New Methods of Field Integration - Dense output and FSAL

Mentor : John Apostolakis

Somnath Banerjee

Runge-Kutta Integration in Geant4

- Geant4 uses Runge-Kutta methods to simulate trajectory of particles
- RK methods are used to integrate first order differential equations : $\boldsymbol{y}' = f(s,\boldsymbol{y})$
- Example : Euler's explicit RK method :

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(s_n,y_n) \; ; \quad h = s_{n+1} - s_n$$

- Efficiency, accuracy and robustness are desirable features from a particular RK method

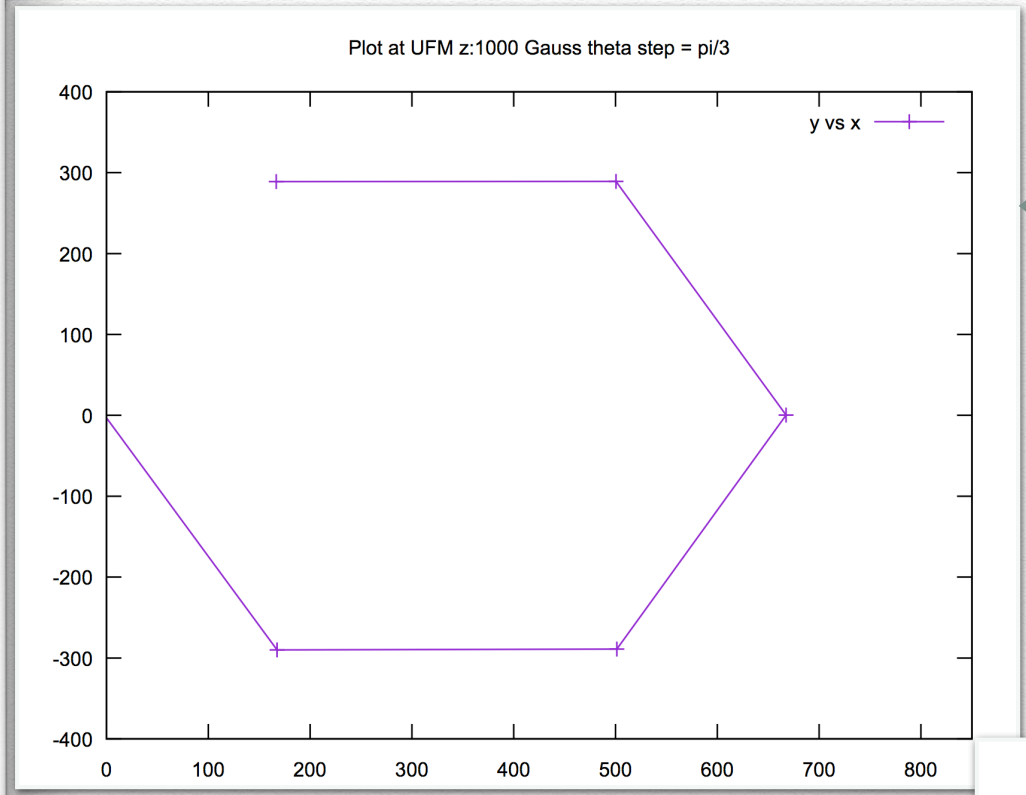
New Steppers

- Added new steppers with additional capabilities
  - Re-use field evaluations (First Same As Last)
  - Interpolation
- Higher order steppers (up to order 8)
  - with higher accuracy
  - built in interpolation schemes

Interpolation

With interpolation, any number of points between two already computed points can be calculated without needing to repeatedly call the Stepper( )

