TU Berlin — Department of Mathematics
Dr. Kersten Schmidt, Dirk Klindworth

winter term 2015/2016

NUMERICS OF PARTIAL DIFFERENTIAL EQUATIONS

Series 8

1. Local degrees of freedom for linear finite elements on triangular meshes

Let Π_K denote the local trial space of linear finite elements on triangular meshes. We consider the three choices of sets of local degrees of freedom

(i) $\Sigma_K = \{v \mapsto v(\mathbf{p}_j(K)) \;\mid\; j = 0,1,2\}$, where $\mathbf{p}_0(K), \dots, \mathbf{p}_2(K)$ are the three vertices of the triangle K,

(ii) $\Sigma_K = \{v \mapsto v(\boldsymbol{\gamma}), v \mapsto \frac{\partial v}{\partial \xi_1}(\boldsymbol{\gamma}), v \mapsto \frac{\partial v}{\partial \xi_2}(\boldsymbol{\gamma})\}$, where $\boldsymbol{\gamma}$ is the barycenter of the triangle K,

(iii) $\Sigma_K = \{v \mapsto \int_{e_j} v \,\mathrm{d}S \;\mid\; j = 0,1,2\}$, where $e_0, \dots, e_2$ are the edges of the triangle K. The finite element that corresponds to this choice of local degrees of freedom is called *Crouzeix-Raviart finite element*.

For each choice

a) verify that the dimension of the set of local degrees of freedom matches the dimension of the local trial space Σ_K,

b) verify that the set of local degrees of freedom is unisolvent, i.e. $\forall v \in \Pi_K : \quad (\ell(v) = 0 \quad \forall \ell \in \Sigma_K) \implies v = 0$,

c) determine where the local degrees of freedom are localized, i.e. localized on vertices or edges of the triangle or otherwise it is localized on the triangle, and

d) check if the finite element that is characterized by the local degrees of freedom is H^1-conforming.

2. $H(\mathrm{div})$-conforming linear finite elements

Let K be a triangle of the mesh $\mathcal{M}$ of a computational domain $\Omega \subset \mathbb{R}^2$. We consider the local trial space

$$\Pi_K = \{\mathbb{R}^2 \ni \boldsymbol{\xi} \mapsto \boldsymbol{\alpha} + \beta\boldsymbol{\xi},\ \boldsymbol{\alpha} \in \mathbb{R}^2, \beta \in \mathbb{R}\}$$

and the set of local degrees of freedom

$$\Sigma_K = \{\mathbf{v} \mapsto \int_{e_j} \mathbf{v} \cdot \mathbf{n} \,\mathrm{d}S \;\mid\; j = 0,1,2\},$$

where $e_0, \dots, e_2$ are the edges of the triangle K.

Show that (K, Π_K, Σ_K) is a unisolvent, $H(\mathrm{div})$-conforming finite element, i. e.

a) verify that the dimension of the set of local degrees of freedom matches the dimension of the local trial space Σ_K,

b) verify that the set of local degrees of freedom is unisolvent, i.e. $\forall \mathbf{v} \in \Pi_K : \quad (\ell(\mathbf{v}) = 0 \quad \forall \ell \in \Sigma_K) \implies \mathbf{v} = 0$, and

c) show that for any edge e_j of K the local degrees of freedom localized on e_j uniquely determine the trace $R_{\mathbf{n}}\mathbf{v}|_{e_j} = \mathbf{v}|_{e_j} \cdot \mathbf{n}$ of $\mathbf{v} \in \Pi_K$ on e_j.

See next page!

3. Convergence rate of the linear FE solver

For the setting in Exercise 3b) in Series 7, i.e.

$$\begin{aligned} -\Delta u + u &= f \quad \text{in } \Omega =]0,1[^2, \\ \mathbf{grad}\, u \cdot \mathbf{n} &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

with $f(x,y) = (8\pi + 1)\cos(2\pi x)\cos(2\pi y)$, compute the discretization error in the energy norm

$$\|e_n\|_{\mathrm{e}} := \sqrt{\mathsf{b}(e_n, e_n)} = \sqrt{\mathsf{b}(u - u_n, u - u_n)},$$

where b is the bilinear form given by

$$\mathsf{b}(u,v) = \int_\Omega \nabla u \cdot \nabla v + uv \,\mathrm{d}\mathbf{x}$$

and u_n is the solution of your linear FE solver.

