical function in MICE itself
    - Optimise solenoid currents
  - Place constraints on beta (& alpha) functions
    - "beta should be 333, alpha 0 in upstream solenoid"
    - "beam shouldn't scrape in the quadrupoles"
  - Eventually… calculate higher order/non-linear terms in optics
    - Get at emittance growth
  - Eventually… ~arbritrary functions for optimisation
  - Other suggestions are welcome!

Class Diagram



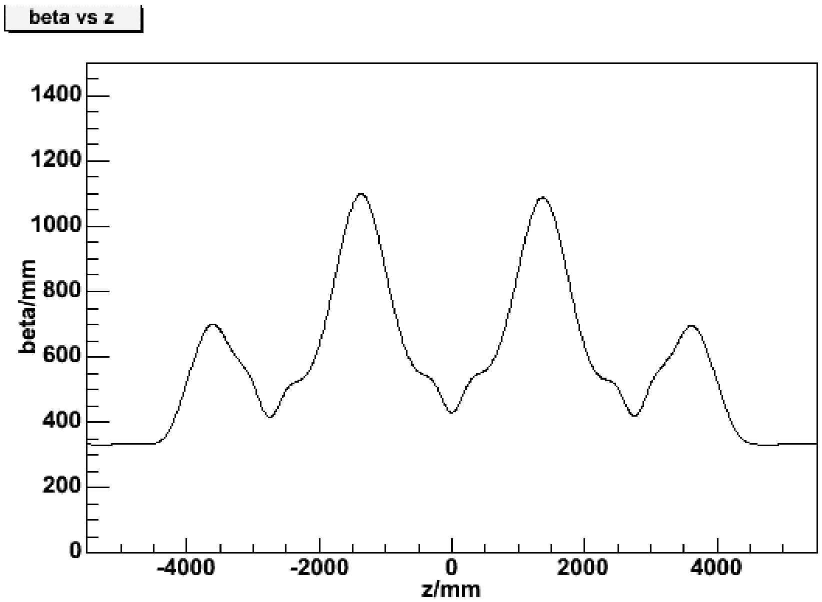
Beta Function Calculation

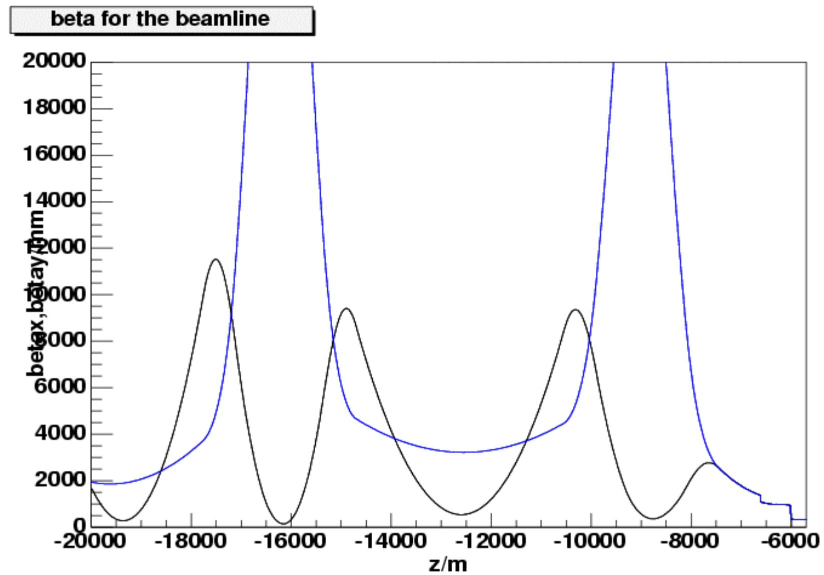
- Hacked from JHC's numerical integration using "Finite difference method"
  - Integrate well known eqn $2\beta\beta'' - (\beta')^2 + 4\beta^2 K^2 - 4 = 0$
  - Faster/more accurate to use Runge-Kutta?
  - Solenoid, Quadrupole provide *K(z)*
    - No fringe fields in quads
- Slightly different in materials
  - Uses things like

$$\varepsilon_{new}{}^2 = \varepsilon_{old}{}^2 + \frac{< d\theta^2 > \beta_{old}\varepsilon_{old} p_z}{m_\mu}$$

