The Polyorder Project

A Unified Computing Framework for Self-Consistent Field Theory

Yi-Xin Liu（刘一新）
lyx@fudan.edu.cn
http://ngpy.org

Department of Marcomolecular Science
Fudan University
Shanghai, China

June 28, 2012

Outline

- The Road to **Polyorder**
- C++ and Object-Oriented Programming (OOP)
- The Design of **Polyorder**
  - Framework
  - `Field`
  - `Updater`
  - `Model`
  - `scft`
  - TODO List
- Utilities
  - `load`, `pi`, `pp`
  - `xscft`, `bscft`, `simmon`
  - `gensym`, **Gyroid**

The Road to **Polyorder**

Follow or fork **Polyorder** at:
https://bitbucket.org/liuyxpp/polyorder

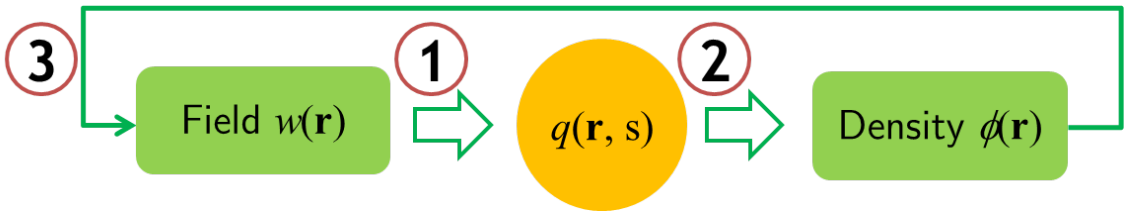
The SCFT Algorithm

Most SCFT equations should be solved numerically.

The set of SCFT equations are highly nonlinear. A common numerical approach is to adapt an iterative algorithm. It mainly contains three parts:

1. Solving modified diffusion equations,
2. Quadrature of propagators along the chain contour, and
3. Updating potential fields with various schemes.

$$\omega_p = \chi_{ps} N \phi_s(\vec{r}) + \sum_{p \neq p'} \chi_{pp'} N \phi_{p'}(\vec{r}) + \eta(\vec{r})$$



$$\frac{\partial q_p}{\partial s} = \nabla^2 q_p - \omega_p q_p$$

$$\phi_p = \frac{\bar{\phi}_p}{Q_p f_p} \int_0^{f_p} ds q_p(\vec{r}, s)\, q_p^*(\vec{r}, f_p - s)$$

An Intuitive Implementation

PSscft of Dr. Wendi Song

The program is written in C++ but with little OO feature.

Implementation details:

- Initialization: `PSscft, init, init2`
- Step 1: `laplace, change3D, calc_q, propag`
- Step 2: `calc_dens`
- Step 3: `renewfield, relax`
- Other: `energy, outdata, close`

This implementation is very specific:

- Must re-compile after modifying any parameters.
- Must re-code almost all the functions for different polymer architectures.
- Difficult to introduce new potential fields (new interactions).
- Difficult to change space dimension.

```
ps.cpp (/export/home/lyx/sandbox)
  function
    PSscft [PSscft]
    calc_dens [PSscft]
    calc_q [PSscft]
    change3D [PSscft]
    close [PSscft]
    draw_color_map [PSscft]
    energy [PSscft]
    init [PSscft]
    init2 [PSscft]
    laplace [PSscft]
    outdata [PSscft]
    propag [PSscft]
    relax [PSscft]
    renewfield [PSscft]
```

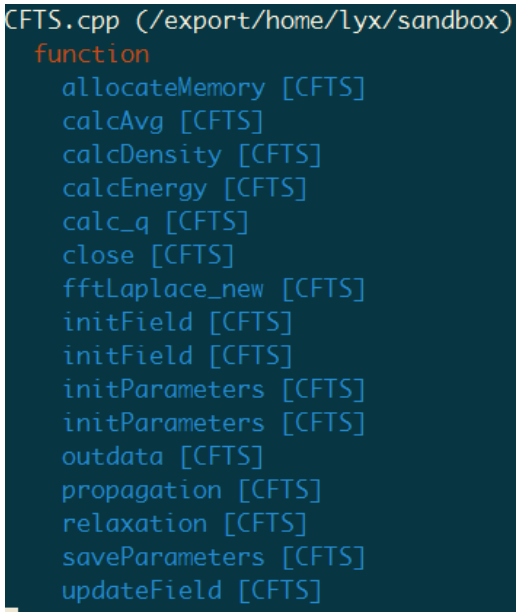
Improved version 1 (2010.3)

CFTS

The program is written in C++ also with little OO feature.

Implementation details:

- Initialization: `allocateMemory`, `initParameters`, `initField`
- Step 1: `fftLaplace_new`, `calc_q`, `propagation`
- Step 2: `calcDensity`
- Step 3: `updateField`, `relaxation`
- Other: `calcAvg`, `calcEnergy`, `outdata`, `saveParameters`, `close`



This implementation is also very specific but with some improvements

- Parameters and data I/O using Matlab MAT file.
- Eliminate the usage of `change3D`.

Improved version 2 (2010.6)

SCFT_nApBSCSalt_CGC_Pseudospectral_1D

The program is written in C++ also with little OO feature.

