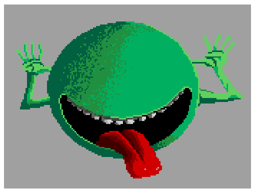
Course 401-0674-00L: Numerical Methods for Partial Differential Equations
Examination, August 13, 2010

Prof. Ralf Hiptmair



Dont't panic !
Good luck !

Duration of examination: 180 minutes

The total number of points is 200. Please pay attention to the number of points awarded for each (sub-)problem. It is roughly correlated with the amount of information your answer should contain. For additional information see the examination instruction sheet.

Problem 1. (Convergence of finite element solutions (15 points))

A student is testing his implementation of a finite element method. On the square domain $\Omega =]0,1[^2$ he considers the 2nd-order elliptic boundary value problem

$$-\Delta u = 1 \quad \text{in } \Omega, \qquad u = \frac{1}{4}\left(1 - \|\boldsymbol{x}\|^2\right) \quad \text{on } \partial\Omega \,. \tag{1}$$

He computes an approximate solutions $u_N \in \mathcal{S}_p^0(\mathcal{M})$ by means of a finite element Galerkin method using linear ($p = 1$) and quadratic ($p = 2$) Lagrangian finite elements on a sequence of triangular meshes $\mathcal{M}$.

The following table lists the measured $H^1(\Omega)$-seminorm of the discretization error as a function of the meshwidth $h_{\mathcal{M}}$.

$h_{\mathcal{M}}$	0.70	0.35	0.17	0.088	0.044	0.022	0.011
$\mathcal{S}_1^0(\mathcal{M})$	0.10	0.051	0.025	0.012	0.0064	0.0032	0.0008
$\mathcal{S}_2^0(\mathcal{M})$	$1.75 \cdot 10^{-16}$	$1.24 \cdot 10^{-15}$	$5.71 \cdot 10^{-15}$	$2.29 \cdot 10^{-14}$	$8.91 \cdot 10^{-14}$	$3.53 \cdot 10^{-13}$	$1.41 \cdot 10^{-12}$

The data of this table are available in the MATLAB data file `cvgtab.mat`.

(1a) (3 points)

Show that $u(\boldsymbol{x}) = \frac{1}{4}(1 - \|\boldsymbol{x}\|^2)$ is the exact solution of (1)

(1b) (8 points)

What kind of convergence (qualitative and quantitative) for linear Lagragian finite elements can be inferred from the error table?

(1c) (4 points)

Explain the striking difference between the behavior of the discretization error for linear and quadratic Lagrangian finite elements.

Problem 2. (Lax-Wendroff scheme (65 points))

The Cauchy problem for a generic scalar conservation law seeks $u : \mathbb{R} \times [0,T] \mapsto \mathbb{R}$, which solves

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial}{\partial x} f(u) &= 0 \quad \text{in } \mathbb{R} \times [0,T] \; , \\ u(x,0) &= u_0(x) \; , \quad x \in \mathbb{R} \; . \end{aligned} \tag{2}$$

We compute approximations $\mu_j^{(k)} \approx u(jh, k\tau)$ on an equidistant space-time mesh with spatial meshwidth $h > 0$ and uniform timestep $\tau > 0$ by means of the so-called Lax-Wendroff scheme

$$\begin{aligned} \mu_j^{(k)} = \mu_j^{(k-1)} - \tfrac{1}{2}\gamma \left(f(\mu_{j+1}^{(k-1)}) - f(\mu_{j-1}^{(k-1)}) \right) + \tfrac{1}{2}\gamma^2 \Big(f'(\tfrac{1}{2}(\mu_{j+1}^{(k-1)} + \mu_j^{(k-1)}))^2 (\mu_{j+1}^{(k-1)} - \mu_j^{(k-1)}) - \\ f'(\tfrac{1}{2}(\mu_j^{(k-1)} + \mu_{j-1}^{(k-1)}))^2 (\mu_j^{(k-1)} - \mu_{j-1}^{(k-1)}) \Big) \; , \quad \gamma := \frac{\tau}{h} \; , \end{aligned} \tag{3}$$

with initial values $\mu_j^{(0)} := u_0(jh)$, $j \in \mathbb{Z}$.