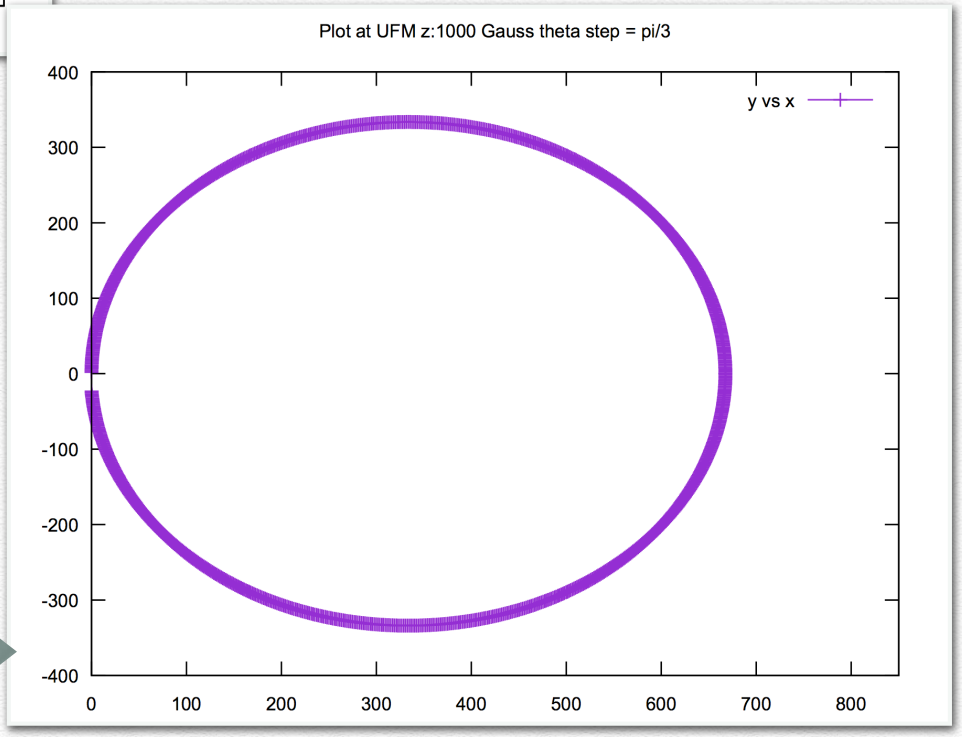
A demo follows :



6 computed points in a circular trajectory

Calling 100 times between each pair of points, to give "Dense" output

Interpolated result



New Tests

testH : first test written for the project

It uses simple field and calls the "*Stepper*( )" method of a stepper class to get trajectory points.

# StepNo	yOut[0]	yErr[0]	yOut-yOutX[0]	yOut[1]	yErr[1]	yOut-yOutX[1]
1	-1.499e-01	5.295e-12	2.064e-12	9.999e+00	-2.277e-10	5.187e-13
2	-5.994e-01	1.212e-11	4.099e-12	1.999e+01	-2.274e-10	1.094e-12
3	-1.348e+00	1.893e-11	6.105e-12	2.996e+01	-2.270e-10	1.734e-12
4	-2.395e+00	2.573e-11	8.082e-12	3.990e+01	-2.263e-10	2.437e-12
5	-3.740e+00	3.250e-11	1.003e-11	4.981e+01	-2.254e-10	3.190e-12
6	-5.382e+00	3.924e-11	1.195e-11	5.968e+01	-2.244e-10	4.000e-12