Implementation details:

- Initialization: `allocateMemory`, `initParameters`, `initField`
- Step 1: `fftLaplace`, `calc_q`, `propagation`
- Step 2: `calcDensity`
- Step 3: `updateField_1S`, `updateField_EM`, `relaxation`
- Other: `calcFieldError`, `findMax`, `calcAvg`, `calcEnergy`, `outData`, `showRange`, `printScreen`, `saveParameters`, `close`

Improvements

- More help functions.
- Support manually select the update scheme.

Things Quickly Mess Up

A new project should be created whenenver there is

- Change of polymer architectures
- Change of components
- Introduction of new interactions
- Introduction of new algorithms
- Change of boundary conditions
- ...

A serious problem inherited is: If you want to revise a common function, such as `outData`, or add a new parameter, you should update all projects simultaneously. It's a disaster!

```
[console 06:00 PM lyx~/project] ls . source_obsolete/
.:
fmg_pbe_1d    fmg_vpbe_2d       sandbox           scft_vfmg_fft_2d
fmg_pbe_2d    fts_dgc_fft_3d    scft_fmg_fft_1d   source_from_others
fmg_pbe_3d    learning_python   scft_fmg_fft_2d   source_obsolete
fmg_q_2d      matlab            scft_fmg_fft_3d
fmg_vpbe_1d   README            scft_fmg_fmg_2d

source_obsolete/:
scft_fmg_fft_2d                   scft_fmg_fft_2d_neutral_batch_cell
scft_fmg_fft_2d_batch_cell        scft_fmg_fft_2d_neutral_batch_fA
scft_fmg_fft_2d_batch_fA          scft_fmg_fft_2d_salt_batch_fA
scft_fmg_fft_2d_neusol_batch_fA   scft_fmg_fft_2d_solvent_batch_fA
scft_fmg_fft_2d_neutral_batch
```

C++ and OOP

Reference

Lippman, S. B.; Lajoie, J.; Moo, B. E.

C++ Primer, 4th Ed. **2005**, Addison-Wesley Professional

What is OOP?

Object-oriented programming

Object-oriented programming (OOP) is a programming paradigm using "objects" – data structures consisting of data fields and methods together with their interactions – to design applications and computer programs. Programming techniques may include features such as data abstraction, encapsulation, messaging, modularity, polymorphism, and inheritance.

Key OOP Features

- Encapsulation
- Inheritance
- Polymorphism

Program is viewed as interacting objects

- Each object contains algorithms to describe its behavior.
- Program design phase involves designing objects and their algorithms.

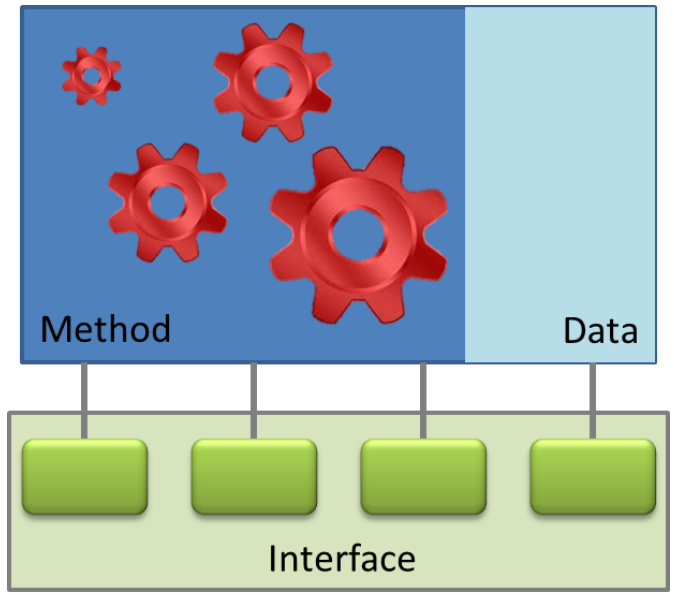
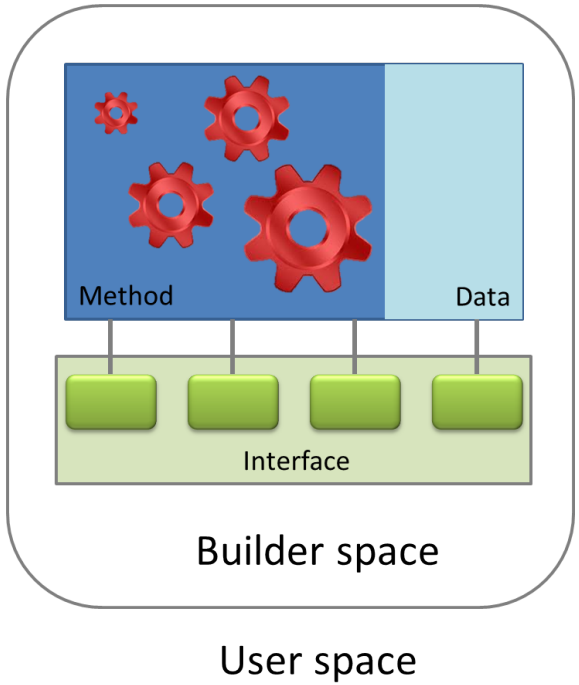


Figure: An object

Encapsulation

- Builder of a concept has detailed view
- User of a concept has abstract view
- Advantages of encapsulation
  - Information hiding
  - Data Protection
  - Consistency
  - Allows change



Inheritance

- Derive a new class from an existing class.
- Create a hierarchy of related classes which share code and interface.
- Represent a "is-a" relationship between objects.

