Prof. R. Hiptmair

Makeup Examination

August 19th, 2010

Problem 1: Kronecker product (30 points)

We consider the MATLAB expression

$$\texttt{y = kron(A,B)*x}, \tag{1}$$

for $n \times n$ dense real matrices stored in `A` and `B`, and a column vector `x` of length n^2, $n \in \mathbb{N}$.

a) **(5 points)** What is the *asymptotic* complexity of the evaluation of this MATLAB expression in terms of the problem size parameter n?

b) **(15 points)** Devise an efficient MATLAB function

```
function y = kronmult(A,B,x)
```

that is algebraically equivalent to the expression (1) above, but enjoys a better asymptotic complexity.

c) **(5 points)** What is the asymptotic complexity of your implementation of `kronmult` in terms of the problem size parameter n? Explain your answer.

d) **(5 points)** What is the asymptotic (in terms of n) complexity of your version of `kronmult`, if `A` and `B` contain sparse $n \times n$ *diagonal* matrices.

Problem 2: Linear least squares problem (20 points)

Input data are two vectors $\mathbf{z}, \mathbf{c} \in \mathbb{R}^n$, $n \in \mathbb{N}$, of measured data. You are expected to compute the two numbers $\alpha^*, \beta^* \in \mathbb{R}$ such that

$$(\alpha^*, \beta^*) = \underset{\alpha,\beta \in \mathbb{R}}{\operatorname{argmin}} \left\| \mathbf{T}_{\alpha,\beta}\mathbf{z} - \mathbf{c} \right\|_2 , \tag{2}$$

with tridiagonal matrix

$$\mathbf{T}_{\alpha,\beta} = \begin{pmatrix} \alpha & \beta & 0 & \dots & 0 \\ \beta & \alpha & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \alpha & \beta \\ 0 & \dots & 0 & \beta & \alpha \end{pmatrix} \in \mathbb{R}^{n,n} .$$

a) **(10 points)** Reformulate (2) as a linear least squares problem in the standard form

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$$

with a suitable matrix $\mathbf{A} \in \mathbb{R}^{m,k}$, $m, k \in \mathbb{N}$, and vectors $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{x} \in \mathbb{R}^k$.

b) **(10 points)** Write a MATLAB function

```
function [alpha,beta] = lsqest(z,c)
```

that computes the values of α^* and β^* according to (2), when `z`, `c` pass the vectors $\mathbf{z}$ and $\mathbf{c}$.

Hint. You may use MATLAB's \-operator for solving a linear least squares problem. For $\mathbf{z} = (1, 2, \dots, 10)^T$, $\mathbf{c} = (10, 9, 8, \dots, 1)^T$ your code should give $\alpha^* \approx -0.4211$, $\beta^* \approx 0.5789$.

Problem 3: Speed of convergence of CG (20 points)

The following is known about the matrix $\mathbf{A} \in \mathbb{R}^{n,n}$:

- $(\mathbf{A})_{i,i} = 5$ for all $1 \le i \le n$,
- $|(\mathbf{A})_{i,j}| \le 1$ for all $1 \le i < j \le n$,
- $\mathbf{A}$ is symmetric and positive definite (s.p.d.),
- each row of $\mathbf{A}$ has at most four non-zero entries.

We consider a linear system of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \in \mathbb{R}^n \; . \tag{3}$$

a) **(7 points)** Appeal to the Gershgorin circle theorem (Lemma 5.1.3 in the lecture material) to find bounds for the largest and smallest eigenvalue of $\mathbf{A}$.

b) **(6 points)** The (non-preconditioned) conjugate gradient method (CG) is applied to solve (3). Give a reasonably sharp bound for the number of CG-steps it takes to reduce the $\mathbf{A}$-norm (energy norm) of the error of the iterates by a factor of 10^6.

c) **(7 points)** Give a general bound (in terms of n and accurate in leading order) of the number of elementary operations (additions/subtractions and multiplications/divisions) that have to be executed in each CG-step.

