NUMERICAL PROGRAMMING 1 (CSE), WS 2014/15

LECTURER: C. LASSER

Contents

Date: updated January 28, 2015.

References

[DR] P. Davis, P. Rabinowitz, Methods of numerical integration (2nd ed.), Dover Publications, 2007.
[D] J. Demmel, Applied Numerical Linear Algebra, SIAM 1997.
[GG] W. Gander, W. Gautschi, Adaptive quadrature – revisited, BIT 40, 84–101, 2000.
[I] A. Iserles, A First Course in the Numerical Analysis of Differential Equations (2nd ed.), Cambridge University Press, 2009.
[K] A. Krommer, C. Ueberhuber, Computational Integration, SIAM, 1998.
[L] C. Lubich, From Quantum to Classical Molecular Dynamics: Reduced Models and Numerical Analysis, EMS, 2008.
[M] C. Moler, Numerical Computing with MATLAB (rev. reprint), SIAM, 2008.
[O] M. Overton, Numerical Computing with IEEE Floating Point Arithmetic, SIAM, 2001.
[S1] G. Stewart, Afternotes on numerical analysis, SIAM, 1996.
[S2] G. Stewart, Afternotes goes to graduate school, SIAM, 1998.
[T] L. N. Trefethen, Approximation theory and approximation practice, SIAM, 2013, `http://www.chebfun.org/ATAP/`.
[TB] L. N. Trefethen, D. Bau, Numerical Linear Algebra, SIAM, 1997.

1. Basics

1.1. Floating point numbers (20.10.)

1.1.1. *Gradual underflow.*

```
x=1; n=1; while (x>0), x=x/2, n=n+1 end,
help realmin, realmin
```

1.1.2. *Overflow.*

```
x=2; y=0; n=1; while (x>y), y=x, x=x^2, n=n+1, end,
help realmax; realmax
```

1.1.3. *Machine precision.*

```
1-(1+1e-16), help eps, eps
```

1.1.4. *Definition.* We use 4 natural numbers to define the set $\mathbb{F}$ of floating point numbers: the base b, the number of digits p (precision), the minimal exponent $e_{\min}$ and the maximal exponent $e_{\max}$. Then, x is a floating point number, if there exists an exponent $e \in \{e_{\min}, \dots, e_{\max}\}$ and digits $d_0, \dots, d_{p-1} \in \{0, \dots, b-1\}$ such that

$$x = \pm b^e \left(d_0 + \frac{d_1}{b} + \frac{d_2}{b^2} + \cdots + \frac{d_{p-1}}{b^{p-1}} \right)$$

For $b = 10$, this means $x = \pm 10^e \times d_0.d_1 d_2 \dots d_{p-1}$.

1.1.5. *Normalization.* For $b = 2$ one requires $d_0 = 1$, resulting in normalized floating point numbers.

1.1.6. *IEEE double format.* $b = 2$, $p = 53$, $e_{\min} = -1022$, $e_{\max} = 1023$.

1.1.7. *Visualization.* Let $b = 2$. The digits of a normalized floating point number define $m \in \{b^p, \dots, 2b^p - b\}$ such that

$$x = \pm m \cdot b^{e-p}.$$

```
b=2; p=3; emin=-1; emax=3; F=0;
for m=b^p:2*b^p-b, F=[F, m*b.^([emin:emax]-p)]; end
plot(F,zeros(length(F)),'r*')
```

1.1.8. *Range.* Let $b = 2$. We have for all normalized floating point numbers x

$$b^{e_{\min}} \le |x| \le b^{e_{\max}}(2 - b^{1-p}).$$

```
2^(-1022), realmin, 2^(1023)*(2-2^(-52)), realmax
```

1.1.9. *Spacing.* $\varepsilon = b^{1-p}$ is the distance from 1 to the next larger $x \in \mathbb{F}$.

```
2^(-52), eps
```

1.1.10. *Subnormal numbers.* Let $b = 2$. Numbers of the form

$$x = \pm b^{e_{\min}} \left(\frac{d_1}{b} + \frac{d_2}{b^2} + \cdots + \frac{d_{p-1}}{b^{p-1}} \right)$$

are called subnormal.

```
b=2; p=3; emin=-1; m=0:b^p-b; F=m*b.^(emin-p);
plot(F,zeros(length(F)),'r*')
```

Literature. [O, §3–4]

1.2. **Conditioning (22.10.)**

1.2.1. *Cancellation.*

```
x=1; n=20; h=10.^(-(1:n)); d=((x+h)-x)./h;
loglog(h,abs(d-1),'*'), grid on
```

First observation: If $h < \varepsilon$, then $1 + h$ equals 1 in floating point arithmetic.

1.2.2. *Definition.* Let $f : \mathbb{R}^d \to \mathbb{R}$ be twice continuously differentiable. Then, for all $x \in \mathbb{R}^d$,

$$f(x + \Delta x) - f(x) = \nabla f(x) \cdot \Delta x + O(\Delta x^2), \qquad \Delta x \to 0.$$

This implies

$$\lim_{\Delta x \to 0} \frac{|f(x + \Delta x) - f(x)|}{|f(x)|} \Big/ \frac{\|\Delta x\|}{\|x\|} = \frac{\|\nabla f(x)\| \cdot \|x\|}{|f(x)|} =: \kappa_f(x).$$

We call $\kappa_f(x)$ the condition number of f in x.

1.2.3. *Multiplication.* Let $\alpha \in \mathbb{R}$ and $f : \mathbb{R} \to \mathbb{R}$, $x \mapsto \alpha x$. Then, for all $x \in \mathbb{R}$

$$\kappa_f(x) = \frac{|\alpha| \cdot |x|}{|\alpha x|} = 1.$$

1.2.4. *Division.* Let $\alpha \in \mathbb{R}$ and $f : \mathbb{R}^* \to \mathbb{R}$, $x \mapsto \alpha/x$. Then, for all $x \in \mathbb{R}^*$,

$$\kappa_f(x) = \frac{|\alpha|}{|x|^2} \cdot \frac{|x|}{|\alpha|/|x|} = 1.$$

1.2.5. *Subtraction.* Let $\alpha \in \mathbb{R}$ and $f : \mathbb{R} \to \mathbb{R}$, $x \mapsto x - \alpha$. Then, for all $x \in \mathbb{R}$,

$$\kappa_f(x) = \frac{1 \cdot |x|}{|x - \alpha|},$$

and $\kappa_f(x) \to \infty$ as $x \to \alpha$. That is, subtraction of nearby numbers is ill-conditioned.

1.2.6. *Rule of thumb.*

$$\log_{10}\left(\frac{|f(x + \Delta x) - f(x)|}{|f(x)|}\right) \approx \log_{10}(\kappa_f(x)) + \log_{10}\left(\frac{\|\Delta x\|}{\|x\|}\right)$$

Therefore, $\log_{10}(\kappa_f(x))$ corresponds to the number of lost "significant digits".

```
-log10(eps), -log10(eps('single'))
```

1.2.7. *Exponential function.* Let $f : \mathbb{R} \to \mathbb{R}$, $x \mapsto \exp(x)$. Then, for all $x \in \mathbb{R}$,

$$\kappa_f(x) = \frac{|\exp(x)| \cdot |x|}{|\exp(x)|} = |x|.$$

```
x=1e-6; y=double(single(x)); diff=abs(x-y)/x; sd=-log10(diff),
diff=abs(exp(x)-exp(y))/exp(x); sde=-log10(diff),
```

1.2.8. *Sine function.* Let $f : \mathbb{R} \to \mathbb{R}$, $x \mapsto \sin(x)$. Then, for all $x \in \mathbb{R}$,

$$\kappa_f(x) = \frac{|\cos(x)| \cdot |x|}{|\sin(x)|}.$$

```
x=pi; y=double(single(x)); diff=abs(x-y)/x; sd=-log10(diff),
diff=abs(sin(x)-sin(y))/abs(sin(x)); sde=-log10(diff)
```

Literature. [O, §11-12]

2. Interpolation

2.1. **Algebraic interpolation (27.10.)**

2.1.1. *Task.* Let $x_0, \dots, x_n$ be distinct points in $[-1, 1]$ and $y_0, \dots, y_n \in \mathbb{R}$. Polynomial interpolation provides *the* polynomial p of degree $\le n$ such that $p(x_j) = y_j$ for all $j = 0, \dots, n$.

2.1.2. *Lagrange polynomials.* Let $\ell_j(x)$ be the polynomial of degree $\le n$ such that $\ell_j(x_k) = \delta_{kj}$, that is, $\ell_j(x) = \prod_{k\neq j}(x - x_k) \Big/ \prod_{k\neq j}(x_j - x_k)$. Then,

$$p(x) = \sum_{j=0}^{n} y_j \ell_j(x).$$

```
x = linspace(-1,1,7); y = [0 0 0 0 0 1 0]; p = polyfit(x,y,6);
xx = linspace(-1,1); yy = polyval(p,xx); plot(xx,yy,x,y,'*')
```

2.1.3. *Conditioning.* Let the nodes $x_0, \dots, x_n$ be given and $p(x; y)$ be the interpolating polynomial for $y = (y_0, \dots, y_n)$. The absolute condition number is

$$\kappa_p(y) = \lim_{\Delta y \to 0} \frac{\|p(\cdot\,; y + \Delta y) - p(\cdot\,; y)\|_\infty}{\|\Delta y\|_\infty} = \sup_{x\in[-1,1]} \sum_{j=0}^{n} |\ell_j(x)| =: \Lambda_n,$$

the Lebesgue constant of the nodes. Always, $\Lambda_n \ge \frac{2}{\pi}\log(n+1) + 0.5$.