  - No kick from B-fields
  - Assumes material is thin (but **not** emittance kick is small)
  - Can step backwards or forwards through the absorber

Examples





- Beta through MICE
  - No material though not difficult to put some in
- Beta through beamline
  - Material, quads, solenoids

Sample output

z   bz   beta   alpha   pz   em   kappa

-5700 0.00384078 333 0 200 6 0.00288059 0 0.00288059

-5690 0.00385122 333 0 200 6 0.00288841 0 0.00288841

-5680 0.00386091 333.045 -0.00224805 200 6 0.00289569 0 0.00289569

-5670 0.0038699 333.132 -0.00434838 200 6 0.00290243 0 0.00290243

-5660 0.00387822 333.258 -0.00630379 200 6 0.00290867 0 0.00290867

…

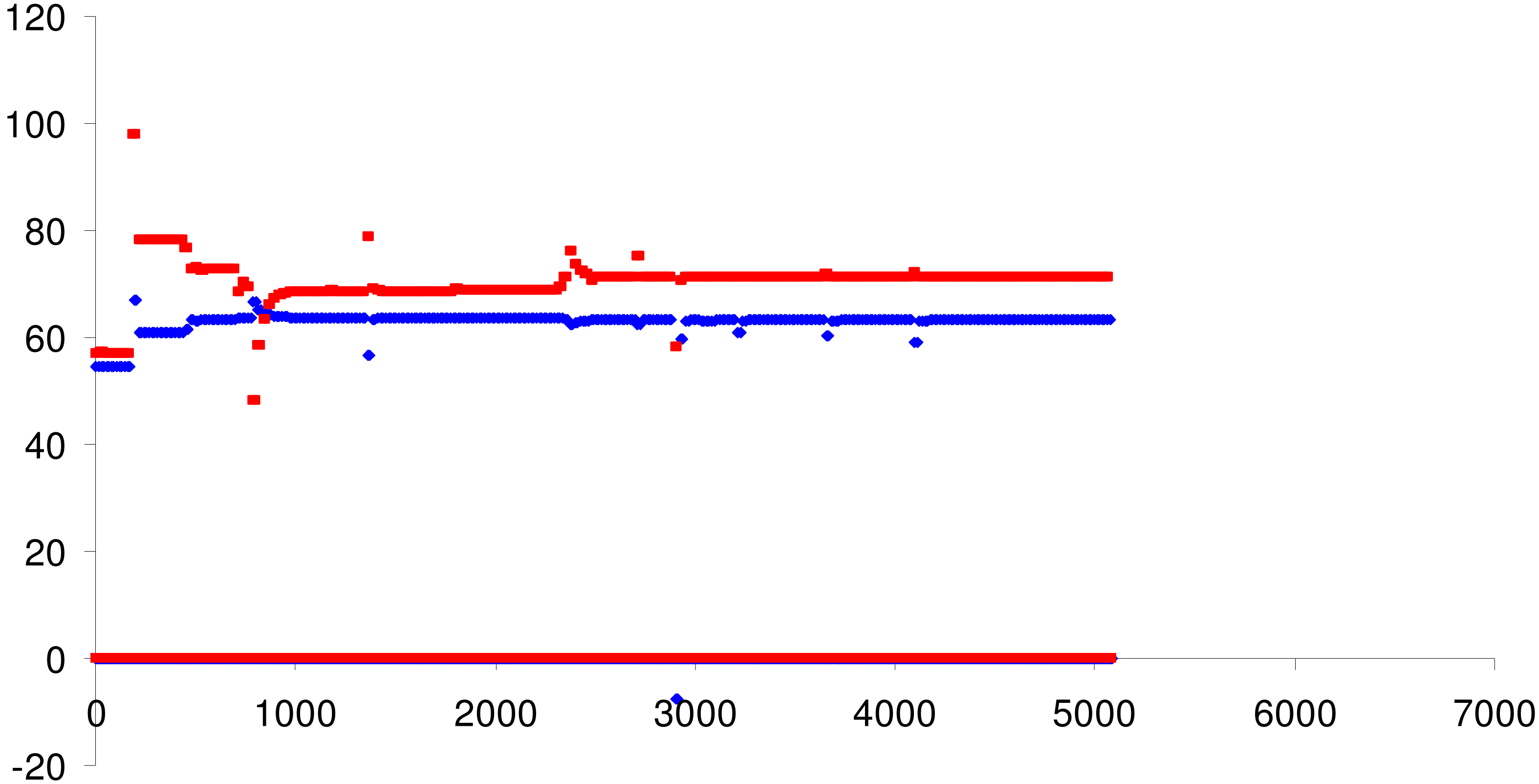
- Output for command beta.print(file)
  - z position
  - on axis B_z
  - beta
  - alpha
  - $\langle p_z \rangle$ of particles
  - emittance of bunch
  - Focusing strength at z (solenoid, quadrupole, solenoid + quadrupole)
- Still quite flexible so may change this

