Re-engineer Propagation of Charged Tracks in Electromagnetic Field

Geant4/Geant V Optimization

Qiuchen Xie

Mentors: Sandro Wenzel, John Apostolakis

Introduction

Geant4

- GEometry ANd Tracking
- Interaction between particle and material

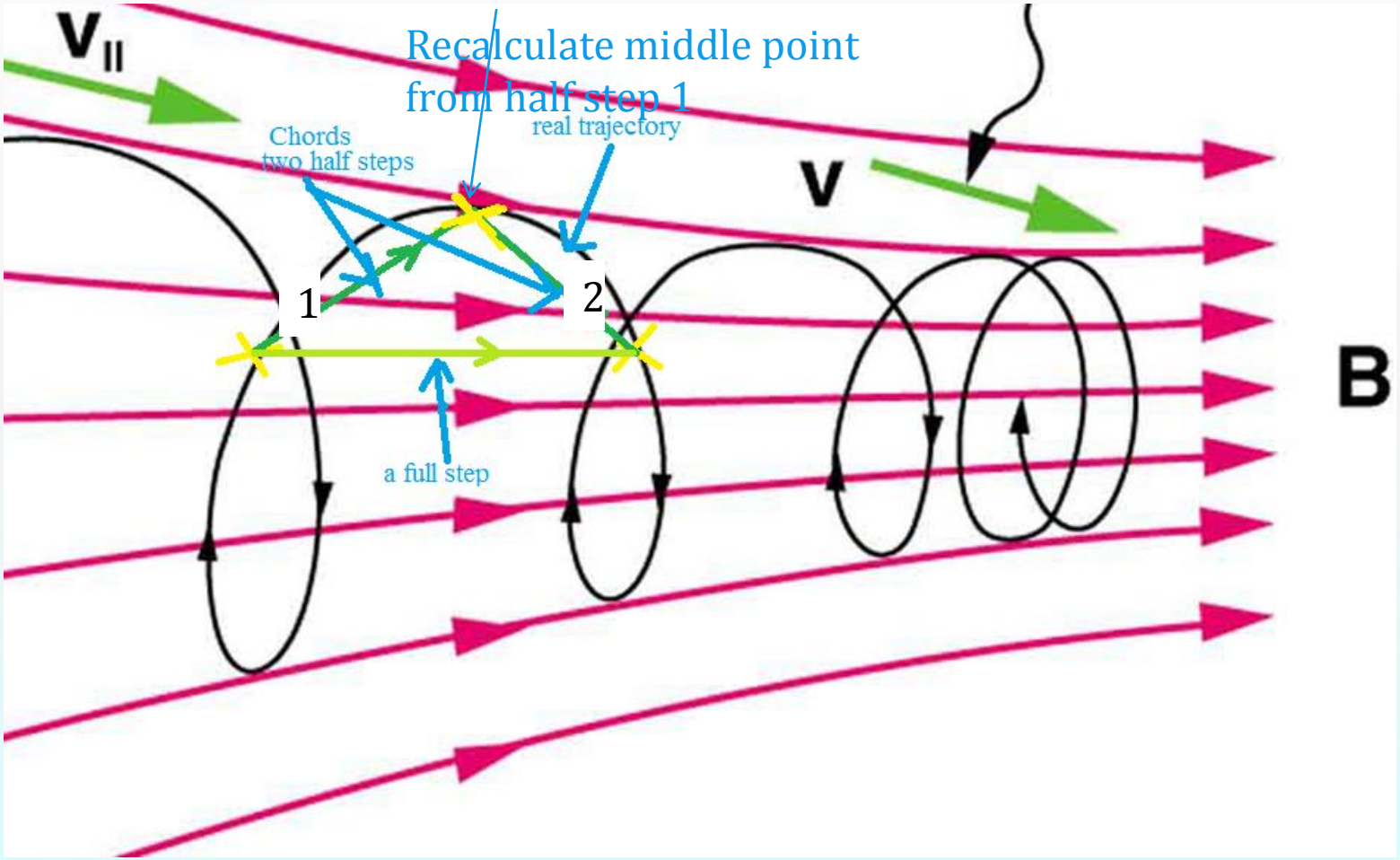
Our Target

- Re-engineering *Source/Geometry/magneticfield*
- Solves differential equations to find path of a particle in a (magnetic) field