2.1.4. *Equidistant nodes.* For equidistant nodes we have $\Lambda_n \ge 2^{n-2}/n^2$.

```
n = [10:10:1000]; c = 2/pi*log(n+1)+1/2; e = 2.^(n-2)./n.^2;
subplot(2,1,1), plot(n,c,'b'), subplot(2,1,2), semilogy(n,e,'r')
```

2.1.5. *Chebyshev nodes.* For the Chebyshev nodes $x_j = \cos(j\pi/n)$, $j = 0, \dots, n$, we have $\Lambda_n \le 1 + \frac{2}{\pi}\log(n+1)$.

```
n = 8; x = cos(pi/n*[0:n]); plot(x,zeros(length(x)),'*')
```

2.1.6. *First barycentric interpolation formula.* We write $\ell_j(x) = \ell(x)\frac{w_j}{x - x_j}$ with $\ell(x) = \prod_{k=0}^{n}(x - x_k)$ the node polynomial and $w_j = \prod_{k\neq j}(x_j - x_k)^{-1}$ the barycentric weight. This gives

$$p(x) = \ell(x) \sum_{j=0}^{n} \frac{y_j w_j}{x - x_j}.$$

The formula is stable and requires $O(n)$ flops for precomputed weights.

```
n = 8; % n = 20; n = 100;
x = cos(pi/n*[0:n]); %x = linspace(-1,1,n+1);
y = 1./(1+25*x.^2);
for j=1:n+1, w(j) = prod(1./(x(j) - x([1:j-1, j+1:end])) ); end
xx = linspace(-1,1); yy = 1./(1+25*xx.^2); l = 1; s = 0;
for j=1:n+1, l = l.*(xx-x(j)); s = s + y(j)*w(j)./(xx-x(j)); end
p = l.*s; plot(xx,yy,xx,p,x,y,'*')
pp = polyfit(x,y,n); yyy=polyval(pp,xx); hold on, plot(xx,yyy,'r')
```

Literature. [T, §5 & §15]

2.2. **Trigonometric interpolation (29.10.)**

2.2.1. *Task.* Let $x_0, \dots, x_{n-1}$ be distinct points in $[0, 2\pi]$ and $y_0, \dots, y_{n-1} \in \mathbb{C}$. Trigonometric interpolation provides *the* trigonometric polynomial

$$p(x) = \sum_{j=0}^{n-1} c_j e^{ijx}$$

such that $p(x_j) = y_j$ for all $j = 0, \dots, n-1$. From now on, $x_j = 2\pi j/n$.

2.2.2. *Conditioning.* As for algebraic interpolation, the absolute condition number is the Lebesgue constant Λ_n, and one can prove $\Lambda_n \leq \frac{2}{\pi}\log(n) + \frac{5}{3}$.

2.2.3. *Orthogonality.* Let $\omega_n = e^{2\pi i/n}$ the nth unit root. We observe $e^{ijx_k} = \omega_n^{jk}$ and

$$\frac{1}{n}\sum_{j=0}^{n-1} \omega_n^{-jk}\omega_n^{jl} = \frac{1}{n}\sum_{j=0}^{n-1} \left(\omega_n^{l-k}\right)^j = \delta_{kl},$$

since we are summing a geometric series.

```
n=20; x=2*pi*[0:(n-1)]/n; y=exp(1i*x);  plot(real(y),imag(y),'*')
```

Therefore, $y_k = \sum_{j=0}^{n-1} c_j \omega_n^{jk}$ and $c_l = \frac{1}{n}\sum_{k=0}^{n-1} \omega_n^{-lk} y_k$.

2.2.4. *Fast Fourier Transform.* Let $n = 2^m$. Since $\omega_{2n}^2 = \omega_n$,

$$c_l = \frac{1}{n}\sum_{k=0}^{n/2-1} \omega_{n/2}^{-lk}\, y_{2k} + \frac{\omega_n^{-l}}{n}\sum_{k=0}^{n/2-1} \omega_{n/2}^{-lk}\, y_{2k+1} = \dots$$

This is the basic observation for the (inverse) FFT, the computation of the coefficient vector $c = (c_0, \dots, c_{n-1})$ from $y = (y_0, \dots, y_{n-1})$ in $O(n\log(n))$ flops.

```
help fft; n = 32; c = zeros(1,n); c(1) = 1; y = fft(c);
subplot(1,2,1), plot([0:n-1],real(y),'b-*'),
subplot(1,2,2), plot([0:n-1],imag(y),'r-*')
```

2.2.5. *Fourier Series.* Let $f : [0, 2\pi] \to \mathbb{C}$ be 2π-periodic, continuously differentiable. Then, $f(x) = \sum_{j=-\infty}^{\infty} \widehat{f}_j e^{ijx}$, where

$$\widehat{f}_j = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-ijx}dx$$

is the jth Fourier coefficient. Let $y_j = f(x_j)$. We observe

$$c_j = \frac{1}{n}\sum_{k=0}^{n-1} y_k\omega_n^{-jk} = \frac{1}{2\pi}\cdot\frac{2\pi}{n}\sum_{k=0}^{n-1} f(x_k)e^{-ijx_k} \approx \widehat{f}_j.$$

2.2.6. *Aliasing.* The interpolation property yields $c_j = \sum_{k=-\infty}^{\infty} \widehat{f}_{j+kn}$.

```
n = 8; x = 2*pi/n*[0:n-1]; y = sin(x) + 3*sin(5*x);
xx = linspace(0,2*pi,100); yy = sin(xx) + 3*sin(5*xx);
z = interpft(y,100); plot(xx,yy,'b-',xx,z,'r-',x,y,'ro')
n = 16; x = 2*pi/n*[0:n-1]; y = sin(x) + 3*sin(5*x);
z = interpft(y,100); hold on, plot(xx,z,'g-',x,y,'go')
```

Literature. [M, Chapter 8]

2.3. **Spline interpolation (3.11.)**

2.3.1. *Linear splines.* Let $x_0, \ldots, x_n \in [a,b]$ with $a = x_0 < \ldots < x_n = b$ and $y_0, \ldots, y_n \in \mathbb{R}$. The continous function $s : [a,b] \to \mathbb{R}$, which is linear on each interval $[x_j, x_{j+1}]$ and satisfies $s(x_j) = y_j$ for all $j = 0, \ldots, n$, is *the* interpolating linear spline:

$$s(x) = y_j + \frac{y_{j+1} - y_j}{x_{j+1} - x_j}(x - x_j), \qquad x \in [x_j, x_{j+1}].$$

```
x = linspace(-2,8,6); y = x.^2 + 10./(sin(x)+1.2);
xx = linspace(-2,8); yy = xx.^2 + 10./(sin(xx)+1.2);
z = interp1(x,y,xx); plot(xx,yy,'b-',xx,z,'r-',x,y,'ro')
```

2.3.2. *Cubic splines.* Twice continuously differentiable functions $s : [a,b] \to \mathbb{R}$, which are cubic polynomials on each interval $[x_j, x_{j+1}]$ and satisfy $s(x_j) = y_j$ for all $j = 0, \ldots, n$, are interpolating cubic splines:

$$s(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \qquad x \in [x_j, x_{j+1}].$$

2.3.3. *Coefficients.* Let $s_j = s''(x_j)$ and $h_j = x_{j+1} - x_j$. Then,

$$s''(x) = \frac{s_j}{h_j}(x_{j+1} - x) + \frac{s_{j+1}}{h_j}(x - x_j), \qquad x \in [x_j, x_{j+1}].$$

and integrating twice,

$$s(x) = a_j + b_j(x - x_j) + \frac{s_j}{6h_j}(x_{j+1} - x)^3 + \frac{s_{j+1}}{6h_j}(x - x_j)^3, \qquad x \in [x_j, x_{j+1}].$$

From $s(x_j) = y_j$, we get $a_j = y_j - \frac{1}{6}s_j h_j^2$. From $s(x_{j+1}) = y_{j+1}$, we get $b_j = \beta_j - \frac{1}{6}(s_{j+1} - s_j)h_j$, $\beta_j = (y_{j+1} - y_j)/h_j$.

2.3.4. *Interior conditions.* Using continuity of $s'(x)$, one obtains that the second derivatives satisfy the tridiagonal system

$$\frac{h_j}{6}s_j + \frac{h_j + h_{j+1}}{3}s_{j+1} + \frac{h_{j+1}}{6}s_{j+2} = \beta_{j+1} - \beta_j, \qquad j = 1, \ldots, n-2.$$

It remains to specify two additional conditions, for example, by choosing s_0 and s_n.

2.3.5. *Natural spline.* Setting the endpoint curvature

$$s_0 = s_n = 0,$$

defines the *natural* spline. Other splines are more accurate, see the next lecture.

2.3.6. *Complete spline.* Setting

$$s'(a) = f'(a), \qquad s'(b) = f'(b),$$

defines the *complete* spline for the function $f : [a,b] \to \mathbb{R}$.

2.3.7. *Not-a-knot conditions.* The not-a-knot condition is

$$s|_{[x_0,x_1]} = s|_{[x_1,x_2]}, \quad s|_{[x_{n-2},x_{n-1}]} = s|_{[x_{n-1},x_n]}.$$

This implies on the left $-h_1 s_0 + (h_0 + h_1)s_1 - h_0 s_2 = 0$ and an analogous condition on the right, since $s''' = (s_{j+1} - s_j)/h_j$ on $[x_j, x_{j+1}]$.

```
z = interp1(x,y,xx,'spline'); hold on, plot(xx,z,'g-')
```

Literature. [S2, Chapter 10 & 11]

2.4. **Interpolation accuracy (5.11.)**

2.4.1. *The task.* Let $x_0, \dots, x_n$ with $a = x_0 < \dots < x_n = b$ and $y_0, \dots, y_n \in \mathbb{R}$. Let $f : [a,b] \to \mathbb{R}$ be such that $f(x_j) = y_j$ for all j. We seek the polynomial p of degree $\leq n$ or a spline s such that

$$p(x_j) = y_j, \qquad s(x_j) = y_j, \qquad (j = 0, \dots, n),$$

and ask for the accuracy.

2.4.2. *Algebraic interpolation.* We prove existence of some $\xi \in [a,b]$ such that

$$f(x) - p(x) = \ell(x) f^{(n+1)}(\xi)/(n+1)!\,, \qquad x \in [a,b],$$

for some $\xi \in [a,b]$ by a 3-step argument.

(1) There is a smooth function $g : [a,b] \to \mathbb{R}$ such that $f(x) - p(x) = g(x)\ell(x)$ for all $x \in [a,b]$.
(2) Study for fixed $x \in [a,b]$ the function $F_x(y) = f(y) - p(y) - g(x)\ell(y)$ and find $\xi \in [a,b]$ such that $F_x^{(n+1)}(\xi) = 0$.
(3) Observe that $\ell^{(n+1)}(x) = (n+1)!$ and conclude the proof.

For Chebyshev nodes, we have spectral convergence w.r.t. $\|\cdot\|_\infty$, see §2.4.5.

