NATIONAL BIOMEDICAL COMPUTATION RESOURCE
Conduct, catalyze and enable multiscale biomedical research

Introduction to Finite Element ToolKit

Yuhui Cheng

ycheng@mccammon.ucsd.edu

NBCR Summer Institute

Aug. 6, 2008

http://mccammon.ucsd.edu/smol/doc/lectures/nbcr080608_1.pdf

Outline

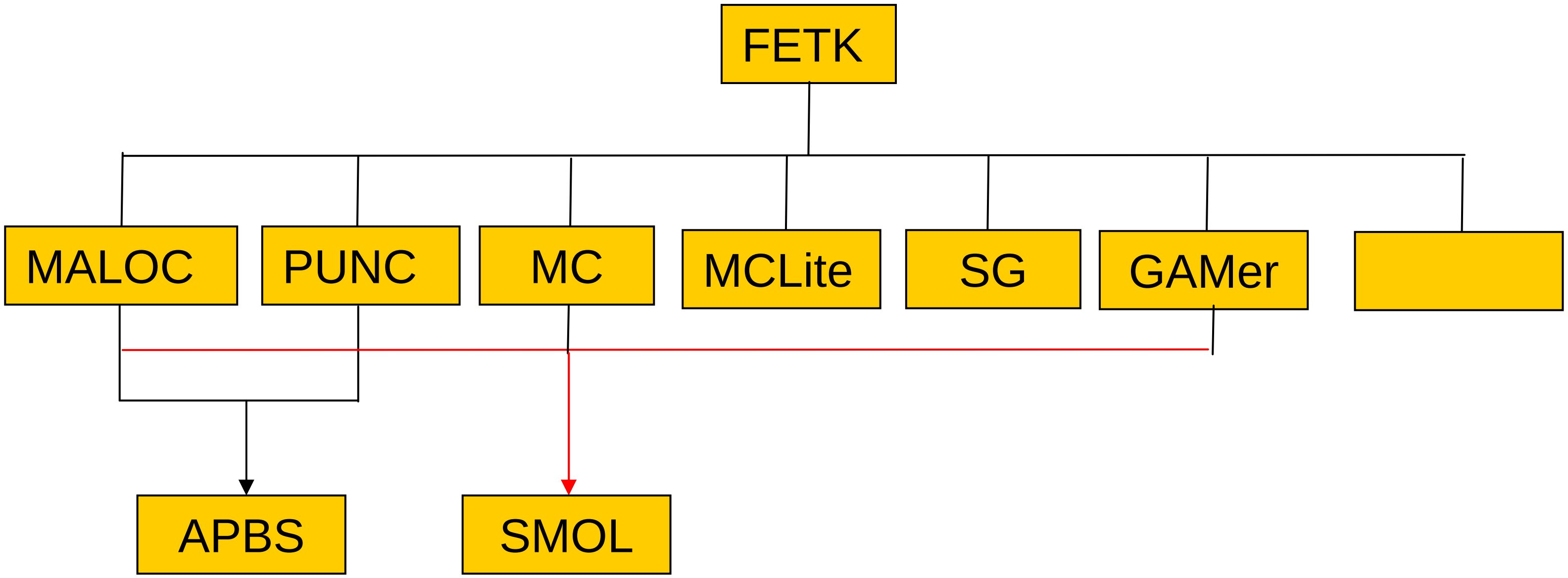
- To introduce the fundamental of the finite element tool kit (FEtk).

- Programming hints for applying the FEtk package.

- Summary remarks.

The contents of the FEtk package

- You can download the package from the below site: http://www.fetk.org



How do we need FETK?