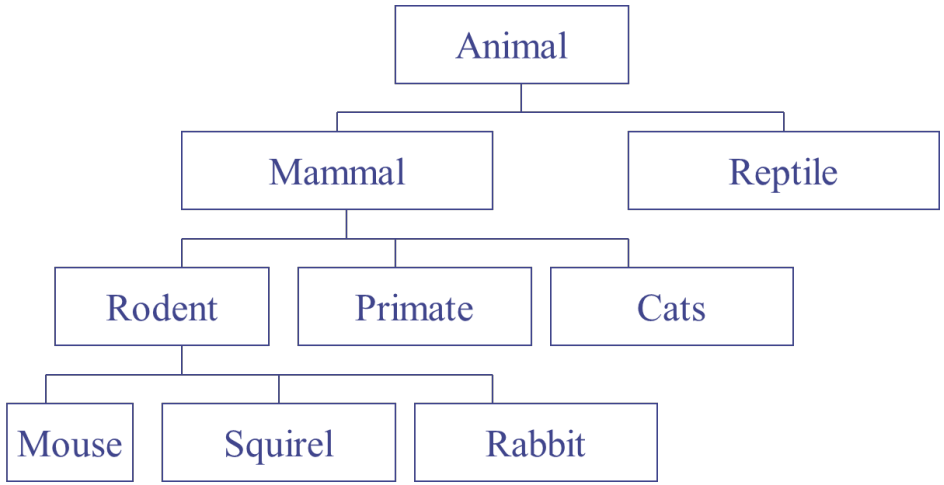
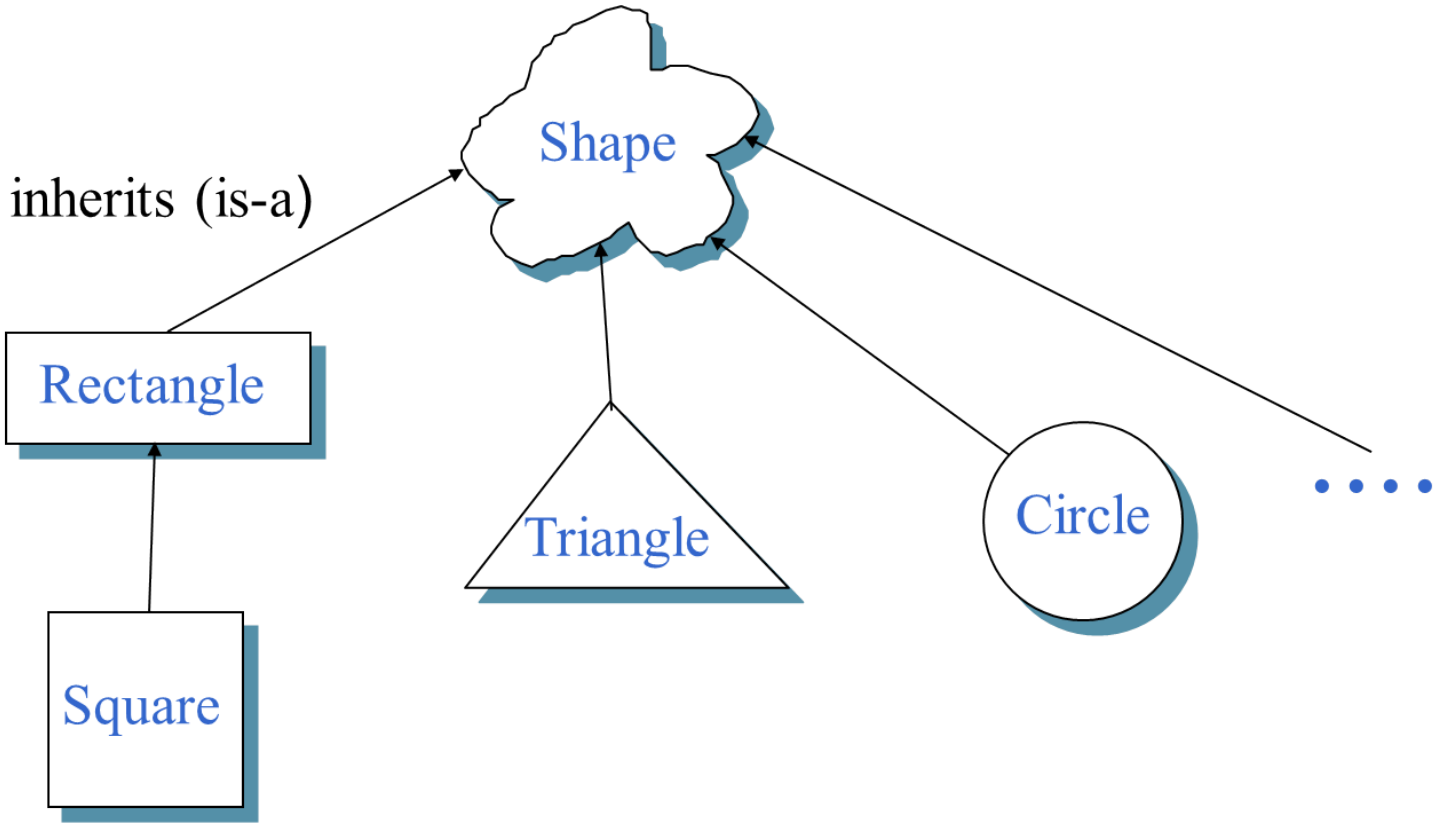


Figure: An inheritance hierarchy

Polymorphism

- The key idea behind OOP.
- derived from a Greek word meaning "many forms".
- Polymorphic objects have same interfaces.
- Dynamic binding.
  - Extensions of the inheritance hierarchy leaves the client's code unaltered.
  - Code is localised - each class is responsible for the meaning of its interfaces.