2.4.3. *Linear splines.* Let $f : [a,b] \to \mathbb{R}$ be twice continuously differentiable and $y_j = f(x_j)$ for all j. Then, there exists $\xi_j \in [x_j, x_{j+1}]$ such that

$$f(x) - s(x) = \frac{1}{2} f''(\xi_j)(x - x_j)(x - x_{j+1}), \qquad x \in [x_j, x_{j+1}].$$

If $h_j = x_{j+1} - x_j$ and $h = \max\{h_0, \dots, h_{n-1}\}$, then

$$\begin{aligned} |f(x) - s(x)| &\leq \tfrac{1}{8} h_j^2 \max_{x \in [x_j, x_{j+1}]} |f''(x)|, \qquad x \in [x_j, x_{j+1}], \\ \|f - s\|_\infty &\leq \tfrac{1}{8} h^2 \|f''\|_\infty. \end{aligned}$$

```
x = linspace(-2,8,6); y = x.^2 + 10./(sin(x)+1.2);
xx = linspace(-2,8); yy = xx.^2 + 10./(sin(xx)+1.2);
z = interp1(x,y,xx); plot(xx,yy,'b-',xx,z,'r-',x,y,'ro')
```

2.4.4. *Cubic splines.* If f is four times continuosly differentiable, then the complete and the not-a-knot cubic spline satisfy

$$\|f^{(r)} - s^{(r)}\|_\infty \leq C_r h^{4-r} \|f^{(4)}\|_\infty, \qquad r = 0, 1, 2.$$

Local error estimates are posssible, too.

2.4.5. *Spectral convergence.* Let $x_j = 2\pi j/n \in [0, 2\pi]$ for $j = 0, \dots, n-1$ and $y_0, \dots, y_{n-1} \in \mathbb{C}$. Let $f : [0, 2\pi] \to \mathbb{C}$ be such that $f(x_j) = y_j$ for all j. The interpolating trigonometric polynomial

$$p(x) = \sum_{j=0}^{n-1} c_j e^{ijx}, \qquad c_j = \frac{1}{n} \sum_{k=0}^{n-1} e^{-ijx_k} y_k,$$

satisfies $p(x_j) = y_j$ for all j, and for all $s \in \mathbb{N}$ there exists a constant $C_s > 0$ such that

$$\|f - p\|_2 \leq C_s n^{-s} \|f^{(s)}\|_2 \qquad (\textit{spectral convergence}).$$

Literature. [S1, §20], [S2, §10 & §11], [L, III.1.3]

3. Quadrature

3.1. **Sum rules (10.11.)**

3.1.1. *The task.* Let $f : [a,b] \to \mathbb{R}$ be continuous. Let $x_0, \dots, x_n \in [a,b]$ and $w_0, \dots, w_n \in \mathbb{R}$. We approximate

$$I(f) = \int_a^b f(x)dx \approx \sum_{j=0}^{n} w_j f(x_j) = Q(f).$$

3.1.2. *Relative condition numbers.* If $\|w\|_1 = |w_0| + \ldots + |w_n|$, then

$$\kappa_I(f) = \frac{(b-a)\|f\|_\infty}{|I(f)|}, \qquad \kappa_Q(f) = \frac{\|w\|_1\|f\|_\infty}{|Q(f)|}.$$

Hence, integration and quadrature of oscillatory functions are ill-conditioned.

3.1.3. *Stability.* If $Q(1) = I(1)$, then we have for the absolute condition numbers

$$\kappa_Q(f) = \|w\|_1 \geq \sum_{j=0}^{n} w_j = Q(1) = I(1) = b - a = \kappa_I(f) \qquad (*)$$

$(*)$ is an equality iff $w_0, \dots, w_n \geq 0$. Therefore, positive weights are preferred.

3.1.4. *Interpolatory rules.* Let p_1, p_2, p_3 be the interpolating polynomials for the nodes $\{\frac{a+b}{2}\}$, $\{a,b\}$, and $\{a, \frac{a+b}{2}, b\}$, respectively. Then,

$$\begin{aligned} M_{[a,b]}(f) &= I(p_1) = (b-a)f(\tfrac{a+b}{2}) \qquad \text{(mid point rule)}, \\ T_{[a,b]}(f) &= I(p_2) = \tfrac{b-a}{2}(f(a)+f(b)), \qquad \text{(trapezoidal rule)} \\ S_{[a,b]}(f) &= I(p_3) = \tfrac{b-a}{6}(f(a)+4f(\tfrac{a+b}{2})+f(b)) \qquad \text{(Simpson's rule)}. \end{aligned}$$

M and T are *both* exact for linear integrands, S for *cubic* ones. For example,

$$M_{[a,b]}(f) - I(f) = -\tfrac{1}{24}(b-a)^3 f''(\xi) \quad \text{for some } \xi \in [a,b].$$

3.1.5. *Sum rules.* Let n be even and $x_j = a + jh$, $j = 0, \dots, n$ with $h = (b-a)/n$ and $f_j = f(x_j)$. The composite rules are

$$\begin{aligned} M_n(f) &= \sum_{j=0}^{n/2-1} M_{[x_{2j}, x_{2(j+1)}]}(f) = \sum_{j=0}^{n/2-1} 2hf_{2j+1}, \\ T_n(f) &= \sum_{j=1}^{n} T_{[x_{j-1}, x_j]}(f) = \tfrac{h}{2}(f_0 + f_n) + \sum_{j=1}^{n-1} hf_j, \\ S_n(f) &= \sum_{j=0}^{n/2-1} S_{[x_{2j}, x_{2(j+1)}]}(f) = \tfrac{h}{3}(f_0 + \sum_{j=0}^{n/2-1} 4f_{2j+1} + \sum_{j=1}^{n/2-1} 2f_{2j} + f_n) \end{aligned}$$

M_n and T_n are both $O(h^2)$, S_n is $O(h^4)$ accurate. For example,

$$T_n(f) - I(f) = \frac{b-a}{12}h^2 f''(\xi) \quad \text{for some } \xi \in [a,b].$$

```
N = [10,100,1000,1e+4]; j=0; a = 1;  %a = 5
for n = N; j = j+1; x = linspace(-a,a,n); y = exp(-x.^2)/sqrt(pi);
T(j) = trapz(x,y);  end,  R=erf(a), loglog(N,abs(T-R),N,(2*a./N).^2)
```

Literature. [S1, §21], [K, §3.4]

3.2. **Gaussian quadrature (12.11.)**

3.2.1. *The task.* Let $w : [a, b] \to [0, \infty[$. We compute

$$I_w(f) = \int_a^b f(x)w(x)dx \approx Q(f) = \sum_{j=0}^{n} w_j f(x_j)$$

for $f : [a, b] \to \mathbb{R}$ continuous. Let $\mathbb{P}_n$ the space of polynomials of degree $\leq n$.

3.2.2. *The weights via Lagrange polynomials.* For all $Q(p) = I_w(p)$ for all $p \in \mathbb{P}_n$ if and only if $w_j = I_w(\ell_j)$ for all $j = 0, \ldots, n$.

3.2.3. *Nonnegative weights.* $Q(p) = I_w(p)$ for all $p \in \mathbb{P}_{2n}$ implies $w_j \geq 0$ for all j, since $w_j = Q(\ell_j^2) = I_w(\ell_j^2)$.

3.2.4. *The idea.* Choose the weights such that Q is exact on $\mathbb{P}_n$. Then determine the nodes nodes such that Q is exact on $\mathbb{P}_{2n+1}$.
Let $p \in \mathbb{P}_{2n+1}$ and $p_{n+1} \in \mathbb{P}_{n+1}$. We write $p = qp_{n+1} + r$ with $q, r \in \mathbb{P}_n$. Then, $I_w(p) = I_w(qp_{n+1}) + Q(r)$ and we obtain $I_w(p) = Q(p)$, if

$$\forall q \in \mathbb{P}_n : I_w(qp_{n+1}) = 0 = Q(qp_{n+1}).$$

This works, if $x_0, \ldots, x_n$ are the roots of p_{n+1} and p_{n+1} is orthogonal on $\mathbb{P}_n$.

3.2.5. *Orthogonal polynomials.* $(p_n)_{n\geq 0}$ is a familiy of orthogonal polynomials, if $p_n \in \mathbb{P}_n$, $p_n \neq 0$ for all n and

$$\forall n \neq m : I_w(p_n p_m) = 0.$$

Each orthogonal family satisfies a 3-term recurrence

$$a_n p_{n+1}(x) = (b_n + c_n x)p_n(x) - d_n p_{n-1}(x), \qquad x \in [a, b],$$

with (a_n), (b_n), (c_n), (d_n) sequences of real numbers.

	$[a, b]$	w	a_n	b_n	c_n	d_n
Legendre P_n	$[-1, 1]$	1	$n+1$	0	$2n+1$	n
Hermite H_n	$]-\infty, +\infty[$	e^{-x^2}	1	0	2	$2n$
Laguerre L_n	$[0, \infty[$	e^{-x}	$n+1$	$2n+1$	-1	n

3.2.6. *Eigenvalue problems.* The 3-tem recurrence allows to characterize the nodes as the eigenvalues of a symmetric tridiagonal eigenvalue problem, while the weights can be obtained from the eigenvectors. For example:

```
N = [1:100]; j = 0; for n=N, j = j+1;
beta = .5./sqrt(1-(2*(1:n)).^(-2));
T = diag(beta,1) + diag(beta,-1); [V,D] = eig(T);
x = diag(D); w = 2*V(1,:).^2; I(j) = w*cos(x); J(j) = w*abs(x); end
semilogy(N,abs(I - 2*sin(1)),'g-o',N,abs(J-1),'r-o')
```

3.2.7. *Radau formulas.* One sets $x_0 = a$ or $x_n = b$, and then constructs Q to be exact on $\mathbb{P}_{2n}$.

3.2.8. *Lobatto formulas.* One sets $x_0 = a$ and $x_n = b$, and then constructs Q to be exact on $\mathbb{P}_{2n-1}$.

