R. Hiptmair
J. Šukys

Numerical Methods for CSE

ETH Zürich
D-MATH

Examination – Solutions

January 31st, 2012

Total points: 180 = 35 + 15 + 35 + 55 + 40.

Problem 1. Structured linear system

[35 points]

(1a) [2 points] $a_j \neq 0,\ j = 1, \dots, n$.

(1b) [8 points] Function:

```
function  A = AstructMat(a)
n = length(a);
A = spdiags(a(:),0,n,n) * spdiags(ones(n,1)*(1:n),(0:n-1),n,n);
```

(1c) [3 points] $O(n^2)$ since $\mathbf{A}$ is triangular.

(1d) [22 points] $\mathbf{A} = \mathbf{DE} = \mathrm{diag}(\mathbf{a}) \cdot \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ 0 & 1 & 2 & \cdots & (n-1) \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & 1 & 2 \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}$.

With Matlab it's easy to find out that

$$\mathbf{A}^{-1} = \mathbf{E}^{-1}\mathbf{D}^{-1} = \begin{pmatrix} 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ & & \ddots & 1 & -2 & 1 \\ \vdots & & & 0 & 1 & -2 \\ 0 & \cdots & & \cdots & 0 & 1 \end{pmatrix} \cdot \mathrm{diag}(a_1^{-1}, \dots, a_n^{-1}).$$

Thus the solution of the LSE is written as a (tridagonal times diagonal times vector) product

$$\mathbf{x} = \mathbf{E}^{-1}\mathbf{D}^{-1}\mathbf{b}, \quad \text{i.e.,} \quad x_j = b_j/a_j - 2b_{j+1}/a_{j+1} + b_{j+2}/a_{j+2}$$

(with the convention $b_{n+1}/a_{n+1} = b_{n+2}/b_{n+2} = 0$):

```
function x = AstructLSE(a,b)
if a
    c = b./a;
    x = c - 2 * [c(2:end);0] + [c(3:end);0;0];
else
    error('The matrix is singular!');
end

```

```
% check with:
% n = 100; a = randn(n,1); b = randn(n,1);
% x = AstructLSE(a,b); norm(b-AstructMat(a)*x)
```

Not requested: it's also easy to prove that the matrix $\mathbf{E}^{-1}$ defined above is indeed the inverse of $\mathbf{E}$:

$$(\mathbf{E})_{i,j} = \begin{cases} j+1-i & j \geq i, \\ 0 & j < i, \end{cases}$$

$$(\mathbf{E}^{-1}\mathbf{E})_{i,j} = \begin{cases} (j+1-i)+(j+1-(i+2))-2(j+1-(i+1)) = 0 & j > i,\ i < n-1; \\ (j+1-i)-2(j+1-(i+1)) = 1 & i = j,\ i < n-1; \\ 1 & i = j \in \{n, n-1\}; \\ 0 & i = n-1,\ j = n; \\ 0 \quad \text{(product of upp. triang. matrices)} & j < i. \end{cases} \tag{1}$$

Problem 2. Best rank-k approximation

[15 points]

(2a) [13 points] Let $\mathbf{A} = \mathbf{U\Sigma V}^T$ be the svd decomposition of $\mathbf{A}$, the three matrices are square and regular, and $\mathbf{\Sigma} = \mathrm{diag}(\sigma_1, \dots, \sigma_n)$. Then $\mathbf{A}^{-1} = \mathbf{V}^{-T}\mathbf{\Sigma}^{-1}\mathbf{U}^{-1} = \mathbf{V\Sigma}^{-1}\mathbf{U}^T$ is a svd decomposition of $\mathbf{A}^{-1}$ with the singular values in reverse order: the largest values in the diagonal matrix $\mathbf{\Sigma}^{-1}$ lie the in the last entries.