FETK has been designed to solve various PDE(s) with finite element method.

$$
\begin{aligned}
-\nabla \cdot (a(x)\nabla u(x)) + b(x, u(x)) &= 0 \ \text{ in } \Omega, \\
u(x) &= g(x) \ \text{ on } \partial\Omega,
\end{aligned}
$$

$$
a : \Omega \mapsto \mathbb{R}^{3\times 3}, \quad b : \Omega \times \mathbb{R} \mapsto \mathbb{R}, \quad g : \partial\Omega \mapsto \mathbb{R}.
$$

Finite element discretization of PDEs

1. Define a function space $V_h=\{v_i\}$ (v_i : piece-wise linear FE basis functions defined over each tetrahedral vertex), and assume the solution to the PDE has the form of

$$u_h(\vec{r}) = \sum_i a_i v_i(\vec{r}),\ u_h \in \overline{u}_h + V_h$$

2. The original problem is equal to solve the below problem:

$$\text{Find } u \in \bar{u} + H_0^1(\Omega) \text{ such that } \langle F(u), v \rangle = 0 \quad \forall v \in H_0^1(\Omega),$$

$$\langle F(u), v \rangle = \int_\Omega (a \nabla u \cdot \nabla v + b(x,u)v)\ dx.$$

The weak form is not linear for u, but linear for v.

Bilinear linearization form of PDE

To apply a Newton iteration, we need to linearize $< F(u), v >$

$$\langle DF(u)w, v\rangle = \frac{d}{dt}\langle F(u+tw), v\rangle = \int_\Omega D\nabla w \cdot \nabla v dx$$

Algorithm 3.2. *(Damped-inexact-Newton)*

- *Given an initial* u
- *While* $(|\langle F(u), v\rangle| > TOL$ for some $v)$ *do:*
  (1) *Find* δ *such that* $\langle DF(u)\delta, v\rangle = -\langle F(u), v\rangle + r, \ \forall v$
  (2) *Set* $u = u + \lambda\delta$
- *end while*

Posteriori ERROR ESTIMATOR

$$\eta_s = \left(h_s^2 \| b(u_h) - \nabla \cdot (a \nabla u_h) \|_{L^2(s)}^2 + \frac{1}{2} \sum_{f \in \mathcal{I}(s)} h_f \| \, [n \cdot (a \nabla u_h)]_f \, \|_{L^2(f)}^2 \right)^{1/2}$$

h_s represent the size of the element.

$f \in I(S)$ denotes a face of simplex

h_s is the size of the face f

$[n \cdot (a \nabla u_h)]_f$ denotes to the ``jump'' term across faces interior to the simplex.

