Scientific Computing I

Winter Semester 2013 / 2014
Prof. Dr. Beuchler
Bastian Bohn and Alexander Hullmann

universitätbonn

Excercise sheet 3.

Closing date **5.11.2013**.

Theoretical exercise 1. (Finite differences [5 points])

The second derivative u'' of a one-dimensional real-valued C^4-function $u(x)$ is to be discretized on a grid with mesh-width h and a finite difference stencil of the form

$$\begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix} .$$

Prove that the maximal consistency order of this finite-difference stencil is 2 and determine the corresponding α, β and γ.

Theoretical exercise 2. (Mixed second derivatives [5 points])

$$\begin{array}{ccc} u_{NW} & u_N & u_{NE} \\ u_W & u_C & u_E \\ u_{SW} & u_S & u_{SE} \end{array}$$

Figure 1: Positions of finite difference points

From the lecture, we already know the second derivatives $u_{xx}(C) \approx \frac{1}{h^2}(u_E + u_W - 2u_C)$ and $u_{yy}(C) \approx \frac{1}{h^2}(u_S + u_N - 2u_C)$. The approximation of *mixed* second derivatives u_{xy} is more difficult.

a) Show that a consistent formula is of the form

$$\frac{1}{4h^2}\begin{bmatrix} -1-\alpha-\beta+\gamma & 2(\alpha-\gamma) & 1-\alpha+\beta+\gamma \\ 2(\alpha+\beta) & -4\alpha & 2(\alpha-\beta) \\ 1-\alpha-\beta-\gamma & 2(\alpha+\gamma) & -1-\alpha+\beta-\gamma \end{bmatrix} \hat{=} \begin{bmatrix} u_{NW} & u_N & u_{NE} \\ u_W & u_C & u_E \\ u_{SW} & u_S & u_{SE} \end{bmatrix}$$

with $\alpha, \beta, \gamma \in \mathbb{R}$.

b) Prove that $u_{NW}, u_{SW}, u_{NE}, u_{SE}$ and u_W, u_S, u_E, u_N cannot have the same sign.

Theoretical exercise 3. (M-Matrices [5 points])

Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be an L_0-Matrix. Show that

a) $\mathbf{A}$ is inverse monotone iff there exists a vector $\mathbf{e} > 0$ with $\mathbf{Ae} > 0$.

b) Then, it holds that

$$\|\mathbf{A}^{-1}\| \leq \frac{\|\mathbf{e}\|}{\min_k(\mathbf{Ae})_k} .$$

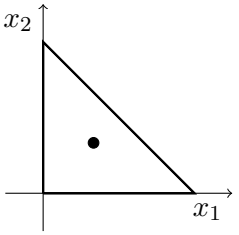
Programming exercise 1. (Numerical quadrature on the reference triangle [10 points])

On last week's exercise sheet you calculated some exact integrals involving linear and quadratic Lagrange polynomials ϕ_α and their derivatives on the reference triangle $\hat{T}$. This week the same integrals will be solved by numerical quadrature routines. A quadrature rule of size N is given by certain weights ω_i and nodes z_i for $i = 0, \ldots, N-1$. The integral of a function $f : \hat{T} \to \mathbb{R}$ is then approximated by

$$\int_{\hat{T}} f(x)\mathrm{d}x \approx \sum_{i=0}^{N-1} \omega_i \cdot f(z_i).$$

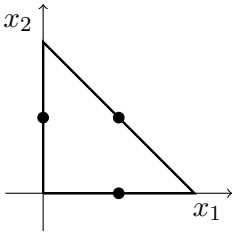
Today the following rules for $N = 1, 3, 7$ (which can be proven to be exact for polynomials of degree up to $1, 2, 5$) will be implemented:

- **Center rule**:



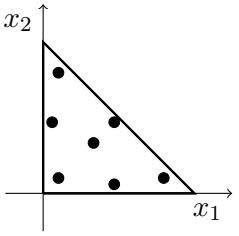
$$z_0 = \left(\tfrac{1}{3}, \tfrac{1}{3}\right)^T$$
$$\omega_0 = \tfrac{1}{2}$$

- **Edge midpoint rule**:



$$z_0 = \left(\tfrac{1}{2}, 0\right)^T,\ z_1 = \left(\tfrac{1}{2}, \tfrac{1}{2}\right)^T,\ z_2 = \left(0, \tfrac{1}{2}\right)^T$$
$$\omega_0 = \omega_1 = \omega_2 = \tfrac{1}{6}$$