Two possible solutions:

$$\mathbf{B} = \operatorname*{argmin}_{\mathbf{M}\in\mathbb{R}^{n,n},\ \mathrm{rank}(\mathbf{M})=k} \left\|\mathbf{A}^{-1} - \mathbf{M}\right\|_2^2 \quad = \quad \mathbf{V}\mathbf{S}_{k,+}\mathbf{U}^T \quad = \quad \mathbf{V}_k\mathbf{S}_{k,-}\mathbf{U}_k^T,$$

where

$$\mathbf{S}_{k,+} := \mathrm{diag}(0, \dots, 0, \sigma_{n+1-k}^{-1}, \dots, \sigma_n^{-1}) \in \mathbb{R}^{n,n}$$

and

$$\mathbf{S}_{k,-} := \mathrm{diag}(\sigma_{n+1-k}^{-1}, \dots, \sigma_n^{-1}) \in \mathbb{R}^{k,k}\ ,$$

$$\mathbf{U}_k := \mathbf{U}_{:,n+1-k:n} \in \mathbb{R}^{n,k}\ , \qquad \mathbf{V}_k = \mathbf{V}_{:,n+1-k:n} \in \mathbb{R}^{n,k}\ .$$

Function:

```
function B = kRankInv(A,k)
n = size(A,1);
[U,S,V] = svd(A);
%     best version:
B = V(:,n+1-k:n)*diag(diag(S(n+1-k:n,n+1-k:n)).^(-1))*U(:,n+1-k:n)';
%     other version:
%B = V * diag([zeros(n-k,1); diag(S(n+1-k:n, n+1-k:n)).^(-1)]) *U';

%     Check code by comparing:
% ( n=10; k=80; )
% A = full(gallery('poisson',n));   B = kRankInv(A,k);   v=eig(A);
% [norm(B-inv(A)),      1/v(k+1)]
```

(2b) [2 points] $\mathbf{B}$ is not unique because the SVD decomposition is not uniquely defined for matrices with repeated singular values. E.g., if $\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, then we can have $\mathbf{B}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ or $\mathbf{B}_2 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$.

Non-uniqueness of $\mathbf{U}$ and $\mathbf{V}$ is **not** sufficient.

Problem 3. Solving an eigenvalue problem with Newton's method

[35 points]

(3a) [10 points]

$$\mathbf{DF}(\mathbf{x}) = \begin{pmatrix} \mathbf{A} - \lambda\mathbf{I} & -\mathbf{x} \\ -\mathbf{x}^\top & 0 \end{pmatrix}.$$

(3b) [5 points]

$$\begin{pmatrix} x^{(k+1)} \\ \lambda^{(k+1)} \end{pmatrix} = \begin{pmatrix} x^{(k)} \\ \lambda^{(k)} \end{pmatrix} - \begin{pmatrix} \mathbf{A} - \lambda\mathbf{I} & -\mathbf{x}^{(k)} \\ -(\mathbf{x}^{(k)})^\top & 0 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{A}\mathbf{x}^{(k)} - \lambda^{(k)}\mathbf{x}^{(k)} \\ 1 - \frac{1}{2}\left\|\mathbf{x}^{(k)}\right\|^2 \end{pmatrix}$$

(3c) [20 points] Function:

```
function [eigvec, eigval] = eignewton (A, x, rtol, atol)

MAXIT = 10000;
x = [x; x.'*A*x/(x.'*x)];
F = @(x) [(A - x(end)*eye(size(A)))*x(1:end-1); 1 - 0.5*norm(x)^2];
DF = @(x) [A - x(end)*eye(size(A)), -x(1:end-1); -x(1:end-1).', 0];

for i = 1:MAXIT
        s = DF(x) \ F(x);
        x = x-s;
        if ((norm(s) < rtol *norm(x)) || (norm(s) < atol))
                break;
        end
end

eigvec = x(1:end-1);
eigval = x(end);
```

Problem 4. Matrix ODE

[55 points]

(4a) [5 points] The general form of the Runge-Kutta schemes is the following (for $s \in \mathbb{N}$):

$$\mathbf{Y}_1 = \mathbf{Y}_0 + h\sum_{i=1}^{s} b_i\mathbf{K}_i, \quad b_i \in \mathbb{R}, \tag{2}$$

with $\mathbf{K}_i$ given by

$$\mathbf{K}_i = f(\mathbf{Y}_0 + h\sum_{j=1}^{i-1} a_{i,j}\mathbf{K}_j), \quad a_{i,j} \in \mathbb{R}. \tag{3}$$

Notice, that

$$\mathbf{K}_1 = -\underbrace{\left(\mathbf{Y}_0 - \mathbf{Y}_0^\top\right)}_{=0 \text{ for } \mathbf{Y}_0 = \mathbf{Y}_0^\top}\mathbf{Y}_0 = 0. \tag{4}$$

Induction: assuming $\mathbf{K}_i = 0$ for all $i < n \leq s$, we obtain

$$\mathbf{K}_{n+1} = f(\mathbf{Y}_0 + h\sum_{j=1}^{n} a_{n+1,j}\mathbf{K}_j) = f(\mathbf{Y}_0) = 0. \tag{5}$$

Hence, $\mathbf{K}_i = 0$ for all $i \leq s$ and $\mathbf{Y}_1 = \mathbf{Y}_0$. By induction, $\mathbf{Y}_k = \mathbf{Y}_0$ for all $k \in \mathbb{N}$.

