Top Math Summer School on

Adaptive Finite Elements: Analysis and Implementation
Organized by: Kunibert G. Siebert

Pedro Morin

Instituto de Matemática Aplicada del Litoral
Universidad Nacional del Litoral

Santa Fe - Argentina

July 28 – August 2, 2008
Frauenchiemsee, Germany

Outline

A Posteriori Error Estimates

We consider the problem

$$\begin{aligned} -\operatorname{div}(a\nabla u) + b\cdot\nabla u + c\,u = f \qquad & \text{in } \Omega, \\ u = g_D \qquad & \text{on } \Gamma_D, \\ a\frac{\partial u}{\partial n} = g_N \qquad & \text{on } \Gamma_N, \end{aligned}$$

The Residual Type A Posteriori Error Estimators are:

$$\eta_{\mathcal{T}}^2(T) = C_1\, h_T^2\, \|r\|_{L_2(T)}^2 + C_2\, h_T\, \|j\|_{L_2(\partial T)}^2$$

Where

$$r = -\operatorname{div}(a\nabla u_{\mathcal{T}}) + b\cdot\nabla u_{\mathcal{T}} + cu_{\mathcal{T}} - f$$

$$j_{|S} = \begin{cases} 0 & \text{if } S\subset\Gamma_D \\ a\nabla u_{\mathcal{T}}\cdot n - g_N & \text{if } S\subset\Gamma_N \\ a^1\nabla u_{\mathcal{T}}^1\cdot n^1 + a^2\nabla u_{\mathcal{T}}^2\cdot n^2 & \text{if } S \text{ is interior} \end{cases}$$

(We have neglected the terms related to the approximation of the Dirichlet boundary condition)

A Posteriori Error Estimates

We consider the problem

$$\begin{aligned} -\operatorname{div}(a\nabla u) + b\cdot\nabla u + c\,u = f \qquad &\text{in } \Omega,\\ u = g_D \qquad &\text{on } \Gamma_D,\\ a\frac{\partial u}{\partial n} = g_N \qquad &\text{on } \Gamma_N, \end{aligned}$$

The Residual Type A Posteriori Error Estimators are:

$$\eta_{\mathcal{T}}^2(T) = C_1\, h_T^2\, \|r\|^2_{L_2(T)} + C_2\, h_T\, \|j\|^2_{L_2(\partial T)}$$

Where

$$r = [-\operatorname{div}(a\nabla u_{\mathcal{T}})+]b\cdot\nabla u_{\mathcal{T}} + cu_{\mathcal{T}} - f \qquad \text{(if } a \text{ is constant)}$$

$$j_{|S} = \begin{cases} 0 & \text{if } S\subset\Gamma_D\\ a\nabla u_{\mathcal{T}}\cdot n - g_N & \text{if } S\subset\Gamma_N\\ a|\nabla u_{\mathcal{T}}^1 - \nabla u_{\mathcal{T}}^2| & \text{if } S \text{ is interior (if } a \text{ is constant)} \end{cases}$$

(We have neglected the terms related to the approximation of the Dirichlet boundary condition)

Upper and Lower Bounds

There exist two constants $0 < c_\ell, C_u < \infty$ such that

$$\|u - u_{\mathcal{T}}\|^2_{H^1(\Omega)} \leq C_u \sum_{T \in \mathcal{T}} \eta^2_{\mathcal{T}}(T)$$

and

$$c_\ell \, \eta^2_{\mathcal{T}}(T) \leq \|u - u_{\mathcal{T}}\|^2_{H^1(\omega_{\mathcal{T}}(T))} + h_T^2 \|r - \overline{r}\|^2_{L^2(\omega_{\mathcal{T}}(T))}$$

IMPLEMENTATION of a Posteriori Error Estimates

We assume that we have a mesh described by a structure with the following fields (among others):

- `mesh.vertex_coordinates`
- `mesh.element_vertices`
- `mesh.element_neighbours`
- `mesh.element_boundary`
- `mesh.n_elem`

