

Course 401-3663-00S: Numerical Solution of
Differential Equations
Examination, 08.08.2012

Prof. Philipp Grohs



Dont't panic!
Good luck!

Duration of examination: 180 minutes

The total number of points is 235. Please pay attention to the number of points awarded for each (sub-)task. It is roughly correlated with the amount of information your answer should contain. For additional information see the examination instruction sheet.

Problem 1. Parabolic timestepping with Crank-Nicolson and implicit Euler [85 points]

Let $\Omega := (0, 1)^2$ and consider the problem

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

where f , g and u_0 are given such $u(t, \mathbf{x}) = \cos(2\pi x_1) \sin(t\pi x_2)$ is the exact solution.

(1a) (5 points) Derive the variational formulation for this parabolic problem.

(1b) (10 points) Show that the initial value problem arising from a spatial discretization of the variational formulation using piecewise linear finite elements with basis functions $\{b_N^i\}_i$ is given by

$$\begin{aligned} M \frac{d}{dt} \vec{\mu}(t) + A \vec{\mu}(t) &= \mathbf{F}(t), \\ M \vec{\mu}(0) &= \vec{\mu}_0, \end{aligned} \tag{1}$$

where $\vec{\mu}(t)$ is the finite element coefficient vector, $\vec{\mu}_0 = \vec{\mu}(0)$, \mathbf{F} is the time-dependent load vector

$$F_i(t) = \int_{\Omega} f(t, \mathbf{x}) b_N^i(\mathbf{x}) d\mathbf{x},$$

and M and A are the mass- and Galerkin matrices respectively

$$M_{ji} = \int_{\Omega} b_N^i(\mathbf{x}) b_N^j(\mathbf{x}) d\mathbf{x}, \quad A_{ji} = \int_{\Omega} \text{grad } b_N^i(\mathbf{x}) \cdot \text{grad } b_N^j(\mathbf{x}) d\mathbf{x}.$$

(1c) (15 points) For an initial value problem

$$\dot{\mathbf{y}} = \mathbf{h}(t, \mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0,$$

let the time-stepping scheme be given for $m = 0, \dots, M$ by the *Crank-Nicolson scheme*

$$\mathbf{y}^{(m+1)} = \mathbf{y}^{(m)} + \frac{1}{2}\Delta t \left(\mathbf{h}(t_m, \mathbf{y}^{(m)}) + \mathbf{h}(t_{m+1}, \mathbf{y}^{(m+1)}) \right),$$

with initial value $\mathbf{y}^{(0)} = \mathbf{y}_0$, time step $\Delta t := T/M$ and time points $t_m := m\Delta t$.

For which Δt is this scheme stable?

HINT: Consider the ODE $\dot{\mathbf{y}} = \lambda \mathbf{y}$ for $\lambda < 0$.

(1d) (10 points) Show that the Crank-Nicolson scheme applied to (1) gives the following linear system:

$$\left(\mathbf{M} + \frac{1}{2}\Delta t \mathbf{A} \right) \vec{\mu}^{(m+1)} = \left(\mathbf{M} - \frac{1}{2}\Delta t \mathbf{A} \right) \vec{\mu}^{(m)} + \frac{1}{2}\Delta t (\mathbf{F}_{m+1} + \mathbf{F}_m), \quad (2)$$

where $\mathbf{F}_k = \mathbf{F}(t_k)$.

(1e) (15 points) Implement (2) arising from a spatial discretization of the variational formulation using piecewise linear finite elements by completing the routine

```
main_crank.
```

Use the supplied function `p12` for any quadrature that you might need. For each level of mesh refinement, use the number of timesteps needed to balance the errors from time and space discretization.

Run your code and plot the convergence. What type of convergence and what order do you observe?

(1f) (15 points) Use the implicit Euler scheme and linear Lagrangian elements to derive a new timestepping scheme for this problem.

Copy your code for `main_crank` to

```
main_implEul,
```

and modify it so that it calculates the convergence rate for the implicit Euler timestepping scheme.

Plot the convergence rate and comment on the type of convergence and the rate.