(4b) [15 points]

Listing 1: Integration of ODE using `ode45`

```
function YT = matode(Y0,T)
% Numerical integration with ode45, matrices have to be vectorized
```

```
n = size(Y0,1);
opts = odeset('abstol',10E-10,'reltol',10E-8,'stats','on');
[~,YT] = ode45(@matodefun,[0 T],reshape(Y0,n*n,1),opts);
YT = reshape(YT(end,:)',n,n);
end

function y = matodefun(t,y)
% Right hand side for matrix valued ODE relying on vectorized matrices
N = length(y);
n = floor(sqrt(N));
Y = reshape(y,n,n);
y = reshape(-(Y-Y')*Y,N,1);
end
```

(4c) [10 points] Applying the chain rule, we obtain

$$\begin{aligned}\frac{d}{dt}(\mathbf{Y}^\top(t)\mathbf{Y}(t)) &= \frac{d}{dt}(\mathbf{Y}^\top(t))\mathbf{Y}(t) + \mathbf{Y}^\top(t)\frac{d}{dt}(\mathbf{Y}(t)) \\ &= (-(\mathbf{Y}(t) - \mathbf{Y}^\top(t))\mathbf{Y}(t))^\top\mathbf{Y}(t) + \mathbf{Y}(t)(-(\mathbf{Y}(t) - \mathbf{Y}^\top(t))\mathbf{Y}(t)) \\ &= -\mathbf{Y}^\top(t)\mathbf{Y}^\top(t)\mathbf{Y}(t) + \mathbf{Y}^\top\mathbf{Y}(t)\mathbf{Y}(t) - \mathbf{Y}^\top\mathbf{Y}(t)\mathbf{Y}(t) + \mathbf{Y}^\top(t)\mathbf{Y}^\top(t)\mathbf{Y}(t) = 0.\end{aligned} \tag{6}$$

(4d) [5 points]

Listing 2: Invariance check for sub-problem (c)

```
function checkinvariant(Y0,T)

YT = matode(Y0,T);

if norm(YT'*YT - Y0'*Y0) < 10*eps
  display('Assertion is true.');
else
  display('Assertion is false.');
end
```

(4e) [10 points]

Listing 3: Discrete gradient rule

```
function YT = matodespr(Y0,T,N)
% Using N equidistant steps of a structure preserving discrete gradient rule to
% the matrix ODE dY/dt = - (Y-Y')Y over [0,T].

rhs = @(Y) -(Y-Y')*Y;

[n,m] = size(Y0); h = T/N;
I = eye(n,n);
YT = Y0;
for j=1:N
  Ys = YT + 0.5*h*rhs(YT);
  DYs = Ys-Ys';
  YT = (I+0.5*h*DYs)\(I-0.5*h*DYs)*YT;
end
```

(4f) [10 points] See Listing 4 and Figure 1.

Listing 4: Script to determine convergence order of `matodespr`

```
function matodecvg ()

n = 3;
```

```
T = 1;
Ns = [10,20,40,80,160,320,640,1280];

[Y0,dummy] = qr(magic(n));
YT = matode(Y0,T);

err = [];
for N = Ns
  Yspr = matodespr(Y0,T,N);
  err = [err, norm(Yspr-YT)];
end

order = 2;
slope = 2*err(1)/(Ns(1)^(-order));
loglog(Ns, slope * Ns.^(-order), 'k-', Ns, err, 'r-o');
xlabel('{\bf No. of steps}','fontsize',14);
ylabel('{\bf error}','fontsize',14);
legend(sprintf('O(N^{-%d})',order),'discrete gradient rule', ...
       'location','best');
print -depsc2 'matodecvg.eps';
```

