

Описание электрического и магнитного полей

- Описание электрического и магнитного полей создаётся при описании геометрии детектора
- Моделирование движения заряженной частицы в поле осуществляется численным решением уравнения движения с учетом величины поля в данной точке

$$m \frac{d\vec{v}}{dt} = q (\vec{E} + \vec{v} \times \vec{B})$$

- Geant4 позволяет описывать как простые (однородные) поля, так и сложные поля, в том числе задаваемые экспериментально измеренными картами напряженности поля
- Поля могут быть как стационарными, так и изменяющимися во времени

Рассматриваются:

- 1) Глобальные поля (во всём геометрическом мировом объёме)
 - Однородные и заданные пользователем
- 2) Локальные поля (в конкретном детекторе/объёме)
 - Однородные и заданные пользователем

Описание электрического и магнитного полей

При расчёте движения траектории заряженной частицы в поле криволинейный трек разбивается на линейные сегменты – хорды.

Один физический шаг может быть разбит на несколько хорд.

В некоторых случаях на одном шаге может быть реализовано несколько спиральных вращений.

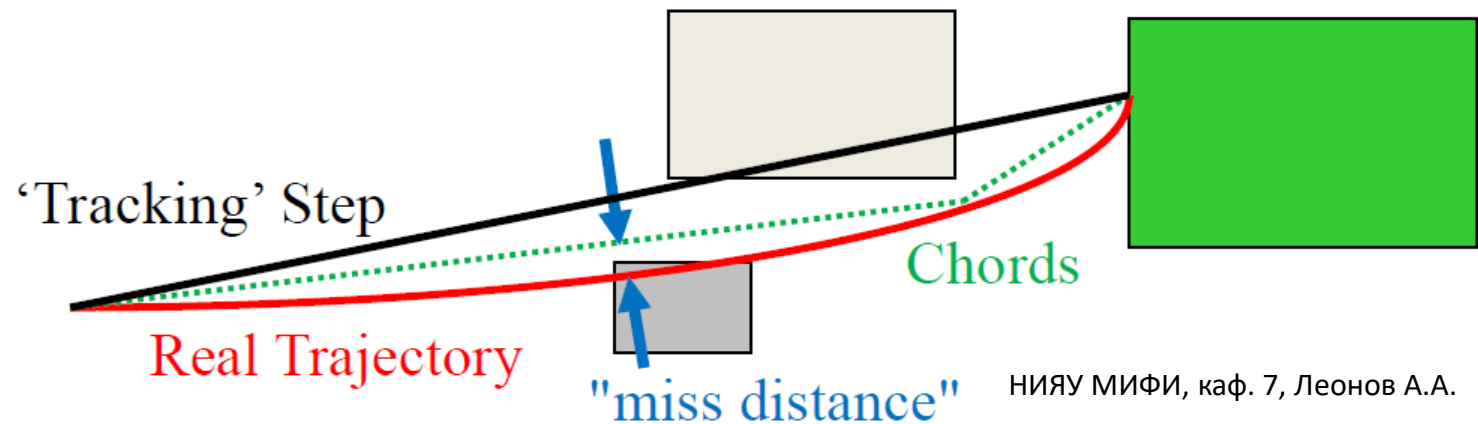
Численное решение (интегрирование) уравнения движения заряженной частицы в поле реализуется по умолчанию с использованием метода Рунге-Кутты.

При моделировании движения частицы в однородном поле, когда известно точное решение, может использоваться helix (спираль).

При моделировании движения частицы в слабо меняющемся поле, может использоваться комбинация Рунге-Кутты и helix

Сегменты аппроксимируют трек с некоторым приближением, выбираются в соответствии с максимально допустимым значением сагитты, которое задаётся пользователем (*miss distance*).

Малые значения *miss distance* сильно увеличивают время расчёта.



Описание электрического и магнитного полей

Точность, с которой определяется пересечение с объёмом, задаётся параметром *delta intersection*.