Our proposal

- Goal: speed up and avoiding virtual functions
- Solution: i) template polymorphism and ii) vectorization

Particle tracks & numerical methods



Classes redesign

Classes in *geometry/magneticfield*

- Steppers, Equations, Field class(es)

Critical for performance

- Stepper() function and its function calls

Progress

- “Evolutionary” redesign
- Plug and play - any stepper, equation
- Added classes see appendix

Template mode

Template classes & template polymorphism

- Template method pattern: CRTP
- Implemented at run-time -> virtual interfaces
- Convert to compile time - use templates
- Vecotorization:

```
typedef TClassicalRK4<Equation_t, 6> StepperRK4;
```

Plug and play- example 0

Pull out

G4MagErrorStepper

←G4ClassicalRK4

G4Mag_UsualEqRhs

G4CachedMagneticField

G4QuadrupoleMagField

Plug in

TMagErrorStepper

← TClassicalRK4

TMagFieldEquation

TCachedMagneticField

TQuadrupoleMagField

With interface:
e.g.

```
typedef TClassicalRK4<Equation_t, 6> StepperRK4_t;
```

Revised design- example 1

- template <class T_Field>

class TMagFieldEquation : public G4Mag_UsualEqRhs {/*…*/}

- template <class T_Equation, int N>

```
class TClassicalRK4 : public TMagErrorStepper <TClassicalRK4<T_Equation, N>, T_Equation, N>
{
//within stepper() function
//use RK4 numerical method to solve for differential equation with a fixed step size
//itself couldn't provide error; thus we need call it from a errorStepper to estimate error
}
```

- //the interface – template parameters

```
typedef TClassicalRK4<Equation_t, 6> StepperRK4_t;
```

Revised design- example 1

- template <class T_Field>

```
class TMagFieldEquation : public G4Mag_UsualEqRhs {/*…*/}
```

- template <class T_Equation, int N>

```
class TClassicalRK4 : public TMagErrorStepper <TClassicalRK4<T_Equation, N>, T_Equation, N>
{
//within stepper() function
//use RK4 numerical method to solve for differential equation with a fixed step size
//itself couldn't provide error; thus we need call it from a errorStepper to estimate error
}
```

- //the interface – template parameters

```
typedef TClassicalRK4<Equation_t, 6> StepperRK4_t;
```

Revised design- example 1

- template <class T_Field>

class TMagFieldEquation : public G4Mag_UsualEqRhs {/*…*/}

- template <class T_Equation, int N>

```
class TClassicalRK4 : public TMagErrorStepper <TClassicalRK4<T_Equation, N>, T_Equation, N>
{
//within stepper() function
//use RK4 numerical method to solve for differential equation with a fixed step size
//itself couldn't provide error; thus we need call it from a errorStepper to estimate error
}
```

- //the interface – template parameters

typedef TClassicalRK4<Equation_t, 6> StepperRK4_t;

