T U T O R I A L

"Numerical Methods for Solving Partial Differential Equations"

to the Lectures on NuPDE

T V Monday, 26 November 2007 (Time: 08:30 – 10:00, Room: T 212)

1.5 FEM for BVPs for second-order ODEs (continued)

25 Write a function `Mult(↓matrix, ↓vector, ↑result)` which computes the matrix-vector product `result`$=K\,u$ of a given tridiagonal matrix K (`matrix`), implemented by the data type `Matrix` (see Exercise 16 in Tutorial III), and of a given vector `vector`$=u$.

If you like, you can then use C^{++}'s operator overloading to allow `x = A * y;`

```
inline Vector operator* (const Matrix& mat, const Vector& vec) {
  Vector res(vec.size ());
  Mult (mat, vec, res);
  return res;
}
```

26 Define a C^{++} class `Preconditioner` which implements the Jacobi Preconditioner $C_h = D_h = \mathrm{diag}\,(K_h)$.

27 Write a function (or a member function of the class `Preconditioner` which solves the linear system

$$C_h\,\underline{w}_h = \underline{r}_h$$

for $C_h = D_h$ (diagonal) and for a given vector $\underline{r}_h$.

28 Write a function `Richardson(↓A, ↕x, ↓b, ↓C, ↕max_iter, ↕tol)` to solve the linear system

$$A\,\underline{x} = \underline{b}$$

by the preconditioned Richardson method:

$$\underline{x}^{(n+1)} = \underline{x}^{(n)} + C^{-1}(\underline{b} - A\,\underline{x}^{(n)})$$

with the stopping criterion

$$\|\underline{r}^{(n)}\|_{\ell_2} = \|\underline{b} - A\,\underline{x}^{(n)}\|_{\ell_2} \le \varepsilon \|\underline{b}\|_{\ell_2}\,,$$

where `A`$=A$, `x`$=\underline{x}^{(0)}$ in input and `x`$=\underline{x}^{(n)}$ in output, `b`$=\underline{b}$, `C`$=C$ and `tol`$=\varepsilon$. In input, `maxiter` is the maximal number of iterations. In output, `maxiter`$=n$ returns the number of iterations needed to satisfy the stopping criterion.

Hint: use the template `Richardson.hpp` *and rewrite it for your own purposes, or use, e. g.,* `std::valarray<double>` *(*`#include <valarray>`*) as a vector class.*

29 Use your program to discretize the following boundary value problem:

Find a function $u(x)$ satisfying

$$\begin{aligned} -u''(x) &= f(x) && x \in \Omega \\ u(x) &= g_D(x) && x \in \Gamma_D \\ \frac{\partial u}{\partial n}(x) &= 0 && x \in \Gamma_N \end{aligned}$$

with the data $f(x) = 8$, $\Omega = (0,\, 1)$, $\Gamma_D = \{0\}$, $g_D(x) = -1$, $\Gamma_N = \{1\}$. Then solve the discretized problem

$$K_h \underline{u}_h = \underline{f}_h$$

by the preconditioned Richardson method with the Jacobi preconditioner $C_h = D_h = \operatorname{diag}(K_h)$.

Sharpness of the condition number bound

We consider the boundary value problem

$$\begin{aligned} u''(x) &= f(x) && x \in (0,\, 1)\,, \\ u(0) &= 0\,, \quad u(1) = 1 \end{aligned}$$

Let K_h denote the $(N_h - 1)$-dimensional stiffness matrix obtained by the finite element method using the Courant elements on a uniform mesh $0 = x_0 < x_1 < \ldots x_{N_h} = 1$, with $h_k = x_k - x_{k-1} = h$ for all $k = 1,\, \ldots,\, N_h$.

In the following two exercises we show that the condition number estimate

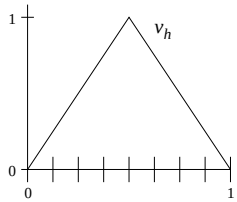
$$\kappa(K_h) = \mathcal{O}(h^{-2})$$

given in the lecture is *sharp*. Furthermore we assume that N_h is **even**.

30* Show that for a special choice of $\underline{v}_h \in \mathbb{R}^{N_h}$,

$$\lambda_{\min}(K_h) = \min_{\underline{v}_h} \frac{(K_h\, \underline{v}_h,\, \underline{v}_h)}{(\underline{v}_h,\, \underline{v}_h)} \le C\, h\,.$$

Hint: Choose $\underline{v}_h$ corresponding to v_h:



31 Show that for a special choice of $\underline{v}_h \in \mathbb{R}^{N_h}$,

$$\lambda_{\max}(K_h) = \max_{\underline{v}_h} \frac{(K_h\, \underline{v}_h,\, \underline{v}_h)}{(\underline{v}_h,\, \underline{v}_h)} \ge c\, h^{-1}\,.$$

Hint: Choose $\underline{v}_h = (1, -1, 1, -1, \ldots 1)$ corresponding to a zig-zag function.

```
// file richardson.hpp

#ifndef __RICHARDSON_H
#define __RICHARDSON_H

// Iterative template routine -- preconditioned Richardson
//
// RICHARDSON solves the linear system Ax=b using
// the preconditioned richardson iteration.
// The returned value indicates convergence within
// max_iter iterations (return value 0)
// or no convergence within max_iter iterations (return value 1)
// Upon successful return (0), the output arguments have the
// following values:
//        x: computed solution
// mat_iter: number of iterations to satisfy the stopping criterion
//      tol: residual after the final iteration

template <class MATRIX, class VECTOR, class PRECONDITIONER, class REAL>
int
RICHARDSON (const MATRIX & A, VECTOR & x, const VECTOR & b,
    const PRECONDITIONER & M, int & max_iter, REAL & tol)
{
  REAL resid;
  VECTOR z(b.size ());
  REAL normb = norm (b);
  VECTOR r = b - A * x;

  if (normb == 0.0) normb = 1;
  resid = norm (r) / normb;

  if (resid <= tol)
    {
      tol = resid;
      max_iter = 0;
      return 0;
    }

  for (int i=1; i<max_iter; i++)
    {
      z = M.solve (r);
      x += z;
      r = b - A * x;
      resid = norm(r) / normb;

      if (resid <= tol)
        {
          tol = resid;
          max_iter = i;
          return 0;
        }
    }

  tol = resid;
  return 1;
}

#endif // __RICHARDSON_H
```