HINT: For an ODE $\dot{\mathbf{y}} = h(t, \mathbf{y})$, the implicit Euler scheme with timestep τ is

$$\mathbf{y}^{(m+1)} = \mathbf{y}^{(m)} + \tau h(t_{m+1}, \mathbf{y}^{(m+1)}).$$

(1g) (15 points) Make a copy of `main_crank` and `main_implEul` called

```
main_crank_nonsmooth
```

and

```
main_implEul_nonsmooth
```

respectively. In these files, use $f \equiv 0$ and use the supplied `u_jump` function for both initial condition and Dirichlet boundary data. This function is

$$u_0(\mathbf{x}) = \begin{cases} 1, & x_1 < x_2, \\ 0, & \text{otherwise,} \end{cases}$$

i.e. it is not smooth. Remove all the L^2 -error computing code, since we no longer have an exact solution. Instead, for each mesh refinement level, plot the solution along the diagonal from $(0, 1)$ to $(1, 0)$ using the LehrFEM function `plotLine_LFE`.

Run both scripts and comment on what you see.

Problem 2. The Euler-Bernoulli equation [100 points]

The scaled Euler-Bernoulli equation in one dimension reads

$$\frac{d^4 u}{dx^4} = q(x) \quad (3)$$

for $x \in (0, L)$, with boundary conditions

$$u(0) = u'(0) = 0, \quad \frac{d^2 u}{dx^2}(L) = M_0, \quad \frac{d^3 u}{dx^3}(L) = -F. \quad (4)$$

(2a) (5 points) Show that the variational formulation for this problem is to find

$$u \in \mathcal{S} = \{f \in H^2(0, L) \mid f(0) = f'(0) = 0\}$$

so that

$$\int_0^L u'' v'' dx = \int_0^L u q dx + F v(L) + M_0 v'(L)$$

for all $v \in \mathcal{S}$.

HINT: The left-hand boundary conditions should be treated as *essential*, that is, a part of the function spaces. The right-hand boundary conditions should be treated as *non-essential*. They will enter the variational formulation as part of the linear functional ℓ .

(2b) (5 points) To solve this problem we will be using piecewise *cubic* finite elements on a uniform grid with meshwidth h . For each element, we will use the degrees of freedom corresponding to the values of the function *and* its derivatives at the endpoints of the element.

Show that on the reference element $(-1, 1)$, the four shape functions

$$\begin{aligned} \hat{b}_1(\xi) &= \frac{1}{4}(1 - \xi)^2(2 + \xi) & \hat{b}_2(\xi) &= \frac{1}{4}(1 - \xi)^2(1 + \xi) \\ \hat{b}_3(\xi) &= \frac{1}{4}(1 + \xi)^2(2 - \xi) & \hat{b}_4(\xi) &= \frac{1}{4}(1 + \xi)^2(\xi - 1) \end{aligned}$$

satisfy the conditions.

HINT: You need to show the following

$$\begin{aligned} \hat{b}_1(-1) &= 1, & \hat{b}_1'(-1) &= 0, & \hat{b}_1(1) &= 0, & \hat{b}_1'(1) &= 0, \\ \hat{b}_2(-1) &= 0, & \hat{b}_2'(-1) &= 1, & \hat{b}_2(1) &= 0, & \hat{b}_2'(1) &= 0, \\ \hat{b}_3(-1) &= 0, & \hat{b}_3'(-1) &= 0, & \hat{b}_3(1) &= 1, & \hat{b}_3'(1) &= 0, \\ \hat{b}_4(-1) &= 0, & \hat{b}_4'(-1) &= 0, & \hat{b}_4(1) &= 0, & \hat{b}_4'(1) &= 1. \end{aligned}$$

(2c) (10 points) Write a MATLAB routine `shap = shap_CFE(x)` that computes the values of the shape functions in the points given in the $N \times 1$ -vector \mathbf{x} (relative to $(-1, 1)$) and returns them in the $N \times 4$ -vector `shap`.

Make a plot of the four shape functions on the reference element.

(2d) (10 points) Let $K = [x_j, x_{j+1}]$ be a given element with $x_{j+1} - x_j = h$. What are the transformed shape functions $b_K^i, i = 1, \dots, 4$, on this element?