$$err = \sqrt{\sum_s \eta_s^2}$$

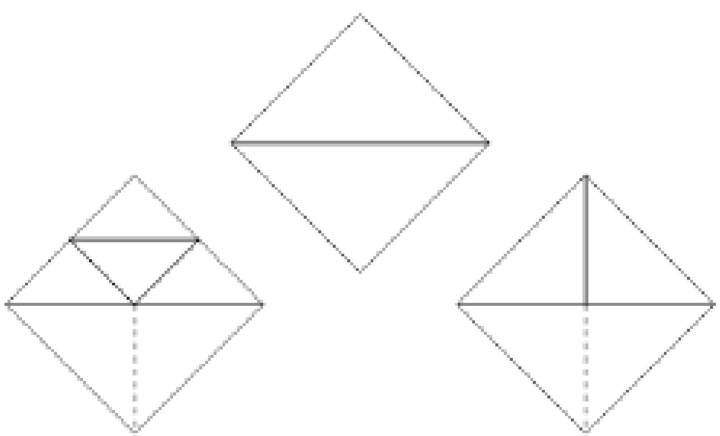
Solve → Estimate → Refine

Mesh Marking and Refinement

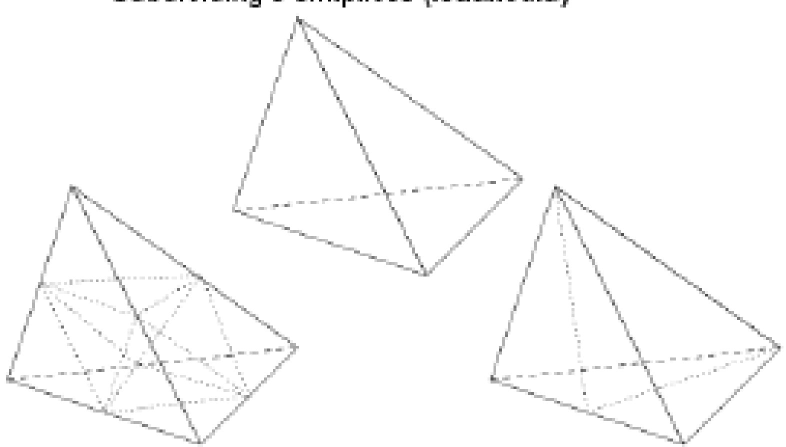
Algorithm 3.1. *(Adaptive multilevel finite element approximation)*

- *While* $(\|u - u_h\|_X > \epsilon)$ *do:*
  (1) *Find* $u_h \in \bar{u}_h + V_h \subset H_0^1(\Omega)$ *such that* $\langle F(u_h), v_h \rangle = 0, \ \forall \ v_h \in V_h \subset H_0^1(\Omega)$.
  (2) *Estimate* $\|u - u_h\|_X$ *over each element.*