Problem 4: "Quadrature of the circle" (40 points)

Given a smooth function $f : [-1, 1] \mapsto \mathbb{R}$, Gaussian quadrature shall be used to approximate the integral

$$I(f) := \int_{-1}^{1} \sqrt{1 - t^2} f(t) \, \mathrm{d}t \; . \tag{4}$$

A MATLAB routine `[x,w]=gaussquad(n)` that computes the nodes (vector `x`) and weights (vector `w`) of n-point Gaussian quadrature on $[-1, 1]$ is supplied in the file `gaussquad.m`.

a) **(12 points)** For $f \equiv 1$ the integral value is $\pi/2$, half of the area of the unit disk. Write a MATLAB routine

`function plotgausserr`

that creates a log-log plot of the quadrature error versus the number $n \in \{1, \ldots, 30\}$ of quadrature points, when Gaussian quadrature on $[-1, 1]$ is used to evaluate the integral for $f = 1$ right away. What kind of convergence do you observe?

Hint: The requested error plot may look like that depicted in Figure 1.

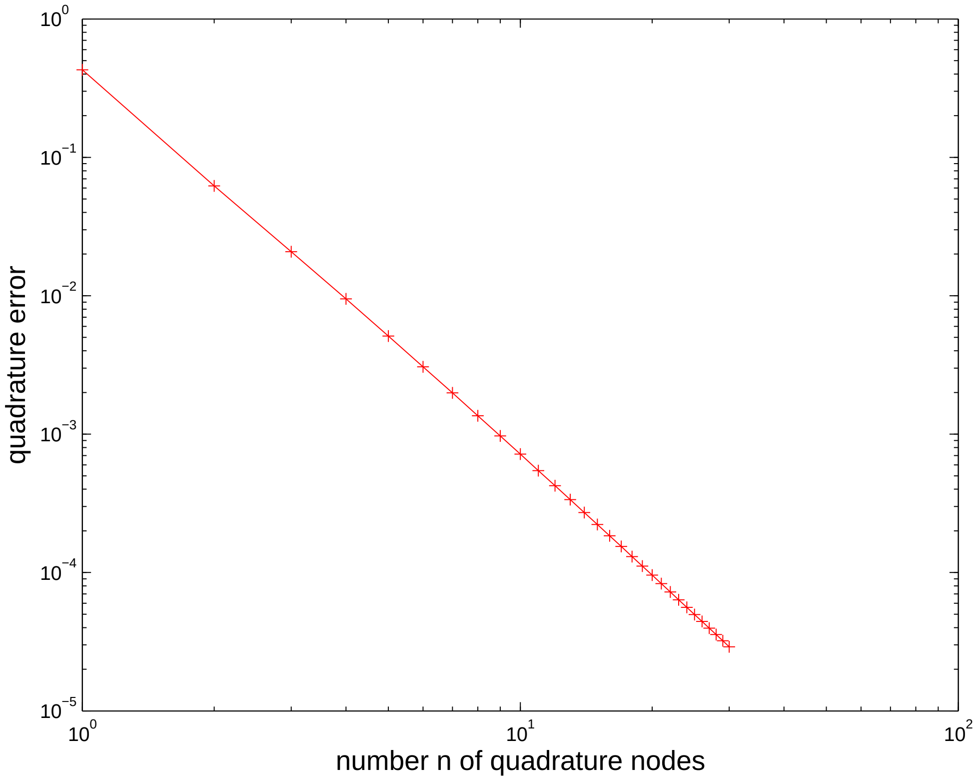

Figure 1: Quadrature error for Gaussian quadrature applied to (4) with $f \equiv 1$.

b) **(8 points)** The file `circquad.m` contains the following MATLAB function

```
function I = circquad(f,n)
% Numerical quadrature for \int_{-1}^{1} \sqrt{1-t^2} f(t) dt
g = @(s) 2*s.^2.*sqrt(2-s.^2).*(f(s.^2-1)+f(1-s.^2));
[x,w]=gaussquad(n)
I = 0.5*dot(w,g(0.5*(x+1)));
```

Write a MATLAB function

`function plotIerr`

that creates a lin-log plot of the quadrature error of `cricquad` versus the number n of quadrature points for $f = 1$ and $n \in \{1, \ldots, 10\}$. What kind of convergence do you observe?

Hint: Your plot may look like that displayed in Figure 2.