Revised design- example 2

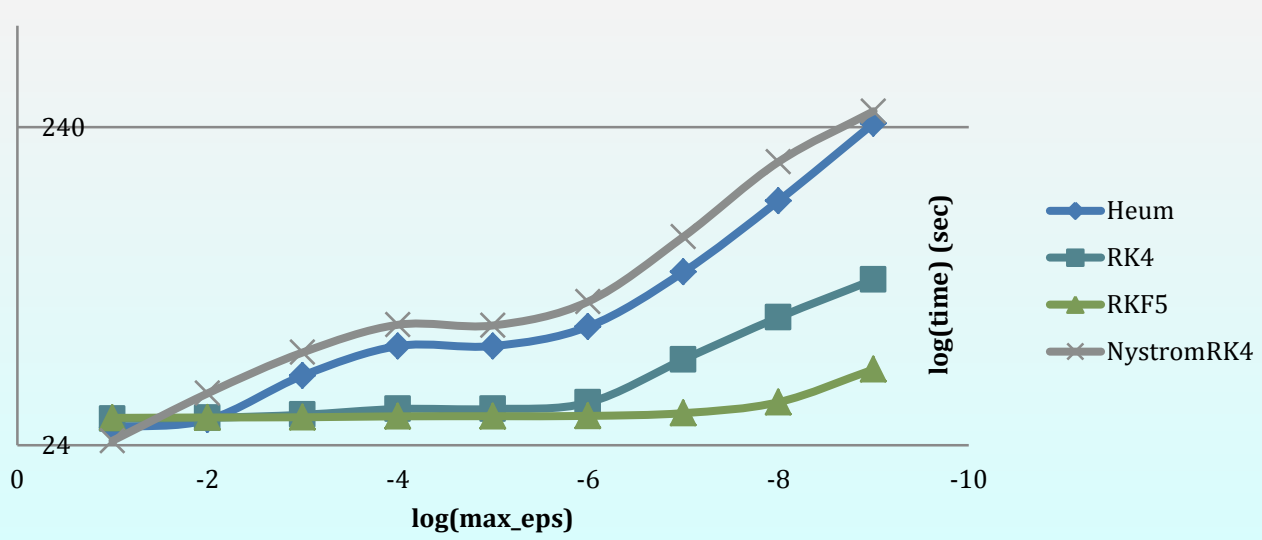
```
template <class T_Stepper, class T_Equation, int N>
class TMagErrorStepper : public G4MagIntegratorStepper {
//…within the dumpStepper() function:
// Do two half steps
static_cast<T_Stepper*>(this)->DumbStepper (yInitial, dydx, halfStep, yMiddle);//determine middle point
TRightHandSide(yMiddle, dydxMid); //re-calculate middle points
static_cast<T_Stepper*>(this)->DumbStepper (yMiddle, dydxMid, halfStep, yOutput);//dtermine end point
//…
// Do a full Step
static_cast<T_Stepper*>(this)->DumbStepper(yInitial, dydx, hstep, yOneStep);//use to compare above & estimate error
//….
}
```

Benchmark and profiler results- Before Changing

Profiling result

- clearly, stepper() function should & could be optimized(Data see appendix)

Benchmark



Benchmark and profiler results- After Changing

- **Profiling result**
- Confirmed there are no function calls in Stepper() - using inline field method
- In unit test TCashKarpRKF45 is 61% vs. 67%
- TClassicalRK4 and TCashKarpRKF45: stepper() improves 10%
- **Benchmark**
- Full G4 application 'NTST' (drift chamber)
- Improved by 5%-10% (using our template classes)
  - Stepper takes 25% of application time

Thank you!

