INTRODUCTION TO MESH ADAPTATION AND MULTILEVEL METHODS

LONG CHEN

1. Introduction to mesh adaptation

We start with a simple motivation in 1D for the use of adaptive procedures. Given $\Omega = (0,1)$, a grid $\mathcal{T}_N = \{x_i\}_{i=0}^N$ of Ω

$$0 = x_0 < x_1 < \cdots x_i < \cdots < x_N = 1$$

and a continuous function $u : \Omega \to \mathbb{R}$, we consider the problem of approximating u by a piecewise constant function u_N over $\mathcal{T}_N$. We measure the error in the maximum norm.

Suppose that u is Lipschitz in $[0,1]$. Consider the approximation

$$u_N(x) := u(x_{i-1}), \text{ for all } x_{i-1} \le x < x_i.$$

If the grid is quasi-uniform in the sense that $h_i = x_i - x_{i-1} \le C/N$ for $i = 1, \cdots N$, then it is easy to show that

$$\|u - u_N\|_\infty \le C N^{-1} \|u'\|_\infty \tag{1}$$

We can achieve the same convergent rate N^{-1} with less smoothness of the function. Suppose $\|u'\|_{L^1} \neq 0$. Let us define a grid distribution function

$$F(x) := \frac{1}{\|u'\|_{L^1}} \int_0^x |u'(t)|\, dt.$$

Then $F : [0,1] \to [0,1]$ is a non-decreasing function. Let $y_i = i/N, i = 0, \cdots, N$ be a uniform grid. We choose x_i such that $F(x_i) = y_i$. Then

$$\int_{x_{i-1}}^{x_i} |u'(t)|\, dt = F(y_i) - F(y_{i-1}) = N^{-1}, \tag{2}$$

and

$$|u(x) - u(x_{i-1})| \le \int_{x_{i-1}}^{x_i} |u'(t)|\, dt \le N^{-1} \|u'\|_{L^1},$$

which leads to the estimate

$$\|u - u_N\|_\infty \le C N^{-1} \|u'\|_{L^1}. \tag{3}$$

We use the following example to illustrate the advantage of (3) over (1). Let us consider the function $u(x) = x^r$ with $r \in (0,1)$. Then $u' \notin L^\infty(\Omega)$ but $u' \in L^1(\Omega)$. Therefore we cannot obtain optimal convergent rate on quasi-uniform grids while we could on the correctly adapted grid. For this simple, one can easily compute

$$x_i = (\frac{i}{N})^{1/r}, \quad \text{for all } 0 \le i \le N.$$

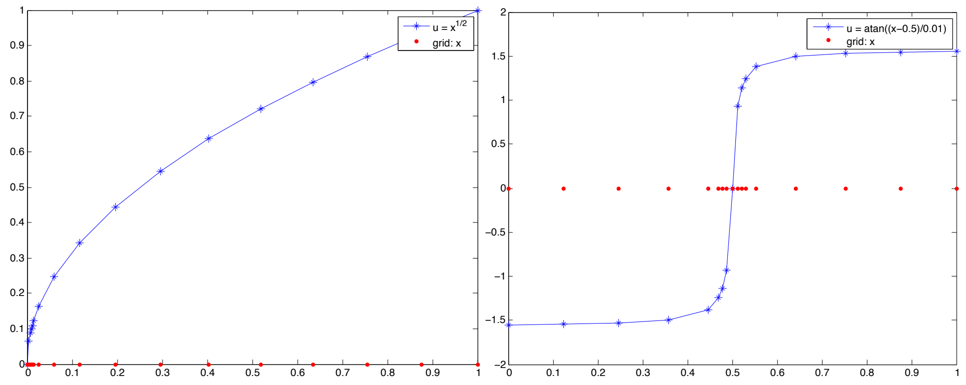
Estimate (3) will hold on the grid $\mathcal{T}_N = \{x_i\}_{i=0}^N$ which has higher density of grid points near the singularity of the u'.

In (2), we choose a grid such that a upper bound of the error is equidistributed. This is instrumental for adaptive finite element methods on solving PDEs.

A possible Matlab code is listed below.

```
function x = equidistribution(M,x)
h = diff(x);
F = [0; cumsum(h.*M)];
F = F/F(end);
y = (0:1/(length(x)-1):1)';
x = interp1(F,x,y);
```

Examples of adaptive grids for two functions with singularity are plotted below.



(a) Adaptive grid for $u(x) = x^(1/2)$

(b) An adaptive grid for $u(x) = \arctan((x - 0.5)/0.01)$

FIGURE 1. Adaptive grids for two functions with singularity.