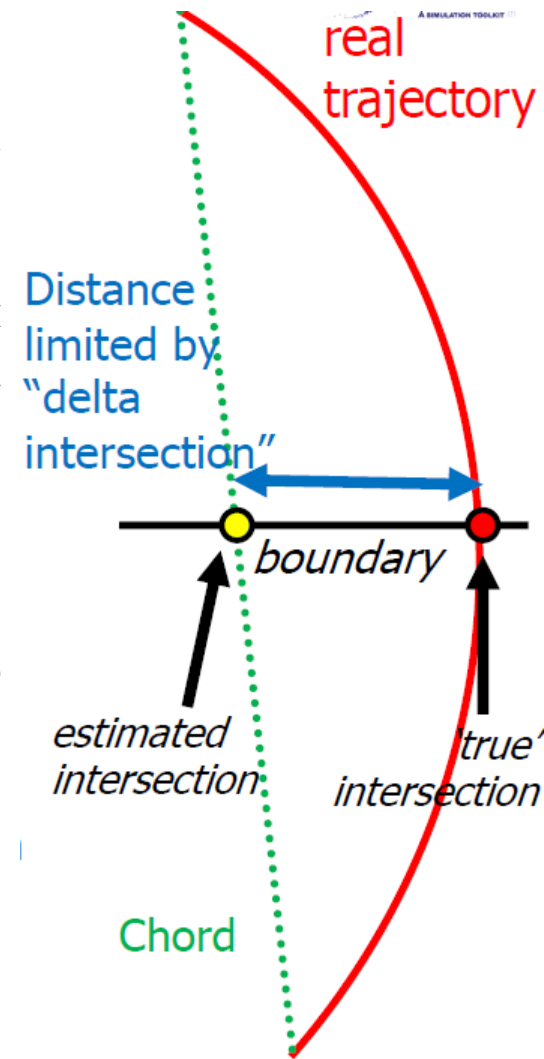
Предел на оценку ошибки определения конечной точки интегрирования на данном физическом шаге, не пересекающем объём, задаётся параметром *delta one step*. Максимальная длина шага, на котором заново вычисляется траектория движения.

delta intersection – более важный параметр.

delta intersection и *delta one step* не являются независимыми, их значения не должны сильно различаться (менее, чем на порядок).

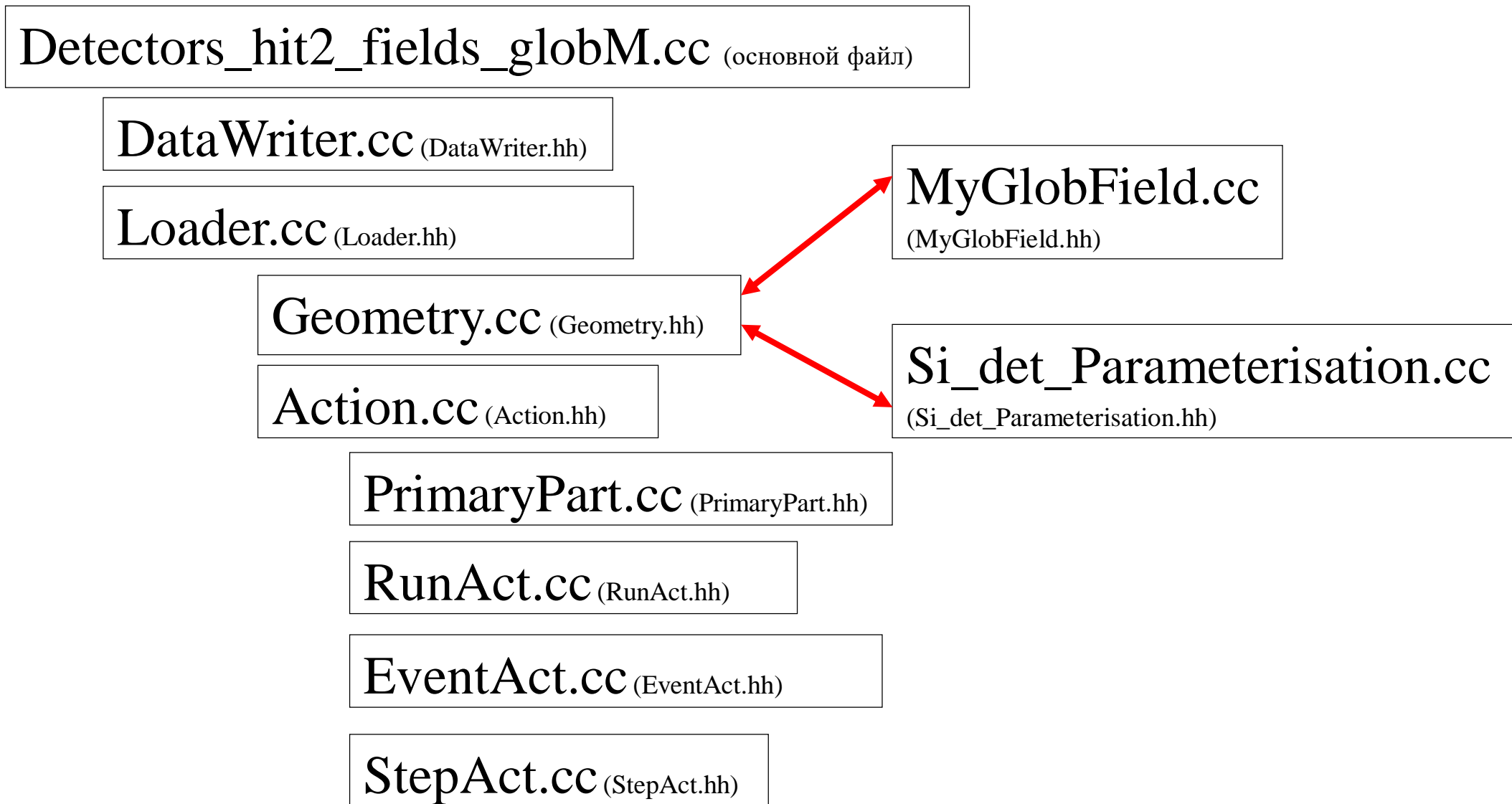
Максимальная и минимальная относительная точность решения уравнения движения: *EpsilonMax* и *EpsilonMin*.

