T U T O R I A L

"Numerical Methods for Solving Partial Differential Equations"

for the Lectures on NuPDE

T VIII Monday, 7 January 2008 (Time: 08:30 – 10:00, Room: T 212)

1.10 Optimal Preconditioning (MDS)

In this tutorial we consider the MDS (multilevel diagonal scaling) preconditioner. The exercises are not as complicated as they appear at first.

Let $\mathcal{T}_l$ be a subdivision of the interval $\Omega = (0,\, 1)$ given by the nodes

$$0 = x_{l,0} \le x_{l,1} \le \cdots \le x_{l,N_l} = 1\,.$$

For a fixed mesh $\mathcal{T}_l$ we derive a new subdivision $\mathcal{T}_{l+1}$ by refinement: We introduce $N_{l+1} = 2\,N_l - 1$ nodes given by

$$x_{l+1,2i} = x_{l,i}\,, \quad x_{l+1,2i+1} = \frac{1}{2}\big(x_{l,i} + x_{l,i+1}\big)\,.$$

We denote the corresponding FEM spaces (using linear Courant elements) by V_l and V_{l+1}. Obviously, we have $V_l \subset V_{l+1}$, and the following relation between the nodal basis functions holds:

$$\varphi_{l,i}(x) = \frac{1}{2}\,\varphi_{l+1,2i-1}(x) + \varphi_{l+1,2i}(x) + \frac{1}{2}\,\varphi_{l+1,2i+1}(x)\,, \quad 0 \le i \le N_l\,.$$

In the following we fix a uniform mesh $\mathcal{T}_1$ and a fixed number of levels $L \ge 1$, and we define $\mathcal{T}_2, \ldots, \mathcal{T}_L$ recursively by refinement as above. The mesh $\mathcal{T}_1$ is called *coarsest grid*, whereas $\mathcal{T}_L$ is the *finest grid*. When comparing two subsequent grids $\mathcal{T}_l$, $\mathcal{T}_{l+1}$, these are called *coarse* and *fine grid*, respectively.

44 Let w_l be a finite element function on the coarse grid $\mathcal{T}_l$:

$$w_l(x) = \sum_{i=0}^{N_l} w_{l,i}\,\varphi_{l,i}(x)\,.$$

Find a representation of w_l using the basis functions $\{\varphi_{l+1,i}\}_i$ associated to the fine grid $\mathcal{T}_{l+1}$ such that

$$w_l(x) = \sum_{i=0}^{N_{l+1}} w_{l+1,i}\,\varphi_{l+1,i}(x)\,.$$

Represent the relation between the coefficient vectors $\underline{w}_{l+1} = (w_{l+1,i})_{i=0,\ldots,N_{l+1}}$ and $\underline{w}_l = (w_{l,i})_{i=0,\ldots,N_l}$ by an $N_{l+1} \times N_l$ matrix I_l^{l+1} such that

$$\underline{w}_{l+1} = I_l^{l+1}\,\underline{w}_l\,.$$

45 Let $R : H^1(0,1) \to \mathbb{R}$ be a continuous linear functional. Furthermore, for the coefficient vector $\underline{r}_{l+1} = (r_{l+1,i})_{i=0,\dots,N_{l+1}}$ defined by the relation

$$r_{l+1,i} := \langle R, \varphi_{l+1,i} \rangle ,$$

we have

$$\langle R, v_{l+1} \rangle = \sum_{i=0}^{N_{l+1}} r_{l+1,i}\, v_{l+1,i} = (\underline{r}_{l+1}, \underline{v}_{l+1})_{\ell_2} .$$

Find a representation of the evaluation of this functional for a finite element function v_l defined on the coarse grid $\mathcal{T}_l$:

$$\langle R, v_l \rangle = \sum_{i=0}^{N_l} r_{l,i}\, v_{l,i} = (\underline{r}_l, \underline{v}_l)_{\ell_2} .$$

Show the following relation between the coefficient vectors $\underline{r}_{l+1} = (r_{l+1,i})_{i=0,\dots,N_{l+1}}$ and $r_l = (r_{l,i})_{i=0,\dots,N_l}$:

$$\underline{r}_l = I_{l+1}^l\, \underline{r}_{l+1} , \qquad \text{with } I_{l+1}^l = (I_l^{l+1})^\top .$$

Hint:

$$(\underline{r}_l, v_l)_{\ell_2} = \langle R, v_l \rangle = (\underline{r}_{l+1}, \underline{v}_{l+1})_{\ell_2} = (\underline{r}_{l+1}, I_l^{l+1} \underline{v}_l)_{\ell_2} .$$