Example: Shape Library



Shape in C++

```
1 class Shape {
2 public:
3     virtual double area() = 0;
4 };
```

Rectangle, Circle

```
class Rectangle : public Shape {
public:
    Rectangle(double h, double w):height_(h),width_(w){}
    double area() const { return height_ * width_; }
private:
    double height_, width_;
};
```

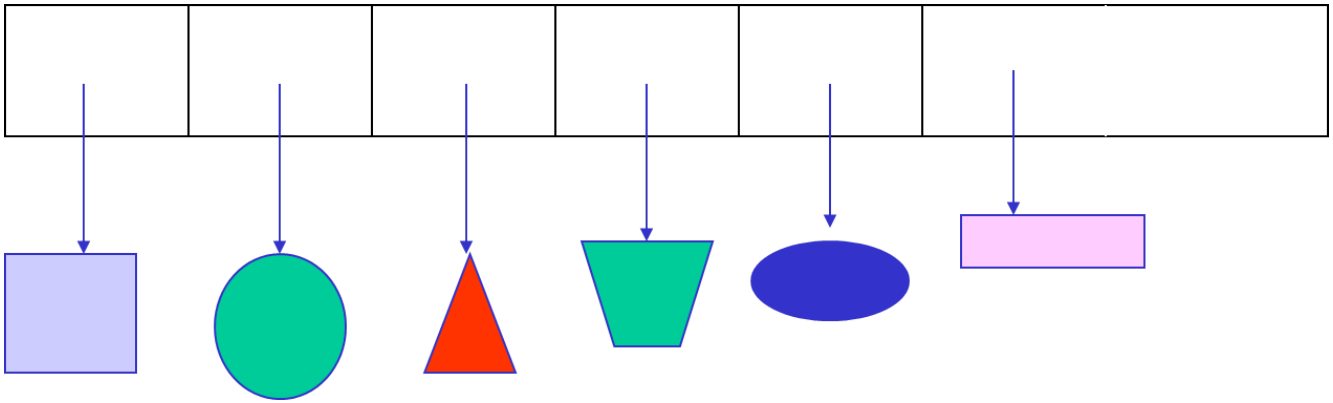
```
class Circle : public Shape {
public:
    Circle(double r):radius_(r){}
    double area() const { return PI * radius_ * radius_; }
private:
    double radius_;
};
```

Square

```
1 class Square : public Rectangle {
2 public:
3     Square(double a):Rectangle(a, a){}
4     double area() const {
5         return this->Rectangle::area();
6     }
7 };
```

The Power of Polymorphism

A list of Shape objects



The following function can calculate the sum of area of all the shapes in the list, even will work when new Shape types are added later on.

```
1 double CalculateAreaSum(Shape *ps, int N){
2     tot_area = 0.0;
3     for(int i=0; i<N; i++){
4         tot_area += ps->area();
5     }
6     return tot_area;
7 }
```

The Design of **Polyorder**

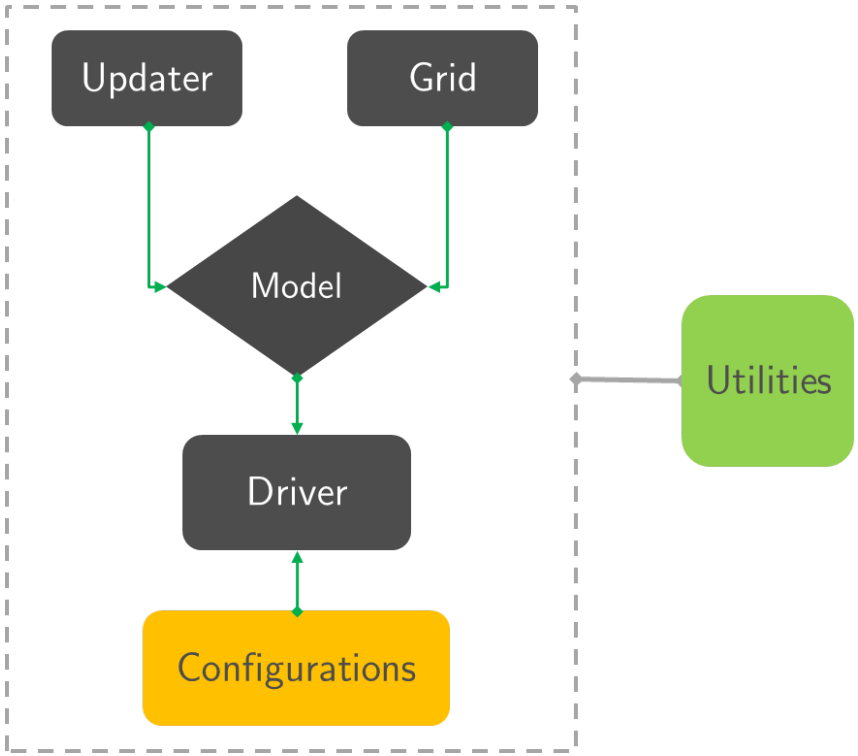
Follow or fork **Polyorder** at:
https://bitbucket.org/liuyxpp/polyorder