  (3) *Initialize two temporary simplex lists as empty:* $Q1 = Q2 = \emptyset$.
  (4) *Place simplices with large error on the "refinement" list* $Q1$.
  (5) *Bisect all simplices in* $Q1$ *(removing them from* $Q1$*),*
  and place any nonconforming simplices created on the list $Q2$.
  (6) $Q1$ *is now empty; set* $Q1 = Q2$, $Q2 = \emptyset$.
  (7) *If* $Q1$ *is not empty, goto (5).*
- *End While.*

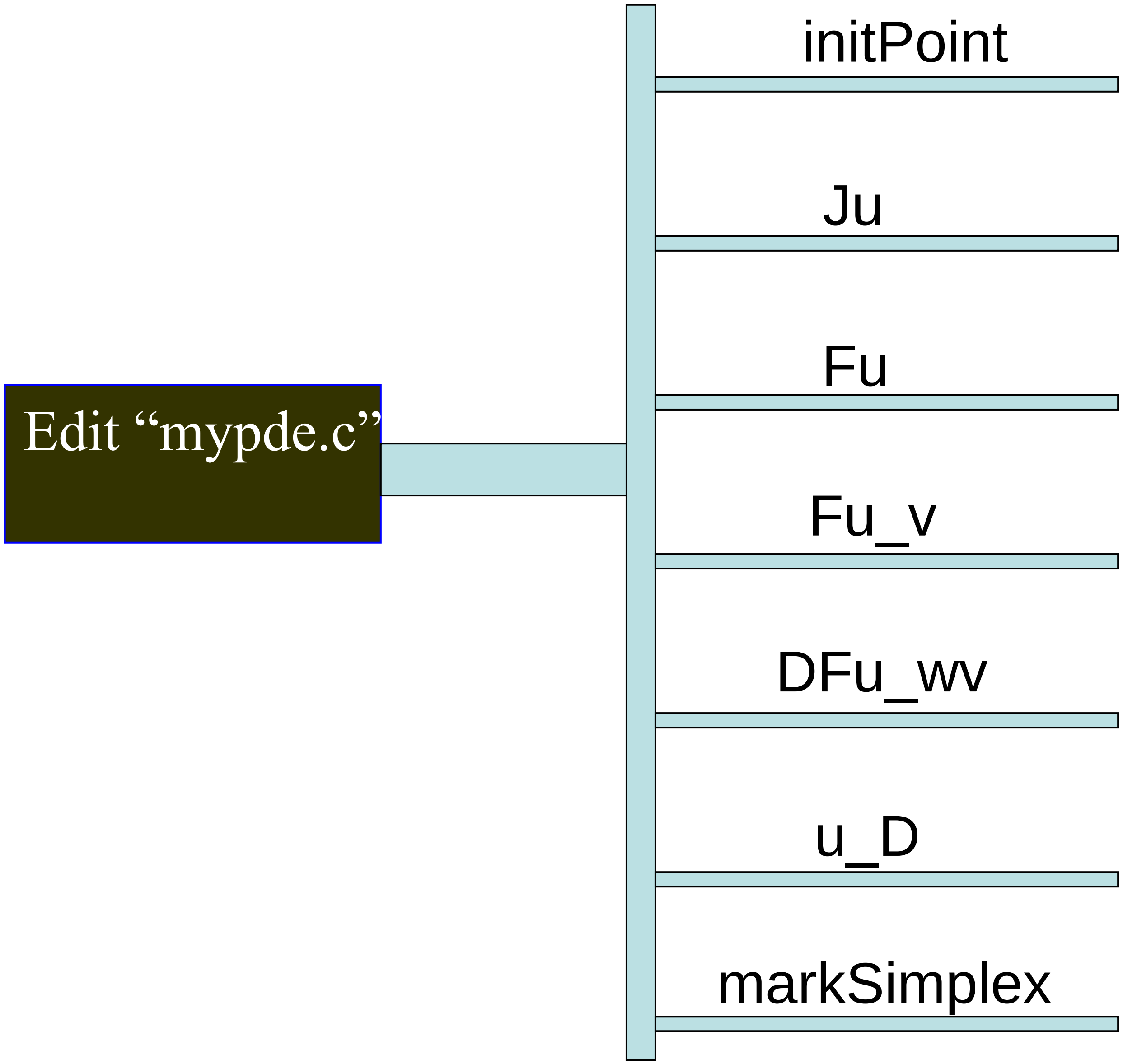
Subdividing 2-simplices (triangles)



