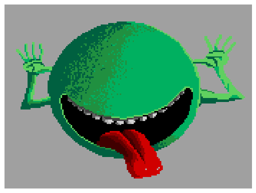
Course 401-3663-00L: Numerical Methods for Partial Differential Equations
Examination, Spring 2011

Prof. Ralf Hiptmair


Dont't panic !
Good luck !

Duration of examination: 180 minutes

Problem 1. (Convection-diffusion problem (55 points))

For $\epsilon > 0$ we consider the one-dimensional convection diffusion problem on $\Omega =]0, 1[$

$$-\epsilon \frac{d^2 u}{dx^2} + \frac{du}{dx} = f(x) \quad \text{in }]0,1[\quad , \quad u(0) = u(1) = 0 \; . \tag{1}$$

The following variational formulation has been suggested for (1): seek $u \in H^2(]0,1[)$ such that

$$\int_0^1 \epsilon \frac{du}{dx}\frac{dv}{dx} + \frac{du}{dx} v \,\mathrm{d}x$$
$$+u(0)v(0) + \epsilon \left(\frac{du}{dx}(0)v(0) - \frac{du}{dx}(1)v(1) + u(0)\frac{dv}{dx}(0) - u(1)\frac{dv}{dx}(1) + \alpha u(0)v(0) + \alpha u(1)v(1) \right)$$
$$= \int_0^1 f v \,\mathrm{d}x \; , \tag{2}$$

for all $v \in H^2(]0,1[)$. Here $\alpha > 0$ is a parameter.

(1a) (5 points) Show that a smooth solution of (1) will also solve (2).

(1b) (20 points) Compute the linear system of equations arising from the Galerkin finite element discretization of (2) by means of piecewise linear Lagrangian finite elements on an equidistant grid with meshwidth $h := \frac{1}{N}$, $N \in \mathbb{N}$. Use the trapezoidal rule for the approximate evaluation of the integrals.

(1c) (15 points) Write a MATLAB function

```
u = solve2pcdbvp(N,epsilon,f_hd)
```

that solves (1) with the discretization introduced in (1b) using N grid cells. The argument `f_hd` is a function handle of type `@(x)` providing the function f. The vector `u` is to return the values of the finite element solution at the nodes of the mesh. Choose $\alpha = \frac{10}{h}$.

Hint. The function `solve2pcdbvpRef.p` supplies a reference implementation of `solve2pcdbvp`.

(1d) (15 points) For $f \equiv 1$, $\epsilon = 0.01$, create a suitable plot of the error norm

$$\mathrm{err}(N) = \left(\frac{1}{N} \sum_{j=1}^{N-1} |(u - u_N)(jh)|^2 \right)^{1/2}$$

and use it to describe qualitatively and quantitatively the convergence of the method in this norm.

Hint. The exact solution in the case $f \equiv 1$ is

$$u(x) = x + \frac{\exp(\frac{x-1}{\epsilon}) - \exp(-\frac{1}{\epsilon})}{\exp(-\frac{1}{\epsilon}) - 1} \; .$$

Problem 2. (Enquist-Osher numerical flux (75 points))

We consider the following Cauchy problem for a scalar conservation law

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial}{\partial x} f(u) = 0 \quad &\text{on } \mathbb{R}\times]0,T[\; , \\ u(x,0) = u_0(x) \quad &\forall x \in \mathbb{R} \; , \end{aligned} \tag{3}$$

where $f : \mathbb{R} \mapsto \mathbb{R}$ is the flux function.

The Cauchy problem (3) should be solved by a fully discrete conservative finite volume method on an equidistant mesh $\mathcal{M} = \{]x_{j-1}, x_j[\colon\; x_j = jh, j \in \mathbb{Z}\}$ with meshwidth $h > 0$ combined with explicit Euler timestepping. We rely on the 2-point Enquist-Osher *numerical flux*

$$F_{\mathrm{EO}}(v,w) := \tfrac{1}{2}(f(v) + f(w)) - \tfrac{1}{2} \int_v^w |f'(\xi)| \, \mathrm{d}\xi \; . \tag{4}$$

(2a) (5 points) Show that the Engquist-Osher flux is consistent with the flux function f.

(2b) (15 points) Show that the Engquist-Osher flux is monotone.

For the remainder of this problem consider the special choice

$$f(u) := \cosh(u) = \frac{1}{2}(e^u + e^{-u}) \; .$$

(2c) (10 points) If u_0 is supported in $[0,1]$ and $-1 \leq u_0(x) \leq 1$ for all $x \in \mathbb{R}$, find the maximal possible support of $u(\cdot,t)$ for time $t > 0$, where $u(x,t)$ solves (3).