Overview

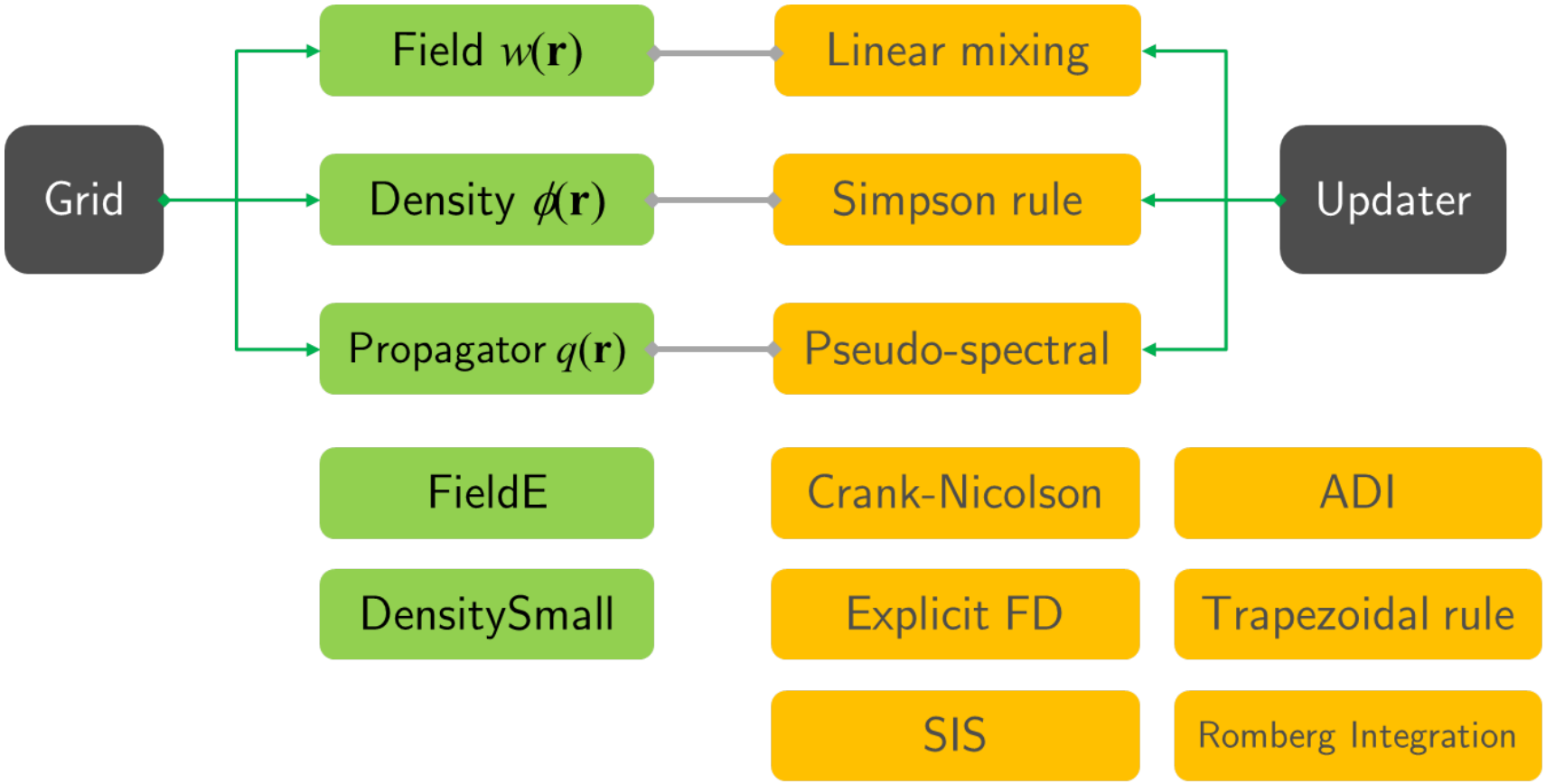
The **Polyorder** project

`Polyorder` is a C++ library which aims to ease the development of polymer self-consistent field theory (SCFT) programs.

The framework



Grid and Updater



Grid

```
class Grid {
public:
    Grid(const UnitCell&, int Lx, int Ly, int Lz);
    // ... More constructors here
    Grid & operator= (const Grid&);
    double & operator() (int ix, int iy, int iz);
    Grid & operator+= (const Grid&);
    // ... More operator overloading here
    const string name() const;
    // ... More parameter interfaces here
    const double mean() const;
    // ... More grid operations here
    virtual void update(); // interface for class hierarchy
protected:
    int Lx_, Ly_, Lz_;
    // ... More members shared with class hierarchy
private:
    // ... Private members and member functions
};
```

Density

```
class Density : public Grid {
public:
    Density(const string, const Config&, const Updater*);
    // ... More constructors here
    Density & operator= (const Density&);
    void update(const Propagator &q, const Propagator &qc);
    void update(const Propagator &q, const Propagator &qc,
                const Updater*);
private:
    Updater *updater_;
    void update_(const Propagator &q, const Propagator &qc);
};
```

Updater

```
class Updater {
public:
    // for Propagator
    virtual void solve(Propagator&, const Grid&);
    // for Density
    virtual void solve(blitz::Array<double,3>, const Propagator&,
                       const Propagator&, double cc) const;
    // for Field
    virtual void solve(Grid&, const Grid&) const;
    virtual Updater *clone() const;
};
```

