R. Hiptmair
A. Moiola

Fall term 2010

Numerical Methods for CSE

ETH Zürich
D-MATH

Examination

August 11th, 2011

Instructions.

Duration of examination: 180 minutes. **Total points: 90.**

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 folder
`/home/exam/resources/Matlab`.
Do not save or modify any file outside this folder.

The course script is available in `/home/exam/resources/NumCSE10slides.pdf`.

The maximum grade can be obtained by solving four problems out of five.

Problem 1 Structured matrix–vector product [16 points]

Consider the real $n \times n$ matrix $\mathbf{A}$ defined by $(\mathbf{A})_{i,j} = a_{i,j} = \min\{i, j\}$, for $i, j = 1, \ldots, n$. The matrix-vector product $\mathbf{y} = \mathbf{A}\mathbf{x}$ can be implemented in Matlab as

$$\mathtt{y = min(ones(n,1)*(1:n), (1:n)'*ones(1,n))*x;} \tag{1}$$

(1a) [2 points] What is the asymptotic complexity (for $n \to \infty$) of the evaluation of the Matlab command displayed above, with respect to the problem size parameter n?

(1b) [6 points] Write an *efficient* Matlab function

`function y = multAmin(x)`

that computes the same multiplication as (1) but with a better asymptotic complexity with respect to n.

HINT: you can test your implementation by comparing the returned values with the ones obtained with code (1).

(1c) [2 points] What is the asymptotic complexity (in terms of problem size parameter n) of your function `multAmin`?

(1d) [2 points] Consider the following Matlab script `multAB.m`:

```
n = 10;
D = [-ones(n,1),[2*ones(n-1,1);1],-ones(n,1)];
B = spdiags(D,[-1,0,1],n,n);
x = rand(n,1);
fprintf('|x-y| = %d\n',norm(multAmin(B*x)-x));
```

Sketch the matrix $\mathbf{B}$ created in line 3 of `multAB.m`.

HINT: this Matlab script is provided as file `multAB.m`.

(1e) [4 points] Several runs of `multAB` from the previous sub-problem produced the following output:

```
>> multAB
|x-y| = 3.587240e-15
>> multAB
|x-y| = 1.864381e-15
>> multAB
|x-y| = 5.324443e-16
```

Explain these results.

HINT: what is the relationship of the matrices $\mathbf{A}$ and $\mathbf{B}$? A fully rigorous proof is not required here.

Problem 2 Modified Newton method [25 points]

For the solution of the non-linear system of equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ (with $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$), the following iterative method can be used:

$$\begin{aligned} \mathbf{y}^{(k)} &= \mathbf{x}^{(k)} + D\mathbf{F}(\mathbf{x}^{(k)})^{-1}\,\mathbf{F}(\mathbf{x}^{(k)}) , \\ \mathbf{x}^{(k+1)} &= \mathbf{y}^{(k)} - D\mathbf{F}(\mathbf{x}^{(k)})^{-1}\,\mathbf{F}(\mathbf{y}^{(k)}) , \end{aligned} \tag{2}$$

where $D\mathbf{F}(\mathbf{x}) \in \mathbb{R}^{n,n}$ is the Jacobian matrix of $\mathbf{F}$ evaluated in the point $\mathbf{x}$.

(2a) [3 points] Show that $\mathbf{x}^{(k)} = \mathbf{x}^{(0)}$ for every $k \in \mathbb{N}$, if $\mathbf{F}(\mathbf{x}^{(0)}) = \mathbf{0}$ and $D\mathbf{F}(\mathbf{x}^{(0)})$ is regular.

(2b) [3 points] Implement a Matlab function

```
function x1 = ModNewtStep( x0, F, DF )
```

that computes a step of the method (2) for a *scalar* function $\mathbf{F}$, that is, for the case $n = 1$.

Here, `F` is a handle to the function $F : \mathbb{R} \mapsto \mathbb{R}$ and `DF` a handle to its derivative $F' : \mathbb{R} \mapsto \mathbb{R}$.