Optimiser

- Optimise *Parameters* (e.g. solenoid currents) using Minuit
  - Takes a list (array) of parameters (doubles)
  - Calculates an arbritrary function that outputs a number
  - Tries to minimise the output of the function
- Interface with Minuit parameters using a C++ object
  - Allows tricks like the same parameter can look at multiple components
    - E.g. we can use the same current for different solenoids
  - Or we can
- Scoring algorithm (which we minimise)
  - use a number of constraints on α and β

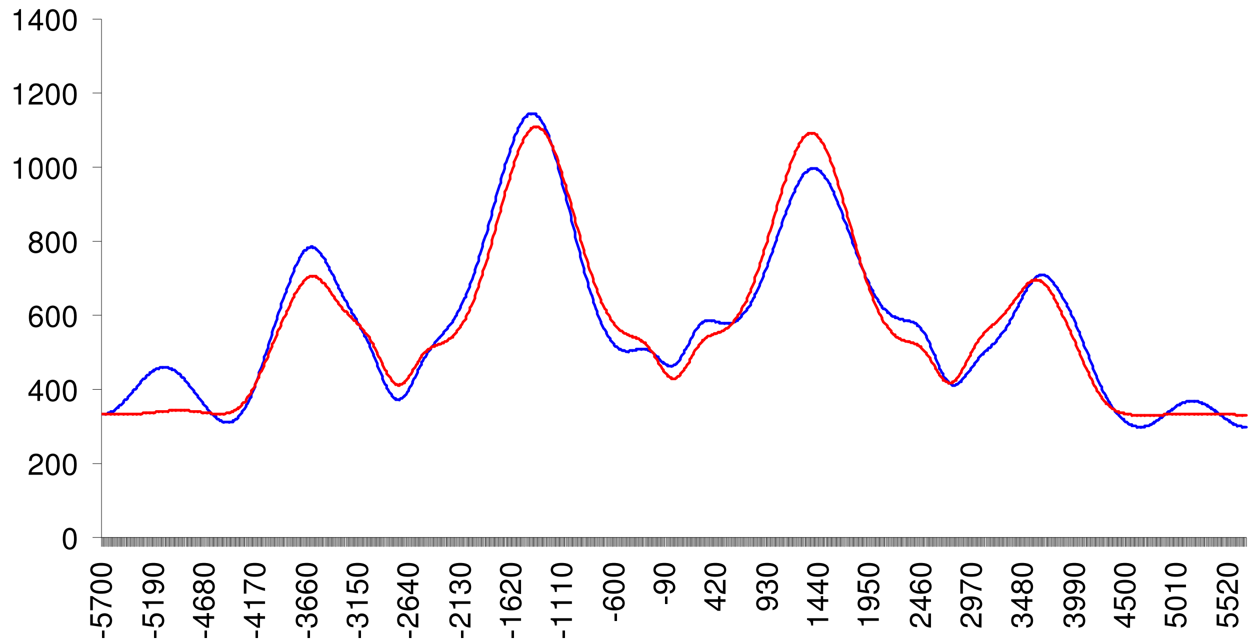
$$score = \sum_{constraints} (\beta_{measured} - \beta_{required})^2 + c \sum_{constraints} (\alpha_{measured} - \alpha_{required})^2$$