PseudoSpectral

```
class PseudoSpectral : Updater {
public:
    PseudoSpectral(const UnitCell&, int Lx, int Ly, int Lz,
                   double ds);
    // ... More constructors
    void solve(Propagator&, const Grid&);
    PseudoSpectral *clone() const;
private:
    blitz::Array<double,3> laplace_;
    double *fftw_in_;
    fftw_complex *fftw_out;
    fftw_plan p_forward_, p_backward_;
};
```

Model

```
class Model {
public:
    Model(const Config&);
    virtual void init(const Config&)=0;
    virtual void reset(const Config&)=0;
    virtual void update()=0;
    virtual double H() const=0;
    virtual double Hw() const=0;
    virtual double Hs() const=0;
    virtual double residual_error() const=0;
    virtual double incomp() const=0;
    virtual void display() const();
    virtual void save(const string)=0;
    virtual void save_model(const string)=0;
    virtual void save_field(const string)=0;
    virtual void save_density(const string)=0;
    virtual void save_q(const string)=0;
};
```

Model_AB: AB diblock copolymers

```
class Model_AB : Model {
public:
    // ... Implement all virtual interface in base class Model
private:
    int NA_, NB_, N_;
    double fA_, fB_, Rg_, a_;
    double chiAB_;
    int Ms_, sA_, sB_;
    double ds_;

    Field *wA_, *wB_;
    Yita *yita_;
    Density *phiA_, *phiB_;
    Propagator *qA_, *qB_, *qAc_, *qBc_;
};
```

Model_AB::update

```
1  void Model_AB::update() {
2      // Step 1
3      qA_->update(*wA_);
4      qB_->set_head(qA_->get_tail());
5      qB_->update(*wB_);
6      qBc_->update(*wB_);
7      qAc_->set_head(qBc_->get_tail());
8      qAc_->update(*wA_);
9
10     // Step 2
11     phiA_->update(*qA_, *qAc);
12     phiB_->update(*qB_, *qBc);
13
14     // Step 3
15     yita_->update(*phiA_ + *phiB_ - 1.0);
16     wA_->update(N_ * chiAB_ * (*phiB_) + *yita_);
17     wB_->update(N_ * chiAB_ * (*phiA_) + *yita_);
18 }
```

Driver class: scft

```
class scft {
public:
    scft(const string config_file, Model *pmodel);
    void run();
private:
    Config _cfg;
    Model *model_;
    int iter_, num_iters_;
    double minH_, minH_var;
    blitz::Array<double,1> residual_error_, H_, incomp_;

    void init_(const string);
    void relax_();
    void save_param_();
    // ... More save methods
    void display(double t) const;
}
```

The Configuration File

Section: Model

The configuration file is a standard ini file, which can be parsed by **SimpleIni** of C++ and **ConfigParser** of Python.

```
[Model]
number_of_component = 5
N = 100
a = 0.7
fA = 0.4
chiN = 20
chiAS = 0.0
chiBS = 0.0
phiC = 0.8
cs = 0.02
alphaA = 0.02
alphaB = 0.0
upsA = -1
upsB = 0
upsP = 1
upsN = -1
epsA = 0.01
epsB = 0.01
epsS = 0.208
Ms = 101
seed =
```

The Configuration File

Section: UnitCell and Grid

```
23 [UnitCell]
24 CrystalSystemType = Cubic
25 SymmetryGroup = Ia-3d
26 a = 6.0
27 b =
28 c = 1.4
29 alpha =
30 beta =
31 gamma =
32 N_list =
33 c_list = 16384,18,5,1
34
35 [Grid]
36 dimension = 3
37 Lx = 32
38 Ly = 32
39 Lz = 32
40 lamA = 0.05
41 lamB = 0.05
42 lamS = 0.05
43 lamP = 1.0
44 lamN = 1.0
45 lamPsi = 0.05
46 lamYita = 10.0
47 v1 = -0.05
48 v2 = 0.05
49 gridType =
50 field_data = /export/home/lyx/opt/lyx/polyorder/fields/3D/Ia-3d/32x32x32/fie
   ld_in.mat
51 gridInitType = Random
52 phasePattern =
```

The Configuration File

Section: Algorithm and SCFT

```
54 [Algorithm]
55 dielectric_constant = 0
56 charge_distribution = 0
57 density_integration = 1
58 fft2mg_mode = 0
59 fft2mg_interp = 1
60
61 [SCFT]
62 base_dir = ./
63 data_file = scft_out
64 param_file = param_out
65 min_iter = 3000
66 max_iter = 20000
67 is_display = true
68 is_save_data = true
69 is_save_q = false
70 display_interval = 50
71 record_interval = 10
72 save_interval = 100000
73 thresh_H = 1.0e-6
74 thresh_residual = 3.0e-7
75 thresh_incomp = 1.0e-7
76
```

The Configuration File

Section: Batch and xscft

```
77 [Batch]
78 name = a
79 min = 2.5
80 step = 0.5
81 max = 8.5
82
83 [xscft]
84 nodeFile = nodes
85 activeBatchPath = /export/home/lyx/simulation/active_batch/
86 exeName = ABSe_ps_mud_pbc_0.6
87 exePath = /export/home/lyx/opt/lyx/polyorder/build/bin/
88 dataPath = /export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/
89 dataPathSuffix = -Ran
90 batchScriptVar = f
91 batchScriptMin = 0.2
92 batchScriptStep = 0.02
93 batchScriptMax = 0.8
94 waitTime = 600
95
```

