T U T O R I A L

"Numerical Methods for Solving Partial Differential Equations"

to the Lectures on NuPDE

T III Monday, 6 November 2006 (Time: 13:45 - 15:15, Room: P 004)

1.3 FEM for BVPs of Second-order ODEs

Let $\Omega = (0,1)$, $\Gamma = \partial\Omega = \{0,1\} = \Gamma_D \cup \Gamma_N$ with $\Gamma_D \cap \Gamma_N = \emptyset$.
Consider the one-dimensional boundary value problem: find $u(x)$ such that

$$\begin{array}{rcll} -u''(x) & = & f(x) & x \in \Omega \\ u(x) & = & g_D(x) & x \in \Gamma_D \\ u'(x) & = & \alpha(x)(g_R(x) - u(x)) & x \in \Gamma_R. \end{array} \tag{1.18}$$

We discretize the problem using the finite element method with Courant elements.
We consider the nodes $0 = x_0 < x_1 < \cdots < x_{N_h-1} < x_{N_h} = 1$ which define a mesh $\mathcal{T}_h$ of Ω made of subintervals $T_k = (x_{k-1}, x_k)$, $k = 1, \ldots, N_h$.
We introduce the finite element space:

$$V_h = \{v_h \in \mathcal{C}(\overline{\Omega}) |\ v_{h|T_k} \in \mathbb{P}_1 \text{ for all } T_k \in \mathcal{T}_h\}$$

whose basis is given by the nodal functions φ_i $(i = 0, \ldots, N_h)$, with $\varphi_i(x_j) = \delta_{ij}$ $(i, j = 0, \ldots, N_h)$.

In the following exercises we start developing a program that will allow us to compute the finite element approximation u_h of the solution u of (1.18).

We denote the input parameters of the functions by $\downarrow$ and the output parameters by $\uparrow$.

13 Write a function `ElementStiffnessMatrix(↓xa,↓xb,↑element_matrix)` that for `xa`= x_{k-1} and `xb`= x_k returns the 2×2 local stiffness matrix `element_matrix`= $K_h^{(k)}$ on the element T_k:

$$K_h^{(k)} = \begin{pmatrix} \int_{T_k} (\varphi'_{k-1}(x))^2 dx & \int_{T_k} \varphi'_{k-1}(x)\varphi'_k(x) dx \\ \int_{T_k} \varphi'_k(x)\varphi'_{k-1}(x) dx & \int_{T_k} (\varphi'_k(x))^2 dx \end{pmatrix}.$$

14 Write a function `ElementLoadVector(↓(*f)(x),↓xa,↓xb,↑element_vector)` that for `xa`= x_{k-1} and `xb`= x_k returns the 2×1 local load vector `element_vector`= $f_h^{(k)}$ on the element T_k:

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

Use the trapezoidal rule to approximate the integrals:

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

15 Define a data type `Mesh` which contains all the information about the mesh $\mathcal{T}_h$.

Hint: use `struct` *in C or* `class` *in C++.*

16 Define an efficient data type `Matrix` for the stiffness matrices K_h exploiting the fact that K_h is tridiagonal.

Hint: use `struct` *in C or* `class` *in C++.*

Consider now $\Gamma_D = \emptyset$, $\Gamma_R = \{0, 1\}$ and $\alpha(x) = 0$ (pure homogenous Neumann case).

17 Write a function `StiffnessMatrix(↓mesh,↑matrix)` that assembles the global $(N_h + 1) \times (N_h + 1)$ stiffness matrix `matrix`$= K_h$ for a given subdivision `mesh`$= \mathcal{T}_h$ of Ω.

Hint: set $K_h = 0$, then start with $K_h^{(0)}$ and use a loop over all the elements T_k to update the matrix K_h. On each T_k, call the function `ElementStiffnessMatrix` *to compute $K_h^{(k)}$ and pay attention to put the entries of $K_h^{(k)}$ at the correct position in K_h.*

18 Write a function `LoadVector(↓(*f)(x),↓mesh,↑vector)` that assembles the global $(N_h + 1) \times 1$ load vector `vector`$= \underline{f}_h$ for a given subdivision `mesh`$= \mathcal{T}_h$ of Ω.

Hint: set $\underline{f}_h = 0$, then start with $\underline{f}_h^{(0)}$ and use a loop over all the elements T_k to update the vector $\underline{f}_h$. On each T_k, call the function `ElementLoadVector` *to compute $\underline{f}_h^{(k)}$ and pay attention to add the entries of $\underline{f}_h^{(k)}$ at the correct position in $\underline{f}_h$.*

Test the data types and the functions that you have implemented taking some simple examples, e.g., consider equidistant nodes x_i and simple functions $f(x)$ ($f(x) = 1$, $f(x) = x \ldots$).