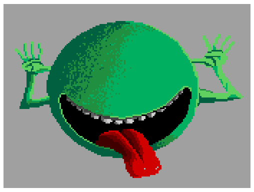
Course 401-0674-00L: NPDE
Midterm, 2013-04-19
Prof. Ralf Hiptmair



Don't panic!
Good luck!

Name
Student number

Points					

Problem 1 Energy Norm and Continuity [4 points]

On a function space V_0 we consider a linear variational problem

$$u \in V_0 : \quad \mathsf{a}(u,v) = \ell(v) \quad \forall v \in V_0, \tag{1.1}$$

with

- a symmetric, positive definite bilinear form $\mathsf{a} : V_0 \times V_0 \mapsto \mathbb{R}$,
- a right hand side linear form $\ell : V_0 \mapsto \mathbb{R}$ that satisfies

  $$|\ell(v)| \leq C_\ell \|v\|_a \quad \text{for some } C_\ell > 0,$$

  where $\|\cdot\|_a$ is the energy norm induced by a.

(1a) [2 points] Show that the solution u of (1.1) satisfies $\|u\|_a \leq C_\ell$.

(1b) **[2 points]** Show that $\frac{1}{2}\mathsf{a}(v,v) - \ell(v) \geq -\frac{1}{2}\|u\|_a^2$ for all $v \in V_0$, when u solves (1.1).

Problem 2 Robin Boundary Conditions [5 points]

We consider the linear 2nd-order elliptic boundary value problem

$$-\Delta u = f \quad \text{in } \Omega \subset \mathbb{R}^d \quad , \quad \mathbf{grad}\, u \cdot \boldsymbol{n} + \lambda u = 0 \quad \text{on } \partial\Omega. \tag{2.1}$$

(2a) **[3 points]** State the variational formulation for (2.1).

HINT: For the derivation you may use Green's formula

$$\int_\Omega \mathbf{j} \cdot \mathbf{grad}\, u + u \operatorname{div} \mathbf{j} \,\mathrm{d}\boldsymbol{x} = \int_{\partial\Omega} u\, \mathbf{j} \cdot \boldsymbol{n} \,\mathrm{d}S, \qquad \begin{array}{l} \mathbf{j} : \Omega \mapsto \mathbb{R}^d, \\ u : \Omega \mapsto \mathbb{R}. \end{array}$$

(2b) [2 points] Based on the variational formulation, argue why (2.1) has a unique (weak) solution.

Problem 3 One-Dimensional Convection [8 points]

Let $\mathcal{M}$ be an equidistant mesh of $]0,1[$ with $M \in \mathbb{N}$ cells and consider the bilinear form

$$\mathsf{b}(u,v) := \int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x}(x)\, v(x)\,\mathrm{d}x, \quad u,v \in H_0^1(]0,1[). \tag{3.1}$$

For its Galerkin discretization we use the space $\mathcal{S}_{1,0}^0(\mathcal{M}) \subset H_0^1(]0,1[)$ of piecewise linear Lagrangian finite elements on $\mathcal{M}$ equipped with the standard nodal basis of "tent functions" arranged in lexicographic order "from left to right" . Let $\mathbf{B}$ denote the Galerkin matrix of b w.r.t. $\mathcal{S}_{1,0}^0(\mathcal{M})$.

(3a) [1 points] Explain why the number of non-zero entries of $\mathbf{B}$ behaves like $\mathcal{O}(M)$ as $M \to \infty$.

(3b) **[3 points]** Compute the element matrix for b for an interior cell of $\mathcal{M}$.

HINT: The element matrices are not symmetric!

(3c) **[4 points]** Compute the Galerkin matrix $\mathbf{B}$.

Problem 4 Efficient Matrix-Vector Multiplication [8 points]

In the course we have seen that the assembly of a finite element Galerkin matrix can be accomplished by means of the MATLAB function `assembleFEMatrix` from Listing 4.1, provided that proper implementations of the functions `getElementMatrix` and `locglobmap` are available

Listing 4.1: Abstract assembly routine for finite element Galerkin matrices

```
function A = assembleFEMatrix(Mesh)
    A = sparse(0,0); maxidx = 0; % Allocate empty sparse matrix

    for k = Mesh.Elements' % loop over all cells
```

```
        % Local computation of element matrix for the k^th cell
        Ak = getElementMatrix(k);

        % Get row vector of global indices
        idx = locglobmap(k,(1:size(Ak,1)));

        % Grow matrix, if necessary
        maxtmp = max([idx,maxidx]);
        if (maxtmp > maxidx), maxidx = maxtmp; A(maxidx,maxidx) = 0; end

        % Add local contributions to global matrix; passing a vector
        % of indices selects submatrix!
        A(idx,idx) = A(idx,idx) + Ak;
    end
end
```

(4a) **[2 points]** Explain why the implementation of `assembleFEMatrix` given in Listing 4.1 performs poorly in terms of runtime.

(4b) **[6 points]** Assuming that the function `assembleFEMatrix` from Listing 4.1 performs assembly correctly, supplement the missing line 6 in the function `amux` of Listing 4.2 so that it returns the result of the multiplication of the Galerkin matrix with a vector $\vec{\xi}$ of suitable length, *without* ever forming the Galerkin matrix.

Listing 4.2: Multiplication of finite element Galerkin matrix with a vector

```
1 function y = amux(Mesh,xi)
2     y = zeros(length(xi));
3     for k = Mesh.Elements'
4         Ak = getElementMatrix(k);
5         idx = locglobmap(k,(1:size(Ak,1)));
6         % ???
7     end
8 end
```