EpsilonMax и *EpsilonMin* имеют приоритет над *delta one step*.



Глобальное однородное магнитное поле

проект **Field_globM**



Глобальное однородное магнитное поле

Geometry.hh

```
#include "MyGlobField.hh"
.....
class Geometry : public G4VUserDetectorConstruction
{
public:
    .....
    MyGlobField* magField;
    .....
}
```

Объект поля - член класса
с геометрией

MyGlobField.hh

```
#include "G4MagneticField.hh"
#include "G4FieldManager.hh"
.....
class MyGlobField
{
public:
    G4MagneticField* globMfield;
    G4FieldManager* globFieldManager;
    MyGlobField();
    ~MyGlobField();
};
```

Менеджер, который
управляет движением
частицы в поле

Geometry.cc

```
Geometry::Geometry(std::ofstream& ofsa)
{
    this->f_geom=&ofsa;
    (*f_geom) << "Hi from Geom!" << std::endl;
    magField = new MyGlobField();
}
```

Объект поля задаётся
в конструкторе класса с геометрией

Объект магнитного поля

Глобальное однородное магнитное поле

MyGlobField.cc

Инициализация глобального менеджера

Глобальный менеджер поля ассоциируется с объёмом мира, существует до задания геометрии, вызывается из менеджера транспортировки.

```
MyGlobField::MyGlobField()
{
    globFieldManager = G4TransportationManager::GetTransportationManager()->GetFieldManager();
    globMfield = new G4UniformMagField(G4ThreeVector(1.*tesla,0.,0.));
    globFieldManager->SetDetectorField(globMfield);
    globFieldManager->CreateChordFinder(globMfield);
}
MyGlobField::~MyGlobField()
{
    delete globMfield;
    globMfield = nullptr;
}
```

Задание магнитного поля

Добавление магнитного поля в глобальный менеджер

Создается объект, рассчитывающий траекторию движения, и добавляется в менеджер

Удаление объекта поля

Визуализация магнитного поля

vis.mac

/vis/scene/add/magneticField

Глобальное однородное электрическое поле

проект **Field_globE**

Geometry.hh

```
#include "MyGlobField.hh"
.....
class Geometry : public G4VUserDetectorConstruction
{
public:
    .....
    MyGlobField* elField;
    .....
}
```

Объект поля - член класса
с геометрией



Geometry.cc

```
Geometry::Geometry(std::ofstream& ofsa)
{
    this->f_geom=&ofsa;
    (*f_geom) << "Hi from Geom!" << std::endl;
    elField = new MyGlobField();
}
```



Объект поля задаётся в
конструкторе класса с геометрией

Глобальное однородное электрическое поле

MyGlobField.cc

Глобальный менеджер

```
MyGlobField::MyGlobField()
{
    globFieldManager = G4TransportationManager::GetTransportationManager()->GetFieldManager();
    globEfield = new G4UniformElectricField(G4ThreeVector(0.0,0.0,1000.0*kilovolt/cm));
    globFieldManager->SetDetectorField(globEfield);
    globEquation = new G4EqMagElectricField(globEfield);
    globStepper = new G4ClassicalRK4(globEquation,8);
    globntgrDriver = new G4MagInt_Driver(0.1*mm, globStepper, globStepper->GetNumberOfVariables());
    globChordFinder = new G4ChordFinder(globntgrDriver);
    globFieldManager->SetChordFinder(globChordFinder);
}

MyGlobField::~MyGlobField()
{
    delete globEfield;    globEfield = nullptr;
    delete globChordFinder; globChordFinder= nullptr;
    delete globStepper;    globStepper = nullptr;
    delete globEquation;    globEquation = nullptr;
}
```