(2c) [9 points] What is the order of convergence of the method?
To investigate it, write a Matlab routine

```
function ModNewtOrder
```

that:

- uses the function `ModNewtStep` from subtask (2b) in order to apply (2) to the following scalar equation
$$\arctan(x) - 0.123 = 0 ;$$
- determines empirically the order of convergence, in the sense of Definition 4.1.14 of the course slides;
- plots the error committed by the method against the number of iterations used, and saves the picture in `ModNewtOrder.eps` .

Use $x_0 = 5$ as initial guess.

HINT 1: the "exact" solution is $x = 0.123624065869274$.
HINT 2: remember that $\arctan'(x) = \frac{1}{1+x^2}$.

(2d) [7 points] Write a Matlab function

```
function x = ModNewtSys( A, c, tol )
```

that provides an efficient implementation of the method (2) for the non-linear system

$$\mathbf{F}(\mathbf{x}) := \mathbf{A}\mathbf{x} + \begin{pmatrix} c_1 e^{x_1} \\ \vdots \\ c_n e^{x_n} \end{pmatrix} = \mathbf{0} ,$$

where $\mathbf{A} \in \mathbb{R}^{n,n}$ is symmetric positive definite, and $c_i \geq 0$, $i = 1, \ldots, n$. Stop the iteration when the Euclidean norm of the increment $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ relative to the norm of $\mathbf{x}^{(k+1)}$ is smaller than the tolerance passed in `tol`.

(2e) **[3 points]** In order to apply the iteration (2), two linear systems have to be solved. What do these systems have in common? How can this fact be used to speed up an implementation of the method for large n?

Problem 3 Quadrature plots [16 points]

We consider three different functions on the interval $I = [0, 1]$:

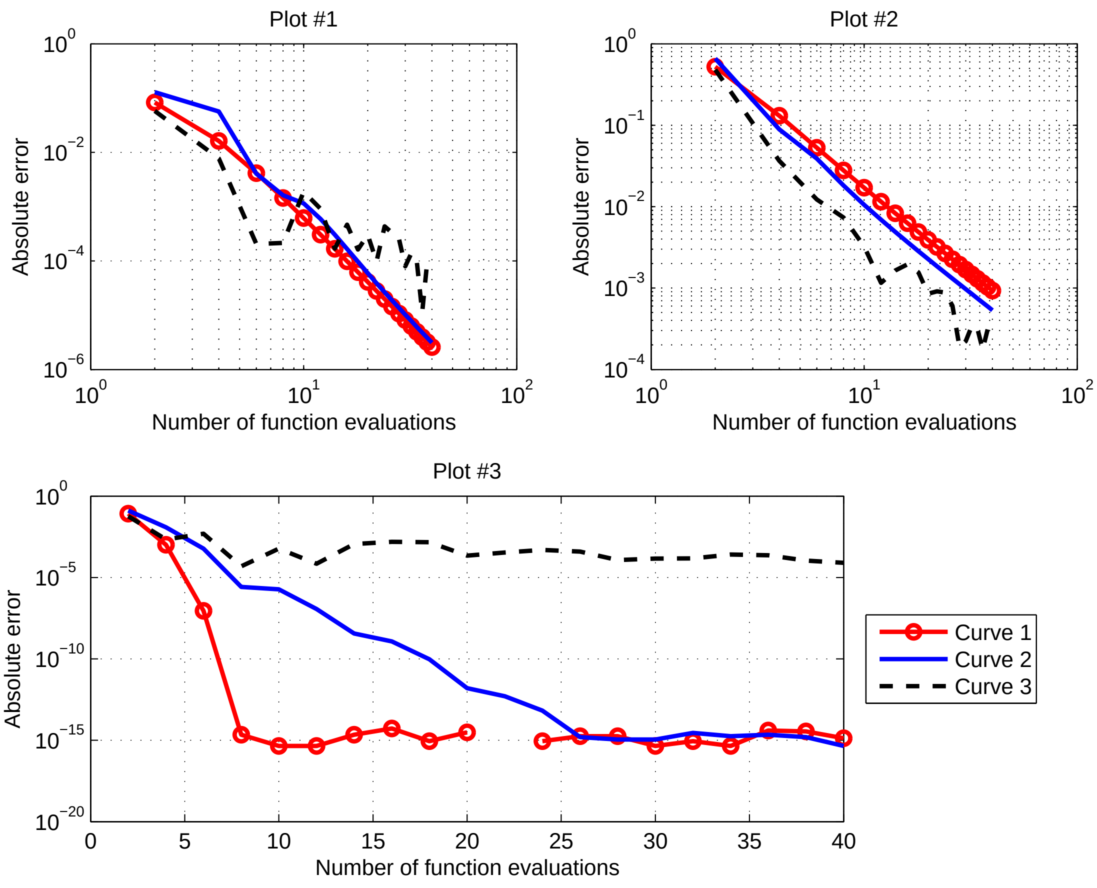
$$\begin{aligned}
&\text{function A:} \quad f_A \in C^\infty(I)\,, \quad f_A \notin \mathcal{P}_k \; \forall\, k \in \mathbb{N}\,;\\
&\text{function B:} \quad f_B \in C^0(I)\,, \quad f_B \notin C^1(I)\,;\\
&\text{function C:} \quad f_C \in \mathcal{P}_{12}\,,
\end{aligned}$$

