R. Hiptmair
J. Šukys

Fall term 2011

Numerical Methods for CSE

ETH Zürich
D-MATH

Examination

August 14th, 2012

Instructions.

Duration of examination: 180 minutes. **Total points: 150**

Concise answers are desirable, but any "yes" or "no" answer requires explaining.

Write Matlab codes as simple as possible and add essential comments. Features of a code that have not been asked for will not earn extra points.

Only the Matlab files that are requested in the problem statement will be corrected. The theoretical parts of the problems have to be solved **on paper**.

All the requested `.m` and `.eps` files (with the correct file names) have to be saved in the folders

`~/resources/Matlab/Task*/`

Do not save or modify any file outside these folders.

The course scripts are available in `~/resources/NumCSE11(_ext).pdf`

Look at the number of points awarded for sub-problems to gauge the desired level of detail for the answer.

Good luck!

Problem 1 Advantages of higher order quadrature rules

[7 points]

It is known that for $f \in C^2(I)$, $I = [0,1]$, the quadrature error of the composite trapezoidal rule

$$E(n) = \int_0^1 f(x)\,\mathrm{d}x - h\left(\frac{1}{2}f(0) + \sum_{i=1}^{n-1} f(ih) + \frac{1}{2}f(1)\right), \qquad h = \frac{1}{n},$$

behaves like $E(n) = O(n^{-2})$, for $n \to \infty$. Thus, the trapezoidal rule could be used to solve any numerical quadrature problem with C^2-integrand up to arbitrary accuracy.

Explain why other higher-order composite quadrature rules are relevant, nevertheless.

Problem 2 Cubic spline interpolation

[8 points]

We consider an interval $[a,b] \subset \mathbb{R}$ equipped with a knot set $\mathcal{M} := \{a = t_0 < t_1 < \ldots < t_{n-1} < t_n = b\}$. Let $p : [a,b] \mapsto \mathbb{R}$ be the restriction of a cubic polynomial to $[a,b]$.

Show that the complete cubic spline interpolant $\in \mathcal{S}_{3,\mathcal{M}}$ of p always agrees with p, but the natural cubic spline interpolant need not.

Problem 3 Newton's method

[25 points]

Let $\mathbf{A} \in \mathbb{R}^{n,n}$ be a large sparse s.p.d. matrix and $\mathbf{b} \in \mathbb{R}^n$. Consider the following non-linear system of equations:

$$\text{seek} \quad \mathbf{x} \in \mathbb{R}^n : \qquad \left(\mathbf{A} + \|\mathbf{x}\|_2^2\,\mathbf{I}\right)\mathbf{x} = \mathbf{b}\,. \tag{1}$$

(3a) [10 points] Give the concrete formula of the Newton iteration for the solution of (1).

(3b) [15 points] Implement a Matlab function

```
function [x, res] = newton(n)
```

which uses the damped Newton method

```
function [x, res] = dampnewton(x0,F,DF,rtol,atol)
```

provided in the file `dampnewton.m` to solve (1) for

```
A = gallery('poisson',n);    b = ones(n*n,1);
```

with initial guess $\mathbf{x}^{(0)} := \mathbf{A}^{-1}\mathbf{b}$. Use relative tolerance (`rtol`) of 10^{-6} and absolute tolerance (`atol`) of 10^{-8}.

Output the norm of the solution vector $\|\mathbf{x}\|_2$ and the final residual `res(end)` for $n = 10$.

Problem 4 Zero finding

[30 points]

The non-linear system of equations (1) may be recast as a non-linear system with $n + 1$ unknowns

$$\text{seek} \quad \mathbf{x} \in \mathbb{R}^n, \lambda \in \mathbb{R}: \quad \begin{cases} (\mathbf{A} + \lambda \mathbf{I})\mathbf{x} = \mathbf{b}, \\ \|\mathbf{x}\|_2^2 - \lambda = 0. \end{cases} \tag{2}$$

(4a) [10 points] Formulate the Newton iteration for (2).

(4b) [7 points] Through the elimination of $\mathbf{x}$ from (2) one arrives at a scalar non-linear equation in the unknown λ, whose solutions agree with the zeros of a function $\psi : \mathbb{R} \mapsto \mathbb{R}$. State this function (you can use inverses of matrices) and implement a Matlab function

```
function y = psi(A,b,x)
```

which returns the value of Ψ at a point `x` for a given matrix $\mathbf{A}$ and right hand side vector $\mathbf{b}$.

(4c) [3 points] Implement a Matlab function

```
function plot_psi(A,b)
```

which plots Ψ for a given matrix $\mathbf{A}$ and right hand side vector $\mathbf{b}$. Plot the function in interval $[-10, 10]$ for the concrete data

```
A = gallery('poisson',n);    b = ones(n*n,1);
```

as in sub-problem (3b).

HINT: A scrambled implementation of the function Ψ is available in the file `psi_ref.p`, which provides the function `psi_ref(A,b,x)`.

(4d) [10 points] Implement a Matlab function

```
function lambda = psi_zero(A,b)
```

which uses Matlab 's function `fzero` to solve the scalar non-linear problem from the previous sub-problem.

Output the obtained λ for the data $\mathbf{A}$ and $\mathbf{b}$ as specified in sub-problem (4c).

HINT: You have to find two values $0 \leq \lambda_a \leq \lambda_b$ such that $\psi(\lambda_a) < 0$ and $\psi(\lambda_b) > 0$.

HINT: A scrambled implementation of the function Ψ is available in the file `psi_ref.p`, which provides the function `psi_ref(A,b,x)`.