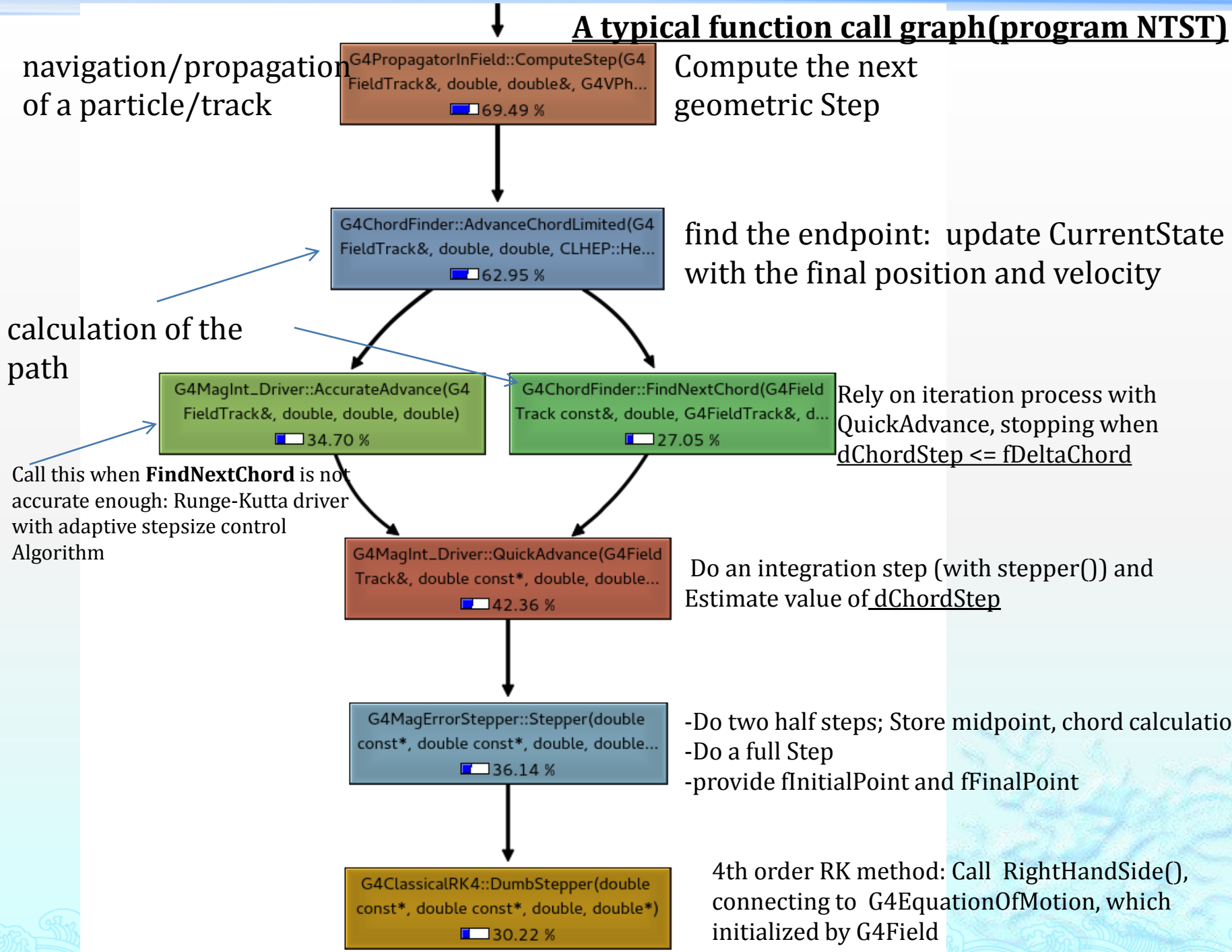
QUESTIONS?

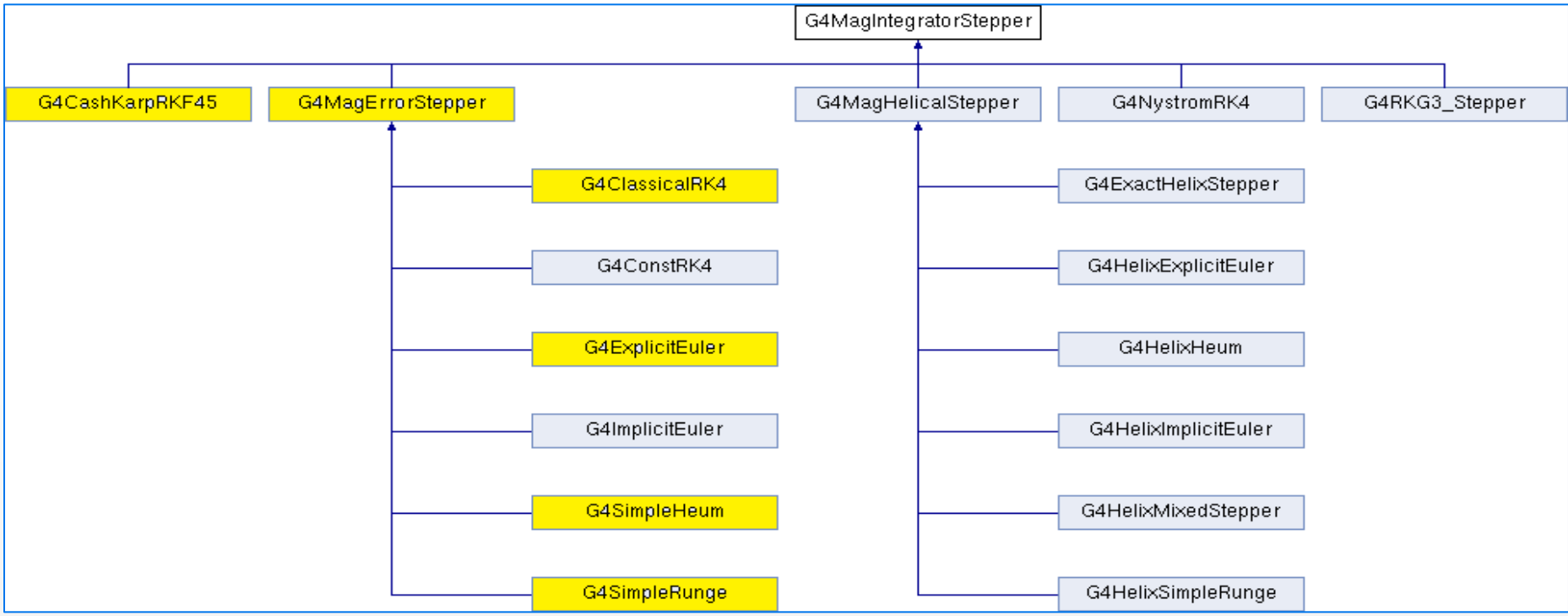
Acknowledgement:
Google Summer of Code 2014
CERN