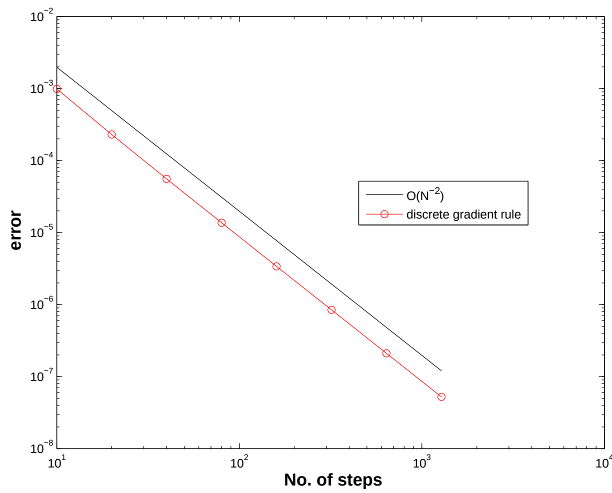

Figure 1: Convergence plot for Problem 4.

Problem 5. Legacy routine

[40 points]

(5a) [5 points] Sparse matrix-vector multiplication is $O(n)$. Each iteration of `pcg` with a sparse matrix is $O(n)$. m such iterations will be $O(nm)$.

(5b) [15 points] Linear convergence. See Listing 5 and Figure 2.

Listing 5: Script to determine convergence properties of `gse`

```
function gsecvg ()

A = gallery('poisson',100);
mm = [1,2,3,4];
style = {'x-b','x-.g', 'x-r','x--k'};
iters = [1:11];
vals = zeros(size(iters));
refsol = gse(A, @(x) A\x, 1e-14, 10000);
figure(1);
```

```
clf;
for curm = mm
        for(itind = 1:length(iters))
                vals(itind) = gse(A, @(x) pcg(A,x,0,curm), 0.01, iters(itind));
        end
        semilogy(iters, abs(vals-refsol), style{curm});
        hold on;
end
xlabel('{\bf # iters}'); ylabel('{\bf |se - se\_exact|}');
legend('m=1','m=2','m=3','m=4');
print -depsc2 'gsecvg.eps';
```

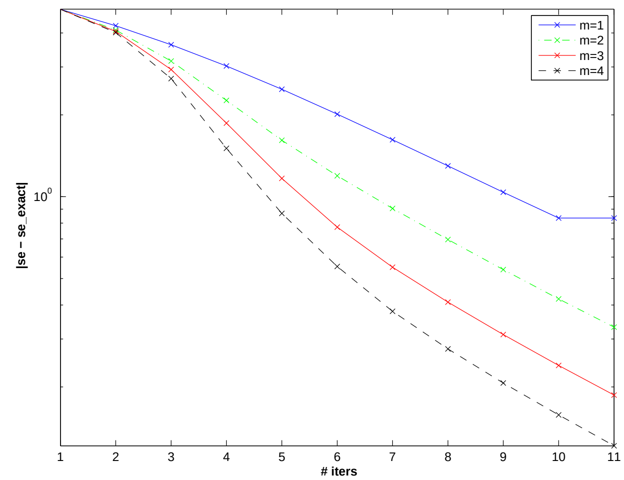

Figure 2: Convergence plot for Problem 5.

(5c) [20 points] The algorithm in the code is the inverse iteration algorithm. The return value is the smallest eigenvalue of a given matrix $\mathbf{A}$.

The loop body is executing the following computations (hat denotes normalized vectors):

$$\begin{aligned}
&\mathbf{v} \longleftarrow \mathbf{A}\hat{\mathbf{z}},\\
&\rho \longleftarrow \mathbf{v}^\top \hat{\mathbf{z}} = \hat{\mathbf{z}}^\top \mathbf{A}\hat{\mathbf{z}}, \qquad (\rho - \text{approximation of eigenvalue associated to } \hat{\mathbf{z}})\\
&\mathbf{r} \longleftarrow \mathbf{v} - \rho\hat{\mathbf{z}} = \mathbf{A}\hat{\mathbf{z}} - \rho\hat{\mathbf{z}},\\
&\mathbf{z} \longleftarrow \hat{\mathbf{z}} - \mathbf{A}^{-1}\mathbf{r} = \hat{\mathbf{z}} - \mathbf{A}^{-1}\mathbf{A}\hat{\mathbf{z}} + \rho\mathbf{A}^{-1}\hat{\mathbf{z}} = \rho\mathbf{A}^{-1}\mathbf{z}, \qquad (\text{inverse iteration})\\
&\hat{\mathbf{z}} \longleftarrow \mathbf{z}.
\end{aligned} \tag{7}$$