Literature. [S1, Chapter 23]

3.3. **Adaptive quadrature (17.11.)**

3.3.1. *The idea.* One approximates

$$I(a,b) = \int_a^b f(x)dx \approx Q_j, \quad j = 1, 2$$

by two different quadratures Q_1, Q_2 and sets $\tau > 0$. If $|Q_1 - Q_2|/|Q_2| < \tau$, then one chooses either Q_1 or Q_2. Otherwise one sets $m = \frac{a+b}{2}$ and continues with $I(a,m)$ and $I(m,b)$ separately.

```
for j=1:100, x1 = linspace(0,1/j,10); x2 = linspace(0,1/j,20);
q1(j) = trapz(x1,sqrt(x1)); q2(j) = trapz(x2,sqrt(x2)); end
loglog([1:100],abs(q1-q2)./abs(q2))
```

3.3.2. *Romberg extrapolation.* Let $Q_1 = T(h)$ and $Q_2 = T(h/2)$ be trapezoidal rules with step size h and $h/2$. There exists $c \in \mathbb{R}$ such that for smooth integrands

$$T(h) - I(a,b) = ch^2 + O(h^3), \qquad h \to 0.$$

Then, $R(h) = (4Q_2 - Q_1)/3$ is $O(h^3)$ accurate.

```
N = [10,100,1000,1e+4]; h = 1./N; I = 0.1; j = 0; for n=N, j = j+1;
x = linspace(0,1,n); q1(j) = trapz(x,x.^9); x = linspace(0,1,2*n);
q2(j) = trapz(x,x.^9);  r(j) = (4*q2(j)-q1(j))/3; end
loglog(N,abs(q1-I),N,abs(q2-I),N,abs(r-I),N,h.^2,'b:',N,h.^3,'r:')
```

3.3.3. *Termination criteria.* Let $0 \neq J \approx |I(a,b)|$ and $J_\tau = J \cdot \tau/\text{eps}$. Stop if

$$J_\tau + (Q_1 - Q_2) = J_\tau \quad \text{or} \quad m \leq a \quad \text{or} \quad m \geq b.$$

The first criterion guarantees, that the subdivided integrals are not negligible for the whole integral, while the last two criteria prevent working on too small subintervals.

3.3.4. *Adaptive Simpson quadrature.* $Q_1 = S(h)$, $Q_2 = S(h/2)$, extrapolation $Q_1 = (16Q_2 - Q_1)/15$ and $n = 8$ Monte-Carlo estimate for J.

```
f = @(x) sqrt(x); tau = 1e-6; quad(f,0,1,tau) - 2/3
f = @(x) 2.0*(x>0.25 & x<0.75); tau = 1e-6; quad(f,0,1,tau) - 1
omega = 1e+1; f = @(x) cos(omega*x); quad(f,0,1) - sin(omega)/omega
```

Literature. [GG]

3.4. **Monte Carlo quadrature (19.11.)**

3.4.1. *The key observation.* Let $\Omega \subset \mathbb{R}^d$, $w : \Omega \to [0, \infty[$ such that $\int_\Omega w(x)dx = 1$. Then,

$$I_w(f) = \int_\Omega f(x)w(x)dx = \mathbb{E}_w(f)$$

and $I_w((f - \mathbb{E}_w(f))^2) = \mathbb{V}_w(f)$.

3.4.2. *Strong law of large numbers.* Let $(X_n)_{n>0}$ be independent identically distributed random variables distributed according to w. Then,

$$\mathbb{P}\Big(\lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} f(X_j) - \mathbb{E}_w(f) \Big) = 1.$$

This motivates $Q_n(f) = \frac{1}{n} \sum_{j=1}^{n} f(X_j)$.

```
n = 1e+1; x = rand(n,1); plot(x,zeros(n,1),'ro'), I = sum(x)/n
```

3.4.3. *Expectation & variance.* We have $\mathbb{E}(Q_n(f)) = \mathbb{E}_w(f)$, $\mathbb{V}(Q_n(f)) = \frac{1}{n}\mathbb{V}_w(f)$,

$$\sqrt{\mathbb{E}([Q_n(f) - \mathbb{E}_w(f)]^2)} = \frac{\sigma_w(f)}{\sqrt{n}}.$$

```
N = [10,100,1000,1e+4,1e+5,1e+6]; j = 0;
for n=N, j = j+1; q(j) = sum(rand(n,1))/n; end,
loglog(N,abs(q-0.5),'b',N,1/2./sqrt(N),'g')
```

3.4.4. *Statistics.* Let $q_1, \dots, q_m$ be independent runs of $Q_n(f)$. Then,

$$\overline{q} = \frac{1}{m} \sum_{j=1}^{m} q_j, \qquad \overline{v} = \frac{1}{m-1} \sum_{j=1}^{m} (q_j - \overline{q})^2$$

are unbiased estimators of $\mathbb{E}_w(f)$ and $\mathbb{V}_w(f)$, respectively. That is, $\mathbb{E}(\overline{q}) = \mathbb{E}(Q_n(f))$ and $\mathbb{E}(\overline{v}) = \mathbb{V}(Q_n(f))$.

```
n = 1e+4; for j=1:10, q(j) = sum(rand(n,1))/n; end,
e = abs(mean(q)-0.5), e_est = sqrt(var(q)), e_theo = 1/2/sqrt(n)
```

3.4.5. *Univariate normal distribution.* If $\mathbb{E}(X) = \mu$ and $\mathbb{V}(X) = \sigma^2$, then

$$w(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\tfrac{1}{2\sigma^2}(x - \mu)^2), \qquad x \in \mathbb{R}.$$

```
n = 100; x = 1+randn(n,1)*2; xr = linspace(min(x),max(x));
plot(x,zeros(n,1),'ro',xr,exp(-1/8*(xr-1).^2)),
n = 1e+4; for j=1:10, x = 1+randn(n,1)*2; q(j) = sum(x)/n; end,
e = abs(mean(q)-1), e_est = sqrt(var(q)), e_theo = 2/sqrt(n)
```

3.4.6. *Multivariate normal distribution.* With $\mathbb{E}(X) = \mu$ and covariance matrix $\Sigma = \mathbb{E}((X - \mu)(X - \mu)^T)$, we have

$$w(x) = (2\pi)^{-d} \det(\Sigma)^{-1/2} \exp(-\tfrac{1}{2}(x - \mu) \cdot \Sigma^{-1}(x - \mu)), \qquad x \in \mathbb{R},$$

```
n = 100; m = 10; q = zeros(2,m);
for j=1:m, x = repmat([1,2],n,1)+randn(n,2)*chol([1,0.5;0.5,2]);
plot(x(:,1),x(:,2),'ro'); q(:,j) = sum(x,1)/n; end,
error = norm(mean(q,2)'-[1,2])
```

Literature. [DR, Chapter 5.9]

4. Linear systems

4.1. **Conditioning (24.11.)**

4.1.1. *Vector norms.* A norm is function $\|\cdot\| : \mathbb{R}^n \to [0,\infty[$ that satisfies for all $x, y \in \mathbb{R}^n$ and all $\alpha \in \mathbb{R}$

$$\|x\| = 0 \text{ iff } x = 0, \qquad \|x+y\| \le \|x\| + \|y\|, \qquad \|\alpha x\| = |\alpha| \cdot \|x\|.$$

Examples are $\|x\|_p = (|x_1|^p + \cdots + |x_n|^p)^{1/p}$ for $1 \le p < \infty$ or $\|x\|_\infty = \max_{1\le j\le n} |x_j|$.

4.1.2. *Matrix norms.* Let $\|\cdot\|$ be a norm on $\mathbb{R}^n$. One defines the induced matrix norm

$$\|A\| := \max_{x\neq 0} \|Ax\|/\|x\| = \max_{\|x\|=1} \|Ax\|, \qquad A \in \mathbb{R}^{n\times n}.$$

In particular, $\|\mathrm{Id}\| = 1$.

4.1.3. *Maximum column and row sum.* Let $A \in \mathbb{R}^{n\times n}$, $c_1, \ldots, c_n \in \mathbb{R}^n$ the columns and $r_1, \ldots r_n \in \mathbb{R}^n$ the rows of A. We have

$$\|A\|_1 = \max_{1\le j\le n} \|c_j\|_1, \qquad \|A\|_\infty = \max_{1\le j\le n} \|r_j\|_1.$$

The first relation follows from $\|Ax\|_1 = \|x_1c_1 + \ldots + x_nc_n\|_1 \le \max_j \|c_j\|_1$ for $\|x\|_1 \le 1$ and $\|e_j\|_1 = 1$, $\|Ae_j\|_1 = \|c_j\|_1$ for the jth unit vector e_j.

4.1.4. *Equivalence of norms.* Let X be a finite-dimensional vector space, i.e. $X = \mathbb{R}^n$ or $X = \mathbb{R}^{n\times n}$. Let $\|\cdot\|_a$ and $\|\cdot\|_b$be two norms on X. There exist $C, c > 0$ such that for all $x \in X$

$$c\|x\|_a \le \|x\|_b \le C\|x\|_a.$$

4.1.5. *Conditioning.* Let $A \in \mathbb{R}^{n\times n}$ invertible, $f : \mathbb{R}^n \to \mathbb{R}^n$, $b \mapsto Ab$, $g : \mathbb{R}^n \to \mathbb{R}^n$, $b \mapsto A^{-1}b$. Then, for all $b \in \mathbb{R}^n$

$$\begin{aligned} \kappa_f(b) &= \lim_{\delta\to 0} \sup_{\|\Delta b\|<\delta} \frac{\|A(b+\Delta b) - Ab\| \cdot \|b\|}{\|Ab\| \cdot \|\Delta b\|} = \frac{\|A\| \cdot \|b\|}{\|Ab\|} \\ \kappa_g(b) &= \frac{\|A^{-1}\| \cdot \|b\|}{\|A^{-1}b\|}. \end{aligned}$$

4.1.6. *Condition of a matrix.* For $A \in \mathbb{R}^{n\times n}$ invertible we set

$$\kappa(A) = \|A^{-1}\| \cdot \|A\|,$$

since $\kappa_f(A^{-1}b) = \|A\| \cdot \|A^{-1}b\|/|b\| \le \kappa(A)$ and $\kappa_g(Ab) = \|A^{-1}\| \cdot \|Ab\|/\|b\| \le \kappa(A)$ for all $b \in \mathbb{R}^n$. In particular, $\kappa(\mathrm{Id}) = 1$.

```
A = [0.2161, 0.1441; 1.2969 0.8648]; b = [0.144; 0.8642]; x = A\b;
bp = b + 1e-8*rand(2,1); xp = A\bp;
k = norm(xp - x)/norm(x)*norm(b)/norm(bp-b),  kappa = cond(A)
```

4.1.7. *Preconditioning.* An invertible $B \in \mathbb{R}^{n\times n}$ with $\kappa(B)$ small, $\kappa(BA) \ll \kappa(A)$ is called a preconditioner for A.

```
A = [0.2161, 0.1441; 1.2969 0.8648]; B = diag(diag(A)); cond(B\A)
```

Literature. [S1, Lecture 16], [TB, Lecture 3]

4.2. **Gaussian elimination (26.11.)**

4.2.1. *Triangular matrices.* Let $L, U \in \mathbb{R}^{n\times n}$ be lower and upper triangular matrices, $l_{ij} = 0$ for $i < j$ and $u_{ij} = 0$ for $i > j$, respectively. Note that $\det(L) = l_{11}\cdots l_{nn}$ and $\det(U) = u_{11}\cdots u_{nn}$. We write in partioned form

$$L = \begin{pmatrix} \lambda & 0 \\ l & L_* \end{pmatrix}, \qquad U = \begin{pmatrix} \mu & U_* \\ 0 & u^T \end{pmatrix}$$

with $L_*, U_* \in \mathbb{R}^{(n-1)\times(n-1)}$ lower and upper triangular, $l, u \in \mathbb{R}^{n-1}$, $\lambda, \mu \in \mathbb{R}$.