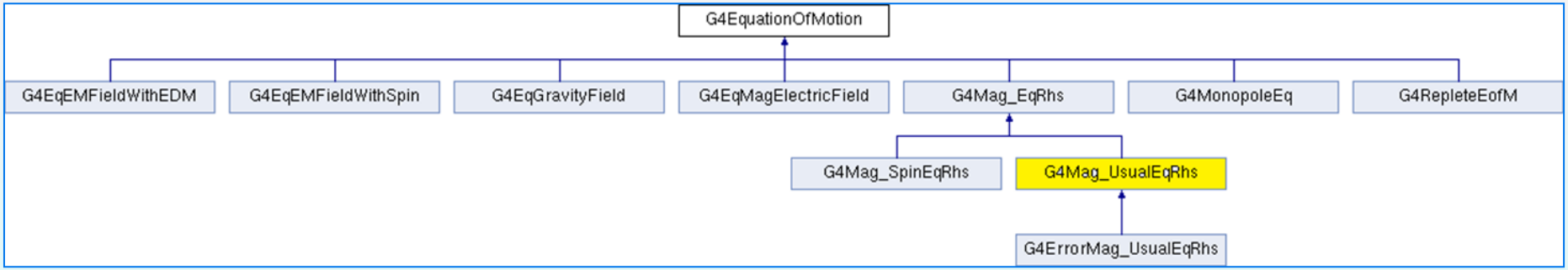
Slide 13-16

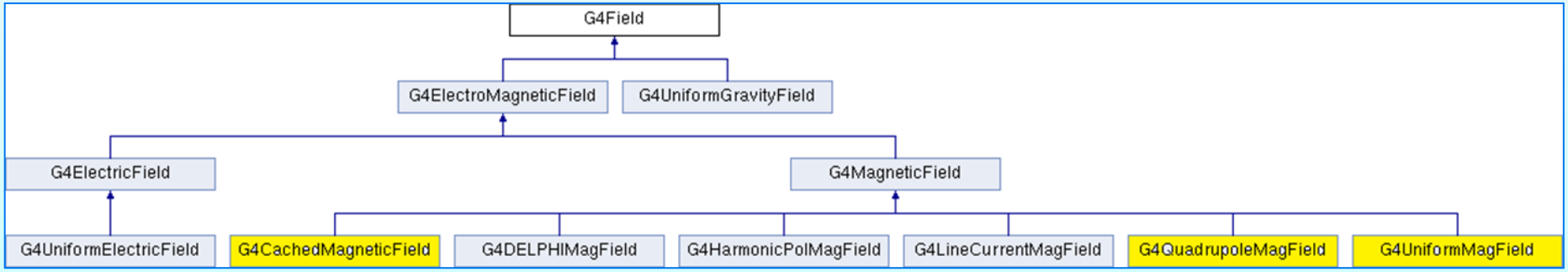
APPENDIX

A typical function call graph(program NTST)




