ÜBUNGEN ZU

NUMERIK PARTIELLER DIFFERENTIALGLEICHUNGEN

für den 23. 11. 2005

Send your programs to `zulehner@numa.uni-linz.ac.at` by 9 a.m.

---

25. Write a function `IR(↓A,↕x,↓b,↓C,↓tau,↕max_iter,↕tol)` that approximately solves the linear system
$$Ax = b$$
by the preconditioned Richardson method
$$x^{(n+1)} = x^{(n)} + \tau\, C^{-1}\left(b - Ax^{(n)}\right)$$
with stopping rule
$$\|r^{(n)}\|_{\ell_2} \le \varepsilon\, \|b\|_{\ell_2}.$$
where `A` $= A$, `x` $= x^{(0)}$ as input, `x` $= x^{(n)}$ as output, `b` $= b$, `C` $= C$, `tau` $= \tau$, `max_iter` = maximal number of iterations as input, `max_iter` $= n$ (the number of iterations needed to satisfy the stopping rule) as output, `tol` $= \varepsilon$.

   Hint: Use the template `ir.h`.

Test your function for a simple example.

26. Use your functions to discretize the following one-dimensional boundary value problem

   Find a function $u(x)$ such that
$$\begin{aligned} -u''(x) &= f(x) \quad x \in \Omega, \\ u(x) &= g_D(x) \quad x \in \Gamma_D, \\ \frac{\partial u}{\partial n}(x) &= g_N(x) \quad x \in \Gamma_N. \end{aligned}$$
with the data
$$f(x) = 8,\ \Omega = (0,1),\ \Gamma_D = \{0\},\ g_D(x) = -1,\ \Gamma_N = \{1\},\ g_N(x) = -4.$$
Then solve the discretized problem
$$K_h \underline{u}_h = \underline{f}_h$$
by the preconditioned Richardson method with Jacobi preconditioner $C_h = D_h = \operatorname{diag}(K_h)$.

```
//*****************************************************************
// Iterative template routine -- Preconditioned Richardson
//
// IR solves the unsymmetric linear system Ax = b using
// Iterative Refinement (preconditioned Richardson iteration).
//
// 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
IR(const Matrix &A, Vector &x, const Vector &b,
   const Preconditioner &M, int &max_iter, Real &tol)
{
  Real resid;
  Vector z;

  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);
    x += z;
    r = b - A * x;

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

  tol = resid;
  return 1;
}
```