IMPLEMENTATION of a Posteriori Error Estimates

We assume that we have a mesh described by a structure with the following fields (among others):

- `mesh.vertex_coordinates`
- `mesh.element_vertices`
- `mesh.element_neighbours`
- `mesh.element_boundary`
- `mesh.n_elem`

We will write the function

```
function global_est = estimate(prob_data, adapt)
```

which receives two data structures with the following information

- `prob_data.f`: the right-hand side function f
- `prob_data.a,b,c`: parameters a, b, c of the elliptic equation (assumed constant)
- `adapt.C`: a vector containing C_1 and C_2 from the definition of the estimators (weights for the interior and jump residual, respectively).

Recall

On an element T, φ_T^i, $i = 1, 2, 3$ denote the local basis functions.

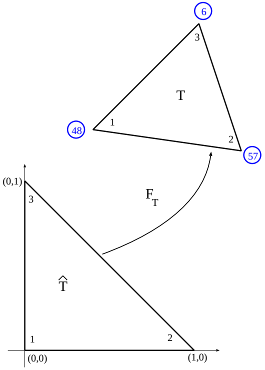
And u_T^i, $i = 1, 2, 3$ denote de nodal values of $u_\mathcal{T}$ at the local nodes.

Then

$$u_\mathcal{T} = \sum_{i=1}^{3} u_T^i \varphi_i$$

And, therefore

$$\begin{aligned}\nabla u_\mathcal{T} &= \nabla \left(\sum_{i=1}^{3} u_T^i \varphi_i \right) \\ &= \sum_{i=1}^{3} u_T^i \nabla \varphi_i \\ &= \begin{bmatrix} \nabla\varphi_1 & \nabla\varphi_2 & \nabla\varphi_3 \end{bmatrix} \begin{bmatrix} u_T^1 \\ u_T^1 \\ u_T^3 \end{bmatrix}\end{aligned}$$

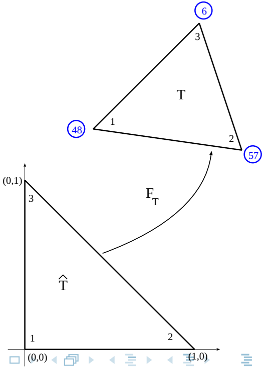


Computation of the gradient of $u_{\mathcal{T}}$

$$\begin{aligned} \hat\phi_1 &= 1 - \hat x_1 - \hat x_2 \\ \hat\phi_2 &= \hat x_1 \\ \hat\phi_3 &= \hat x_2 \end{aligned} \qquad \Longrightarrow \qquad \hat\nabla\hat\phi_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \quad \hat\nabla\hat\phi_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \hat\nabla\hat\phi_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Then $\hat\phi_i = \varphi_i \circ F_T.$ and $\nabla\varphi_i = B^{-T}\hat\nabla\hat\phi_i$

$$\begin{aligned} \nabla u_{\mathcal{T}} &= [\nabla\varphi_1 \quad \nabla\varphi_2 \quad \nabla\varphi_3] \begin{bmatrix} u_T^1 \\ u_T^1 \\ u_T^3 \end{bmatrix} \\ &= B^{-T}[\nabla\hat\phi_1 \quad \nabla\hat\phi_2 \quad \nabla\hat\phi_3] \begin{bmatrix} u_T^1 \\ u_T^1 \\ u_T^3 \end{bmatrix} \\ &= B^{-T} \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_T^1 \\ u_T^1 \\ u_T^3 \end{bmatrix} \\ &= B^{-T}\texttt{grd\_bas\_fcts} \begin{bmatrix} u_T^1 \\ u_T^1 \\ u_T^3 \end{bmatrix} \end{aligned}$$