Note, that with Galerkin orthogonality and the symmetry of the bilinear form

$$\begin{aligned} \mathsf{b}(u - u_n, u - u_n) = \mathsf{b}(u - u_n, u) = \mathsf{b}(u, u - u_n) = \ell(u) - \ell(u_n) \\ = \int_\Omega f u \,\mathrm{d}\mathbf{x} - \int_\Omega f u_n \,\mathrm{d}\mathbf{x}. \end{aligned}$$

The first expression $\ell(u)$ can be computed analytically (in case of a known solution) whereas the latter term is simply the scalar product of solution vector $\mathbf{u}$ and vector $\mathbf{f}$ of the right hand side

$$\int_\Omega f u_n \,\mathrm{d}\mathbf{x} = \sum_{k=1}^{N} (\mathbf{u})_k \int_\Omega f q_{n,P_k} \,\mathrm{d}\mathbf{x} = \mathbf{u} \cdot \mathbf{f}.$$

Compute the energy error for (at least) six different mesh widths (e. g. $h_i = h_0\, 2^{-i/2}$, $i = 0, 1, 2, \ldots$). Plot the energy error over the actual mesh width in double-logarithmic scaling. Use your function `max_mesh_width`, that you implemented in Series 5, in order to determine the actual maximal mesh width of an unstructured mesh. What is the observed convergence rate?

See next page!

4. Linear FE solver for Poisson problems with Neumann boundary conditions

Let $\Omega \subset \mathbb{R}^2$ be an open, bounded Lipschitz domain. Given the functions $f \in L^2(\Omega)$ and $g \in H^{-1/2}(\partial\Omega)$, we are interested in the solution of

$$-\Delta u = f \qquad \text{in } \Omega, \tag{1a}$$

$$\mathbf{grad}\, u \cdot \mathbf{n} = g \qquad \text{on } \partial\Omega, \tag{1b}$$

$$\int_\Omega u \,\mathrm{d}\mathbf{x} = 0, \tag{1c}$$

with the compatibility condition

$$\int_\Omega f \,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} g \,\mathrm{d}S(\mathbf{x}) = 0. \tag{2}$$

As discussed in Series 5 we consider a mixed variational formulation for the implementation, i.e. find $u \in H^1(\Omega)$ and the Lagrange multiplier $\lambda \in \mathbb{R}$ such that

$$\mathsf{b}(u, v) + \lambda \int_\Omega v(\mathbf{x})\mathrm{d}\mathbf{x} = \ell(v) \qquad \forall v \in H^1(\Omega) \tag{3a}$$

$$\int_\Omega u(\mathbf{x})\mathrm{d}\mathbf{x} \qquad\qquad = 0. \tag{3b}$$

a) Using a FE discretization the mixed variational formulation (3) leads to a linear system of the form

$$\begin{pmatrix} \mathbf{A} & \boldsymbol{m} \\ \boldsymbol{m}^\top & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \lambda_n \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}. \tag{4}$$

Now suppose that $g = 0$. Write the main code for solving (1) including

- definition of the source term f as Python function or Python's `lambda` function,
- mesh generation for the square $\Omega =]0, 1[^2$ in dependence of the maximal mesh width h_0,
- computation of the stiffness matrix $\mathbf{A}$,
- computation of the load vector $\mathbf{f}$ in dependence of the source term f, the order n of the numerical integration,
- computation of the vector $\mathbf{m}$,
- solution of the linear system (4) using Scipy's `spsolve`, and
- graphical representation of the solution with Matplotlib's `plot_trisurf`.

b) Let $\Omega =]0, 1[^2$ be the computational domain, $h_0 = 0.1$ the maximal mesh width and $n = 3$ the order of the numerical quadrature. Choose $f(x, y)$ such that $u(x, y) = \cos(\pi x)\cos(\pi y)$ is the analytical solution of (1). Compute and plot your FE approximation u_n of u and its discretization error $e_n = u - u_n$. Verify that the approximated Lagrange multiplier λ_n is reasonable close to the exact value λ.
Hint: Note that here $g = 0$ and the vector $\boldsymbol{m}$ can be obtained using your function for the load vector.