In this problem we consider the special flux function $f(u) = \exp(u)$.

(2a) (10 points) Assume that there are $A, B \in \mathbb{R}$, $A < B$, such that

$$u_0(x) = A \quad \text{for } x \le 0 \; , \quad A \le u_0(x) \le B \quad \text{for } 0 < x < 1 \; , \quad u_0(x) = B \quad \text{for } x \ge 1 \; . \tag{4}$$

Sketch the maximal possible set $\{(x,t) \in \mathbb{R} \times [0,1] : \; B < u(x,t) < A\}$ in the $x-t$-plane.

(2b) (20 points) Implement a MATLAB function

```
muend = lwsol(u0,T,M)
```

that uses the Lax-Wendroff scheme (3) to solve (2) for $f(u) = \exp(u)$ over the time period $[0,T]$ under the assumption (4) with $0 \le A < B \le 1$. Use spatial meshwidth $h = e\tau$, $e := \exp(1)$.

The arguments have the following meaning:

`u0`	:	handle to the function $u_0 : \mathbb{R} \mapsto \mathbb{R}$,
`T`	:	final time $T > 0$,
`M`	:	number of timesteps.

In `muend` the function is to return the approximations $\mu_j^{(M)}$ for $j = \lfloor \frac{-3T}{h} \rfloor, \dots, \lceil \frac{3T+1}{h} \rceil$. Here $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ give the integer precursor and successor, respectively, of a real number.

(2c) (7 points) Use your implementation of `lwsol` to solve the Riemann problem in the MATLAB script `prob2c.m`.

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} \exp(u) = 0 \quad \text{in } \mathbb{R} \times [0,1] \; , \quad u(x,0) = \begin{cases} 0 & \text{for } x < 0 \; , \\ 1 & \text{for } x \ge 0 \; . \end{cases} \tag{5}$$

Also compute the error norm

$$\mathrm{err}(M) = h \sum_{j\in\mathbb{Z}} |\mu_j^M - u(jh,1)| \tag{6}$$

for $M = 20, 40, 80, 160, 320, 640$, create a log-log plot of $\mathrm{err}(M)$ and determine the rate of convergence.

Hint: The analytic solution is a rarefaction wave given by

$$u(x,t) = \begin{cases} 0 & \text{for } x < t \ , \\ \log(\frac{x}{t}) & \text{for } t < x < et \ , \\ 1 & \text{for } x > et \ . \end{cases} \tag{7}$$

If you do not trust your implementation of `lwsol` you may resort to the reference implemenetation `lwsol_Ref` in `lwsol_Ref.p`.

(2d) (8 points) Repeat the evaluations of the previous sub-problem for the C^1 initial data

$$u_0(x) = \begin{cases} \sin^2(\frac{\pi}{2}x) & \text{for } 0 \le x \le 1 \ , \\ 0 & x < 0, \\ 1 & x > 1. \end{cases} \tag{8}$$

Implement this in `prob2d.m`.

Hint: The "exact" solution $x \mapsto u(x,1)$ is provided by the MATLAB function `u = exactsol(x)` that expects a vector `x` of locations and the returns very accurate approximations to the point values of the exact solution there.

(2e) (10 points) Derive the Godunov numerical flux function for (2) and the flux function $f(u) = \exp(u)$.

(2f) (10 points) Implement a MATLAB-function

```
muend = godsol(u0,T,M)
```

that does the simulation requested in sub-problem (2b) based on a conservative finite volume method and Godunov numerical flux with explicit Euler timestepping.