Exercise 1.1. We consider piecewise linear approximation in this exercise.

(1) Let u_I be the nodal interpolation of u on a grid, i.e., u_I is piecewise linear and $u_I(x_i) = u(x_i)$ for all i. Prove that if $|u''(x)|$ is monotone decreasing in (x_{i-1}, x_i), then for $x \in (x_{i-1}, x_i)$

$$|(u - u_I)(x)| \le \left(\int_{x_{i-1}}^{x_i} |u''(s)|^{1/2} \, \mathrm{d}s \right)^2 .$$

(2) Give the condition on the grid and the function such that the following estimate holds and prove your result.

$$\|u - u_I\|_\infty \le C \|u\|_{1/2} N^{-2}.$$

When applied to numerical solution of PDEs, the function u and its derivatives are unknown. Only an approximated solution to the function u at grid points is available and a good approximation of derivatives or a upper bound of the error should be computed by a post-processing procedure. Another difficulty is the mesh requirement in two and higher dimensions. The mesh refinement or moving is much complicated in higher dimensions.

2. Introduction to Multilevel Methods: Hierarchical Basis

We compute an approximate solution of the Poisson equation

$$-u''(x) = f(x), x \in (0,1), \quad u(0) = u(1) = 0. \tag{4}$$

Let $N = 2^J$. We choose a uniform grid $\mathcal{T}_J = \{0 = x_0 < x_1 < \cdots < x_{N+1} = 1\}$ with $x_i = i/(N+1)$ for $i = 0, ..., N+1$ and approximate the solution by $u_{\mathcal{T}}$ in a continuous and piecewise linear function space $\mathcal{V}_{\mathcal{T}}$ satisfying boundary condition $u_{\mathcal{T}}(0) = u_{\mathcal{T}}(1) = 0$ such that

$$a(u_{\mathcal{T}}, v) = a(u, v) = (f, v), \qquad \text{for all } v \in \mathcal{V}_{\mathcal{T}},$$

where

$$a(u,v) = \int_0^1 u'v' \, \mathrm{d}x, \quad (f,v) = \int_0^1 fv \, \mathrm{d}x.$$

So $u_{\mathcal{T}}$ is the projection of u into the finite element space $\mathbb{V}_{\mathcal{T}}$ in the $a(\cdot,\cdot)$ inner product.

The nodal basis of $\mathbb{V}_{\mathcal{T}}$ is $\{\phi_i\}_{i=1}^N$ with $\phi_i \in \mathbb{V}_{\mathcal{T}}, \phi_i(x_j) = \delta_{i,j}$. Then $u_N = \sum_{i=1}^N u_i\phi_i$ with unknown coefficients u_i. Let

$$a_{ij} = a(\phi_j, \phi_i), \text{ and } f_i = (f, \phi_i).$$

Then using integration by part, we can compute the solution by solving the following linear algebraic equation

$$\begin{aligned} a_{11}u_1 + a_{12}u_2 + \cdots + a_{1N}u_N &= f_1, \\ a_{21}u_1 + a_{22}u_2 + \cdots + a_{2N}u_N &= f_2, \\ &\cdots\cdots\cdots\cdots\cdots \\ a_{N1}u_1 + a_{N2}u_2 + \cdots + a_{NN}u_N &= f_N, \end{aligned}$$

or in short $A\boldsymbol{u} = \boldsymbol{f}$ with obvious notation.

The matrix A is sparse, i.e., only $3N$ nonzero entries. But A^{-1} is a dense matrix. The computation of A^{-1} using Gauss elimination requires $\mathcal{O}(N^3)$ operations. We shall present an $\mathcal{O}(N)$ algorithm using multilevel methods.

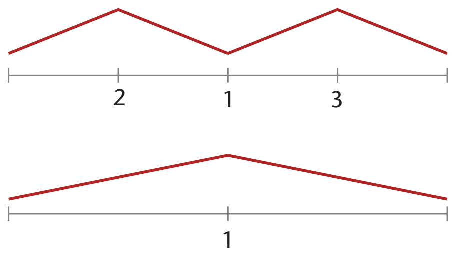


FIGURE 2. Hierarchical basis

Let $\mathcal{T}_k = \{0 = x_0 < x_1 < \cdots < x_{2^k+1} = 1\}$ be a uniform grid in the kth level. We re-order the vertices in $\mathcal{T}_J$ such that $(2^k, 2^{k+1}-1) \in \mathcal{T}_k$ for $k = 1, \cdots J$; See Figure 2. We can easily verify that $a(\phi_1^1, \phi_2^2) = a(\phi_1^1, \phi_3^2) = a(\phi_2^2, \phi_3^2) = 0$ and in general the HB basis