where $\mathcal{P}_k$ is the space of the polynomials of degree at most k defined on I. The following quadrature rules are applied to these functions:

- quadrature rule A, global Gauss quadrature;
- quadrature rule B, composite trapezoidal rule;
- quadrature rule C, composite 2-point Gauss quadrature.

The corresponding absolute values of the quadrature errors are plotted against the number of function evaluations in Figure 1. Notice that only the quadrature errors obtained with an even number of function evaluations are shown. You can see the same figure in the files `ExQuadPlot.jpg` / `ExQuadPlot.eps`.

Figure 1: Problem 3, quadrature convergence plots for different functions and different rules.



(3a) [8 points] Match the three plots (plot #1, #2 and #3) with the three quadrature rules (quadrature rule A, B, and C). Justify your answer.

HINT: notice the different axis scales in the plots.

(3b) [8 points] The quadrature error curves for a particular function f_A, f_B and f_C are plotted in the same style (curve 1 as red line with small circles, curve 2 means the blue solid line, curve 3 is the black dashed line). Which curve corresponds to which function (f_A, f_B, f_C)? Justify your answer.

Problem 4 System of ODEs [13 points]

Consider the initial value problem

$$\begin{aligned} 2\ddot{u}_1 - \ddot{u}_2 &= u_1(u_2+u_1)\ , \\ -\ddot{u}_{i-1} + 2\ddot{u}_i - \ddot{u}_{i+1} &= u_i(u_{i-1}+u_{i+1}) \qquad i = 2,\dots,n-1\ , \\ 2\ddot{u}_n - \ddot{u}_{n-1} &= u_n(u_n+u_{n-1})\ , \\ u_i(0) &= u_{0,i} \qquad i=1,\dots,n\ , \\ \dot{u}_i(0) &= v_{0,i} \qquad i=1,\dots,n\ , \end{aligned} \tag{3}$$

in the time interval $[0, T]$.

(4a) [9 points] Implement a Matlab function

```
function [Tout, Uout] = MyOde ( T, u0, v0 )
```

that uses Matlab 's `ode45` numerical integrator to solve the IVP described above. Here, `T`> 0 is a scalar, and `u0` and `v0` are n-dimensional column vectors. The return value `Tout` should provide the vector of times as returned by `ode45`, whereas `Uout` is a matrix whose columns contain approximations of $(u_1(t), \dots, u_n(t))$ at those times.

HINT: in case this helps you, you might rephrase (on paper) the initial value problem (3) in a different way.