Problem 3. (Parabolic evolution problem (120 points))

On the spatial domain $\Omega =]0,1[^2$ we consider the parabolic initial boundary value problem

$$\begin{aligned} \frac{\partial u}{\partial t} - \Delta u &= f && \text{on } \Omega\times]0,T[\ , \\ u &= 0 && \text{on } \partial\Omega\times]0,T[\ , \\ u(\cdot,0) &= u_0 && \text{in } \Omega \ , \end{aligned} \tag{9}$$

where the initial data u_0 and the source function $f = f(\boldsymbol{x},t)$ are given.

For the spatial finite element Galerkin discretization of (9) we employ parametric bilinear Lagrangian finite elements on general *quadrilateral* meshes like the one depicted in Figure 1.

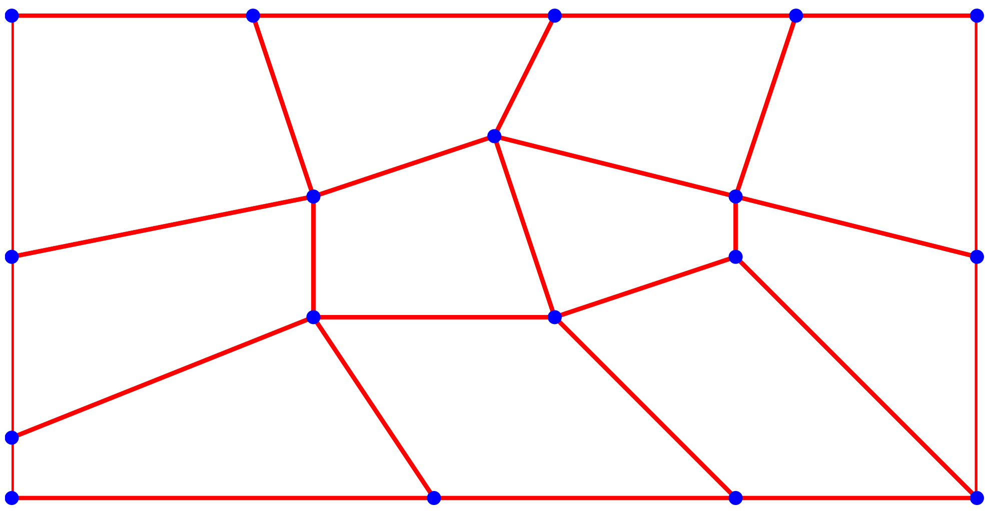


Figure 1: Example of a general quadrilateral mesh

(3a) (5 points) Derive the spatial variational formulation of the parabolic evolution problem (9).

(3b) (3 points) Give the formula of a bilinear transformation $\mathbf{\Phi}_K : \widehat{K} \mapsto K$ that takes the unit square $\widehat{K} =]0,1[^2$ to a general quadrilateral with the vertices $\boldsymbol{a}^1, \boldsymbol{a}^2, \boldsymbol{a}^3, \boldsymbol{a}^4 \in \mathbb{R}^2$ (numbered counterclockwise, see Figure 2).