HINT: Normal affine transformation applies. You must scale the second and fourth shape function to preserve the conditions

$$\begin{aligned} b_K^1(x_j) &= 1, & b_K^1(x_{j+1}) &= 0, & b_K^1(x_j) &= 0, & b_K^1(x_{j+1}) &= 0, \\ b_K^2(x_j) &= 0, & b_K^2(x_{j+1}) &= 0, & b_K^2(x_j) &= 1, & b_K^2(x_{j+1}) &= 0, \\ b_K^3(x_j) &= 0, & b_K^3(x_{j+1}) &= 1, & b_K^3(x_j) &= 0, & b_K^3(x_{j+1}) &= 0, \\ b_K^4(x_j) &= 0, & b_K^4(x_{j+1}) &= 0, & b_K^4(x_j) &= 0, & b_K^4(x_{j+1}) &= 1. \end{aligned}$$

(2e) (10 points) The second derivatives of the shape functions *on the reference element* are

$$\begin{aligned} \hat{b}_1''(\xi) &= \frac{3}{2}\xi & \hat{b}_2''(\xi) &= \frac{3}{2}\xi - \frac{1}{2} \\ \hat{b}_3''(\xi) &= -\frac{3}{2}\xi & \hat{b}_4''(\xi) &= \frac{3}{2}\xi + \frac{1}{2} \end{aligned}$$

Show that the element stiffness matrix for an element K of size h is

$$\mathbf{A}_K = \frac{1}{h^3} \begin{pmatrix} 12 & 6h & -12 & 6h \\ 6h & 4h^2 & -6h & 2h^2 \\ -12 & -6h & 12 & -6h \\ 6h & 2h^2 & -6h & 4h^2 \end{pmatrix}.$$

HINT: Remember to that the transformation scales the second derivatives by $(2/h)^2$. Note also the scaling factor introduced in (2f).

HINT: You only need three integrals (of 1, ξ and ξ^2). The rest is simple algebra.

Given a mesh

$$\mathcal{M} = \{0 = x_0, x_1, \dots, x_N = L\}$$

made up of elements $K_j = [x_j, x_{j+1}]$, for $j = 0, \dots, N-1$, we define the global basis functions b_j^1 and b_j^2 for $j = 0, \dots, N$ as

$$b_j^1(x) = \begin{cases} b_{K_{j-1}}^3(x), & x \in K_{j-1} \\ b_{K_j}^1(x), & x \in K_j \end{cases}$$

and

$$b_j^2(x) = \begin{cases} b_{K_{j-1}}^4(x), & x \in K_{j-1} \\ b_{K_j}^2(x), & x \in K_j. \end{cases}$$

In this way, shape functions 1 and 3 on two consecutive elements are joined to form one basis function b_j^1 , much like how piecewise linear tent functions work. The same works with shape

functions 2 and 4 forming b_j^2 . Note that the basis functions are piecewise polynomial and *globally continuously differentiable*.

We will employ the basis numbering according to the listing

$$\{b_0^1, b_0^2, b_1^1, b_1^2, \dots, b_N^1, b_N^2\}.$$

(2f) (15 points) Write a MATLAB function

$$A = \text{assemMat_CFE}(L, N)$$

that takes in the domain size parameter L and a number of *elements* N , and returns the global stiffness matrix A of size $2(N+1) \times 2(N+1)$.

(2g) (15 points) Write a MATLAB function

$$L = \text{assemLoad_CFE}(L, N, q, F, M_0)$$

that takes in the same parameters as before, and in addition a function handle to q (which you can assume is vectorizable, i.e. it can take vector arguments, not just doubles), as well as the boundary conditions F and M_0 , and returns the column load vector of size $2(N+1)$.

HINT: Use two-point Gaussian quadrature to evaluate the integrals you need. Relative to $(0, 1)$, it has points $\pm 1/\sqrt{3}$ and weights 1. Do not forget to transform.

(2h) (15 points) Write a MATLAB function

$$[xpts, U] = \text{eval_CFE}(L, N, u, K)$$

that takes in the previously mentioned arguments L and N , as well as a coefficient vector u of size $2(N+1)$ and a positive integer K , and outputs a list of $KN+1$ equally spaced x -points in $xpts$ and the corresponding values of the function u in U .