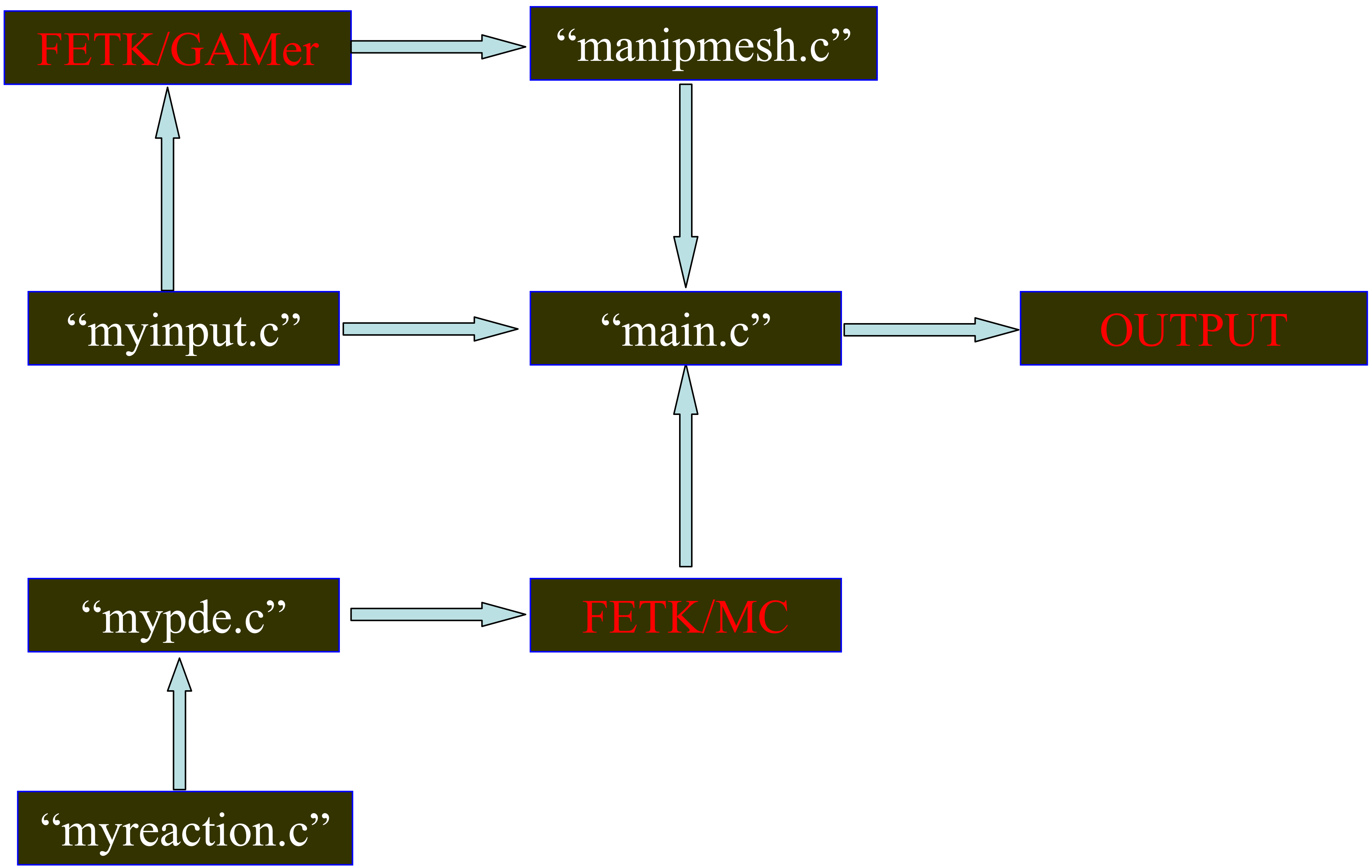
Subdividing 3-simplices (tetrahedra)