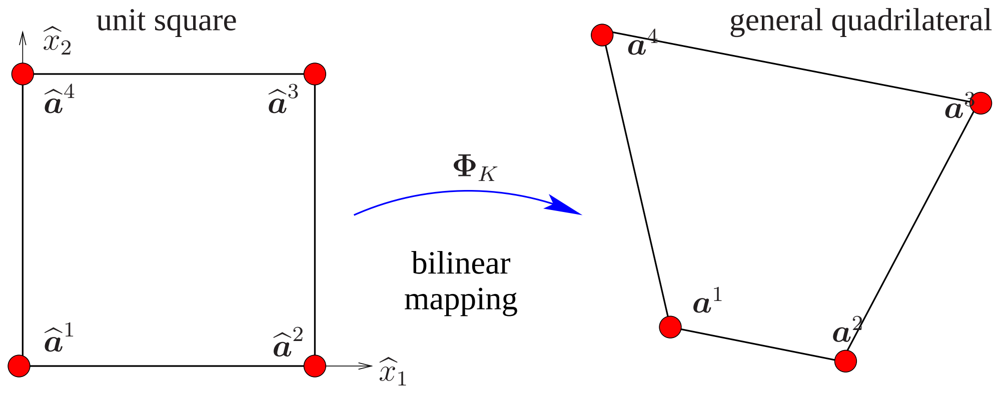


Figure 2: Bilinear mapping taking the unit square to a general quadrilateral

(3c) (5 points) We write $\widehat{b}^i$, $i = 1, \ldots, 4$, for the bilinear local shape functions on $\widehat{K}$ and $b^i_K := (\mathbf{\Phi}_K^{-1})^* \widehat{b}^i$ for those on the generic quadrilateral K. Show that

$$\mathbf{grad}_{\boldsymbol{x}}\, b^i_K(\boldsymbol{x}) = \mathbf{M}(\mathbf{\Phi}_K^{-1}(\boldsymbol{x}))^{-T}\, \mathbf{grad}_{\widehat{\boldsymbol{x}}}\, \widehat{b}^i(\mathbf{\Phi}_K^{-1}(\boldsymbol{x})) \;, \tag{10}$$

with a $\mathbb{R}^{2,2}$-valued function

$$\mathbf{M}(\widehat{\boldsymbol{x}}) := \big(\widehat{x}_2(\boldsymbol{a}^1 - \boldsymbol{a}^2 + \boldsymbol{a}^3 - \boldsymbol{a}^4) + (\boldsymbol{a}^2 - \boldsymbol{a}^1), \quad \widehat{x}_1(\boldsymbol{a}^1 - \boldsymbol{a}^2 + \boldsymbol{a}^3 - \boldsymbol{a}^4) + (\boldsymbol{a}^4 - \boldsymbol{a}^1)\big) \;. \tag{11}$$

(3d) (15 points) Complete the implementation of the LehrFEM-MATLAB function

```
function Aloc =
               STIMA_Lapl_BFE(Vertices,ElemInfo,QuadRule,varargin)
```

that computes the element matrix corresponding to the bilinear form $\mathsf{a}_\Delta(u,v) := \int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, v \,\mathrm{d}\boldsymbol{x}$ for parametric bilinear Lagrangian finite elements on a general quadrilateral based on a local quadrature rule.

Here, the 2×4 matrix `Vertices` passes the coordinates of $\boldsymbol{a}^1, \boldsymbol{a}^2, \boldsymbol{a}^3, \boldsymbol{a}^4$ and `QuadRule` is a structure storing the weights and nodes of a quadrature rule on $\widehat{K}$ on the fields `QuadRule.w` and `QuadRule.x`, respectively. The other arguments need not be used.

Hint: Supplied is the function `grad_shap_BFE` that takes a 2-vector $\widehat{\boldsymbol{x}} \in \widehat{K}$ of coordinates and returns the gradients $\mathbf{grad}_{\widehat{\boldsymbol{x}}}\, \widehat{b}^i(\widehat{\boldsymbol{x}})$, $i = 1, \ldots, 4$, in one column vector with the components of the i-th gradient stored in coefficients $2i - 1$ and $2i$.

Hint: A reference implementation is supplied through the function `STIMA_Lapl_BFE_Ref` in the file `STIMA_Lapl_BFE_Ref.p`, but the source code is not accessible.

(3e) (12 points) Complete the implementation of the LehrFEM-MATLAB function

```
function floc = Load_BFE(Vertices,f,QuadRule),
```

that returns the element load vector corresponding to the linear form $\ell(v) := \int_\Omega f v \,\mathrm{d}\boldsymbol{x}$ for parametric bilinear Lagrangian finite elements on a general quadrilateral based on a local quadrature rule `QuadRule`. The arguments `Vertices` and `QuadRule` match those of `STIMA_Lapl_BFE`, whereas `f` contains a handle of type `@(x)` to the function $f : \Omega \mapsto \mathbb{R}$.