HINT: Remember to scale the shape functions 2 and 4 by $h/2$.

(2i) (15 points) Write a MATLAB script

$$\text{main_CFE}$$

to solve the problem (3)-(4) with the conditions $L = 1$, $M_0 = -2\pi$, $F = -\pi^3$ and

$$q(x) = -4\pi^3 \cos(\pi x) + \pi^4 x \sin(\pi x)$$

using $N = 10, 20, \dots, 100$. Evaluate the solutions using `eval_CFE` with $K = 11$, and use these points as quadrature rules to compute the L^2 -error for each mesh. The exact solution is

$$u(x) = x \sin(\pi x).$$

Plot the convergence rate of the L^2 -error and comment on the type of convergence and its rate.

Problem 3. One-dimensional conservation law [50 points]

In this problem we will develop a relatively simple solver for the one-dimensional conservation law

$$\frac{du}{dt} + \frac{d}{dx}(u^4) = 0 \quad (5)$$

with flux function $f(u) = u^4$ and initial condition $u(0, x) = u_0(x)$.

(3a) (10 points) Write a MATLAB function

`characteristics(a, b, T, N, u0)`

that takes in a function u_0 and plots characteristics for it in the (x, t) -plane. The plot should be restricted to the x -interval $[a, b]$ and the t -interval $[0, T]$, showing characteristics starting in N different equispaced points along the x -axis.

Produce plots for both $u_0(x) = e^{-x^2}$ and

$$u_0 = \begin{cases} 1, & -\frac{1}{2} < x < \frac{1}{2} \\ 0, & \text{otherwise.} \end{cases}$$

Why can you not use these plots to reconstruct the solution $u(x, t)$?

We will be solving (5) using ordinary finite volume methods to track the evolution of the *cell averages* $\mu_j(t)$, defined for some grid $\{x_j\}_j$ as

$$\mu_j(t) = \frac{1}{x_{j+1} - x_j} \int_{x_j}^{x_{j+1}} u(t, x) dx.$$

The ODE in question is

$$\frac{d\mu_j}{dt} = -\frac{1}{h} [F(\mu_j, \mu_{j+1}) - F(\mu_{j-1}, \mu_j)], \quad (6)$$

where F is a numerical flux function.

(3b) (10 points) We will be using the *Lax-Friedrich* numerical flux function

$$F_{\text{LF}}(v, w) = \frac{1}{2} (f(v) + f(w)) - \frac{1}{2} (w - v) \max_{u \in [v, w]} |f'(u)|.$$

Implement a MATLAB function

`F = Flf(v, w)`

that computes the Lax-Friedrich flux function for two *scalar* (that is, not vectorized) inputs v and w , using the flux function $f(u) = u^4$.

(3c) (10 points) Implement a MATLAB function

`F = Flfvec(mu)`

That takes in a column vector of cell averages $\vec{\mu}$ and outputs the quantities

$$F(\mu_j, \mu_{j+1}) - F(\mu_{j-1}, \mu_j)$$

for each j . Assume that $\vec{\mu}$ extends indefinitely to the left and right with zero.

HINT: You can use `arrayfun` to vectorize your implementation from (3b).

(3d) (10 points) For timestepping, we will use the explicit Heun scheme (also known as improved Euler), which is given by the Butcher table

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}, \quad (7)$$

Implement a MATLAB function

```
mufinal = heun(mu0, h, T, M)
```

that performs M timesteps of the Heun method using the initial vector of cell averages μ_0 and final time T . The spatial resolution h is also given.

(3e) (10 points) To test your implementation, write a MATLAB script

```
main
```

That runs the timestepping for $T = 1$ with $M = 100$ timesteps using the initial condition

$$u_0(x) = \begin{cases} 1, & -\frac{1}{2} \leq x \leq \frac{1}{2} \\ 0, & \text{otherwise.} \end{cases}$$

Use a mesh covering the interval $[-2, 2]$ with 80 cells. Plot the final distribution.

Problem References

[NPDE] [Lecture Slides](#) for the course “Numerical Methods for Partial Differential Equations”, SVN revision # 54024.

Last modified on April 11, 2013