T U T O R I A L

"Numerical Methods for Solving Partial Differential Equations"

to the Lectures on NuPDE

T VIII Thursday, 14 December 2006 (Time: 8:30 - 10:00, Room: HS 13)

1.8 FEM for BVPs of Second-order ODEs

37 Write a function `CG(↓A,↕x,↓b,↓C,↕max_iter,↕tol)` to solve the linear system $Ax = b$ by the preconditioned conjugate gradient (PCG) method with stopping rule

$$\|r^{(n)}\|_{\ell_2} \leq \varepsilon \|b\|_{\ell_2}.$$

The arguments of the function are to be understood as follows: `A`= A, `x`= $x^{(0)}$ as input and `x`= $x^{(n)}$ as output, `b`= b, `C` is the preconditioner C, `max_iter` is the maximal number of iterations as input while, as output, it is equal to the number of iterations n needed to satisfy the stopping criterion, and `tol`= ε.

Use the template `cg.h`.

We consider now the MDS (Multilevel Diagonal Scaling) preconditioner.
Let $\mathcal{T}_l$ be a subdivision of the interval $\Omega = (0, 1)$ given by the nodes:

$$0 = x_{l,0} < x_{l,1} < \ldots < x_{l,N_l} = 1\,.$$

Let $\mathcal{T}_{l+1}$ be a refined subdivision of Ω obtaining by halving the elements of $\mathcal{T}_l$. Considering a numbering of the nodes from left to right, the nodes $x_{l+1,i}$ of $\mathcal{T}_{l+1}$ are

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

Thus, we can derive the following relation between the nodal basis functions of the Courant element with respect to the grids $\mathcal{T}_l$ and $\mathcal{T}_{l+1}$:

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

38 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 associated to the fine grid $\mathcal{T}_{l+1}$, i.e., compute the coefficients $w_{l+1,i}$ 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 $\underline{w}_l = \{w_{l,i}\}_{i=0,\ldots,N_l}$ in the form

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

for an appropriate $N_{l+1} \times N_l$ matrix I_l^{l+1}.

39 Let $R : H^1(0,1) \to \mathbb{R}$ be a continuous linear functional, such that, given a finite element function v_{l+1} on the fine grid $\mathcal{T}_{l+1}$,

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

where $\underline{r}_{l+1} = \{r_{l+1,i}\}_{i=0,\ldots,N_{l+1}}$ and $r_{l+1,i} = \langle R, \varphi_{l+1,i} \rangle$. 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 $\underline{r}_{l+1} = \{r_{l+1,i}\}_{i=0,\ldots,N_{l+1}}$ and $\underline{r}_l = \{r_{l,i}\}_{i=0,\ldots,N_l}$:

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

Hint:

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

40 Let K_l and K_{l+1} be the stiffness matrices on the grid $\mathcal{T}_l$ and $\mathcal{T}_{l+1}$, respectively. Show that there holds:

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

Hint: Given two finite element functions w_h and v_h, on the coarse grid $\mathcal{T}_l$ we have

$$a(w_h, v_h) = (K_l \underline{w}_l, \underline{v}_l)_{\ell_2} \,,$$

while on the fine grid $\mathcal{T}_{l+1}$ we have

$$a(w_h, v_h) = (K_{l+1} \underline{w}_{l+1}, \underline{v}_{l+1})_{\ell_2} \,.$$

41 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.

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

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

43 Implement the MDS preconditioner for a hierarchy of grids $\mathcal{T}_1, \ldots, \mathcal{T}_L$:

1. if we have only one grid $\mathcal{T}_1$ ($L = 1$), then the MDS preconditioner coincides with the Jacobi preconditioner

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

2. for a hierarchy of two grids $\mathcal{T}_1$, $\mathcal{T}_2$ ($L = 2$), the correction $\underline{w}_2$ obtained by MDS for a given residual $\underline{r}_2$ is given by the sum of the correction obtained by the Jacobi preconditioner on the fine grid for the residual $\underline{r}_2$, and the (prolongated) correction obtained by the Jacobi preconditioner on the coarse grid for the (restricted) residual $\underline{r}_1$:

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

with

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

3. for a hierarchy of L grids $\mathcal{T}_1, \ldots, \mathcal{T}_L$, the correction $\underline{w}_L$ obtained by MDS for a given residual $\underline{r}_L$ is given by the sum of the correction obtained by the Jacobi preconditioner on the fine grid $\mathcal{T}_L$ for the residual $\underline{r}_L$, and the (prolongated) correction obtained by the Jacobi preconditioner on the coarse grid $\mathcal{T}_{L-1}$ for the (restricted) residual $\underline{r}_{L-1}$:

$$\underline{w}_L = C_L^{-1}\underline{r}_L = D_L^{-1}\underline{r}_L + I_{L-1}^L\underline{w}_L\,,$$

with

$$\underline{w}_{L-1} = C_{L-1}^{-1}\underline{r}_{L-1} \quad \text{for } \underline{r}_{L-1} = I_L^{L-1}\underline{r}_L\,.$$

Hint: Use a recursive function like

```
MDS(l,r,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;
   }
  }
```

```
//*****************************************************************
// Iterative template routine -- CG
// CG solves the symmetric positive definite linear
// system Ax=b using the Conjugate Gradient method.
// CG follows the algorithm described on p. 15 in the
// SIAM Templates book.
// The return value indicates convergence within max_iter (input)
// iterations (0), or no convergence within max_iter iterations (1).
// Upon successful return, output arguments have the following values:
//        x  --  approximate solution to Ax = b
// max_iter  --  the number of iterations performed before the
//               tolerance was reached
//      tol  --  the residual after the final iteration
//*****************************************************************

template < class Matrix, class Vector, class Preconditioner, class Real >
int
CG(const Matrix &A, Vector &x, const Vector &b,
   const Preconditioner &M, int &max_iter, Real &tol)
{
  Real resid;
  Vector p, z, q;
  Vector alpha(1), beta(1), rho(1), rho_1(1);

  Real normb = norm(b);
  Vector r = b - A*x;

  if (normb == 0.0)
    normb = 1;

  if ((resid = norm(r) / normb) <= tol) {
    tol = resid;
    max_iter = 0;
    return 0;
  }

  for (int i = 1; i <= max_iter; i++) {
    z = M.solve(r);
    rho(0) = dot(r, z);

    if (i == 1)
      p = z;
    else {
      beta(0) = rho(0) / rho_1(0);
      p = z + beta(0) * p;
    }
```

```
    q = A*p;
    alpha(0) = rho(0) / dot(p, q);

    x += alpha(0) * p;
    r -= alpha(0) * q;

    if ((resid = norm(r) / normb) <= tol) {
      tol = resid;
      max_iter = i;
      return 0;
    }

    rho_1(0) = rho(0);
  }

  tol = resid;
  return 1;
}
```