TODO List

- Functional aspect
  - Boundary conditions
  - Error control and smart stop criterion
  - Construct `Model` objects from configuration file
  - More `Updater`s
  - GUI?
- Implementation aspect
  - Abstract Energy calculation
  - Abstract Error calculation
  - Improve `Updater`
  - Space dimension as template
  - Expression template

```
1 Field wA_, wB_;
2 Yita yita_;
3 Density phiA_, phiB_;
4 wA_ = N_ * chiAB_ * phiA_ + yita_;
5 wB_ = N_ * chiAB_ * phiB_ + yita_;
```

Utilities

Follow or fork **Polyorder** at:
https://bitbucket.org/liuyxpp/polyorder

load

`load` is a Perl script that enumerates the CPU loading and the number of free cores of each given nodes. Use

```
$ load -h
$ ...
```

to check more options.

```
[console 03:36 PM lyx~/opt/lyx/scripts] load -a
        NODE    CPU%    FREE CORES
c0101: No route to host
        c0101   0       8
        c0102   89.4    1
        c0103   24.9    6
        c0104   87.9    1
        c0105   96.5    0
        c0106   88.1    1
        c0107   77.7    2
        c0108   77.2    2
        c0109   92.3    0
        c0110   97.4    0
        c0111   89.2    1
        c0112   97.5    0
        c0113   99.3    0
        c0114   76.5    2
        c0115   88      1
        c0116   65.5    3
c0101: No route to host

The first available node in {c0101 c0102 c0103 c0104 c0105 c0106 c0107 c0108 c01
09 c0110 c0111 c0112 c0113 c0114 c0115 c0116}: [c0101]
Total free cores: [28]
```

pi

`pi` is a Perl script that lists all processes of a user on each given nodes. Use

```
$ pi -v
$ ...
```

to check more options.

```
[console 03:50 PM lyx~/opt/lyx/scripts] pi -u xienan
The processes of user [xienan] are:

        NODE    PROGRAM         PID     CPU%
        c0102   ../../../a2dcyl_v5.out  631     99
        c0102   ../../../a2dcyl_v5.out  799     98
        c0102   ../../../a2dcyl_v5.out  1816    99
        c0102   ../../../a2dcyl_v5.out  3108    99
        c0102   ../../../a2dcyl_v5.out  29710   98
        c0102   ../../../a2dcyl_v5.out  30893   98
        c0102   ../../../a2dcyl_v5.out  32061   98

        Total processes of xienan: [7]. Of which:
        7 at node [c0102]
```

pp

pp is a Perl script that finds the full path of the executable file of a process with given PID. Use

```
$ pp -v
$ ...
```

to check more options.

```
[console 03:57 PM lyx~/opt/lyx/scripts] pp -i 18466 -n 9
The bin path of process 18466 on NODE [9] is:
/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-annealed/fA/e0.208eA0.01fC0.8k2
0kAS0.0kBS0.0p0.02s0.02-Ran/fA0.26/f0.26e0.208eA0.01fC0.8k20kAS0.0kBS0.0p0.02s0.
02
```

xscft

`xscft` is a Python script that automatically submits SCFT tasks. It use the same configuration file as **Polyorder**. Use

```
$ xscft -h
$ ...
```

to check more options.

```
/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8k20
kAS0.0kBS0.0p0.02s0.02-Ran/fA0.36/f0.36e0.208eA0.01fC0.8k20kAS0.0kBS0.0p0.02s0.0
2was submitted to node c0116 with PID 1100
/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8k20
kAS0.0kBS0.0p0.02s0.02-Ran/fA0.38/f0.38e0.208eA0.01fC0.8k20kAS0.0kBS0.0p0.02s0.0
2was submitted to node c0103 with PID 27963
/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8k20
kAS0.0kBS0.0p0.02s0.02-Ran/fA0.4/f0.4e0.208eA0.01fC0.8k20kAS0.0kBS0.0p0.02s0.02w
as submitted to node c0106 with PID 31928
/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8k20
kAS0.0kBS0.0p0.02s0.02-Ran/fA0.42/f0.42e0.208eA0.01fC0.8k20kAS0.0kBS0.0p0.02s0.0
2was submitted to node c0103 with PID 29910
/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8k20
kAS0.0kBS0.0p0.02s0.02-Ran/fA0.44/f0.44e0.208eA0.01fC0.8k20kAS0.0kBS0.0p0.02s0.0
2was submitted to node c0104 with PID 28218
/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8k20
kAS0.0kBS0.0p0.02s0.02-Ran/fA0.46/f0.46e0.208eA0.01fC0.8k20kAS0.0kBS0.0p0.02s0.0
2was submitted to node c0103 with PID 31513

No free node available. Waiting for 600 seconds to try again.
Waiting....!
```

bscft

`bscft` is a Python script that performs basic analysis for SCFT batch tasks. It also use the same configuration file as **Polyorder**. Use

```
$ bscft -h
$ ...
```