$$\{\bar{\phi}_1, \cdots, \bar{\phi}_N\} := \{\phi_1^1, \cdots \phi_{2^k}^k, \cdots \phi_{2^{k+1}-1}^k, \cdots\}$$

form an A-orthogonal basis of $\mathbb{V}_{\mathcal{T}}$. If we expand $u_N = \sum_{i=1}^{J} \bar{u}_i \bar{\phi}_i$ using the A-orthogonal basis, the coefficients $\bar{u}_i$ can be computed as

$$\bar{u}_i = \frac{a(u_N, \bar{\phi}_i)}{a(\bar{\phi}_i, \bar{\phi}_i)} = \frac{a(u, \bar{\phi}_i)}{a(\bar{\phi}_i, \bar{\phi}_i)} = \frac{(f, \bar{\phi}_i)}{a(\bar{\phi}_i, \bar{\phi}_i)}, i = 1, \cdots N.$$

The computation of $a(\bar{\phi}_i, \bar{\phi}_i)$ is easy. Suppose $\bar{\phi}_i = \phi_j^k$. Then $a(\bar{\phi}_i, \bar{\phi}_i) = 1/h_k = 2^k$. The computation of $(f, \bar{\phi}_i)$ is subtle. For example, the evaluation of $(f, \bar{\phi}_1) = (f, \phi_1^1)$ requires $\mathcal{O}(N)$ operations! Each fine grid point will be used repeatedly in each level since we have to go through all grid points to compute an accurate approximation of the integral. Therefore, the naive calculation of $(f, \bar{\phi}_i)$ for $i = 1, \cdots, N$, is an $\mathcal{O}(N \log N)$ algorithm.

If we compute (f, ϕ_i) in the nodal basis, every grid point is used only 3 times and thus can be finished in $\mathcal{O}(N)$ operations. We then use the relation between fine level and coarse level to compute $(f, \bar{\phi}_i)$. As an example, we look at an example between two levels. Suppose $(f, \phi_i^2), i = 1, 2, 3$ are available. Using the relation

$$\phi_1^1 = \phi_1^2 + 0.5\phi_2^2 + 0.5\phi_3^2,$$

we can compute

$$(f, \phi_1^1) = (f, \phi_1^2) + 0.5(f, \phi_2^2) + 0.5(f, \phi_3^2).$$

In general, we can compute $\bar{\boldsymbol{f}}$ from $\boldsymbol{f}$ by the following $\mathcal{O}(N)$ algorithm. It can be thought as a basis transformation from the nodal basis to the hierarchical basis.

Before we present the subroutine, let us discuss in detail the data structure. The node is now indexed in a multilevel way. We introduce a pointer `s` to record the ending location of each level. Therefore `s(k-1)+1:s(k)` will be nodes in k-th level only. For each node `i`, we denote by `l(i)` and `r(i)` the left and right neighbor of `i`. In the natural lexicographical ordering `l(i)=i-1` but not the case in the hierarchical ordering.

```
function fbar = NB2HB(f)
for k = J-1:1
    for i = s(k-1)+1: s(k)
        f(i) = f(i) + 0.5*f(l(i)) + 0.5*f(r(i));
    end
end
fbar = f;
```

The coefficients vector $\bar{\boldsymbol{u}}$ is just a scaling of $\bar{\boldsymbol{f}}$.

```
function ubar = scaling(fbar)
for k = 1:J
    for i = s(k-1)+1: s(k)
        ubar(i) = 2^k*fbar(i);
    end
end
```

After we get $\bar{\boldsymbol{u}}$, we can compute $\boldsymbol{u}$ by the transformation from hierarchical basis to nodal basis. This time we scan from the coarse to fine. The hierarchical basis in the coarse grid will contribute to the function value at fine grid points. Using data structure discussed before, it can be implemented by the following subroutine.

```
function u = HB2NB(ubar)
u = ubar;
for k = 2:J
    for i = s(k-1)+1:s(k)
```

```
        u(l(i)) = u(l(i)) + 0.5*u(i);
        u(r(i)) = u(r(i)) + 0.5*u(i);
    end
end
```

Exercise 2.1. Figure out the formulae for `s(k)`, `l(i)` and `r(i)` in the algorithms and write a program using the hierarchical basis to solve the Poisson equation on $(0,1)$ with Dirichlet or Neumann boundary condition.