Hint. The flux function $f(u) = \cosh(u)$ and its dervative are plotted in Fig. 1

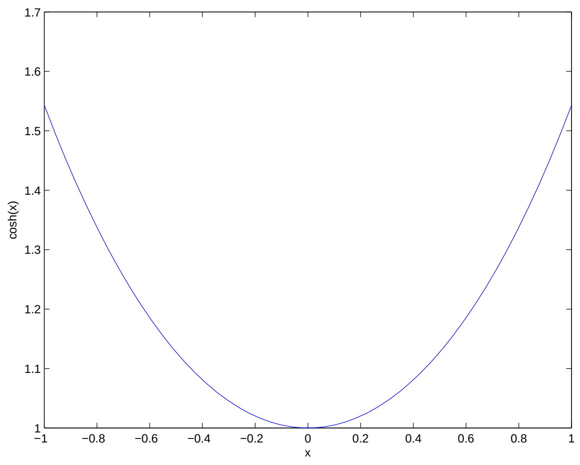
(2d) (10 points) Determine the CFL-condition (maximal timestep as a function of spatial mesh width h) for the conservative finite volume method for (3) introduced above, if it is known that $A \leq u_0(x) \leq B$ for all $x \in \mathbb{R}$, with $A, B \in \mathbb{R}$, $A < B$.

(2e) (10 points) Write a MATLAB function

```
nf = eonf(v,w)
```

that implements the Engquist-Osher numerical flux function F_{EO} according to (4) for the flux $f(u) = \cosh(u)$.

Hint. A reference implementation is given as `eonfRef`.



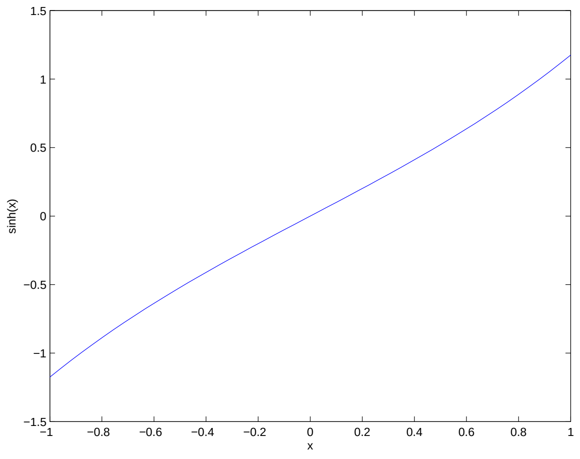


Figure 1: Graphs for flux function $f(u) = \cosh(u)$ (left) and $f'(u) = \sinh(u)$ (right)

(2f) (15 points) Complete the MATLAB function

```
ufinal = solveCP(a,b,N,u0,T)
```

in the file `solveCP.m` that solves the Cauchy problem (3) with the conservative finite volume method described above up to final time $T > 0$. The computations should be done on the spatial interval $[a, b]$ on a grid with nodes $x_j = a + \frac{b-a}{N}(j - \frac{1}{2})$, $j = 1, \ldots, N$. The row vector `u0` passses the cell averages of u_0 and the scalar `T` the final time. Constant continuation of u_0 outside $[a, b]$ is assumed. The row vector `unfinal` returns the approximate cell averages of $u(\cdot, T)$. Use the maximal timestep possible according to the CFL condition, see (2d).

(2g) (10 points) Use the function `solveCP` to solve (3) for initial data

$$u_0(x) = \begin{cases} 1 & \text{for } 0 < x \leq 1 \ , \\ -1 & \text{elsewhere,} \end{cases}$$

and final time $T = 1$. Use $[-1.2, 2.2]$ as computational interval and $N = 100$. The cell avarages of u_0 can be approximated by its values at the cell centers. Plot the approximate cell averages at final time.
Hint. A reference implementation of `solveCP` is provided as `solveCPRef`.

Problem 3. (Transport problem (65 points))

The MATLAB functions

```
data = semiLagr_setup(mesh,v_hd)
u1 = semiLagr_step(u0,tau,data)
```

implement the semi-Lagrangian discretization of the transient convection-diffusion problem

$$\begin{aligned} \frac{\partial u}{\partial t} - \Delta u + \mathbf{v}(\boldsymbol{x}) \cdot \mathbf{grad}\, u = 0 \quad & \text{in } \Omega \times]0, T[\ , \\ u = 0 \quad & \text{on } \partial\Omega \times]0, T[\ , \\ u(\boldsymbol{x}, 0) = u_0(\boldsymbol{x}) \ , \quad & \boldsymbol{x} \in \Omega \ , \end{aligned} \tag{5}$$

Here $\Omega \subset \mathbb{R}^2$ is a polygonal computational domain implicitly described by the LehrFEM mesh data structure for a triangular mesh $\mathcal{M}$ passed as the `mesh` argument. The parameter `v_hd` provides a handle to the continuous stationary velocity vector field $\mathbf{v} = \mathbf{v}(\boldsymbol{x})$ (column vector !).

Using this information the function `semiLagr_setup` performs setup computations and stores their results in `data`. The function `semiLagr_step` carries out a single timestep with timestep size $\tau > 0$. The column vector argument `u0` passes the vertex values of a finite element function $\in \mathcal{S}^0_{1,0}(\mathcal{M})$ and the vertex values of the approximate solution after time τ are returned in the column vector `u1`. The numbering of vertices is given by their index in the `Coordinates` field of the `mesh` data structure.