(4b) [4 points] Create a routine

```
function PlotOde
```

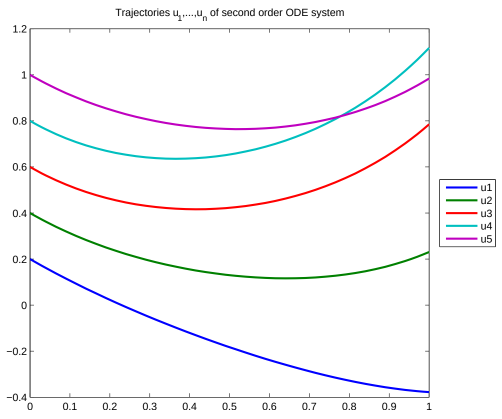
that plots the components of the solution obtained in the previous subtask with

$$n = 5\ , \qquad u_{0,i} = i/n\ , \qquad v_{0,i} = -1\ ,$$

in the interval $[0, T] = [0, 1]$. Save the picture as `PlotOde.eps`.

HINT: a reference solution plot can be seen in Figure 2

Figure 2: Problem 4, reference solution plot.



Problem 5 Least squares fitting of a quadratic functional [20 points]

We want to determine the characteristic matrix $\mathbf{M} \in \mathbb{R}^{2,2}$, $\mathbf{M} = \mathbf{M}^T$, of the homogeneous quadratic functional

$$\Phi_{\mathbf{M}}(\mathbf{z}) := \mathbf{z}^T \mathbf{M} \mathbf{z} \;, \quad \mathbf{z} \in \mathbb{R}^2 \;, \tag{4}$$

from given values y_i, $i = 1, \ldots, N$, $N \in \mathbb{N}$, $N > 3$, "measured" at points $\mathbf{z}_i$, $i = 1, \ldots, N$. This can be done by solving the least squares problem

$$\mathbf{M} = \operatorname*{argmin}_{\mathbf{P} \in \mathbb{R}^{2,2},\, \mathbf{P} = \mathbf{P}^T} \sum_{i=1}^{N} \left(\Phi_{\mathbf{P}}(\mathbf{z}_i) - y_i \right)^2 \;. \tag{5}$$

However, one may insist on getting a positive definite matrix $\mathbf{M}$, which is not guaranteed in (5). Therefore we may also consider the alternative least squares problem

$$\mathbf{C} = \operatorname*{argmin}_{\mathbf{R} \in UT(2)} \sum_{i=1}^{N} \left(\Phi_{\mathbf{R}^T \mathbf{R}}(\mathbf{z}_i) - y_i \right)^2 \;, \tag{6}$$

where $UT(2)$ denotes the space of upper triangular 2×2 matrices. Then we recover $\mathbf{M}$ as $\mathbf{C}^T \mathbf{C}$.

(5a) [4 points] Show that (5) is a linear least squares problem of the form

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \to \min \;.$$

Determine the matrix $\mathbf{A}$ and the vector $\mathbf{b}$. What is the concrete vector of unknowns $\mathbf{x}$ in (5)?

(5b) [4 points] Write a Matlab function

```
function M = QuadFit( Z, y ),
```

which takes the points $\mathbf{z}_i$ (as column of the $2 \times N$-matrix `Z`) and the values y_i (as components of the N-vector `y`) as arguments and returns the solution of (5) in the 2×2 matrix `M`.

(5c) [4 points] Show that (6) is a non-linear least squares problem

$$\|\mathbf{F}(\mathbf{x})\|_2^2 \to \min$$

for a suitable function $\mathbf{F} : UT(2) \mapsto \mathbb{R}^N$. Give a detailed formula for $\mathbf{F}$.

(5d) [8 points] Implement a Matlab function

```
function R = NlQuadFit( X, y, R0 ),
```

that uses the Gauss–Newton method with initial guess $\mathbf{R}_0 \in UT(2)$ (passed in `R0`) to solve (6). The meaning of the other arguments is the same as for sub-problem (5b).

HINT 1: represent the upper triangular matrix $\mathbf{R}$ with the vector $\mathbf{r} = (R_{1,1}, R_{1,2}, R_{2,2})^T$.
HINT 2: you can test the correctness of your code by choosing a matrix of random values for `Z` and a fixed s.p.d. matrix $\mathbf{M}$, and creating the vector `y` as $y_i := \Phi_{\mathbf{M}}(\mathbf{z}_i)$.