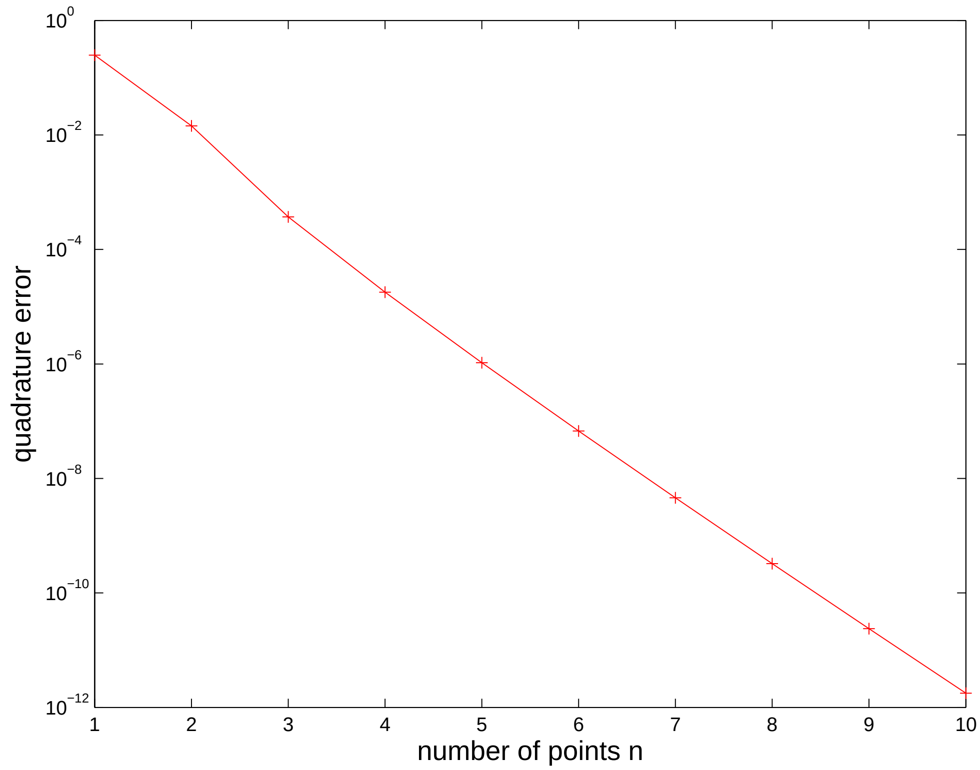

Figure 2: Quadrature error for `circquad`

c) **(10 points)** Obviously, `circquad` applies Gaussian quadrature to the integral

$$\int_0^1 2x^2\sqrt{2-x^2}\,\left(f(x^2-1)+f(1-x^2)\right)\,\mathrm{d}x\ . \tag{5}$$

Show in detail that (4) and (5) give the same value for every f.

d) **(10 points)** Explain why `circquad` achieves a much better accuracy with the same number of f-evaluations compared to straightforward Gaussian quadrature applied to (4).

Problem 5: SVD of a circulant matrix (25 points)

The circulant matrix

$$\mathbf{C} := \begin{pmatrix} u_0 & u_1 & u_2 & \cdots & & & \cdots & u_{n-1} \\ u_{n-1} & u_0 & \ddots & & & & & u_{n-2} \\ u_{n-2} & \ddots & \ddots & & & & & \vdots \\ \vdots & & & & & & & \\ & & & & & & & \\ \vdots & & & & & & \ddots & \vdots \\ u_2 & & & & & \ddots & \ddots & u_1 \\ u_1 & u_2 & \dots & & & \cdots & u_{n-1} & u_0 \end{pmatrix} \in \mathbb{R}^{n,n}$$

is defined by the generating vector $\mathbf{u} := (u_0,\dots,u_{n-1})^T \in \mathbb{R}^n$.

a) **(15 points)** Implement an efficient MATLAB function

```
s = svcirc(u)
```

that computes the sorted singular values of the circulant matrix $\mathbf{C}$, when supplied with the generating vector $\mathbf{u}$.

Hint: Remember that the columns of the Fourier matrix provide a complete orthogonal basis of eigenvectors for any circulant matrix.

Hint: `sort(x,'descend')` sorts the vector `x` in descending order.

b) **(5 points)** Write a MATLAB test routine

```
function svcirctest(u)
```

that uses the built-in MATLAB function `svd()` to validate the correctness of your implementation of `svcirc` by plotting the absolute error of the singular values over their index for a random generating vector $u \in \mathbb{R}^{10}$.

Hint: A circulant matrix can be built by the MATLAB command `gallery('circul',u)`.

c) **(5 points)** What is the asymptotic complexity of `svcirc` in terms of the problem size parameter n?