Объект электрического поля

Добавление электрического поля в менеджер

Уравнение движение в электрическом и магнитном полях

Метод Рунге-Кутты для решения уравнения движения

Контроль ошибки интегрирования, минимальный шаг 0.1 мм

Создается объект, рассчитывающий траекторию движения

Созданный объект добавляется в глобальный менеджер

Удаление объекта поля и объектов для расчёта движения в поле

Вычисление координат, импульса, времени, энергии

Глобальное однородное гравитационное поле

проект **Field_gravity**

Geometry.hh

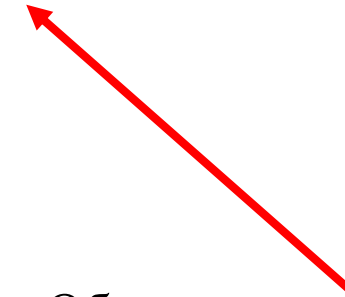
```
#include "MyGlobField.hh"
.....
class Geometry : public G4VUserDetectorConstruction
{
public:
    .....
    MyGlobField* gravField;
    .....
}
```

Объект поля - член класса
с геометрией



Geometry.cc

```
Geometry::Geometry(std::ofstream& ofsa)
{
    this->f_geom=&ofsa;
    (*f_geom) << "Hi from Geom!" << std::endl;
    gravField = new MyGlobField();
}
```



Объект поля задаётся в
конструкторе класса с геометрией

Глобальное однородное гравитационное поле

MyGlobField.cc

```
MyGlobField::MyGlobField()
```

```
{
```

```
using StepperType = G4ClassicalRK4;
```

```
globFieldManager = G4TransportationManager::GetTransportationManager()->GetFieldManager();
```

```
globGravfield = new G4UniformGravityField();
```

```
G4RepleteEofM* equation = new G4RepleteEofM(globGravfield);
```

```
globFieldManager->SetDetectorField(globGravfield);
```

```
StepperType* stepper = new StepperType(equation,8);
```

```
G4double minStep = 0.01*mm;
```

```
G4ChordFinder* chordFinder = nullptr;
```

```
auto intgrDriver = new G4IntegrationDriver<StepperType>(minStep, stepper, stepper->GetNumberOfVariables());
```

```
chordFinder = new G4ChordFinder(intgrDriver);
```

```
G4double deltaChord = 3.0*mm;
```

```
chordFinder->SetDeltaChord(deltaChord);
```

```
G4double deltaIntersection = 0.1*mm;
```

```
globFieldManager->SetDeltaIntersection(deltaIntersection);
```

```
G4double deltaOneStep = 0.01*mm;
```

```
globFieldManager->SetAccuraciesWithDeltaOneStep(deltaOneStep);
```

```
G4double epsMax = 1.0e-4;
```

```
G4double epsMin = 2.5e-7;
```

```
globFieldManager->SetMinimumEpsilonStep(epsMin);
```

```
globFieldManager->SetMaximumEpsilonStep(epsMax);
```

```
globFieldManager->SetChordFinder(chordFinder);
```

```
}
```

Задание типа для метода интегрирования

Глобальный менеджер

Объект гравитационного поля

Уравнение движение в электрическом, магнитном и гравитационном полях

Добавление гравитационного поля в менеджер

Метод Рунге-Кутты для решения уравнения движения

Контроль ошибки интегрирования, минимальный шаг 0.01 мм

Создается объект, рассчитывающий траекторию движения

Установка параметров точности интегрирования уравнения движения

Объект, рассчитывающий траекторию движения, добавляется в глобальный менеджер

Локальное однородное магнитное поле

проект **Field_locM**

MyGlobField.cc

```
MyLocField::MyLocField()
```

```
{
```

```
myEMfield = new G4UniformMagField(G4ThreeVector(100.*tesla,0.,0.));
```

```
myFieldManager = new G4FieldManager(myEMfield);
```

```
}
```

```
MyLocField::~~MyLocField()
```

```
{
```

```
delete myEMfield;
```

```
myEMfield = nullptr;
```

```
}
```

Задание однородного магнитного поля

Geometry.cc

```
Geometry::Geometry(std::ofstream& ofsa)
```

```
{
```

```
this->f_geom=&ofsa;
```

```
(*f_geom) << "Hi from Geom!" << std::endl;
```

```
magField = new MyGlobField();
```

```
}
```

Объект поля задаётся в конструкторе класса с геометрией

Задание менеджера для управления движением в поле

Удаление объекта поля

```
G4VPhysicalVolume* Geometry::Construct()
```

```
{
```

```
.....
```

```
G4bool allLocal = true;
```

```
Pb_log->SetFieldManager(this->magField->myFieldManager, allLocal);
```

```
.....
```

```
}
```