Hint: You may use the supplied function `shap_BFE` that takes a $m \times 2$-matrix of coordinates of points $\widehat{\boldsymbol{x}}_j \in \widehat{K}$, $j = 1, \ldots, m$, and returns an $m \times 4$-matrix of values $\widehat{b}^i(\widehat{\boldsymbol{x}}_j)$, $i = 1, \ldots, 4$, $j = 1, \ldots, m$.

Hint: A reference implementation is supplied through the function `Load_BFE_Ref` in the file `Load_BFE_Ref.p`, but the source code is not accessible.

(3f) (15 points) A *convex* quadrilateral with vertices $\boldsymbol{a}^1, \boldsymbol{a}^2, \boldsymbol{a}^3, \boldsymbol{a}^4 \in \mathbb{R}^2$ (numbered counter-clockwise) can be split into two triangles with vertices $\boldsymbol{a}^1, \boldsymbol{a}^2, \boldsymbol{a}^3$ and $\boldsymbol{a}^1, \boldsymbol{a}^3, \boldsymbol{a}^4$. The *piecewise linear* "tent functions" associated with the vertices of these triangles can serve as local shape functions on the quadrilateral.

Relying on the supplied LehrFEM-MATLAB function

```
Aloc = STIMA_Lapl_LFE(Vertices,varargin),
```

which computes the element stiffness matrix for $-\Delta$ and triangular linear Lagrangian finite elements, complete the LehrFEM-MATLAB function

```
function Aloc = STIMA_Lapl_BFEsplit(Vertices,varargin)
```

that computes the element matrix corresponding to the bilinear form $\mathsf{a}_\Delta(u,v) := \int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, v \,\mathrm{d}\boldsymbol{x}$ based on the piecewise linear local shape function described above.

Here, the 2×4 matrix `Vertices` passes the coordinates of $\boldsymbol{a}^1, \boldsymbol{a}^2, \boldsymbol{a}^3, \boldsymbol{a}^4$, whereas the variable `varargin` can be ignored.

Hint: A reference implementation is supplied through the function `STIMA_Lapl_BFEsplit_Ref` in the file `STIMA_Lapl_BFEsplit_Ref.p`, but the source code is not accessible.

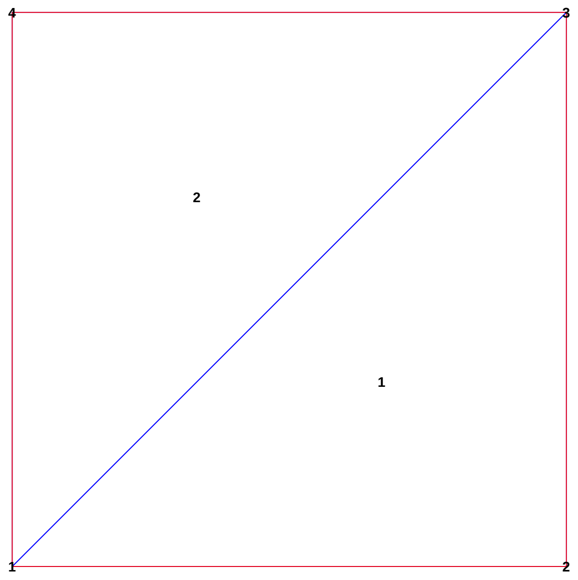


Figure 3: A split quadrilateral.

(3g) (5 points) Calculate a tight bound for the number of non-zero entries of the stiffness matrix for $-\Delta$ (with homogeneous Dirichlet boundary conditions) discretized by means of paremetric bilinear Lagrangian finite elements on a general quadrilateral mesh. The bound should be stated in terms of the numbers of cells, (interior) edges, and (interior) vertices of the mesh.