4.2.2. *Forward substitution.* Let L, U be lower and uper triangular invertible matrices, that is $l_{ii} \neq 0$ and $u_{ii} \neq 0$ for all i. We write $Lx = b$ as

$$\begin{pmatrix} \lambda & 0 \\ l & L_* \end{pmatrix}\begin{pmatrix} x_1 \\ x_* \end{pmatrix} = \begin{pmatrix} \lambda x_1 \\ x_1 l + L_* x_* \end{pmatrix} = \begin{pmatrix} \beta \\ b_* \end{pmatrix},$$

and observe that $L_* x_* = b_* - x_1 l$ is a lower triangular system in $n-1$ dimensions.

```
n = 100; L = tril(rand(n))+eye(n); spy(L);
b = rand(n,1); xx = L\b;
for i=1:n, x(i)=b(i)/L(i,i); b(i+1:n)=b(i+1:n)-x(i)*L(i+1:n,i);
end, error = norm(x-xx)
```

4.2.3. *Stability & operation count.* Forward and back substitution are stable algorithms for solving triangular systems using $\sum_{i=1}^{n}(2(n-i)+1) = n^2$ flops.

4.2.4. *LU decomposition.* Let $A \in \mathbb{R}^{n\times n}$ be invertible admitting a decomposition $A = LU$ with L lower unit triangular ($l_{ii} = 1$ for all i) and U upper triangular,

$$\begin{pmatrix} \alpha & a^T \\ b & A_* \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ l & L_* \end{pmatrix}\begin{pmatrix} \mu & u^T \\ 0 & U_* \end{pmatrix} = \begin{pmatrix} \mu & u^T \\ \mu l & lu^T + L_* U_* \end{pmatrix}.$$

We oberve $u = a$, $l = b/\mu$ and $L_* U_* = A_* - lu^T$.

```
n=100; A=rand(n); LL=tril(A,-1)+eye(n); UU=triu(A)+eye(n);
A=LL*UU; L=eye(n); U=zeros(n,n);
for i=1:n, U(i,i:n)=A(i,i:n); L(i+1:n,i)=A(i+1:n,i)/A(i,i);
A(i+1:n,i+1:n)=A(i+1:n,i+1:n)-L(i+1:n,i)*U(i,i+1:n); end,
norm(L*U-LL*UU)
```

4.2.5. *Operation count.* $\sum_{i=1}^{n}(2(n-i)^2 + (n-i)) \approx \int_0^n 2x^2 dx = \frac{2}{3}n^3$ flops. Solving $Ax = b$ via $Ly = b$, $Ux = y$ is dominated by the cost for the LU decomposition.

4.2.6. *Pivoting.* If A is invertible, then the maximal entry in magnitude of each column vector $a_{i:n,i}$ does not vanish. We permute rows to divide by this maximal entry. This also yields $|l_{i,j}| \leq 1$ for all i, j, which is important for stability, since then

$$\kappa_p(L) = \|L\|_p \cdot \|L^{-1}\|_p \leq n \cdot 2^{n-1}, \qquad p = 1, \infty.$$

4.2.7. *Cholesky decomposition.* If $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite, that is, $A = A^T$ and $x^T Ax > 0$ for all $x \neq 0$, then there exists a lower triangular matrix with positive diagonal entries such that $A = LL^T$, the Cholesky decomposition. For stability, however, pivoting is recommended.

Literature. [TB, Lecture 20 & 21]

4.3. **QR decomposition (1.12.)**

4.3.1. *Orthonormal basis.* A set of vectors $q_1, \dots, q_n \in \mathbb{R}^n$ forms an orthonormal basis of $\mathbb{R}^n$, if

$$q_j^T q_k = \delta_{jk}, \qquad j,k = 1,\dots,n.$$

In particular, $\|q_j\|_2 = \sqrt{q_1^2 + \dots + q_n^2} = 1$ for all j, and for all $x \in \mathbb{R}^n$,

$$x = (q_1^T x)q_1 + \dots + (q_n^T x)q_n, \qquad \|x\|_2 = \sqrt{(q_1^T x)^2 + \dots + (q_n^T x)^2}.$$

4.3.2. *Orthogonal matrix.* If the columns or the rows of a matrix $Q \in \mathbb{R}^{n\times n}$ form an orthonormal basis of $\mathbb{R}^n$, then Q is called an orthogonal matrix. This is the case, if and only if $QQ^T = \mathrm{Id} = Q^TQ$. Each orthogonal matrix Q satisfies

$$\|Q\|_2 = 1, \qquad \kappa_2(Q) = 1.$$

4.3.3. *Gaussian elimination.* The lower unit triangular matrix with $l_{ij} = -1$ below the diagonal satisfies $\kappa_{1,\infty}(L) = n \cdot 2^{n-1}$. It is not stable under smart pivoting.

```
n = 100; L = -tril(ones(n),-1) + eye(n); cond(L,1) , n*2^(n-1),
[L,U] = lu(L*U); cond(L)
```

4.3.4. *Householder reflectors.* Let $v \in \mathbb{R}^n$, $v \neq 0$. The Householder reflector

$$H_v = \mathrm{Id} - 2vv^T/\|v\|_2^2$$

is symmetric and orthogonal. Since $x \mapsto (v^Tx)v/\|v\|_2^2$ orthogonally projects on $\mathrm{span}(v)$, $x \mapsto H_v x$ reflects about the hyperplane, whose vectors are orthogonal to v.

4.3.5. *Householder elimination.* Let $\|x\|_2 = 1$ and $v = x \pm e_1$. Then, $\|v\|_2^2 = 2(1 \pm x_1)$ and $H_v x = \mp e_1$. Let $x \neq 0$ and $v = x/\|x\|_2 \pm e_1$. Then,

$$H_v x = \|x\|_2 H_v x/\|x\|_2 = \mp\|x\|_2 e_1$$

eliminates below the first entry.

4.3.6. *QR decomposition.* Let $A \in \mathbb{R}^{n\times n}$ be invertible. Then, there exist a unique orthogonal matrix Q and an upper triangular matrix R with positive diagonal entries such that $A = QR$.

4.3.7. *Householder triangularization.* The Q factor can be built multipliying matrices of the form

$$\begin{pmatrix} \mathrm{Id}_k & 0 \\ 0 & H_{v_{n-k}} \end{pmatrix}, \qquad k = 0,\dots,n-1.$$

The operation count is $\frac{4}{3}n^3$ flops, which is twice as much as for the LU decomposition. Solving $Ax = b$ by back substitution of $Rx = Q^Tx$ is dominated by the cost for the QR decomposition.

4.3.8. *Gram-Schmidt.* Let $a_1, \dots, a_n$ and $q_1, \dots, q_n$ be the columns of A and Q, respectively. The q_j build an orthonormal basis of $\mathbb{R}^n$ such that

$$\mathrm{span}(a_1,\dots,a_k) = \mathrm{span}(q_1,\dots,q_k), \qquad k = 1,\dots,n.$$

since $A = QR$, that is, $a_k = Q(r_{1k},\dots,r_{kk},0,\dots,0)^T = r_{1k}q_1 + \dots + r_{kk}q_k$.

Literature. [S2, Lectures 5 & 8]

4.4. **Least squares problems (8.12.)**

4.4.1. *Householder triangularization.* Let $A \in \mathbb{R}^{n\times n}$ be invertible. We seek an orthogonal Q and an upper triangular R such that $A = QR$. Let a be the first column of A and $v = a/\|a\|_2 + \mathrm{sgn}(a_1)e_1$. Then, $\|v\|_2^2 = 2 + 2|a_1|/\|a\|_2 > 0$ and

$$H_v A = \begin{pmatrix} \alpha & b^T \\ 0 & A_* \end{pmatrix}$$

with $\alpha = -\mathrm{sgn}(\mathrm{a}_1)\|a\|_2$, $b \in \mathbb{R}^{n-1}$ and $A_* \in \mathbb{R}^{(n-1)\times(n-1)}$ invertible. Analogous eliminations produce vectors $v_{n-k} \in \mathbb{R}^{n-k}$ such that Q^T is a product of matrices

$$\begin{pmatrix} \mathrm{Id}_k & 0 \\ 0 & H_{v_{n-k}} \end{pmatrix}, \qquad k = 0, \dots, n-1.$$

4.4.2. *Rectangular matrices.* Let $A \in \mathbb{R}^{m\times n}$, $m \geq n$, have n independent columns. Then there exists a unique decomposition $A = QR$ with an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ and $R \in \mathbb{R}^{m\times n}$, $r_{jj} > 0$ and $r_{ij} = 0$ for all $i > j$,

$$A = \begin{pmatrix} \times & \times \\ \times & \times \\ \times & \times \end{pmatrix} = \begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{pmatrix} \begin{pmatrix} \times & \times \\ 0 & \times \\ 0 & 0 \end{pmatrix} = QR.$$

4.4.3. *Least squares problem.* Let A as above and $b \in \mathbb{R}^m$. We seek the unique $x \in \mathbb{R}^n$ such that

$$\|Ax - b\|_2^2 = \sum_{j=1}^{m} \left((Ax)_j - b_j\right)^2 = \min!$$

This x is the orthogonal projection of b onto the linear span of the columns of A.