Example - two parameter optimisation



- Current evolution over time
  - Currents converge pretty quickly
  - Processing time ~ minute (on a single Imperial cpu)
  - But fairly simple problem

Beta function



- Beta started as blue
  - Finished as red
  - Pretty simple problem

Sample code

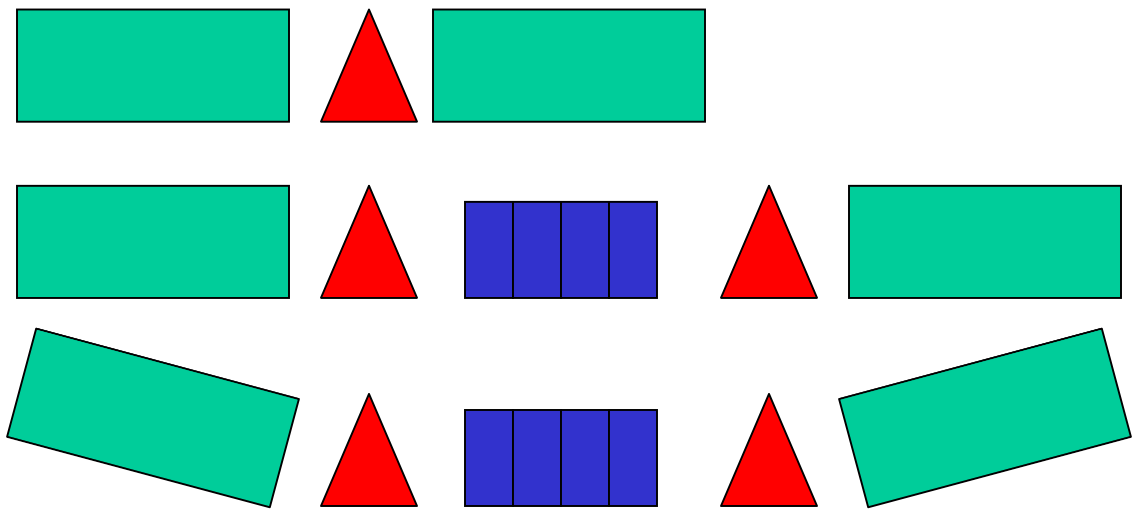
- Interface still requires user to go into code
  - User interface better?
- Takes advantage of c++ “black box” architecture
  - Commands like
    - Beta.AddSolenoid(1260, -5201, 64.44, 50, 255);
    - Minimiser.AddBetaConstraint(333, -5500);
    - Beta.AddQuadrupole(660, zParameter, currentParameter, 178.2);
  - Placing emphasis on usability and extensability
    - Needs feedback from users

Things Optics can do

- Flexible Parameter definition
  - Multiparameter optimisation
  - Single parameter to multiple objects
  - Quad currents, positions
  - Use free parameters or constrained parameters
  - Solenoid currents
- Constraints on beta, alpha
- Materials
  - Specify by x0, dEdz, or material name
  - Calculate length given some desired emittance
- Solenoids + fringe fields
- Field maps
- Quads - no fringe fields

