TMA4220
Numerical solution of partial differential equations
by element methods

Suggested solutions to Problem Set 7[1]

[1]Made by Einar Rønquist, Department of Mathematical Sciences, NTNU, N-7491 Trondheim, Norway.

Exercise 11

(February 17, 2003)

Innerproducts:

$$2 \quad \begin{cases} \underline{r}_{k-1}^T \underline{r}_{k-1} \\ \underline{p}_k^T (\underline{A}\underline{p}_k) \qquad (\underline{w} \equiv \underline{A}\underline{p}_k) \end{cases}$$

Multiplication of a vector with a scalar

$$3 \quad \begin{cases} \beta_k \underline{p}_{k-1} \\ \alpha_k \underline{p}_k \\ \alpha_k (\underline{A}\underline{p}_k) \qquad (\underline{w} \equiv \underline{A}\underline{p}_k) \end{cases}$$

Addition of two vectors:

$$3 \quad \begin{cases} (\underline{r}_{k-1}) + (\beta_k \underline{p}_{k-1}) \\ (\underline{u}_{k-1}) + (\alpha_k \underline{p}_k) \\ (\underline{r}_{k-1}) + (-\alpha_k \underline{A}\underline{p}_k) \end{cases}$$

Exercise 12

(February 17, 2003)

Since we use a one-point Gauss-Legendre formula to evaluate $l(v)$, we expect the quadrature error to be $\mathcal{O}(h^2)$. From the last column of Table 1:

$$\frac{1.24 \cdot 10^{-4}}{3.10 \cdot 10^{-5}} = 4.0$$
$$\frac{3.10 \cdot 10^{-5}}{7.74 \cdot 10^{-6}} = 4.0$$
$$\text{etc.}$$

$\Rightarrow$ error $\sim \mathcal{O}(h^2)$.

Exercise 13

(February 17, 2003)

Since $(\underline{r}^T \underline{r})^{1/2} < 10^{-10}$, we expect the incomplete iteration error to be much smaller than the quadrature errors. Even the quadrature errors are much smaller than the discretization error $|u - u_h|_{H'(\Omega)}$. The current tolerance is probably too small, something which makes the iterative solution process more costly than it needs to be.

For many problems, it suffices to balance the incomplete iteration error with the discretization error. However, it is not trivial to obtain good estimates for these.

Exercise 14

(February 17, 2003)

In the CG-algorithm, we only need to store 4 vectors, each of length N. Hence, $M_{CG} \simeq 4N$. Note that this estimate assumes that we do *not* explicitly contruct the stiffness matrix, but only implement a routine to compute the *action* of the stiffnesss matrix upon a vector. This can be realized pratically by computing the matrix-vector products locally in each element, and then assemble all the local contributions into the global vector.

If we use a banded, direct solver, the storage requirement is $\mathcal{O}(bN)$. Note that, in this case, we explicitly construct the stiffness matrix and store it in banded form (i.e., we only store the matrix elements inside a band around the main diagonal).

A comparison of memory requirement in $\underline{\mathbb{R}}^d$ then gives:

d	M_{CG}	M_{LU}
1	$\mathcal{O}(N)$	$\mathcal{O}(N)$
2	$\mathcal{O}(N)$	$\mathcal{O}(N_1 \cdot N)$
3	$\mathcal{O}(N)$	$\mathcal{O}(N_1^2 \cdot N)$

$\uparrow$
scalable