Задание магнитного поля во всех дочерних объёмах

Привязка магнитного поля к логическому объёму

Локальное однородное электрическое поле

проект **Field_locE**

MyGlobField.cc

```
MyLocField::MyLocField():myEMfield(0), myEquation(0), myStepper(0), myntgrDriver(0), myChordFinder(0), myFieldManager(0)
{
    myEMfield    = new G4UniformElectricField(G4ThreeVector(0.0,100000.*kilovolt/cm,0.0));
    myEquation   = new G4EqMagElectricField(myEMfield);
    myStepper    = new G4ClassicalRK4(myEquation,8);
    myntgrDriver = new G4MagInt_Driver(0.001*mm, myStepper, myStepper->GetNumberOfVariables());
    myChordFinder = new G4ChordFinder(myntgrDriver);
    myFieldManager = new G4FieldManager(myEMfield,myChordFinder,true);
}

MyLocField::~MyLocField()
{
    delete myChordFinder; myChordFinder= nullptr;
    delete myStepper;    myStepper = nullptr;
    delete myEquation;   myEquation = nullptr;
    delete myEMfield;    myEMfield = nullptr;
}
```

Задание электрического поля

Уравнение движение в электрическом и магнитном полях

Метод Рунге-Кутты для решения уравнения движения

Контроль ошибки интегрирования, минимальный шаг 0.001 мм

Создается объект, рассчитывающий траекторию движения

Созданный объект добавляется в менеджер

Удаление объекта поля и объектов для расчёта движения в поле

Локальное однородное электрическое поле

Geometry.cc

```
Geometry::Geometry(std::ofstream& ofsa)
{
    this->f_geom=&ofsa;
    (*f_geom) << "Hi from Geom!" << std::endl;
    elField = new MyGlobField();
}
```

*Объект электрического поля задаётся
в конструкторе класса с геометрией*

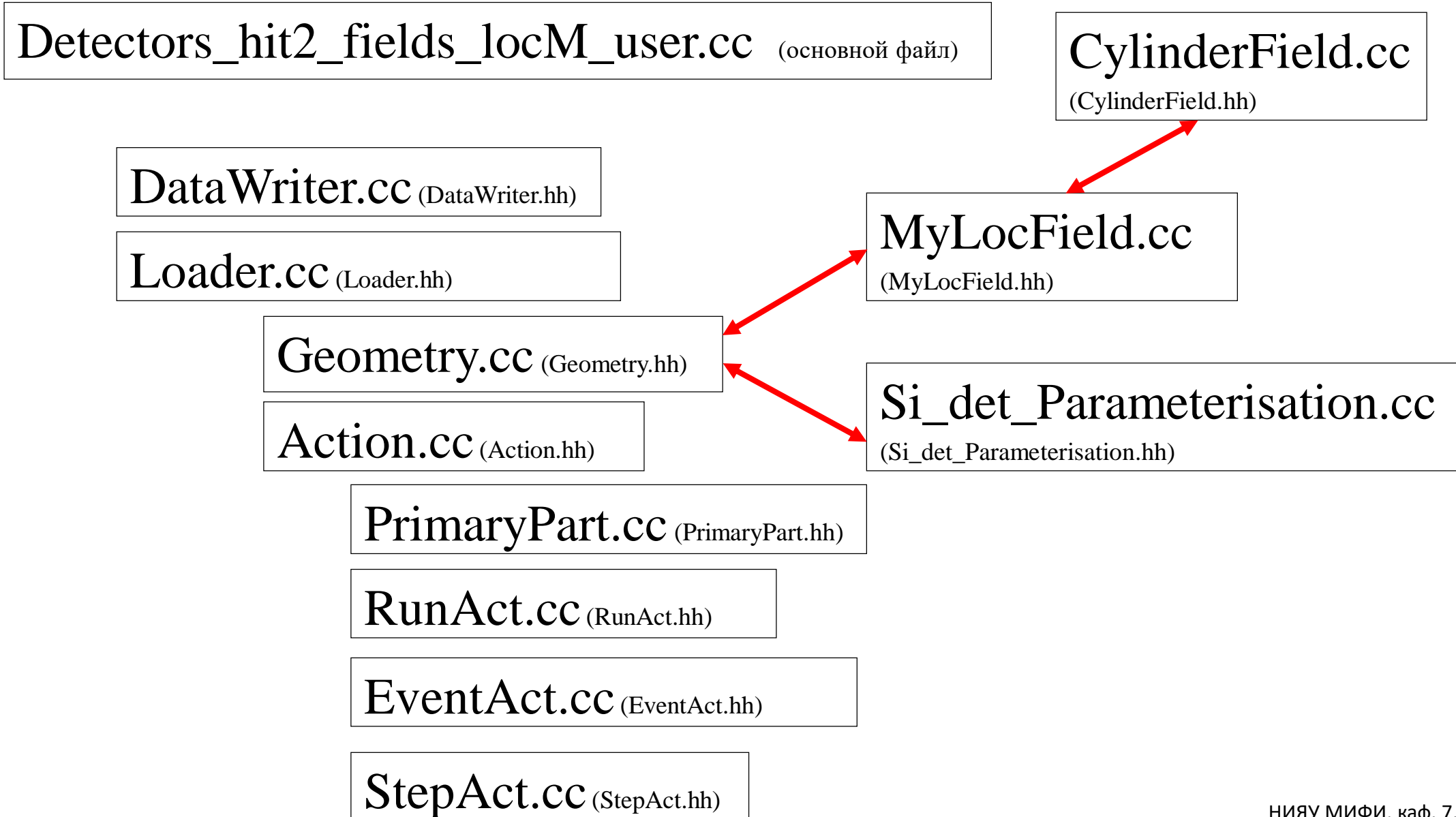
```
G4VPhysicalVolume* Geometry::Construct()
{
    .....
    G4bool allLocal = true;
    Pb_log->SetFieldManager(this->elField->myFieldManager, allLocal);
    .....
}
```

