ÜBUNGEN ZU

NUMERIK PARTIELLER DIFFERENTIALGLEICHUNGEN

für den 7. 12. 2005

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

---

33. Write a function `CG(↓A,↕x,↓b,↓C,↕max_iter,↕tol)` that approximately solves the linear system

$$Ax = b$$

by the preconditioned conjugate gradient method with stopping rule

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

where `A` $= A$, `x` $= x^{(0)}$ as input, `x` $= x^{(n)}$ as output, `b` $= b$, `C` $= C$, `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 `cg.h`.

34. Write a function `G(↓A,↕x,↓b,↓C,↕max_iter,↕tol)` that approximately solves the linear system

$$Ax = b$$

by the preconditioned gradient method with stopping rule

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

where `A` $= A$, `x` $= x^{(0)}$ as input, `x` $= x^{(n)}$ as output, `b` $= b$, `C` $= C$, `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 `cg.h` with `beta(0)` $= 0$.

Test your functions for a simple example.

35. 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 gradient method and preconditioned conjugate gradient method with Jacobi preconditioner $C_h = D_h = \operatorname{diag}(K_h)$.

(a) How does the number of iterations n depend on the step size h and on the number of N_h of unknowns, respectively?

(b) How does the cpu time t depend on the step size h and on the number of N_h of unknowns, respectively?

```
//*****************************************************************
// 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;
}
```