4.4.4. *Least squares via QR.* If $A = QR$, then

$$\begin{aligned} \|Ax - b\|_2^2 &= \|Rx - Q^T b\|_2^2 \\ &= \|R(1:n,:)x - (Q^T b)(1:n)\|_2^2 + \|(Q^T b)(n+1:m)\|_2^2, \end{aligned}$$

such that it remains to solve a $n \times n$ upper triangular system for determining x,

$$R(1:n,:)x = (Q^T b)(1:n).$$

4.4.5. *Normal equation.* x is the orthogonal projection of b onto the column span of A, iff the normal equation is satisfied,

$$A^T(Ax - b) = 0, \qquad A^T A x = A^T b.$$

4.4.6. *Cholesky decomposition.* Since $A^T A$ is symmetric and positive definite, one uses Cholesky decomposition: $A^T A x = LL^T x = A^T b$ is solved via

$$Ly = A^T b, \qquad L^T x = y.$$

```
d = 1e-7; A = [1,1; d,0;0,d]; b = [2;d;d]; x = [1;1];
x1 = A\b; norm(x1-x)/norm(x)
[Q,R] = qr(A);  b2 = Q'*b; x2 = R(1:2,:)\b2(1:2); norm(x2-x)/norm(x)
x3 = (A'*A)\(A'*b); norm(x3-x)/norm(x), cond(A'*A)
```

Literature. [S2, Lecture 8]

5. Iterative solvers

5.1. Jacobi & Gauß–Seidel iteration (10.12.)

5.1.1. *Jacobi iteration.* We write the ith row of the linear system $Ax = b$ as $x_i = (b_i - \sum_{j\neq i} a_{ij}x_j)/a_{ii}$. We construct a sequence $(x^k)_{k>0}$ in $\mathbb{R}^n$ by the Jacobi iteration

$$x_i^{k+1} = (b_i - \sum_{j\neq i} a_{ij}x_j^k)/a_{ii}$$

and hope for $\lim_{k\to\infty} x^k = A^{-1}b$.

5.1.2. *Gauß–Seidel iteration.* The Gauß–Seidel iteration is defined by

$$x_i^{k+1} = (b_i - \sum_{j<i} a_{ij}x_j^{k+1} - \sum_{j>i} a_{ij}x_j^k)/a_{ii}.$$

5.1.3. *Matrix formulation.* Let $A = D - L - U = \mathrm{diag}(A) + \mathrm{tril}(A,-1) + \mathrm{triu}(A,1)$. We write the Jacobi iteration as

$$x^{k+1} = D^{-1}b + D^{-1}(L+U)x^k.$$

The Gauß-Seidel iteration takes the form

$$x^{k+1} = D^{-1}(b + Lx^{k+1} + Ux^k) = (D-L)^{-1}b + (D-L)^{-1}Ux^k.$$

```
n = 100; A = rand(n); for i=1:n, A(i,i)= sum(abs(A(i,:))); end,
b = rand(n,1); D = diag(diag(A)); L = -tril(A,-1); U =-triu(A,1);
m = 10; x = rand(n,1); y = x;
for k=1:m, k; x=D\b + D\(L+U)*x; y=(D-L)\b + (D-L)\U*y;
norm(x-A\b), norm(y-A\b), pause, end
```

5.1.4. *Splitting.* Let $A = M - N$ with M invertible. We write $Ax = (M-N)x = b$ as $x = M^{-1}b + M^{-1}Nx$ and derive a so-called stationary iteration

$$x^{k+1} = M^{-1}b + M^{-1}Nx^k.$$

The Jacobi and Gauß-Seidel iteration are stationary with $A = D - (L+U)$ and $A = (D-L) - U$, respectively.

5.1.5. *Convergence.* We write $x^{k+1} - x = M^{-1}N(x^k - x) = (M^{-1}N)^{k+1}(x^0 - x)$. Sufficient conditions for convergence are

$$\lim_{k\to\infty} (M^{-1}N)^k = 0, \qquad \|M^{-1}N\| < 1.$$

```
norm(D\(L+U)), norm((D-L)\U)
```

5.1.6. *Strictly diagonally dominant matrices.* $A \in \mathbb{R}^{n\times n}$ is called strictly diagonally dominant, if

$$|a_{ii}| > \sum_{j\neq i} |a_{ij}|, \qquad i = 1,\ldots,n.$$

For these matrices one can prove, that a splitting $A = M - N$ provides a convergent stationary iteration, if $\mathrm{diag}(N) = 0$ and $|a_{ij}| = |m_{ij}| + |n_{ij}|$ for all i, j. This implies convergence of the Jacobi and Gauß–Seidel iteration.

Literature. [S2, Lecture 25]

5.2. **Conjugate gradient (15.12.)**

5.2.1. *Energy functional.* Let $A \in \mathbb{R}^{n\times n}$ be symmetric, positive definite, $b \in \mathbb{R}^n$, and $x_* = A^{-1}b$. Consider the energy functional

$$\phi(x) = \tfrac{1}{2}(x - x_*)^T A(x - x_*), \qquad x \in \mathbb{R}^n.$$

We have $\phi(x) \geq 0$ for all x and $\phi(x) = 0$ iff $x = x_*$. We seek the minimizer of ϕ.

5.2.2. *The idea.* Given an iterate $x_k \in \mathbb{R}^n$ and a search direction $s_k \in \mathbb{R}$, we determine the unique $\alpha_k \in \mathbb{R}$ with

$$\phi(x_k + \alpha_k s_k) = \min_{\alpha\in\mathbb{R}} \phi(x_k + \alpha s_k),$$

and set $x_{k+1} = x_k + \alpha_k s_k$.

5.2.3. *The length.* Consider the kth residual $r_k = b - Ax_k$. We minimize

$$\alpha \mapsto \phi(x_k + \alpha s_k) = \|A^{-1/2}r_k - \alpha A^{1/2}s_k\|_2^2.$$

The normal equation $s_k^T A^{1/2}A^{1/2}s_k\alpha = s_k^T A^{1/2}A^{-1/2}r_k$ of this least squares problem is uniquely solved by $\alpha_k = (s_k^T r_k)/(s_k^T A s_k)$.

5.2.4. *A-conjugacy and orthogonality.* From $r_{k+1} = r_k - \alpha_k A s_k$ follows $s_k^T r_{k+1} = 0$. Minimizing

$$a \mapsto \phi(x_k + \alpha_k s) = \|A^{-1/2}r_k - \alpha_k A^{1/2}s\|_2^2,$$

we obtain $\alpha_k A s_k = r_k$ as well as $s_{k-1}^T A s_k = 0$ and $s_j^T r_{k+1} = 0$ for $j \leq k$.

5.2.5. *The search direction.* We start with $s_1 = r_1$. The educated ansatz $s_{k+1} = r_{k+1} - \beta_k s_k$ provides

$$\operatorname{span}\{r_1, \ldots, r_{k+1}\} = \operatorname{span}\{s_1, \ldots, s_{k+1}\} = \operatorname{span}\{r_1, Ar_1, \ldots, A^k r_1\},$$

and $\beta_k = (s_k^T A r_{k+1})/(s_k^T A s_k)$. Moreover, the r_k are mutually orthogonal.

5.2.6. *Krylov spaces.* Let $A \in \mathbb{R}^{n\times n}$ and $x \in \mathbb{R}^n$. $\operatorname{span}\{x, Ax, \ldots, A^{k-1}x\}$ is called the kth Krylov space associated with A and x.

5.2.7. *The coefficients of CG.* From $s_k = r_k - \beta_k s_{k-1}$ we learn $s_k^T r_k = \|r_k\|_2^2$. From $r_{k+1} = r_k - \alpha_k A s_k$ we deduce $\|r_{k+1}\|_2^2 = -\alpha_k s_k^T A r_{k+1}$. Hence,

$$\alpha_k = \|r_k\|_2^2/(s_k^T A s_k), \qquad \beta_k = -\|r_{k+1}\|_2^2/\|r_k\|_2^2.$$

```
n = 100; A = rand(n); A = A'*A + eye(n); b = rand(n,1); xx = A\b;
m = 30; x = rand(n,1); r = zeros(n,m); r(:,1) = b-A*x; s = b-A*x;
for k=1:m, alpha = norm(r(:,k))^2/(s'*A*s); x = x+ alpha*s;
e = norm(x-xx), pause,  r(:,k+1) = r(:,k)-alpha*A*s;
beta = -norm(r(:,k+1))^2/norm(r(:,k))^2; s = r(:,k+1)-beta*s; end
cond(A,2)
```

5.2.8. *Convergence.* One can prove for $e_k = x_k - x_*$

$$\|e_k\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|e_1\|_A$$

with $\|y\|_A = \sqrt{y^T A y}$ for $y \in \mathbb{R}^n$ and $\kappa = \kappa_2(A)$.

Literature: [S2, Lecture 21–22]

5.3. **Newton's method (17.12.)**

5.3.1. *The task.* We compute zeros of $f : \mathbb{R} \to \mathbb{R}$, that is, x_* with $f(x_*) = 0$.

5.3.2. *Conditioning.* We use the norm

$$\|\Delta f\|_\infty = \sup\{|\Delta f(x)| : x \in \mathbb{R}\}.$$

We assume that $f + \Delta f$ has a zero $\widetilde{x}_*$ with $\widetilde{x}_* \to x_*$ as $\|\Delta f\|_\infty \to 0$. Since $f(\widetilde{x}_*) = -\Delta f(\widetilde{x}_*)$ and $f(\widetilde{x}_*) = f'(x_*)(\widetilde{x}_* - x_*) + O(|\widetilde{x}_* - x_*|^2)$, we obtain

$$\kappa_{x_*}(f) = \lim_{\delta \to 0} \sup_{\|\Delta f\|_\infty < \delta} \frac{|\widetilde{x}_* - x_*| \cdot \|f\|_\infty}{\|\Delta f\|_\infty \cdot |x_*|} = \frac{\|f\|_\infty}{|f'(x_*)| \cdot |x_*|}.$$

Therefore, zeros with small first derivative are badly conditioned.