Задание магнитного поля во всех дочерних объёмах

Привязка электрического поля к логическому объёму

Заданное пользователем локальное магнитное поле

проект Field_locM_user



Заданное пользователем локальное магнитное поле

CylinderField.hh

```
class CylinderField : public G4MagneticField
{
public:
    CylinderField(G4double radius, G4double det_length, G4double By);
    ~CylinderField();
    virtual void GetFieldValue(const G4double Point[4], G4double *pField)
const;
private:
    G4double fradius; G4double fdet_length;
    G4double fBy;
};
```

*постоянное магнитное поле fBY
вдоль оси Y внутри цилиндра с
радиусом $fradius$ и высотой
 $2*fdet_length$*

*Вектор из шести компонент:
первые три – x, y, z компоненты
вектора магнитного поля;
следующие три – x, y, z компоненты
вектора электрического поля;*

*Три пространственные координаты и
время, для которых задаются вектора
поля $pField$*

Для задания собственного магнитного поля создаётся класс – наследник абстрактного класса `G4MagneticField`.
Должен быть переопределён чисто виртуальный метод `GetFieldValue()`.

Public Member Functions

	<code>G4MagneticField ()</code>
virtual	<code>~G4MagneticField ()</code>
	<code>G4MagneticField (const G4MagneticField &r)</code>
<code>G4MagneticField &</code>	<code>operator= (const G4MagneticField &p)</code>
<code>G4bool</code>	<code>DoesFieldChangeEnergy () const</code>
virtual void	<code>GetFieldValue (const G4double Point[4], G4double *Bfield) const =0</code>


Заданное пользователем локальное магнитное поле

CylinderField.cc

```
CylinderField::CylinderField(G4double radius, G4double det_length, G4double By)
```

```
{  
  fradius=radius;  
  fdet_length=det_length;  
  fBy=By;  
}
```

*Постоянное магнитное поле вдоль оси Y внутри
цилиндра с радиусом fradius и высотой 2*fdet_length*



```
void CylinderField::GetFieldValue(const G4double Point[4], G4double *pField) const  
{  
  if (abs(Point[1])<fdet_length)&(powf((powf(Point[0],2.))+powf(Point[2],2.)),0.5)<fradius))  
  {  
    pField[1]=fBy;  
  }  
  else  
  {  
    pField[1]=0.;  
  }  
  pField[0]=0.; pField[2]=0.; pField[3]=0.; pField[4]=0.; pField[5]=0.;  
}
```


Заданное пользователем локальное магнитное поле

MyLocField.cc

```
MyLocField::MyLocField()
{
myEMfield    = new CylinderField(100.*mm,40.*mm,100.*tesla);
myFieldManager = new G4FieldManager(myEMfield);
}

MyLocField::~~MyLocField()
{
delete myEMfield;
myEMfield = nullptr;
}
```

Geometry.cc

```
G4VPhysicalVolume* Geometry::Construct()
{
.....
G4bool allLocal = true;
Pb_log->SetFieldManager(this->magField->myFieldManager, allLocal);
.....
}
```

Задание цилиндрического магнитного поля

Задание менеджера для управления движением частиц в поле

Geometry.hh

```
#include "MyLocField.hh"
.....
class Geometry : public G4VUserDetectorConstruction
{
public:
.....
MyLocField* magField;
.....
}
```

Объект поля - член класса с геометрией

Задание магнитного поля во всех дочерних объёмах

Привязка магнитного поля к логическому объёму

Заданное пользователем глобальное магнитное поле

MyLocField.cc

```
MyLocField::MyLocField()
{
    globFieldManager = G4TransportationManager::GetTransportationManager()->GetFieldManager();
    globfield = new CylinderField(100.*mm,40.*mm,25.*tesla);
    globFieldManager->SetDetectorField(globMfield);
    globFieldManager->CreateChordFinder(globMfield);
}
MyLocField::~~MyLocField()
{
    delete globMfield;
    globMfield = nullptr;
}
```