46* Let K_l and K_{l+1} be the stiffness matrices on the grid $\mathcal{T}_l$ and $\mathcal{T}_{l+1}$, respectively. Show the relation

$$K_l = I_{l+1}^l\, K_{l+1}\, I_l^{l+1} = (I_l^{l+1})^\top\, K_{l+1}\, I_l^{l+1} .$$

47 Write a function `RefineUniform(↓coarsemesh, ↑finemesh)`, which computes the refined grid $\mathcal{T}_{l+1}$ (`finemesh`) starting from the coarse mesh $\mathcal{T}_l$ (`coarsemesh`) as described above.

48 Write a function `Prolongate(↓coarsevector, ↑finevector)` which computes $\underline{w}_{l+1} = I_l^{l+1}\, w_l$, where `coarsevector`=$\underline{w}_l$ and `finevector`=$\underline{w}_{l+1}$.

Write a function `Restrict(↓finevector, ↑coarsevector)` which computes $\underline{w}_l = I_{l+1}^l\, \underline{w}_{l+1}$, where `finevector`=$\underline{w}_{l+1}$ and `coarsevector`=$\underline{w}_l$.

49 Implement the MDS preconditioner C_{MDS}^{-1} for a hierarchy of recursively refined grids $\mathcal{T}_1, \dots, \mathcal{T}_L$, i. e., implement the operation

$$\underline{w}_L = C_{\mathrm{MDS}}^{-1}\, \underline{r}_L .$$

For $l = 1, \dots, L$ we define $D_l = \operatorname{diag}(K_l)$.

1. If there is only one grid ($L = 1$), then the MDS preconditioner coincides with the Jacobi preconditioner, i. e.,

$$\underline{w}_1 = D_1^{-1}\, \underline{r}_1 .$$

2. For two grids $L = 2$, the correction $\underline{w}_2$ is defined as the sum of
   - the correction obtained by the Jacobi preconditioner applied to the residual $\underline{r}_2$ on the fine grid,
   - and the (prolongated) correction obtained by the Jacobi preconditioner on the coarse grid for the (restricted) residual $\underline{r}_1$, i. e.,

$$\underline{w}_2 = D_2^{-1}\,\underline{r}_2 + I_1^2\,\underline{w}_1\,,$$

with

$$\underline{w}_1 = D_1^{-1}\,\underline{r}_1, \qquad \text{where } \underline{r}_1 = I_2^1\,\underline{r}_2\,.$$

3. For a hierarchy of grids $\mathcal{T}_1, \ldots, \mathcal{T}_L$ we have the recursive definition:

$$\begin{aligned} C_{\mathrm{MDS},1}^{-1} &= D_1^{-1}\,, \\ C_{\mathrm{MDS},l}^{-1} &= D_l^{-1} + I_{l-1}^{l}\,C_{\mathrm{MDS},l-1}\,I_l^{l-1}\,. \end{aligned}$$

and $C_{\mathrm{MDS}} := C_{\mathrm{MDS},L}$.

Hint: The operation $\underline{w}_l = C_{\mathrm{MDS},l}^{-1}\,\underline{r}_l$ for $l > 1$ is equivalent to

$$\begin{aligned} \underline{r}_{l-1} &= I_l^{l-1}\,\underline{r}_l\,, \\ \underline{w}_{l-1} &= C_{\mathrm{MDS},l-1}^{-1}\,\underline{r}_{l-1}\,, \\ \underline{w}_l &= I_{l-1}^{l}\,\underline{w}_{l-1}\,. \end{aligned}$$

Hence, use a recursive function like

```
void MDS (int l, const Vector& r, Vector& w)
{
  . . .
  if (l == 1)
    {
       w = JacobiPreconditioner.solve (l, r);
    }
  else
    {
       w = JacobiPreconditioner.solve (l, r);
       Restrict (r, r_coarse);
       MDS (l-1, r_coarse, w_coarse);
       Prolongate (w_coarse, w_fine);
       w += w_fine;
    }
}
```

50 Embed your MDS code in a class `MDSPreconditioner` where you have a member function `solve` similar to the one in your existing Jacobi preconditioner class, e. g., `Vector MDSPreconditioner::solve (const Vector& r)`.

Consider a FEM discretized boundary value problem of your choice and solve it using the preconditioned CG method using the MDS preconditioner.