R. Hiptmair
A. Moiola

Fall term 2010

Numerical Methods for CSE

ETH Zürich
D-MATH

Examination

February 10th, 2011

Instructions.

Duration of examination: 180 minutes. **Total points: 65.**

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.

Problem 1 and Problem 2 have to be solved on paper, no Matlab files for these problems will be considered in the correction.

In Problem 3, Problem 4 and Problem 5 **only the Matlab files that are requested in the problem statement will be corrected**. The theoretical parts of these problems have to be solved on paper.

In Problem 3 a few Matlab "p-files" are provided. You can use them to skip a subtask and proceed with the solution of the following ones. They are not available with Octave.

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!

Problem 1 Advantages of conjugate gradient method [8 points]

Give at least four circumstances when the iterative conjugate gradient (CG) solver may be preferred to a direct solver based on Cholesky decomposition for the symmetric, positive definite linear system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{n,n}$, $\mathbf{b} \in \mathbb{R}^n$.

Problem 2 Cholesky and QR decomposition [14 points]

(2a) [4 points] Show that, for every matrix $\mathbf{A} \in \mathbb{R}^{m,n}$ such that $\mathrm{rank}(\mathbf{A}) = n$, the product matrix $\mathbf{A}^T\mathbf{A}$ admits a Cholesky decomposition.

(2b) [6 points] The following Matlab function is given:

```
function [Q,R] = altqr(A)
R = chol(A'*A);
Q = (R'\A')';
```

Prove that, for every matrix $\mathbf{A}$ as in subtask (2a), `altqr(A)` and `qr(A,0)` will produce the same output, if there are no roundoff errors.

(2c) [4 points] Let ϵ denote the machine precision (`eps` in Matlab). Why does the command `altqr(A)` fail for

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ \frac{1}{2}\sqrt{\epsilon} & 0 \\ 0 & \frac{1}{2}\sqrt{\epsilon} \end{pmatrix} \quad ?$$

Problem 3 Quadrature [16 points]

Given a smooth, odd function $f : [-1, 1] \to \mathbb{R}$, consider the integral

$$I := \int_{-1}^{1} \arcsin(t)\ f(t)\,\mathrm{d}t. \tag{1}$$

We want to approximate this integral using global Gauss quadrature. The nodes (vector `x`) and the weights (vector `w`) of n-point Gaussian quadrature on $[-1, 1]$ can be computed using the provided Matlab routine `[x,w]=gaussquad(n)` (in the file `gaussquad.m`).

(3a) [4 points] Write a Matlab routine

`function GaussConv(f_hd)`

that produces an appropriate convergence plot of the quadrature error versus the number $n = 1, \ldots, 50$ of quadrature points. Here, `f_hd` is a handle to the function f.

Save your convergence plot for $f(t) = \sinh(t)$ as `GaussConv.eps`.

HINT 1: use the Matlab command `quad` with tolerance `eps` to compute a reference value of the integral.

HINT 2: if you cannot implement the quadrature formula, you can resort to the Matlab function

`function I = GaussArcSin(f_hd, n)`

provided in implemented `GaussArcSin.p` that computes n-points Gauss quadrature for the integral (1). Again `f_hd` is a function handle to f.

(3b) [1 point] Which kind of convergence do you observe?

(3c) [3 points] Transform the integral (1) into an equivalent one with a suitable change of variable so that Gauss quadrature applied to the transformed integral converges much faster.

(3d) [4 points] Now, write a Matlab routine

`function GaussConvCV(f_hd)`

which plots the quadrature error versus the number $n = 1, \ldots, 50$ of quadrature points for the integral obtained in the previous subtask.

Again, choose $f(t) = \sinh(t)$ and save your convergence plot as `GaussConvCV.eps`.

HINT: In case you could not find the transformation, you may rely on the function

`function I = GaussArcSinCV(f_hd, n)`

implemented in `GaussArcSinCV.p` that applies n-points Gauss quadrature to the transformed problem.

(3e) [4 points] Which kind of convergence is achieved? Explain the difference between the results obtained in subtasks (3a) and (3d).

Problem 4 Exponential integrator [16 points]

A step with size h of the so-called *exponential Euler* single step method for the ODE $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$ with continuously differentiable $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ reads

$$\mathbf{y}_1 = \mathbf{y}_0 + h\, \varphi\big(h\mathbf{D}\mathbf{f}(\mathbf{y}_0)\big)\, \mathbf{f}(\mathbf{y}_0), \tag{2}$$

where $\mathbf{D}\mathbf{f}(\mathbf{y}) \in \mathbb{R}^{d,d}$ is the Jacobian of $\mathbf{f}$ at $\mathbf{y} \in \mathbb{R}^d$, and the matrix function $\varphi : \mathbb{R}^{d,d} \to \mathbb{R}^{d,d}$ is defined as $\varphi(\mathbf{Z}) = (\exp(\mathbf{Z}) - \mathbf{Id})\, \mathbf{Z}^{-1}$. Here $\exp(\mathbf{Z})$ is the matrix exponential of $\mathbf{Z}$, a special function $\exp : \mathbb{R}^{d,d} \to \mathbb{R}^{d,d}$.