5.3.3. *Newton's method.* Starting from x_0, we construct a sequence $x_1, x_2, \ldots$ with $\lim_{k\to\infty} x_k = x_*$. Newton's method defines

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

This can be motivated via $0 = f(x_*) = f(x_k) + f'(x_k)(x_* - x_k) + O(|x_* - x_k|^2)$.

5.3.4. *Quadratic convergence.* If $f'(x_*) \neq 0$ and x_0 is sufficiently close to x_*, then $\lim_{k\to\infty} x_k = x_*$ and

$$\lim_{k\to\infty} \frac{x_{k+1} - x_*}{(x_k - x_*)^2} = \frac{f''(x_*)}{2f'(x_*)}.$$

5.3.5. *Fixed point methods.* Let $\varphi(x) = x - f(x)/f'(x)$. Newton's method is $x_{k+1} = \varphi(x_k)$ and computes a fixed point of φ, that is, x_* with

$$\varphi(x_*) = x_*.$$

5.3.6. *Systems.* For zeros of $f : \mathbb{R}^n \to \mathbb{R}^n$, Newton's method is

$$x_{k+1} = x_k - f'(x_k)^{-1} f(x_k),$$

where $f'(x_k) \in \mathbb{R}^{n\times n}$ is the Jacobian of f at x_k. This requires two steps: solve the linear system $f'(x_k)y_k = f(x_k)$, update $x_{k+1} = x_k - y_k$.

```
x = linspace(-2,2); [X1,X2] = meshgrid(x); Y = sin(pi/2*X1) + X2.^3;
N = @(x) x - [2*x(1), 2*x(2); pi/2*cos(pi/2*x(1)), 3*x(2)^2]\...
[x(1)^2 + x(2)^2-1; sin(pi/2*x(1)) + x(2)^3];
contour(X1,X2,Y), colorbar, hold on, t = linspace(0,2*pi);
plot(cos(t),sin(t),'-r'), x = [1.8;-1.8]; plot(x(1),x(2),'*r')
for k=1:10, x = N(x); plot(x(1),x(2),'*r'), pause, end
```

5.3.7. *Termination.* For x_k sufficiently close to x_*,

$$\frac{1}{4\kappa} \cdot \frac{\|x_k - x_*\|}{\|x_k - x_0\|} \leq \frac{\|f(x_k)\|}{\|f(x_0)\|} \leq 4\kappa \cdot \frac{\|x_k - x_*\|}{\|x_k - x_0\|}$$

with $\kappa = \kappa(f'(x_*))$. One might terminate, if

$$\|f(x_k)\| \leq \tau_{\text{rel}} \|f(x_0)\| + \tau_{\text{abs}}.$$

Alternatively, one might terminate, if $\|y_k\| \leq \tau_{\text{rel}} \|x_k - x_0\|$, since $\|x_k - x_*\| = \|y_k\| + O(\|x_k - x_*\|^2)$.

Literature. [S1, Lecture 2,3 & 5]

6. Eigenvalues

6.1. **Power iterations (7.1.)**

6.1.1. *The task.* Let $A \in \mathbb{C}^{n\times n}$. We are interested in computing eigenvalues $\lambda \in \mathbb{C}$ and eigenvectors $x \in \mathbb{C}^n \setminus \{0\}$ of A. An eigenpair (x, λ) of A is defined by the relation

$$Ax = \lambda x.$$

In the following, we always work with normalized eigenvectors, $\|x\|_2 = 1$. We call an eigenvalue λ dominant, if $|\lambda| > |\mu|$ for all other eigenvalues μ.

6.1.2. *Power iteration.* We assume that $A \in \mathbb{C}^{n\times n}$ has a basis of eigenvectors $x_1, \ldots, x_n \in \mathbb{C}^n$, that is, we can write any $v \in \mathbb{C}^n$ as $v = \alpha_1 x_1 + \ldots + \alpha_n x_n$. Then,

$$A^k v = \alpha_1 \lambda_1^k x_1 + |\lambda_1|^k \left(\alpha_2 \frac{\lambda_2^k}{|\lambda_1|^k} x_2 + \ldots + \alpha_n \frac{\lambda_n^k}{|\lambda_1|^k} x_n \right).$$

If λ_1 is dominant and $\alpha_1 \neq 0$, then $\operatorname{dist}(A^k v, \operatorname{span}(x_1)) = O(|\lambda_2/\lambda_1|^k)$, where we have numbered $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$.

```
A=-[1 2;3 4]; [V,D]=eig(A);
plot([0,V(1,2),-V(1,2)],[0,V(2,2),-V(2,2)],'r-'),
axis([-1 1 -1 1]), hold on, v=[1;0]; plot(v(1),v(2),'o'),
for k=1:3, v=A*v/norm(A*v); plot(v(1),v(2),'o'), pause, end
```

6.1.3. *Rayleigh quotient.* Let $A \in \mathbb{C}^{n\times n}$. For $x \in \mathbb{C}^n$ we set $r(x) = (x^*Ax)/(x^*x)$. If (x, λ) is an eigenpair of A, then $r(x) = \lambda$. Let $y \in \mathbb{C}^n$ with $\|y\|_2 = 1$ and $y^*x = 0$. Then, for all $h > 0$

$$r(x + hy) = \frac{\lambda + hx^*Ay + h^2}{1 + h^2} = \lambda + O(h).$$

If $A = A^*$, then $r(x + hy) = \lambda + O(h^2)$.

```
A=[1 2;3 4]; ll=eig(A), v=[1;0];
for k=1:10, v=A*v/norm(A*v); l(k)=v'*A*v; end,
A=[1 2;2 3]; lls=eig(A), v=[1;0];
for k=1:10, v=A*v/norm(A*v); ls(k)=v'*A*v; end,
semilogy([1:10],abs(l-ll(2)),[1:10],abs(ls-lls(2)))
```

6.1.4. *Inverse iteration.* Let $A \in \mathbb{C}^{n\times n}$ be invertible. (x, λ) is an eigenpair of A if and only if (x, λ^{-1}) is an eigenpair of A^{-1}. The inverse iteration

$$v_k = A^{-1}v_{k-1}/\|A^{-1}v_{k-1}\|_2 = A^{-k}v_0/\|A^{-k}v_0\|_2, \qquad k \geq 1,$$

approximates the eigenvector x_n of A with convergence rate $O(|\lambda_n/\lambda_{n-1}|^k)$.

```
A=[1 2;3 4]; ll=eig(A); v=[1;0];
for k=1:10, v=A\v; v=v/norm(v); l(k)=v'*A*v; end
semilogy([1:10],abs(l-ll(1)),'r')
```

6.1.5. *Shifted inverse iteration.* Let $\sigma \in \mathbb{C}$. The shifted inverse iteration

$$v_k = (A - \sigma \mathrm{Id})^{-1} v_{k-1}, \qquad v_k = v_k/\|v_k\|_2,$$

approximates the eigenpair (x, λ) with $|\lambda - \sigma| < |\mu - \sigma|$ for all other eigenvalues μ.

Literature. [TB, Lecture 27]

6.2. **QR iteration (12.1.)**

6.2.1. *Unitary transformation.* Let $A, Q \in \mathbb{C}^{n\times n}$ and Q unitary, that is, $Q^*Q = \mathrm{Id} = QQ^*$. Then, Q^*AQ has the same eigenvalues as A, since $Ax = \lambda x$ is equivalent to $Q^*AQQ^*x = \lambda Q^*x$.

6.2.2. *Schur decomposition.* Let (q_1, λ_1) be an eigenpair of $A \in \mathbb{C}^{n\times n}$, $\|q_1\|_2 = 1$. Let $Q = (q_1, Q_\diamond) \in \mathbb{C}^{n\times n}$ be unitary. We compute

$$Q^*AQ = \begin{pmatrix} q_1^*Aq_1 & q_1^*AQ_\diamond \\ Q_\diamond^*Aq_1 & Q_\diamond^*AQ_\diamond \end{pmatrix} = \begin{pmatrix} \lambda_1 & q_1^*AQ_\diamond \\ 0 & Q_\diamond^*AQ_\diamond \end{pmatrix}.$$

The eigenvalues of $Q_\diamond^*AQ_\diamond$ and λ_1 are the eigenvalues of A. The process can be continued to obtain an upper triangular matrix $Q^*AQ = T$ whose diagonal elements are the eigenvalues of A.

6.2.3. *Hermitian matrices.* If $A = A^* \in \mathbb{C}^{n\times n}$ is hermitian, then the Schur decomposition $Q^*AQ = T$ is hermitian, too, and T is a diagonal matrix whose diagonal elements are the eigenvalues of A.

6.2.4. *Left eigenvectors.* Let $A \in \mathbb{C}^{n\times n}$. The eigenvalues of A^* are the complex conjugate eigenvalues of A. Eigenvectors of A^* are called left eigenvectors of A, since $A^*x = \overline{\lambda}x$ is equivalent to $x^*A = \lambda x^*$.

6.2.5. *Schur decomposition, repeated.* Let (q_n, λ_n) be a left eigenpair of $A \in \mathbb{C}^{n\times n}$, $\|q_n\|_2 = 1$. Let $Q = (Q_\diamond, q_n) \in \mathbb{C}^{n\times n}$ be unitary. We have

$$Q^*AQ = \begin{pmatrix} Q_\diamond^*AQ_\diamond & Q_\diamond^*Aq_n \\ 0 & \lambda_n \end{pmatrix}.$$

6.2.6. *QR iteration idea.* Let $\kappa \in \mathbb{C}$ and $A - \kappa\mathrm{Id} = QR$. Then,

$$q_n^* = (Qe_n)^* = e_n^*Q^* = e_n^*R(A - \kappa\mathrm{Id})^{-1} = r_{nn}e_n^*(A - \kappa\mathrm{Id})^{-1}$$

Since $\|q_n\|_2 = 1$, we have $q_n^* = e_n^*(A - \kappa\mathrm{Id})^{-1}/\|e_n^*(A - \kappa\mathrm{Id})^{-1}\|_2$. That is, the nth column of Q is an approximate left eigenvector of A and can be used for an approximate Schur decomposition of A.