Future - Longitudinal Cooling

- Aim to write proposal for longitudinal cooling in MICE
  - Single wedge - MICE “IV.wedge”
  - Wedges and RF - MICE “V.wedge”
  - Wedges, RF and Tilted solenoids (ambitious) MICE “VII.wedge”!
- Optics and simulation study
  - Dispersion, longitudinal dynamics

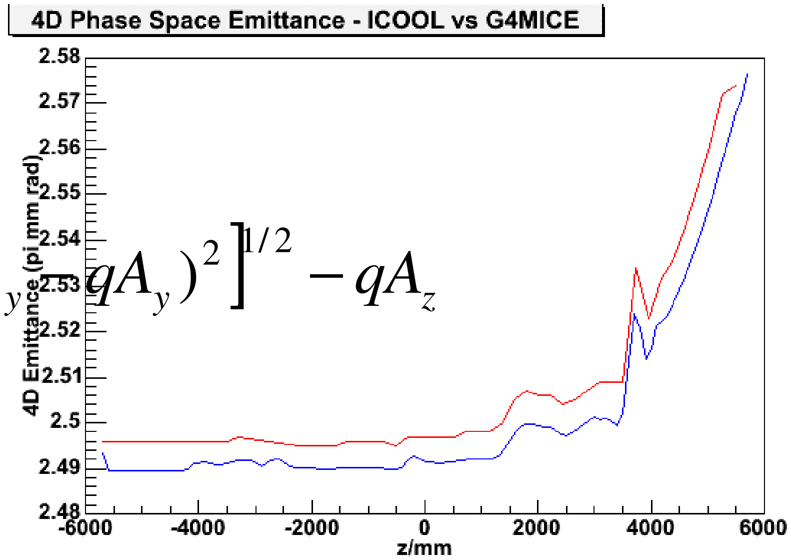


Future - Non-linear dynamics

- Calculate emittance increase from non-linearities in the Hamiltonian
  - Taylor expansion
  - Lie expansion (Dragt et al)

$$H = \left[(p_t + q\phi)^2 - m^2 - (p_x - qA_x)^2 - (p_y - qA_y)^2\right]^{1/2} - qA_z$$

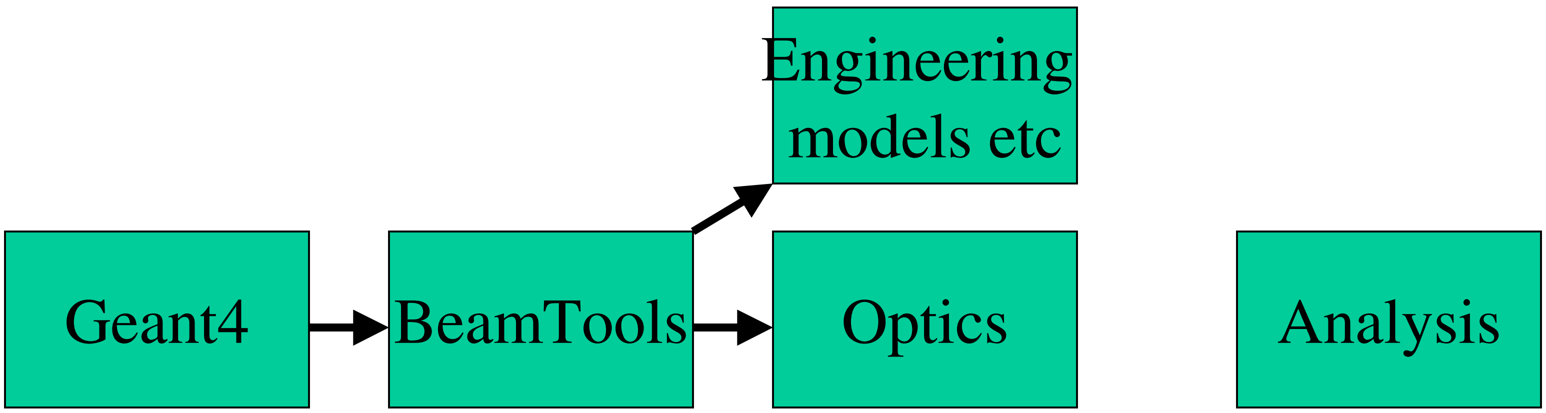
$$A_\phi \approx \frac{r}{2}B(z) - \frac{r^3}{16}B''(z) + \ldots$$