Инициализация глобального менеджера

Задание однородного магнитного поля в цилиндре

Добавление магнитного поля в глобальный менеджер

Создается объект, рассчитывающий траекторию движения, и добавляется в глобальный менеджер

Удаление объекта поля

Заданное пользователем глобальное магнитное поле

проект **Field_locM_userG**

CylinderField.hh

```
class CylinderField : public G4MagneticField
{
public:
    CylinderField(G4double radius, G4double det_length, G4double Bx);
    ~CylinderField();
    virtual void GetFieldValue(const G4double Point[4], G4double *pField) const;
private:
    G4double fradius;
    G4double fdet_length;
    G4double fBx;
};
```

*Постоянное магнитное поле fBx
вдоль оси X внутри цилиндра с
радиусом $fradius$ и высотой
 $2*fdet_length$*

Объект поля - член класса с геометрией

Geometry.cc

```
Geometry::Geometry(std::ofstream& ofsa)
{
    this->f_geom=&ofsa;
    (*f_geom) << "Hi from Geom!" << std::endl;
    magField = new MyLocField();
}
```

*Объект магнитного поля задаётся в
конструкторе класса с геометрией*

Geometry.hh

```
#include "MyLocField.hh"
.....
class Geometry : public G4VUserDetectorConstruction
{
public:
    .....
    MyLocField* magField;
    .....
```


Заданное пользователем глобальное магнитное поле

CylinderField.cc

```
CylinderField::CylinderField(G4double radius, G4double det_length, G4double Bx)
```

```
{  
  fradius=radius;  
  fdet_length=det_length;  
  fBx=Bx;  
}
```

*Постоянное магнитное поле вдоль оси X внутри
цилиндра с радиусом fradius и высотой 2*fdet_length*



```
CylinderField::~~CylinderField(){}  
  
void CylinderField::GetFieldValue(const G4double Point[4], G4double *pField) const
```

```
{  
  if ((abs(Point[0])<fdet_length)&(powf((powf(Point[1],2.))+powf(Point[2],2.)),0.5)<fradius))  
  {  
    pField[0]=fBx;  
  }  
  else  
  {  
    pField[0]=0.;  
  }  
  pField[1]=0.; pField[2]=0.; pField[3]=0.; pField[4]=0.; pField[5]=0.;  
}
```

Заданное пользователем локальное электрическое поле

проект **Field_locE_user**

CylinderField.hh

```
class CylinderField : public G4ElectricField
{
public:
    CylinderField(G4double voltage, G4double rmin, G4double rmax, G4double det_length);
    ~CylinderField();
    virtual void GetFieldValue(const G4double Point[4], G4double *pField) const;
private:
    G4double fvoltage;
    G4double frmin;
    G4double frmax;
    G4double fdet_length;
};
```

электрическое поле внутри цилиндрического конденсатора с приложенным напряжением fvoltage;
*длина конденсатора 2*fdet_length;*
радиусом внутреннего цилиндрического электрода frmin;
радиусом внешнего цилиндрического электрода frmax.

Заданное пользователем локальное электрическое поле

MyField.cc

```
MyField::MyField():myEMfield(0), myEquation(0), myStepper(0), myntgrDriver(0), myChordFinder(0), myFieldManager(0)
{
myEMfield    = new CylinderField(powf(10.,9.)*volt,1.*mm,40.*mm,400.*mm);
myEquation   = new G4EqMagElectricField(myEMfield);
myStepper    = new G4ClassicalRK4(myEquation,8);
myntgrDriver = new G4MagInt_Driver(0.001*mm, myStepper, myStepper->GetNumberOfVariables());
myChordFinder = new G4ChordFinder(myntgrDriver);
myFieldManager = new G4FieldManager(myEMfield,myChordFinder,true);
}
```

Задание электрического поля в цилиндрическом конденсаторе

Уравнение движение в электрическом и магнитном полях

Метод Рунге-Кутты для решения уравнения движения

Контроль ошибки интегрирования, минимальный шаг 0.001 мм

Создается объект, рассчитывающий траекторию движения

Созданный объект добавляется в создаваемый менеджер

CylinderField.cc

```
CylinderField::CylinderField(G4double voltage, G4double rmin, G4double rmax, G4double det_length)
{
fvoltage=voltage;
frmin=rmin;
frmax=rmax;
fdet_length=det_length;
}
```