(3a) (10 points) Use the two functions to solve (5) on the unit disc $\Omega := \{\boldsymbol{x} \in \mathbb{R}^2 : \|\boldsymbol{x}\| \leq 1\}$ with

$$\mathbf{v}(\boldsymbol{x}) = \begin{pmatrix} -x_2 \\ x_1 \end{pmatrix} , \quad \boldsymbol{x} \in \Omega .$$

and final time $T = 2\pi$. To that end write a MATLAB function

`ufinal = solverot(u0_hd,N)` ,

which relies on N uniform timesteps of the semi-Lagrangian method and reads a triangular mesh from `CircMesh.mat` (use MATLAB's `load` function to import the `mesh`).

The argument `u0_hd` is a handle to a real valued function on Ω that provides u_0. The function returns the values of the finite element solution at $t = T$ and at interior vertices.

Hint. The supplied function `idof = get_Int_DOF(mesh)` will give you the indices of the interior vertices of the mesh.

(3b) (5 points) For a stationary velocity field $\mathbf{v} \in (H^1(\Omega))^2$ we consider the evolution problem

$$\begin{aligned} \frac{\partial u}{\partial t} - \Delta u + \operatorname{div}(\mathbf{v}(\boldsymbol{x})u) &= 0 \quad \text{in } \Omega\times]0,T[\;, \\ u &= 0 \quad \text{on } \partial\Omega\times]0,T[\;, \\ u(\boldsymbol{x},0) &= u_0(\boldsymbol{x}) \;, \quad \boldsymbol{x} \in \Omega \;. \end{aligned} \tag{6}$$

Convert it into an initial-boundary value problem for the PDE

$$\frac{\partial u}{\partial t} - \Delta u + \mathbf{v}(\boldsymbol{x}) \cdot \mathbf{grad}\, u - c(\boldsymbol{x})u = 0 \;, \tag{7}$$

with suitable coefficient function $c : \Omega \mapsto \mathbb{R}$, which is to be stated in terms of $\mathbf{v}$.

Hint. Apply the product rule to $\operatorname{div}(\mathbf{v}\, u)$.

(3c) (15 points) Which evolution problems (over small time intervals) have to be solved in each step, when split-step timestepping based on the *Strang splitting* is applied to the evolution problem

$$\begin{aligned} \frac{\partial u}{\partial t} = \underbrace{\Delta u - \mathbf{v}(\boldsymbol{x}) \cdot \mathbf{grad}\, u}_{=:g(u)} + \underbrace{c(\boldsymbol{x})u}_{=:h(u)} &\quad \text{in } \Omega\times]0,T[\;, \\ u = 0 &\quad \text{on } \partial\Omega\times]0,T[\;, \\ u(\boldsymbol{x},0) = u_0(\boldsymbol{x}) \;, &\quad \boldsymbol{x} \in \Omega \;. \end{aligned} \tag{8}$$

Here the decomposition of the right-hand side of the PDE indicated by the underbraces defines the splitting to be used. The timestep size should be denoted by $\tau > 0$.

(3d) (20 points) Write a MATLAB function

```
u1 = reaction_step(mesh,u0,tau,c_hd)
```

that solves the variational evolution problem: seek $t \mapsto u_N(t) \in \mathcal{S}^0_{1,0}(\mathcal{M})$

$$\int_\Omega \frac{\partial u_N}{\partial t} v_N \mathrm{d}\boldsymbol{x} = \int_\Omega c(\boldsymbol{x}) u_N \, v_N \, \mathrm{d}\boldsymbol{x} \quad \forall v_N \in \mathcal{S}^0_{1,0}(\mathcal{M}) \;, \tag{9}$$

over one timestep of length $\tau > 0$ using the explicit midpoint rule, a 2-stage explicit Runge-Kutta method described by the Butcher scheme

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array} \quad . \tag{10}$$

The integrals in (10) should be approximated by means of the local vertex based quadrature rule ("2D trapezoidal rule").

The argument `mesh` should pass a LehrFEM mesh data structure. The column vector `u0` contains the nodal values for u_N before the timestep, `u1` those after the timestep. `c_hd` is the function handle for $\mathbf{c} = \mathbf{c}(\boldsymbol{x})$.

Hint. Again use the function `idof = get_Int_DOF(mesh)` to find the indices of the interior vertices.

(3e) (15 points) Implement a MATLAB function

```
ufinal = solvetrp(mesh,v_hd,c_hd,u0_hd,N) ,
```

which relies on N uniform steps of the Strang-splitting split-step method discussed in (3c) to solve (8) approximately with $T = 1$. The parameter `v` provides a handle to the continuous stationary velocity vector field $\mathbf{v} = \mathbf{v}(\boldsymbol{x})$ (column vector !) and `c_hd` a handle to $\mathbf{c} = \mathbf{c}(\boldsymbol{x})$. The argument `u0_hd` is a handle to a real valued function on Ω that provides u_0.

Hint. Use the functions `semiLagr_setup` (once in the beginning), `semiLagr_step` and `reaction_step`. For the latter function a reference implementation is provided in `reaction_stepRef.p`. Pay attention to an efficient "leapfrog-style" implementation of the Strang-splitting.

References

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