c) Add a function `elemLoadNeumann` to your Python module `FEM.py` that returns the element vector related to the Neumann boundary data.

```
#
# elemLoadNeumann(p, n, g)
#
```

See next page!

```
# returns the element vector related to the Neumann boundary
# data
#   int_I g v ds
# for linear FEM on the straight boundary edge I.
#
# input:
# p - 2x2 matrix of the coordinates of the nodes on the boundary
#     edge
# n - order of the numerical quadrature
# g - Neumann data as standard Python function or Python's
#     lambda function
#
# output:
# gK - element vector (2x1 array)
#
```

For the numerical quadrature choose the Gauß-Legendre rule, whose weights and abscissas for integration over the interval $[-1, 1]$ can be obtained using Numpy's `leggauss` function `numpy.polynomial.legendre.leggauss`.

d) Add a function `loadNeumann` to your Python module `FEM.py` that returns the vector related to the Neumann boundary data.

```
#
# loadNeumann(p, be, n, g)
#
# returns the vector related to the Neumann boundary data
#   int_dOmega g v ds
# for linear FEM on straight boundary edges.
#
# input:
# p  - Nx2 matrix with coordinates of the nodes
# be - Bx2 matrix with the indices of the nodes of boundary edges
# n  - order of the numerical quadrature
# g  - Neumann data as standard Python function or Python's
#      lambda function
#
# output:
# LoadNeumann - Nx1 vector as numpy-array
#
```

e) Write the main code for solving (1) with non-vanishing Neumann data g including

- definition of the source term f and the Neumann data g as Python function or Python's `lambda` function,
- mesh generation for the unit circle $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : |\mathbf{x}| < 1\}$ in dependence of the maximal mesh width h_0 (to this end, create a triangular mesh of a polygon as approximation to a circle using Gmsh),
- computation of the stiffness matrix $\mathbf{A}$,
- computation of the load vector $\mathbf{f}$ in dependence of the source term f, the Neumann data g and the order n of the numerical integration,
- computation of the vector $\mathbf{m}$,

See next page!

- solution of the linear system (4) using Scipy's `spsolve`, and
- graphical representation of the solution with Matplotlib's `plot_trisurf`.

f) Let the unit circle $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : |\mathbf{x}| < 1\}$ be the computational domain, $h_0 = 0.1$ the maximal mesh width and $n = 3$ the order of the numerical quadrature. Choose $f(x, y)$ such that $u(\mathbf{x}) = x_1 \sin(\pi|\mathbf{x}|)$, i. e. $u(r, \theta) = r \cos\theta \sin(\pi r)$, is the analytical solution of (1). Compute and plot your FE approximation u_n of u and its discretization error $e_n = u - u_n$.
Hint: Note that $\nabla \cdot \mathbf{n} \equiv \frac{\partial}{\partial r}$ *on* $\partial\Omega$.

To be handed in by: January 5th, 2016, 10:15 a.m.

Your solutions of the theoretical exercises 1 and 2 have to be handed in in paper form or as PDF file send to `klindworth@math.tu-berlin.de`!
Send your Python module `FEM.py`, the `.msh`-files of all used Gmsh meshes (if any), and the Python scripts `series8_3.py`, `series8_4b.py` and `series8_4f.py`, that do all steps explained in exercise 3, 4b) and 4f) without any need of interaction when executed from the command line of the operating system, to `klindworth@math.tu-berlin.de`!
The code must be well documented using prologue and inline comments! The plots must have titles and labels!

There will be no tutorial class on December 10th and December 17th. Instead there will be a voluntary but recommended consultation in MA 366 starting at 2:30 p.m. Bring your own laptop! In this consultation you can continue developing your code while Dirk Klindworth will be available for questions.

This exercise series will be discussed in the tutorial class on January 7th, 2016, 2:15 p.m. in A 052.

Coordinator: Dirk Klindworth, MA 365, 030/314-25192, `klindworth@math.tu-berlin.de`
Website: `http://www.tu-berlin.de/?NumPDE`