Problem 5 The leapfrog single step method for 2nd-order ODEs

[30 points]

Somebody claims that the following timestepping scheme (with uniform timestep $h > 0$)

$$\begin{cases} \dfrac{\mathbf{v}_{j+\frac{1}{2}} - \mathbf{v}_{j-\frac{1}{2}}}{h} = \mathbf{f}(\mathbf{y}_j) \\ \dfrac{\mathbf{y}_{j+1} - \mathbf{y}_j}{h} = \mathbf{v}_{j+\frac{1}{2}} \end{cases} \quad \text{with} \quad \frac{1}{2}(\mathbf{v}_{\frac{1}{2}} + \mathbf{v}_{-\frac{1}{2}}) = \dot{\mathbf{y}}(0)\ , \tag{3}$$

can be used to solve the initial value problem for the 2nd-order ODE

$$\ddot{\mathbf{y}} = \mathbf{f}(\mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0, \quad \dot{\mathbf{y}}(0) = \mathbf{v}_0, \tag{4}$$

where $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ is supposed to be a continuous function.

(5a) [10 points] Implement an *efficient* (memory usage must not depend on the parameter `N`) Matlab function

```
function yfinal = lf (f, y0, T, N)
```

that uses `N` uniform timesteps of (3) to solve (4) with $\mathbf{v}_0 = 0$ on the time interval $[0, T]$, and returns an approximation for $\mathbf{y}(T)$. Here `f` passes a handle of type `@(y)` to the function $\mathbf{f}$.

HINT: For the first sub-step (i.e. to obtain $\mathbf{v}_{\frac{1}{2}}$), use (3) to derive $\mathbf{v}_{\frac{1}{2}} = \mathbf{v}_0 + \frac{h}{2}\mathbf{f}(\mathbf{y}_0)$.

(5b) [10 points] Use the adaptive Matlab built-in integrator `ode45` with absolute tolerance 10^{-10} and relative tolerance 10^{-8} in a Matlab function

```
function yfinal = lfode45(f,y0,T)
```

that also computes an approximation of $\mathbf{y}(T)$ for the solution $\mathbf{y}(t)$ of (4) with $\mathbf{v}_0 = 0$ and $\mathbf{y}_0$ passed in `y0`.

HINT: Rewrite (4) as system of first order ordinary differential equations.

(5c) [10 points] Implement a Matlab function

```
function order = lforder (f,y0,T)
```

which determines and outputs the order of convergence of the error $\|\mathbf{y}(T) - \mathbf{y}^{(N)}\|_2$ for the timestepping scheme (3) by comparing with the result from `lfode45`. Use numbers `N` of time steps `N` equal to $2^{4,\ldots,9}$.

Compute the order for $d = 1$, $\mathbf{f}(y) = \sin(y)$, $\mathbf{y}(0) = 1$, and $T = 1$.

HINT: Scrambled implementations of `lf(f,y0,T,N)` and `lfode45(f,y0,T,N)` are available through `lf_ref.p` and `lfode45_ref.p`.

Problem 6 Matrix equation

[20 points]

Consider the non-linear matrix equation

$$\mathbf{X} \in \mathbb{R}^{n,n} - \textbf{upper triangular}: \quad \mathbf{X}^\top \mathbf{A}\mathbf{X} = \mathbf{S} \tag{5}$$

for given *s.p.d.* matrices $\mathbf{S} \in \mathbb{R}^{n,n}$, $\mathbf{A} \in \mathbb{R}^{n,n}$.

(6a) [10 points] Show existence and uniqueness of a solution $\mathbf{X}$ of (5).

HINT: Cholesky decomposition.

(6b) [7 points] Implement a Matlab function

```
function X = mateqsolve(A,S)
```

that computes the solution of (5).

(6c) [3 points]

What is the asymptotic complexity in terms of the problem size parameter n of your implementation of `mateqsolve`?

HINT: if you cannot determine the asymptotic complexity analytically, you can obtain it numerically by conducting `tic-toc` run time measurements. In that case, please use template `mateqsolve_comp`.

HINT: A scrambled implementation of `mateqsolve(A,S)` is available in `mateqsolve_ref.p`.

Problem 7 Special matrices

[30 points]

Let the matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ be defined as

$$\mathbf{A} = \left[f\left(\frac{i+j}{n}\right) \right]_{i,j=1}^{n} , \tag{6}$$

where $f : \mathbb{R} \to \mathbb{R}$ is an arbitrary function.

(7a) [3 points] How many different numerical values can the elements of $\mathbf{A}$ attain at most?

(7b) [5 points] Find a permutation matrix $\mathbf{P} \in \mathbb{R}^{n,n}$ such that matrix $\mathbf{PA}$ is a Toeplitz matrix.

(7c) [2 points] What is the asymptotic complexity of the Matlab command `y = A*x` in terms of the problem size parameter n, for a given vector $\mathbf{x} \in \mathbb{R}^n$?

(7d) [20 points] Implement an *efficient* Matlab function

```
function y = multA(n,f,x)
```

which computes the matrix vector product $\mathbf{y} = \mathbf{Ax}$ with an asymptotic computational effort of $O(n \log(n))$. Here `f` passes a handle to the function f.

Compute the norm $\|\mathbf{y}\|_2$ of the vector `y` from `multA` for $n = 10$, $\mathbf{x} = (1, 2, \ldots, 10)^\top$ and $f(x) = \sin(x)$.

HINT: Multiplication with a Toeplitz matrix can be realized by a multiplication with a circulant matrix.