G4MagIntegratorStepper
G4CashKarpRKF45
G4MagErrorStepper
G4MagHelicalStepper
G4NystromRK4
G4RKG3_Stepper
G4ClassicalRK4
G4ConstRK4
G4ExplicitEuler
G4ImplicitEuler
G4SimpleHeum
G4SimpleRunge
G4ExactHelixStepper
G4HelixExplicitEuler
G4HelixHeum
G4HelixImplicitEuler
G4HelixMixedStepper
G4HelixSimpleRunge


G4EquationOfMotion
G4EqEMFieldWithEDM
G4EqEMFieldWithSpin
G4EqGravityField
G4EqMagElectricField
G4Mag_EqRhs
G4MonopoleEq
G4RepleteEofM
G4Mag_SpinEqRhs
G4Mag_UsualEqRhs
G4ErrorMag_UsualEqRhs


G4Field
G4ElectroMagneticField
G4UniformGravityField
G4ElectricField
G4MagneticField
G4UniformElectricField
G4CachedMagneticField
G4DELPHIMagField
G4HarmonicPolMagField
G4LineCurrentMagField
G4QuadrupoleMagField
G4UniformMagField

Revised design- new interface

```
typedef TCachedMagneticField<TQuadrupoleMagField> Field_t;
typedef TMagFieldEquation<Field_t> Equation_t;
typedef TCashKarpRKF45<Equation_t, 6> Stepper_t;
typedef TClassicalRK4<Equation_t, 6> StepperRK4_t;

//the rest part would be same

Equation_t *tEquation = new Equation_t(&tMagField);

case 14: pStepper = new Stepper_t(tEquation); break;
```

	1 G4ImplicitEuler		4 Classic RK4		8 G4CashKarpRKF45		13 G4NystromRK4	
testPropagateMagField	Time (s)	Percentage	Time(s)	Percentage	Time(s)	Percentage	Time(s)	Percentage
Total user time (Quadropole field) (s)	**0.11**		0.10		0.05		0.05	
G4MagErrorStepper::Stepper(%)	0.06	56.0	0.07	77.0				
StepperChoice::Stepper(%)	0.04	38.0	0.06	65.0	0.03	67	0.02	38
G4Mag_UsualEqRhs::EvaluateRhsGivenB(%)	0.01	7.6	0.01	14.0	0.004	8.1		
G4CacheMagneticField::GetFieldValue(%)	0.01	5.7	0.01	10.0	0.03	7.3	0.003	6.5
OneGoodStep: pow(%)	0.02	0.2	0.01	0.079	0.004	0.083	0.01	0.23
ComputeStep- Quadropole field (cycle)	49259		38406		20174		22964	
ComputeStep- uniform field (cycle)	54125		8287		6648		13615	
testPropagateSpin								
Total user time (Quadropole field) (s)	**2.14**		**0.18**		**0.08**		**Not Applicable**	
G4MagErrorStepper::Stepper(%)	1.43	67	0.1386	77				
StepperChoice::Stepper(%)	1.02	48	0.1206	67	0.06	69		
G4Mag_SpinEqRhs::EvaluateRhsGivenB(%)	0.64	30	0.0414	23	0.02	21		
OneGoodStep: pow(%)	0.26	12						
ComputeStep- Quadropole field (cycle)								
ComputeStep- uniform field (cycle)								
NTST-NTST-bench1								
Stepper			4 Classic RK4					
Epsilon-Max			0.01					
			Time	Percentage				
Total user time (no profiling - benchmark run)								
Total user time (with profiling)			31.5					
G4PropatorInField::ComputeStep			20.0	**64.0**	63.5%			
G4ChordFinder:AdvanceChordLimited			10.0	33.0	31.7%			
G4MultiLevelLocator::EstimateIntersectionPoint			6.9	**22.0**	21.9%			
StepperChoice::Stepper			7.9	25.0	25.1%			
G4Mag_UsualEqRhs::EvaluateRhsGivenB			1.6	5.2	5.1%			
G4Transportation::PostStepDoIt()			8.5	27.0	27.0%			
ComputeStep- Quadropole field (cycle)								
ComputeStep- uniform field (cycle)				176.2				