Problem 6: Solving an implicit ODE (40 points)

For a Lipschitz continuous function $g : [0, \infty] \mapsto [0, \infty]$, we consider the scalar implicit initial value problem

$$\dot{y}e^{\dot{y}} = g(y) \quad , \quad y(0) = y_0 > 0 \ . \tag{6}$$

a) **(20 points)** Write a MATLAB function

```
function fy = impoderhs(g,y)
```

that uses Newton's method to evaluate the right hand side f of the ODE $\dot{y} = f(y)$ that is equivalent to the ODE of (6).

Use $\log(g(y))$ as initial guess and stop the iteration, once the relative size of the Newton correction is below 10^{-6}.

Hint: A (hidden) reference implementation of `impoderhs` is given in MATLAB function `impoderhs_ref` (in the file `impoderhs_ref.p`, which serves exactly the same purpose an `.m`-file, but conceals the source code).

b) **(10 points)** Design a MATLAB function

```
function [t,y] = odeimpl(g,T,y0)
```

that uses the MATLAB standard integrator `ode45` with absolute tolerance 10^{-7} and relative tolerance 10^{-5} to solve (6) on $[0, T]$. The return values are those of `ode45`.

c) **(10 points)** Write a MATLAB function

```
function plotivpsol
```

that solves the initial value problem (6) for the concrete $g(y) = \frac{y}{1+y^2}$ and $y_0 = \frac{1}{2}$ over the time interval $[0, 4]$. Plot both $y(t)$ and $\dot{y}(t)$ in one chart.

Hint: Your solution should look like the plot shown in Figure 3.

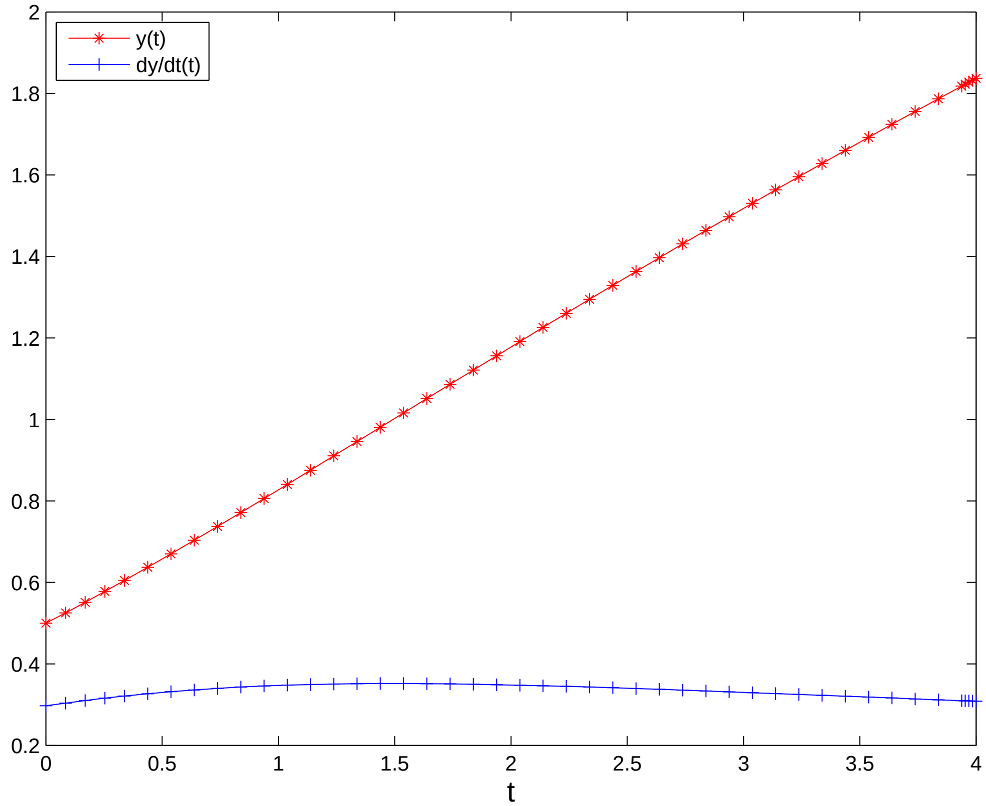

Figure 3: Solution of IVP of Problem 6(c)