file estimate.m

```
function global_est = estimate(prob_data, adapt)
%   computes the residual type a posteriori error estimators
%   for the elliptic problem

global mesh uh
n_elem = mesh.n_elem;
grduh = zeros(n_elem, 2);

grd_bas_fcts = [ -1 -1 ; 1 0 ; 0 1 ]' ;

for el = 1:n_elem
  v_elem = mesh.elem_vertices(el, :);
  v1 = mesh.vertex_coordinates( v_elem(1), : )' ;
  v2 = mesh.vertex_coordinates( v_elem(2), : )' ;
  v3 = mesh.vertex_coordinates( v_elem(3), : )' ;
  B = [ v2-v1 , v3-v1 ];

  grduh(el, :) = ( (B') \ (grd_bas_fcts*uh(v_elem)) )';
end
```

file `estimate.m`, computation of interior residual

```
for el = 1:n_elem
  v_elem = mesh.elem_vertices(el, :);  v1 = ... ;  v2 = ... ;  v3 = ...
  B = [ v2-v1 , v3-v1 ];  el_area = abs(det(B))/2; h_T = sqrt(el_area);

  % local degrees of freedom
  uh1 = uh(v_elem(1));  uh2 = uh(v_elem(2));  uh3 = uh(v_elem(3));

  %   midpoints of the sides
  m12 = (v1 + v2)/2;  m23 = (v2 + v3)/2;  m31 = (v3 + v1)/2;
  %   values of rhs f at midpoints
  f12 = feval(prob_data.f, m12); f23 = feval(prob_data.f, m23);  f31 =
  %   finite element function at midpoints
  uh12 = (uh1 + uh2)/2;  uh23 = (uh2 + uh3)/2;  uh31 = (uh3 + uh1)/2;

  r12 = prob_data.b*grduh(el,:)' + prob_data.c*uh12 - f12;
  r23 = prob_data.b*grduh(el,:)' + prob_data.c*uh23 - f23;
  r31 = prob_data.b*grduh(el,:)' + prob_data.c*uh31 - f31;
```

file estimate.m, computation of jump residual

```
for el = 1:n_elem
  [...]
  % now the jumps
  jump_res2 = 0;
  for side = 1:3
     vec = [0 0];
     if (mesh.elem_boundaries(el, side) == 0)     % interior side
        vec = (grduh(el,:) - grduh(mesh.elem_neighbours(el,side),:))...
              * prob_data.a ;
     elseif (mesh.elem_boundaries(el, side) < 0)   % Neumann side
        switch (side)
          case 1,     tangential = v3 - v2;
          case 2,     tangential = v1 - v3;
          case 3,     tangential = v2 - v1;
        end  % now rotate clockwise to get an outer normal
       normal = [tangential(2) ; -tangential(1)]/norm(tangential);
       vec = prob_data.a*grduh(el,:)*normal - 0 '; % g_N = 0
     end
     jump_res2 = jump_res2 + vec*vec';
  end
```

file `estimate.m`, summing up

```
function global_est = estimate(prob_data, adapt)

[...] --> computation of gradients at each element

mesh.estimator = zeros(n_elem, 1);
for el = 1:n_elem
  v_elem = mesh.elem_vertices(el, :);

  [...] --> computation of integrands of interior and jump residuals

  mesh.estimator(el) = sqrt(  adapt.C(1) * h_T^2                         ...
                                  * el_area * (r12^2+r23^2+r31^2)/3 ...
                            + adapt.C(2) * h_T * h_T * jump_res2);
end

mesh.est_sum2 = mesh.estimator' * mesh.estimator;
mesh.max_est  = max(mesh.estimator);

global_est = sqrt(mesh.est_sum2);
```

Marking Strategies

There are several popular marking strategies

Global Refinement (GR)

$$\mathcal{M}_k = \mathcal{T}_k$$

Not very intelligent!

Marking Strategies

There are several popular marking strategies

Maximum Strategy (MS)

Choose a parameter $0 < \gamma < 1$ and let

$$\mathcal{M}_k := \{T \in \mathcal{T}_k : \eta_k(T) > \gamma\, \eta_{k,\mathsf{max}}\}$$

where $\eta_{k,\mathsf{max}} := \max_{T \in \mathcal{T}_k} \eta_k(T)$

Marking Strategies

There are several popular marking strategies

Equidistribution Strategy (ES)

Let $\varepsilon > 0$ be a desired tolerance for $\displaystyle\sum_{T\in\mathcal{T}_k} \eta_k^2(T)$.