(3h) (15 points) Complete the missing lines in the routine

```
err = L2Err_BFE(Mesh,u,QuadRule,FHandle,varargin)
```

that computes an approximation of $\|u_N - u\|_{L^2(\Omega)}$ for $u_N \in \mathcal{S}^0_{1,0}(\mathcal{M})$ and $u \in C^0(\overline{\Omega})$ by means of the local quadrature rule passed in the argument `QuadRule` (see sub-problem (3d) for further explanations). The argument `u` contains the coefficients of the nodal basis representation of $u_N \in \mathcal{S}^0_{1,0}(\mathcal{M})$, $\mathcal{M}$ a general quadrilateral mesh.

Hint: Further information can be found in the comments inside the function template.

Hint: A reference implementation is supplied through the function `L2Err_BFE_Ref` in the file `L2Err_BFE_Ref.p`, but the source code is not accessible.

(3i) (5 points) With stiffness matrix $\mathbf{A}$ and mass matrix $\mathbf{M}$ we can formally write the spatially semi-discrete version of (9) as

$$\mathbf{M}\frac{d\vec{\boldsymbol{\mu}}}{dt}(t) - \mathbf{A}\vec{\boldsymbol{\mu}}(t) = \vec{\boldsymbol{\varphi}}(t) \quad , \quad \vec{\boldsymbol{\mu}}(0) = \vec{\boldsymbol{\mu}}_0 \; . \tag{12}$$

State a fully discrete version that uses implicit Euler timestepping with uniform timestep $\tau > 0$.

(3j) (10 points) Explain why it is not advisable to use *explicit* Runge-Kutta single-step timestepping methods for (12) with a spatial discretization with piecewise linear Langrangian finite elements.

(3k) (10 points) Implement the timestepping scheme from sub-problem (3i) in

`u = Parb_Evl_BFE(Mesh,u0,FHandle,T,Nsteps)`,

where `Mesh` is a LehrFEM mesh data structure for a quadrilateral mesh, `u0` the coefficient vector for the intial data, `FHandle` of type `@(x,t)` passes the time-dependent right hand side, `T` gives the final time, and `Nsteps` the number of timesteps. The function is to return the finite element solution at final time (encoded by its basis coefficients, of course).

Parts of the code that compute the matrices $\mathbf{A}$ and $\mathbf{M}$ are already supplied. The function `assemLoad_BFE` can be used to obtain $\vec{\boldsymbol{\varphi}}(t)$.

Hint: Further information can be found in the comments inside the function template. You may use the reference implementations of the MATLAB functions from earlier sub-rpoblems, in case you do not trust your implementations. A reference implementation of the currently requested function is available through `Parb_Evl_BFE_Ref`.

(3l) (10 points) For $u_0(\boldsymbol{x}) = \sin(\pi x_1)\sin(\pi x_2)$, $f \equiv 0$ we find the exact solution

$$u(\boldsymbol{x}, t) = e^{-2\pi^2 t}\sin(\pi x_1)\sin(\pi x_2), \quad (x_0, x_1)^T \in \Omega =]0,1[^2. \tag{13}$$

for (9). Extend the MATLAB script `main_BFE.m` to study the convergence (qualitatively and quantitatively) of the *spatial* discretization on a sequence of regularly refined quadrilateral meshes. To that end examine the discretization error $\|u_N(T) - u(T)\|_{L^2(\Omega)}$ for final time $T = 0.1$. Choose the timestep small enough such that the temporal discretization error will never dominate. What rate of convergence do you observe?

Hint: You may use `Parb_Evl_BFE_Ref`. Look at the code template in `main_BFE.m` for further information.

(3m) (10 points) How should the (uniform) timestep for the implicit Euler method be linked to the meshwidth in order to obtain optimal convergence of $\|u_N(T) - u(T)\|_{L^2(\Omega)}$ with *best possible efficiency*.

Hint: Take into account the convergence of the spatial discretization studied in sub-problem (3l).

References

[NPDE] Lecture slides for course "Numerical Methods for Partial Differential Equations", Subversion Revision