- 7 **point rule**:



$$z_0 = \left(\tfrac{6-\sqrt{15}}{21}, \tfrac{6-\sqrt{15}}{21}\right)^T,\ z_1 = \left(\tfrac{9+2\sqrt{15}}{21}, \tfrac{6-\sqrt{15}}{21}\right)^T,$$
$$z_2 = \left(\tfrac{6-\sqrt{15}}{21}, \tfrac{9+2\sqrt{15}}{21}\right)^T,\ z_3 = \left(\tfrac{6+\sqrt{15}}{21}, \tfrac{9-2\sqrt{15}}{21}\right)^T,$$
$$z_4 = \left(\tfrac{6+\sqrt{15}}{21}, \tfrac{6+\sqrt{15}}{21}\right)^T,\ z_5 = \left(\tfrac{9-2\sqrt{15}}{21}, \tfrac{6+\sqrt{15}}{21}\right)^T,$$
$$z_6 = \left(\tfrac{1}{3}, \tfrac{1}{3}\right)^T$$
$$\omega_0 = \omega_1 = \omega_2 = \tfrac{155-\sqrt{15}}{2400},$$
$$\omega_3 = \omega_4 = \omega_5 = \tfrac{155+\sqrt{15}}{2400},$$
$$\omega_6 = \tfrac{9}{80}$$

Tasks:

a) [4 points] Create an `enum RuleName` which stands for the integration rules (i.e. `CENTER_RULE`, `EDGE_MIDPOINTS_RULE` or `SEVEN_POINT_RULE`). Implement the following functions (create a namespace `IntegrationRule` if you are programming in C++, see e.g. `http://www.cplusplus.com/doc/tutorial/namespaces/`).

- `int determineSizeOfRule(RuleName rule)` – returns N for the given `rule`.
- `void getQuadratureWeightsAndNodes(RuleName rule, double* weights, double** nodes, int size)` – returns the `weights` and `nodes` for the given `rule`. `size` is the size of the array `weights` and half of the size of the array `nodes`. It should be equal to N for the corresponding rule.

b) [6 points] Enhance the class `Basis` from last week. To this end, add the following information to the header-file.

- `bool doNumericalQuadrature` – this variable determines if integrals should be computed by hand or by numerical quadrature.
- `IntegrationRule::RuleName quadratureRule` – this is the rule which is used for numerical quadrature.

Implement the following member functions for the `Basis` class:

- `void enableQuadrature(IntegrationRule::RuleName rule)` – enables numerical quadrature instead of direct integral computation for the given `rule`.
- `void disableQuadrature()` – disables numerical quadrature.

Enhance the existing member functions from last weeks code. To this end, check the `doNumericalQuadrature` variable. If it is not set, execute the direct computation from last week. If it is set, calculate the corresponding integrals by numerical quadrature instead. The following member functions have to be enhanced:

- `double calcMassMatrixEntry(int i, int j, double factor)` – computes

$$\texttt{factor} \cdot \int_{\hat{T}} \phi_{\texttt{i}}(x)\phi_{\texttt{j}}(x)\mathrm{d}x.$$

- `double calcLoadVectorEntry(int i, double factor)` – computes

$$\texttt{factor} \cdot \int_{\hat{T}} \phi_{\texttt{i}}(x)\mathrm{d}x.$$

- `double calcStiffnessMatrixEntry(int i, int j, double** A, double d11, double d22, double factor)` – computes

$$\texttt{factor} \cdot \int_{\hat{T}} (\texttt{A} \cdot \nabla\phi_{\texttt{i}}(x))^T \cdot \begin{pmatrix} \texttt{d11} & 0 \\ 0 & \texttt{d22} \end{pmatrix} \cdot \left(\texttt{A} \cdot \nabla\phi_{\texttt{j}}(x)\right) \mathrm{d}x.$$

Test your implementation:

- For each of the three quadrature rules from above calculate the 6×6 mass matrix, the 6-dimensional load vector and the 6×6 stiffness matrix for the quadratic Lagrangian basis. The parameters are the same as last week, i.e. $\texttt{factor} = 1, \texttt{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ and $\texttt{d11} = 2$, $\texttt{d22} = 1$.

Feel free to use your own code from last week's exercise or the incomplete code from the website.