ÜBUNGEN ZU

NUMERIK PARTIELLER DIFFERENTIALGLEICHUNGEN

für den 9. 11. 2005

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

A first task of the programming exercises is the developement of a program for solving the following one-dimensional boundary value problem:

Let $\Omega = (0,1)$, $\Gamma = \partial\Omega = \{0,1\} = \Gamma_D \cup \Gamma_N$ with $\Gamma_D \cap \Gamma_N = \emptyset$.

Find a function $u(x)$ such that

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

This problem is discretized by the finite element method with the Courant element: The nodes

$$0 = x_0 < \ldots < x_{N_h} = 1$$

define a subdivision $\mathcal{T}_h$ of $[0,1]$ with sub-intervals $T_k = (x_{k-1}, x_k)$ for $k = 1, \ldots, N_h$. The nodal basis is given by $\{\varphi_i : i = 0, 1, \ldots, N_h\}$ with $\varphi_i \in V_h = \{v \in C(\overline{\Omega}) : v|_T \in P_1 \text{ for all } T \in \mathcal{T}_h\}$ and $\varphi_i(x_j) = \delta_{ij}$.

In the following exercises input parameters of functions are denoted by '$\downarrow$', output parameters by '$\uparrow$' and input/output parameters by '$\updownarrow$'.

13. Write a function `ElementStiffnessMatrix(↓xa,↓xb,↑element_matrix)` which, for `xa` $= x_{k-1}$ and `xb` $= x_k$ returns the 2-by-2 element stiffness matrix `element_matrix` $= K_h^{(k)}$ for the element T_k, given by

$$K_h^{(k)} = \begin{pmatrix} \displaystyle\int_{T_k} \varphi_{k-1}'(x)^2 \, dx & \displaystyle\int_{T_k} \varphi_{k-1}'(x)\varphi_k'(x) \, dx \\ \displaystyle\int_{T_k} \varphi_k'(x)\varphi_{k-1}'(x) \, dx & \displaystyle\int_{T_k} \varphi_k'(x)^2 \, dx \end{pmatrix}.$$

14. Write a function `ElementLoadVector(↓(*f)(x),↓xa,↓xb,↑element_vector)` which, for `xa` $= x_{k-1}$ and `xb` $= x_k$, returns the 2-dimensional element load vector `element_vector` $= \underline{f}_h^{(k)}$ for the element T_k, given by

$$\underline{f}_h^{(k)} = \begin{pmatrix} \displaystyle\int_{T_k} f(x)\varphi_{k-1}(x) \, dx \\ \displaystyle\int_{T_k} f(x)\varphi_k(x) \, dx \end{pmatrix}.$$

Use the trapezoidal rule

$$\int_a^b g(x)\, dx \approx \frac{b-a}{2}\,[g(a)+g(b)]$$

for approximating the integrals.

15. Define a data type `Mesh` for subdivisions (meshes) by using `struct` in C (or `class` in C++). The data type must contain all information about a subdivision $\mathcal{T}_h$.

16. Define an efficient data type `Matrix` for stiffness matrices K_h (in the one-dimensional case) by using `struct` in C (or `class` in C++).

    Hint: K_h is a tridiagonal matrix.

17. Write a function `StiffnessMatrix(↓mesh,↑matrix)` which assembles the (global) stiffness matrix `matrix` $= K_h$ for a given subdivision `mesh` $= \mathcal{T}_h$.

    For assembling K_h, start with $K_h = 0$ and use a loop over all elements to successively update K_h. On each element T_k, call the function `ElementStiffnessMatrix` to compute $K_h^{(k)}$ and update K_h by adding the entries of $K_h^{(k)}$ at the appropriate positions.

    Consider only the case of the following boundary conditions: $\Gamma_D = \emptyset$, $\Gamma_N = \{0, 1\}$ and $g_N = 0$.

18. Write a function `LoadVector(↓(*f)(x),↓mesh,↑vector)` which assembles the (global) load vector `vector` $= \underline{f}_h$ for a given subdivision `mesh` $= \mathcal{T}_h$.

    For assembling $\underline{f}_h$, start with $\underline{f}_h = 0$ and use a loop over all elements to successively update $\underline{f}_h$. On each element T_k, call the function `ElementLoadVector` to compute $\underline{f}_h^{(k)}$ and update $\underline{f}_h$ by adding the entries of $\underline{f}_h^{(k)}$ at the appropriate positions.

    Consider only the case of the following boundary conditions: $\Gamma_D = \emptyset$, $\Gamma_N = \{0, 1\}$ and $g_N = 0$.

Test your data types and functions for a simple example.