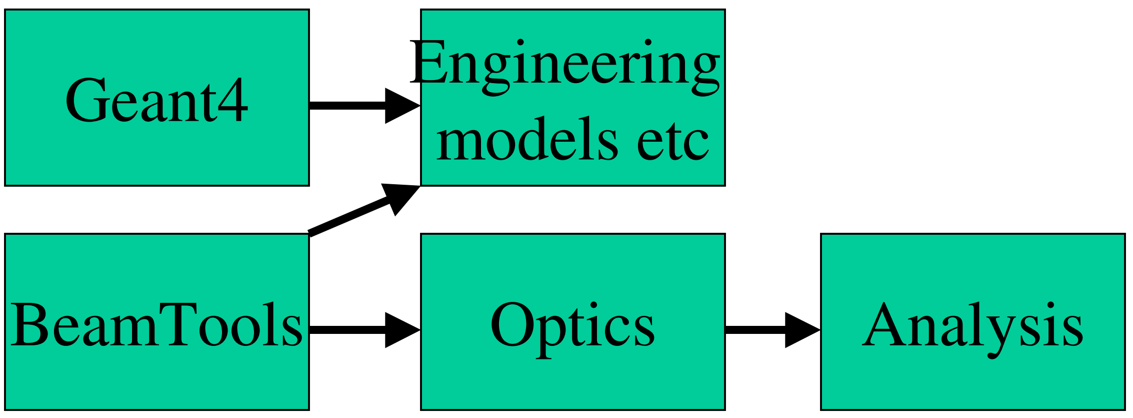


- Famous plot by Bravar (not shown)
  - Symmetric?
  - Software induced?
    - Runga-Kutte (GEANT3/4) is non-Symplectic
  - Measurable?
  - Calculable?

A brief technical note

- Analysis will need access to the Optics package…
  - But shouldn't be able to see Geant4





Conclusions

- We have a great new tool in G4MICE
  - Lots of nice functions
  - Reusable/extendable
- New users and developers are welcome
  - Software development should be done in the G4MICE framework for the common good!
  - chris.rogers@imperial.ac.uk
- Some interesting projects for the future

Lie Algebra "expanded"

- Define a transfer map by $\underline{U}_{final} = \underline{\underline{M}}\,\underline{U}_{start}$
- Then $\underline{\underline{M}}$ can be found using the usual Taylor expansion or using a definition like

$$\underline{\underline{M}} = \exp(:f:)$$

- Where *:f:* is defined by

$$:f: = \sum_i (\partial f / \partial x_i)(\partial / \partial p_i) - (\partial f / \partial p_i)(\partial / \partial x_i)$$

- Then the moments at the end of the channel can be found using

$$m_\alpha^{end} = \sum_\beta D_{\alpha\beta}(\underline{\underline{M}}) m_\beta^{start}$$

- Which can be used to get the change in emittance

Lie Algebra "expanded" 2

- Calculate $\underline{\underline{M}}$ using

$$\frac{d\underline{\underline{M}}}{dz} = \underline{\underline{M}} :-H:$$

- Some nice features
  - $\underline{\underline{M}}$ can be truncated to nth order
  - Truncated terms are independent and symplectic
- Not sure how to calculate $D_{\alpha\beta}$