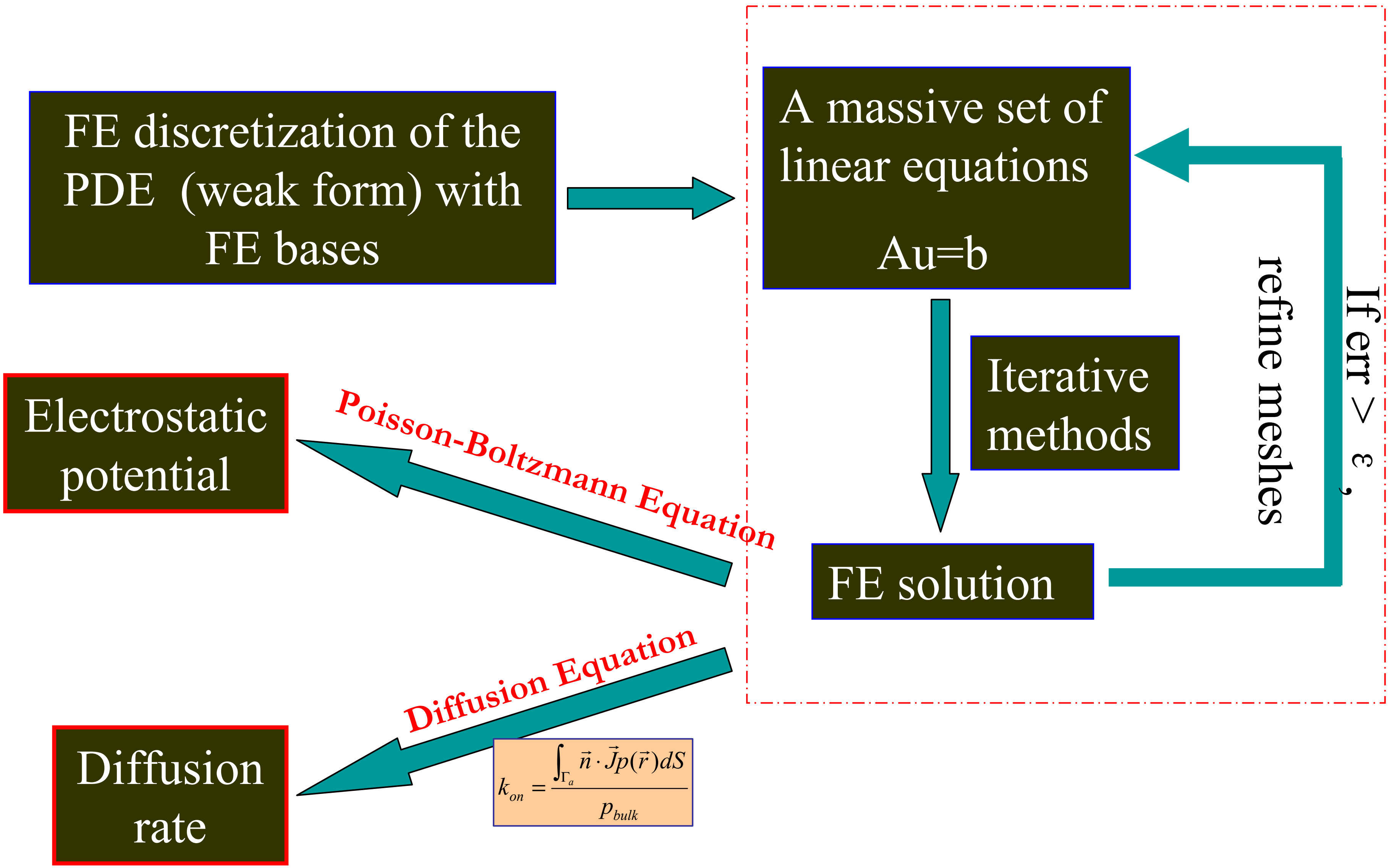
FETK Programming



FETK Programming



Solving Electrostatics and Diffusion by FEtk



Specific Equations in Computational Molecular Modeling

Poisson-Boltzmann Equation

$$-\nabla \cdot \varepsilon(x)\nabla U(x) + \overline{\kappa}^2(x)\sinh U(x) - \frac{4\pi e_c^2}{kT}\sum_i z_i \delta(x - x_i) = 0$$

Smoluchowski Diffusion Equation

$$-\nabla \cdot D e^{-\beta U(\vec{r})} \nabla e^{\beta U(\vec{r})} p(\vec{r}, t \mid \vec{r}_0, t_0) + \frac{\partial p(\vec{r}, t \mid \vec{r}_0, t_0)}{\partial t} = 0$$

$$D'(\vec{r}) = D e^{-\beta U(\vec{r})} \qquad p'(\vec{r}, t) = e^{\beta U(\vec{r})} p(\vec{r}, t \mid \vec{r}_0, t_0)$$

$$-\nabla \cdot D'(\vec{r}) \nabla p'(\vec{r}, t) + e^{-\beta U(\vec{r})} \frac{\partial p'(\vec{r}, t)}{\partial t} = 0$$

Set up your own project with FETK

1. Write down the PDE strong forms, and then derive the weak form.
2. Edit “mypde.c” template in FETK for your PDE. Be careful with the boundary setup.
3. Write your main program and read your initial parameters into FETK.
4. Test your solver with analytical result.
5. You have done with your work.

Pro and Con of FETK

FETK is an excellent finite element package, in which you can find lots of useful subroutines for your numerical experiment. It's convenient to program a new PDE solver based on it. It's robust for complicated geometries and PDEs. But the finite element has its own limits. It need unstructured tetrahedral meshes, which is time-consuming for generating. To obtain very accurate output, the mesh refinement is always a challenging work.