Заданное пользователем локальное электрическое поле

CylinderField.cc

```
void CylinderField::GetFieldValue(const G4double Point[4], G4double *pField) const
```

```
{  
  G4double radius=0., fieldMag=0., fi=0.; G4ThreeVector RadField;
```

```
  if ((Point[2]>-fdet_length)&(Point[2]<fdet_length))
```

```
  {  
    radius=powf((powf(Point[0],2.)+powf(Point[1],2.)),0.5);
```

```
    if ((radius>frmin)&(radius<frmax))
```

```
    {  
      fieldMag = fvoltage/(radius*(std::log(frmin/frmax)));
```

```
      fi=atan2(Point[1],Point[0]);
```

```
      RadField = G4ThreeVector(fieldMag*cos(fi),fieldMag*sin(fi),0.);
```

```
    }
```

```
  else
```

```
  {  
    RadField = G4ThreeVector(0.,0.,0.);
```

```
  }
```

```
}
```

```
else
```

```
{  
  RadField = G4ThreeVector(0.,0.,0.);
```

```
}
```

```
pField[0]=0.; pField[1]=0.; pField[2]=0.; pField[3]=RadField.x(); pField[4]=RadField.y(); pField[5]=RadField.z();
```

```
}
```

*Вектор из шести компонент:
первые три – x, y, z компоненты
вектора магнитного поля;
следующие три – x, y, z компоненты
вектора электрического поля;*

*Три пространственные координаты и
время, для которых задаются вектора
поля pField*

*электрическое поле внутри цилиндрического
конденсатора с заданным напряжением*

$$E = \frac{U}{r \times \ln \frac{R_2}{R_1}}$$

Заданное пользователем локальное электрическое поле

Geometry.cc

Geometry.hh

```
#include "MyField.hh"
```

```
.....
```

```
class Geometry : public G4VUserDetectorConstruction
```

```
{
```

```
public:
```

```
.....
```

```
MyField* elField;
```

```
.....
```

```
}
```

*Объект поля - член класса
с геометрией*

```
Geometry::Geometry(std::ofstream& ofsa)
```

```
{
```

```
this->f_geom=&ofsa;
```

```
(*f_geom) << "Hi from Geom!" << std::endl;
```

```
elField = new MyField();
```

```
}
```

*Объект электрического поля задаётся
в конструкторе класса с геометрией*

```
G4VPhysicalVolume* Geometry::Construct()
```

```
{
```

```
.....
```

```
G4bool allLocal = true;
```

```
Pb_log->SetFieldManager(this->elField->myFieldManager, allLocal);
```

```
.....
```

```
}
```

Привязка электрического поля к логическому объёму

*Задание электрического поля
во всех дочерних объёмах*

Заданное пользователем глобальное электрическое поле

проект Field_locE_userG

MyField.cc

Инициализация глобального менеджера

```
MyField::MyField()
```

```
{
```

```
globFieldManager = G4TransportationManager::GetTransportationManager()->GetFieldManager();
```

```
globEfield = new CylinderField(powf(10.,8.5)*volt,1.*mm,100.*mm,400.*mm);
```

```
globFieldManager->SetDetectorField(globEfield);
```

```
globEquation = new G4EqMagElectricField(globEfield);
```

```
globStepper = new G4ClassicalRK4(globEquation,8);
```

```
globntgrDriver = new G4MagInt_Driver(0.1*mm, globStepper, globStepper->GetNumberOfVariables());
```

```
globChordFinder = new G4ChordFinder(globntgrDriver);
```

```
globFieldManager->SetChordFinder(globChordFinder);
```

```
}
```

```
MyField::~~MyField()
```

```
{
```

```
delete globEfield; globEfield = nullptr;
```

```
delete globChordFinder; globChordFinder= nullptr;
```

```
delete globStepper; globStepper = nullptr;
```

```
delete globEquation; globEquation = nullptr;
```

```
}
```

Задание электрического поля в цилиндрическом конденсаторе

Добавление электрического поля в глобальный менеджер

Уравнение движение в электрическом и магнитном полях

Метод Рунге-Кутты для решения уравнения движения

Контроль ошибки интегрирования, минимальный шаг 0.1 мм

Создается объект, рассчитывающий траекторию движения

Созданный объект добавляется в глобальный менеджер

Удаление объекта поля и объектов для расчёта движения в поле

Заданное пользователем глобальное электрическое поле

Geometry.hh

```
#include "MyField.hh"
.....
class Geometry : public G4VUserDetectorConstruction
{
public:
    .....
    MyField* elField;
    .....
}
```

*Объект поля - член класса
с геометрией*

Geometry.cc

```
Geometry::Geometry(std::ofstream& ofsa)
{
    this->f_geom=&ofsa;
    (*f_geom) << "Hi from Geom!" << std::endl;
    elField = new MyField();
}
```

*Объект электрического поля задаётся
в конструкторе класса с геометрией*

```
G4VPhysicalVolume* Geometry::Construct()
{
    .....
    G4bool allLocal = true;
    Pb_log->SetFieldManager(this->elField->myFieldManager, allLocal);
    .....
}
```