Downside : Very big single cc file containing all of the program

RKTest

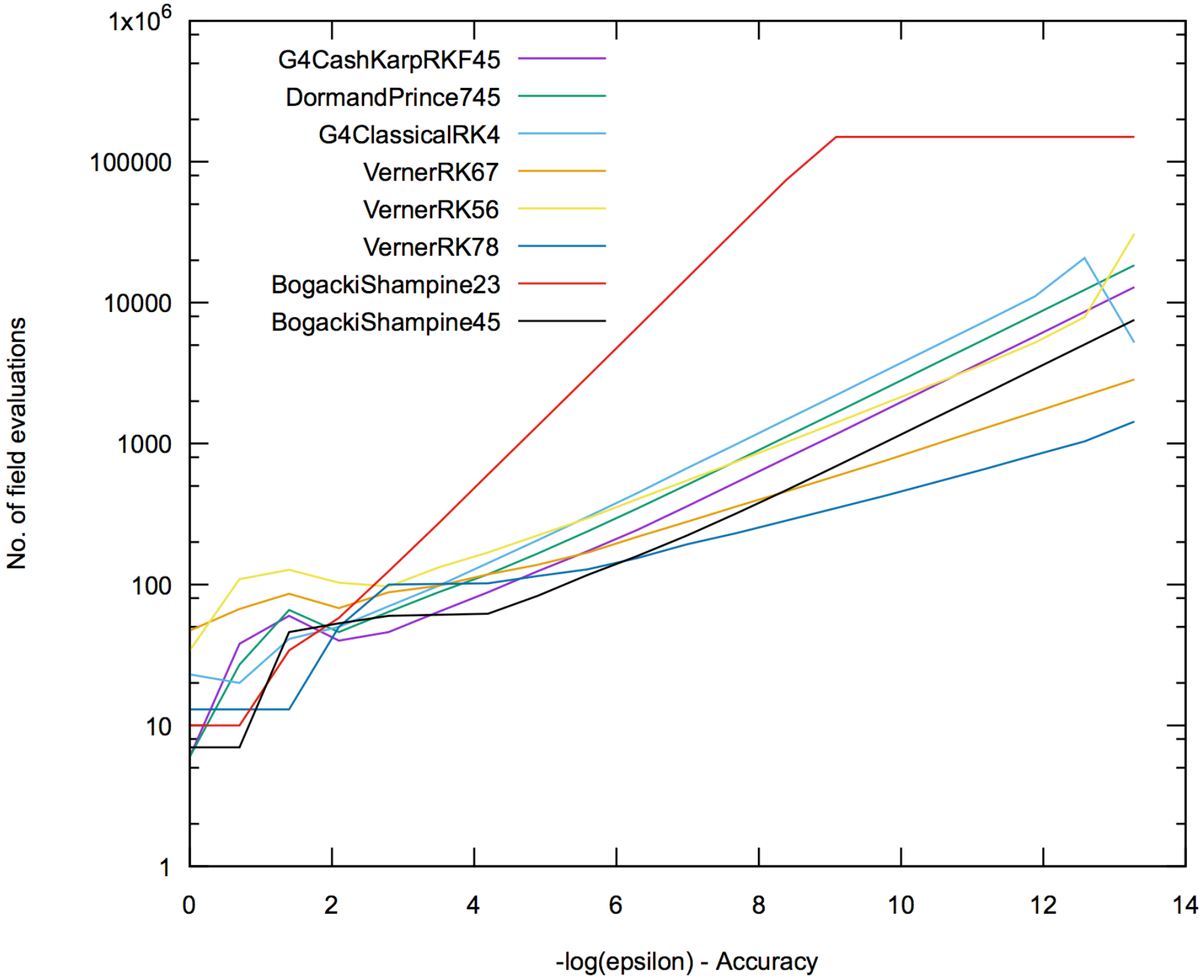
- One stop class for all the tests
- Can do testH-like output, test performance, interpolation error

```
#include “RKTest.hh”
#include “DormandPrince745.hh”

RKTest myTest;
G4double field_factor = 0.1;
myTest.testG4Stepper<DormandPrince745>(“umf”, field_factor);
```

```
# -log10(eps)     func_evals
          -0            10
     0.69897            10
     1.39794            10
     2.09691            10
     2.79588            10
     3.49485            10
     4.19382            10
     4.89279            10
     5.59176            10
```

Test - Results


1x10^6
100000
10000
1000
100
10
1
No. of field evaluations
G4CashKarpRKF45
DormandPrince745
G4ClassicalRK4
VernerRK67
VernerRK56
VernerRK78
BogackiShampine23
BogackiShampine45
0
2
4
6
8
10
12
14
-log(epsilon) - Accuracy

Challenges

- Accommodate FSALDriver or another implementation to handle FSAL
- Integrate interpolation (in steppers) with the Geant4 suite
- Fix bugs and inconsistencies of the current work
- Evaluate the actual efficiency of the new Steppers over the old ones

What I learnt

- ☑ About Geant4 and simulation software in general
- ☑ About Numerical methods especially for solving differential equations
- ☑ Unit conventions in particle physics
- ☑ That, C++ feels bad if you define templated functions of a class in a separate file

Thank You

and all the best for future