The function φ is implemented in the provided file `phim.m`.

(4a) [2 points] Show that the exponential Euler single step method defined in (2) solves the initial value problem

$$\dot{\mathbf{y}} = \mathbf{A}\,\mathbf{y}\,, \quad \mathbf{y}(0) = \mathbf{y}_0 \in \mathbb{R}^d\,, \qquad \mathbf{A} \in \mathbb{R}^{d,d}\,,$$

exactly.

HINT: the solution of the IVP is $\mathbf{y}(t) = \exp(\mathbf{A}t)\mathbf{y}_0$. You may assume that $\mathbf{A}$ is regular.

(4b) [2 points] Write a Matlab function

```
function y1 = ExpEulStep(y0,f,df,h)
```

that implements (2). Here `f` and `df` are handles to the ODE right-hand side function $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ and its Jacobian, respectively.

HINT: use the supplied function `phim.m`.

(4c) [5 points] What is the order of the single step method (2)? To investigate it, write a Matlab routine

```
function ExpIntOrder
```

that applies it to the scalar logistic ODE

$$\dot{y} = y\,(1-y)\,, \quad y(0) = 0.1\,,$$

in the time interval $[0, 1]$ and plots the error at the final time against the stepsize h.

Save the convergence plot in `ExpIntOrder.eps`.

HINT: the exact solution is

$$y(t) = \frac{y(0)}{y(0) + \big(1 - y(0)\big)e^{-t}}.$$

(4d) [4 points] Write a Matlab function

```
function yOut = ExpIntSys(n,N,T)
```

that applies N uniform steps of the exponential Euler method to solve the initial value problem

$$\begin{aligned} \dot{\mathbf{y}} &= -\mathbf{A}\,\mathbf{y} + \mathbf{g}(y), \\ \mathbf{y}_j(0) &:= j/d, && j = 1,\ldots,d, \\ \mathbf{g} : \mathbb{R}^d \to \mathbb{R}^d \qquad & \big(\mathbf{g}(\mathbf{y})\big)_j := (\mathbf{y}_j)^3, && j = 1,\ldots,d, \end{aligned}$$

over $[0,T]$, where the matrix $\mathbf{A} \in \mathbb{R}^{d,d}$ is obtained as

$$\texttt{A} = \texttt{gallery}(\texttt{'poisson'}, \texttt{n});$$

and $d = \texttt{n}^2$.

The output variable `yOut` should be an $(N+1) \times d$-matrix with an approximation of $\mathbf{y}((j-1)\frac{T}{N})$ in the j-th row.

Choose $T = 1$, $\texttt{n} = 5$ ($d = 25$), $N = 100$, and plot the computed state value at the final time $(\mathbf{y}(T))_j$ against $j = 1,\ldots,d$; save the plot as `ExpIntSys.eps`.

HINT: for *Octave* users, the Poisson matrix $\mathbf{A}$ can be generated by the following commands:

```
B = spdiags([-ones(n,1),2*ones(n,1),-ones(n,1)],[-1,0,1],n,n);
A = kron(B,speye(n))+kron(speye(n),B);
```

(4e) [3 points] Write a Matlab routine

`function ExpIntErr`

that compares the results obtained in the previous subtask with the ones obtained using `ode45` with default tolerances. Compute the relative error in 2-norm at time $T = 1$ with $n = 5$ ($d = 25$) and $N = 100$ steps of the exponential Euler method, when the result from `ode45` is accepted as "exact solution".

Problem 5 Matrix least squares in Frobenius norm [11 points]

Consider the following problem:

$$\text{given } \mathbf{z} \in \mathbb{R}^n, \mathbf{g} \in \mathbb{R}^n, \quad \text{find} \quad \mathbf{M}^* = \operatorname*{argmin}_{\mathbf{M}\in\mathbb{R}^{n,n},\, \mathbf{Mz}=\mathbf{g}} \|\mathbf{M}\|_F \ , \tag{3}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix.

(5a) [6 points] Reformulate the problem as an equivalent standard constrained least squares problem

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^N,\, \mathbf{Cx}=\mathbf{d}} \|\mathbf{Ax}-\mathbf{b}\|_2 \ ,$$

for suitable matrices $\mathbf{A}$, $\mathbf{C}$ and vectors $\mathbf{b}$ and $\mathbf{d}$. These matrices and vectors have to be specified based on $\mathbf{z}$ and $\mathbf{g}$.

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

`function M = MinFrob(z,g)`

that computes the solution of the minimization problem (3) given the vectors $\mathbf{z}, \mathbf{g} \in \mathbb{R}^n$.

HINT: the Matlab command `kron` comes handy.