Choose a parameter $0 < \theta < 1$ (usually $\theta \approx 1$), and let

$$\mathcal{M}_k := \left\{ T \in \mathcal{T}_k : \eta_k^2(T) > \theta^2 \frac{\varepsilon}{\#\mathcal{T}_k} \right\}$$

Marking Strategies

There are several popular marking strategies

Modified Equidistribution Strategy (MES)

Choose a parameter $0 < \theta < 1$ (usually $\theta \approx 1$), and let

$$\mathcal{M}_k := \left\{ T \in \mathcal{T}_k : \eta_k^2(T) > \theta^2 \frac{\sum_{T \in \mathcal{T}} \eta_{\mathcal{T}}^2(T)}{\# \mathcal{T}_k} \right\}$$

Marking Strategies

There are several popular marking strategies

Dörfler's Strategy, or Guaranteed Error Reduction Strategy (GERS)

Let $\mathcal{M}_k \subset \mathcal{T}_k$ satisfy

$$\sum_{T \in \mathcal{M}_k} \eta_k^2(T) \geq (1 - \theta_*)^2 \sum_{T \in \mathcal{T}_k} \eta_k^2(T) \tag{1}$$

For optimality, $\mathcal{M}_k$ should be the smallest subset of $\mathcal{T}_k$ satisfying (1). Thus the elements with largest indicators should be chosen.

Marking Strategies

There are several popular marking strategies

Dörfler's Strategy, or Guaranteed Error Reduction Strategy (GERS)

Let $\mathcal{M}_k \subset \mathcal{T}_k$ satisfy

$$\sum_{T \in \mathcal{M}_k} \eta_k^2(T) \geq (1-\theta_*)^2 \sum_{T \in \mathcal{T}_k} \eta_k^2(T) \tag{1}$$

To avoid sorting, we do it by *layers*:

$\eta_{\texttt{max}} := \max_{T\in\mathcal{T}_k} \eta_k(T)$
$\gamma := 1$
`choose a parameter` $\nu > 0$ `small`
`sum := 0`
`while (` `sum <` $(1-\theta_*)^2 \sum_{T\in\mathcal{T}_k} \eta_k^2(T)$ `)`
  $\gamma = \gamma - \nu$
  $\mathcal{M}_k := \{T \in \mathcal{T}_k : \eta_k(T) > \gamma\, \eta_{\texttt{max}}\}$
  `sum :=` $\sum_{T\in\mathcal{M}_k} \eta_k^2(T)$
`end`

Implementation of Marking Strategies, file mark_elements.m

```
function mark_elements(adapt)
% Possible strategies are
% GR: global (uniform) refinement,
% MS: maximum strategy,
% GERS: guaranteed error reduction strategy (D\"orfler's)

global mesh

mesh.mark = zeros(mesh.n_elem, 1);

switch adapt.strategy
case 'GR'
  mesh.mark = adapt.n_refine * ones(size(mesh.estimator));
case 'MS'
  f = find(mesh.estimator > adapt.MS_gamma * mesh.max_est);
  mesh.mark( f ) = adapt.n_refine;
case 'GERS'
   [...]
end
```

Implementation of Marking Strategies, file `mark_elements.m`

```
function mark_elements(adapt)
[...]
global mesh

mesh.mark = zeros(mesh.n_elem, 1);

switch adapt.strategy
[...]
case 'GERS'
  est_sum2_marked = 0;
  threshold = (1 - adapt.GERS_theta_star)^2 * mesh.est_sum2;
  gamma = 1;
  while (est_sum2_marked < threshold)
    gamma = gamma - adapt.GERS_nu;
    f = find(mesh.estimator > gamma * mesh.max_est);
    mesh.mark(f) = adapt.n_refine;
    est_sum2_marked = sum((mesh.estimator(f)).^2);
  end
end
```