to check more options.

```
/export/home/lyx/simulation/scft_pe_3d/nApB-ceps-annealed/fA/e0.208eA0.01fC0.8k2
0kAS0.0kBS0.0p0.02s0.02-Ran/fA0.2                                              \
/export/home/lyx/simulation/scft_pe_3d/nApB-ceps-annealed/fA/e0.208eA0.01fC0.8k2
0kAS0.0kBS0.0p0.02s0.02-Ran/fA0.22                                             |
/export/home/lyx/simulation/scft_pe_3d/nApB-ceps-annealed/fA/e0.208eA0.01fC0.8k2
0kAS0.0kBS0.0p0.02s0.02-Ran/fA0.24                                             |
/export/home/lyx/simulation/scft_pe_3d/nApB-ceps-annealed/fA/e0.208eA0.01fC0.8k2
0kAS0.0kBS0.0p0.02s0.02-Ran/fA0.26                                             |
/export/home/lyx/simulation/scft_pe_3d/nApB-ceps-annealed/fA/e0.208eA0.01fC0.8k2
0kAS0.0kBS0.0p0.02s0.02-Ran/fA0.28                                             /
/export/home/lyx/simulation/scft_pe_3d/nApB-ceps-annealed/fA/e0.208eA0.01fC0.8k2
0kAS0.0kBS0.0p0.02s0.02-Ran/fA0.3                                              \
/export/home/lyx/simulation/scft_pe_3d/nApB-ceps-annealed/fA/e0.208eA0.01fC0.8k2
0kAS0.0kBS0.0p0.02s0.02-Ran/fA0.32                                             \
/export/home/lyx/simulation/scft_pe_3d/nApB-ceps-annealed/fA/e0.208eA0.01fC0.8k2
0kAS0.0kBS0.0p0.02s0.02-Ran/fA0.34                                             \
/export/home/lyx/simulation/scft_pe_3d/nApB-ceps-annealed/fA/e0.208eA0.01fC0.8k2
0kAS0.0kBS0.0p0.02s0.02-Ran/fA0.36                                             /
/export/home/lyx/simulation/scft_pe_3d/nApB-ceps-annealed/fA/e0.208eA0.01fC0.8k2
0kAS0.0kBS0.0p0.02s0.02-Ran/fA0.38                                             /
```

simmon

`simmon` is a Perl script that monitors the active SCFT tasks and performs basic analysis using `bscft` when a batch task is done. No options are available.

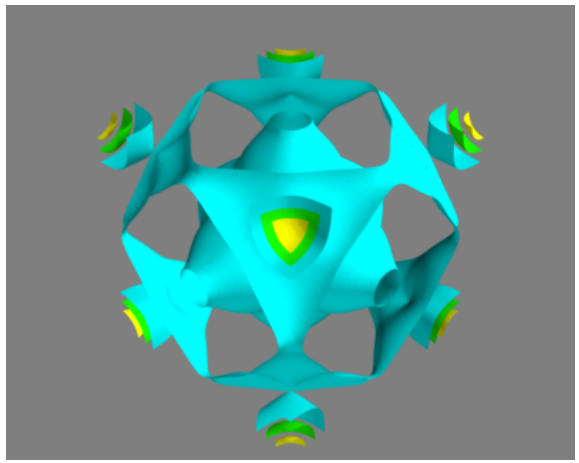
```
-->/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8
k20kAS0.0kBS0.0p0.02s0.02-Ran/fA0.3
-->/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8
k20kAS0.0kBS0.0p0.02s0.02-Ran/fA0.32
-->/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8
k20kAS0.0kBS0.0p0.02s0.02-Ran/fA0.34
-->/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8
k20kAS0.0kBS0.0p0.02s0.02-Ran/fA0.36
-->/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8
k20kAS0.0kBS0.0p0.02s0.02-Ran/fA0.38
-->/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8
k20kAS0.0kBS0.0p0.02s0.02-Ran/fA0.4
-->/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8
k20kAS0.0kBS0.0p0.02s0.02-Ran/fA0.42
-->/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8
k20kAS0.0kBS0.0p0.02s0.02-Ran/fA0.44
-->/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8
k20kAS0.0kBS0.0p0.02s0.02-Ran/fA0.46
-->/export/home/lyx/simulation/scft_pe_3d/nA0B-ceps-smeared/fA/e0.208eA0.01fC0.8
k20kAS0.0kBS0.0p0.02s0.02-Ran/fA0.48
batchdir-20120626-160246 is still active!

Waiting for 600 seconds to check again.
Waiting....-
```

gensym

`gensym` is a Python script that generates patterns according to space group symmetry using **Gyroid** software package. It also use the same configuration file as **Polyorder**. Use

```
$ gensym -h
$ ...
```

to check more options.



Symmetry Group: $\mathrm{Ia\bar{3}d}$, Grid: $64 \times 64 \times 64$

Gyroid

Gyroid is a Python package that generates symmetry adapted basis functions (SABF) based on the space group of a unit cell.

Typical usage

```
$ python
>>> import gyroid as gy
>>> import numpy as np
>>> N1, N2, N3 = 32, 32, 32
>>> uc = gy.UnitCell(3)
>>> group = gy.Group(3, gy.BRAVAIS, uc.shape, 'Ia-3d')
>>> grid = gy.Grid(np.array([N1,N2,N3], group)
>>> basis = gy.Basis(group, grid)
>>> gy.render_structure_3d(basis, grid, N1, N2, N3, 1.0)
>>> exit()
$ ...
```

Follow or fork **Gyroid** at: https://bitbucket.org/liuyxpp/gyroid

Thanks!