Outline

Code `afem.m`

In this code we implement the loop:

$$\textsc{Solve} \longrightarrow \textsc{Estimate} \longrightarrow \textsc{Mark} \longrightarrow \textsc{Refine}$$

Code afem.m

In this code we implement the loop:

$$\textsc{Solve} \longrightarrow \textsc{Estimate} \longrightarrow \textsc{Mark} \longrightarrow \textsc{Refine}$$

The file afem.m consists essentially of the following steps

```
% initialization
  read initial data and parameters --> init_data
  load the mesh from the folder set in domain
  perform global_refinements global refinements
```

Code afem.m

In this code we implement the loop:

$$\textsc{Solve} \longrightarrow \textsc{Estimate} \longrightarrow \textsc{Mark} \longrightarrow \textsc{Refine}$$

The file afem.m consists essentially of the following steps

```
% initialization
  read initial data and parameters --> init_data
  load the mesh from the folder set in domain
  perform global_refinements global refinements
% adaptive method
  repeat
    assemble system and solve--> assemble_and_solve
                % this is the old fixed_mesh/fem.m
    compute the estimators --> estimate(prob_data, adapt)
    if tolerance is achieved STOP
    otherwise mark elements for refinement --> mark_elements(adapt)
    refine the marked elements --> refine_elements
  end repeat
```

Code afem.m

In this code we implement the loop:

$$\textsc{Solve} \longrightarrow \textsc{Estimate} \longrightarrow \textsc{Mark} \longrightarrow \textsc{Refine}$$

The file `afem.m` consists essentially of the following steps

```
% initialization
  read initial data and parameters --> init_data
  load the mesh from the folder set in domain
  perform global_refinements global refinements
% adaptive method
  repeat
    assemble system and solve--> assemble_and_solve
                % this is the old fixed_mesh/fem.m
    compute the estimators --> estimate(prob_data, adapt)
    if tolerance is achieved STOP
    otherwise mark elements for refinement --> mark_elements(adapt)
    refine the marked elements --> refine_elements
  end repeat
```

Code `afem.m`

In this code we implement the loop:

SOLVE $\longrightarrow$ ESTIMATE $\longrightarrow$ MARK $\longrightarrow$ REFINE

The file `afem.m` consists essentially of the following steps

```
% initialization
  read initial data and parameters --> init_data
  load the mesh from the folder set in domain
  perform global_refinements global refinements
% adaptive method
  repeat
    assemble system and solve--> assemble_and_solve
              % this is the old fixed_mesh/fem.m
    compute the estimators --> estimate(prob_data, adapt)
    if tolerance is achieved STOP
    otherwise mark elements for refinement --> mark_elements(adapt)
    refine the marked elements --> refine_elements
  end repeat
```

Code afem.m

In this code we implement the loop:

$$\textsc{Solve} \longrightarrow \textsc{Estimate} \longrightarrow \textsc{Mark} \longrightarrow \textsc{Refine}$$

The file afem.m consists essentially of the following steps

```
% initialization
  read initial data and parameters --> init_data
  load the mesh from the folder set in domain
  perform global_refinements global refinements
% adaptive method
  repeat
    assemble system and solve--> assemble_and_solve
               % this is the old fixed_mesh/fem.m
    compute the estimators --> estimate(prob_data, adapt)
    if tolerance is achieved STOP
    otherwise mark elements for refinement --> mark_elements(adapt)
    refine the marked elements --> refine_elements
  end repeat
```

Code afem.m

In this code we implement the loop:

SOLVE $\longrightarrow$ ESTIMATE $\longrightarrow$ MARK $\longrightarrow$ REFINE

The file afem.m consists essentially of the following steps

```
% initialization
  read initial data and parameters --> init_data
  load the mesh from the folder set in domain
  perform global_refinements global refinements
% adaptive method
  repeat
    assemble system and solve--> assemble_and_solve
              % this is the old fixed_mesh/fem.m
    compute the estimators --> estimate(prob_data, adapt)
    if tolerance is achieved STOP
    otherwise mark elements for refinement --> mark_elements(adapt)
    refine the marked elements --> refine_elements
  end repeat
```