6.2.7. *Similarity transformation by RQ.* $A - \kappa\mathrm{Id} = QR$ is equivalent to $RQ = Q^*(A - \kappa\mathrm{Id})Q = Q^*AQ - \kappa\mathrm{Id}$ and $Q^*AQ = RQ + \kappa\mathrm{Id}$.

```
A=ones(3,3)+diag([1,2,3]), eig(A),
for k=1:10, [Q,R]=qr(A); A=R*Q, pause, end
```

6.2.8. *Rayleigh quotient shift.* Let $A^{(k)}$ be the kth iterate of the QR iteration. Then,

$$\kappa = e_n^*A^{(k)}e_n = a_{nn}$$

is viewed as an approximate eigenvalue.

```
A=ones(3,3)+diag([1,2,3]); lr = eig(A); l=[];
while length(A(:))>1, kappa=A(end,end);
[Q,R]=qr(A-kappa*eye(size(A))); A=R*Q+kappa*eye(size(A)), pause
if norm(A(end,1:end-1))<1e-4, l=[l;A(end,end)];
A=A(1:end-1,1:end-1); end, end, l=[l;A(end,end)], lr
```

Literature. [S2, Lecture 12 & 15]

6.3. **Singular value decomposition (14.1.)**

6.3.1. *Hermitian matrices.* For any hermitian $A \in \mathbb{C}^{n\times n}$ there exists a unitary $U \in \mathbb{C}^{n\times n}$ and a diagonal $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^{n\times n}$ with

$$A = U\Lambda U^*.$$

The λ_i are the eigenvalues of A, the columns of U are eigenvectors of A.

6.3.2. *Singular value decomposition.* For any $A \in \mathbb{C}^{m\times n}$ with $m \geq n$, there exist unitary matrices $U \in \mathbb{C}^{m\times m}$, $V \in \mathbb{C}^{n\times n}$ and a diagonal rectangular matrix $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n) \in \mathbb{R}^{m\times n}$ such that

$$A = U\Sigma V^*.$$

The σ_i are called the singular values of A. The typical ordering is $\sigma_1 \geq \ldots \geq \sigma_n \geq 0$.

6.3.3. *Economy size.* For any $A \in \mathbb{C}^{m\times n}$ with $m \geq n$, there exist $U \in \mathbb{C}^{m\times n}$, $V \in \mathbb{C}^{n\times n}$ with $U^*U = \mathrm{Id}_n = V^*V$ and a diagonal $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n) \in \mathbb{R}^{n\times n}$ such that

$$A = U\Sigma V^*.$$

6.3.4. *Least squares problem.* Let $A \in \mathbb{C}^{m\times n}$ with $m \geq n$, and n independent columns. For $b \in \mathbb{C}^m$ we seek $x \in \mathbb{C}^n$ such that $\|Ax - b\|_2$ is minimal. We use an svd of A to obtain

$$\begin{aligned} \|Ax-b\|_2^2 &= \|U\Sigma V^*x - b\|_2^2 = \|\Sigma V^*x - U^*b\|_2^2 \\ &= \|\Sigma(1:n,:)V^*x - (U^*b)(1:n)\|_2^2 + \|(U^*b)(n+1:m)\|_2^2. \end{aligned}$$

Hence, $x = V\Sigma(1:n,:)^{-1}(U^*b)(1:n)$.

6.3.5. *Low rank approximation.* We rewrite an svd of $A = U\Sigma V^*$ with the unitary matrices $U = (u_1, \ldots, u_m)$ and $V = (v_1, \ldots, v_n)$ as

$$A = (\sigma_1 u_1, \ldots, \sigma_n u_n)(v_1, \ldots, v_n)^* = \sum_{i=1}^{n} \sigma_i u_i v_i^*.$$

The truncation $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^*$ has rank k and satisfies $\|A - A_k\|_2 = \sigma_{k+1}$.

```
load clown.mat, colormap('gray'), image(X), [U,S,V] = svd(X);
k = 20; Y = U(:,1:k)*S(1:k,1:k)*V(:,1:k)';
figure(2), colormap('gray'), image(Y)
```

6.3.6. *Two steps.* Most svd algorithms combine two steps: First, construct unitary matrices $U_1 \in \mathbb{C}^{m\times m}$ and $V_1 \in \mathbb{C}^{n\times n}$ such that $B = U_1 A V_1^*$ is bidiagonal, that is, all nonzero entries are on the diagonal and the first superdiagonal. Second, compute the svd of $B = U_2\Sigma V_2^*$ such that

$$A = (U_1^*U_2)\Sigma(V_1^*V_2)^*.$$

6.3.7. *Bidiagonal singular value decomposition.* For $B \in \mathbb{C}^{m\times n}$ the matrices BB^* and B^*B are hermitian and tridiagonal. The square roots of their nonzero eigenvalues are the nonzero singular values of B. There are algorithms for computing the svd of B without explicitly forming BB^*, B^*B.

Literature. [D, Chapter 3 & 5]

7. Ordinary differential equations

7.1. Euler methods (19.1.)

7.1.1. *First order ode.* Let $f : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$, $t_0 \in \mathbb{R}$, and $y_0 \in \mathbb{R}^d$. We seek a function $y : \mathbb{R} \to \mathbb{R}^d$ with

$$y'(t) = f(t, y(t)), \qquad y(t_0) = y_0 \qquad (*)$$

If f is Lipschitz continuous, that is, if there exists $L > 0$ such that $\|f(t,x) - f(t,y)\| \leq L\|x - y\|$ for all $t \in \mathbb{R}$, $x, y \in \mathbb{R}^d$, then the Picard–Lindelöf Theorem guarantees existence and uniqueness of the solution y.

7.1.2. *The evolution map.* We study $\Phi_{t,t_0} : \mathbb{R}^d \to \mathbb{R}^d$, $\Phi_{t,t_0}(y_0) = y(t)$ for $t \in \mathbb{R}$. We observe for all $\Delta y \in \mathbb{R}^d$

$$\Phi_{t,t_0}(y_0 + \Delta y) - \Phi_{t,t_0}(y_0) = D\Phi_{t,t_0}(y_0)\Delta y + O((\Delta y)^2).$$

The Wronskian $W(t, t_0) = D\Phi_{t,t_0}(y_0) \in \mathbb{R}^{d\times d}$ satisfies the variational equation

$$\partial_t W(t, t_0) = D_y f(t, y(t))W(t, t_0), \qquad W(t_0, t_0) = \mathrm{Id}.$$

7.1.3. *Conditioning.* We call

$$\kappa(t) = \|W(t, t_0)\|$$

the condition number of the ode $(*)$. It measures the sensitivity of the solution with respect to initial perturbations and depends on $D_y f$.

7.1.4. *Time discretization.* Let $h > 0$ and $t_n = t_0 + nh$ for $n \geq 0$. We construct sequences $(y_n)_{n\geq 0}$ in $\mathbb{R}^d$ with

$$y_n \approx y(t_n).$$

7.1.5. *Forward Euler.* We approximate

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(s, y(s))ds \approx y(t_n) + hf(t_n, y(t_n)),$$

and set $y_{n+1} = y_n + hf(t_n, y_n)$ for $n \geq 0$.

```
f = @(t,x) x.*(1-x); [t,yy]=ode45(f,[0:5],1/10); y(1)=1/10;
h=1; for n=1:5/h, t(n+1)=t(n)+h; y(n+1)=y(n)+h*f(t(n),y(n)); end,
plot([0:5],y,'r*-',[0:5],yy,'b')
h=1/2; for n=1:5/h, t(n+1)=t(n)+h; y(n+1)=y(n)+h*f(t(n),y(n)); end,
hold on, plot([0:0.5:5],y,'g-*')
```

7.1.6. *Backward Euler.* We approximate

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(s, y(s))ds \approx y(t_n) + hf(t_{n+1}, y(t_{n+1})),$$

and set $y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$ for $n \geq 0$.

```
for n=1:5/h, t(n+1)=t(n)+h;
y(n+1)=fsolve(@(x) x-y(n)-h*f(t(n+1),x),y(n)); end,
plot([0:0.5:5],y,'m-*')
```

7.1.7. *Order.* Both Euler methods are of order one, since $y(t_n) - y_n = O(h^2)$.

Literature: [I, 1.1 & 1.2]

7.2. **Runge–Kutta methods (21.1.)**

7.2.1. *Runge–Kutta methods.* We approximate the solution of $y'(t) = f(t, y(t))$, $y(t_0) = y_0$ at times $t_n = t_0 + nh$ via

$$y(t_{n+1}) = y(t_n) + h\int_0^1 f(t_n + hs, y(t_n + hs))ds \approx y_{n+1} = y_n + h\sum_{j=1}^{\nu} b_j f(t_n + hc_j, \xi_j)$$

with $\xi_j = y_n + h\sum_{k=1}^{\nu} a_{jk} f(t_n + hc_k, \xi_k) \approx y(t_n + hc_j)$ and $b, c \in \mathbb{R}^\nu$, $A \in \mathbb{R}^{\nu\times\nu}$. The method is explicit, if A is strictly lower triangular.

7.2.2. *Explicit two stage methods.* Let $\nu = 2$ and $A = [0, 0; \alpha, 0]$. We set $c_1 = 0$, $f_n = f(t_n, y_n)$ and compute

$$\begin{aligned} y_{n+1} &= y_n + hb_1 f_n + hb_2 f(t_n + hc_2, y_n + h\alpha f_n) \\ &= y_n + h(b_1 + b_2)f_n + h^2(b_2c_2\partial_t f(t_n, y_n) + b_2\alpha f_n \cdot \partial_y f(t_n, y_n)) + O(h^3). \end{aligned}$$

For an order two method we have the Butcher tableaux

0	0	0
α	α	0
	$1 - 1/(2\alpha)$	$1/(2\alpha)$

Choosing $\alpha = 1/2$, we obtain Heun's method $y_{n+1} = y_n + hf(t_n + \frac{h}{2}, y_n + \frac{h}{2}f_n)$.

```
f = @(t,x) -x + 2*exp(-t)*cos(2*t); t(1)=0; y(1)=0; h=1/50;
for n=1:(10/h), t(n+1)=t(n)+h; xi2=y(n)+h/2*f(t(n),y(n));
y(n+1)=y(n)+h*f(t(n)+h/2,xi2); end, x = exp(-t).*sin(2*t);
semilogy(t,abs(y-x),'b',t,h^2*ones(size(t)),'r')
```

7.2.3. *Embedding.* Let (A, b, c) have ν stages and $(\tilde{A}, \tilde{b}, \tilde{c})$ have $\tilde{\nu} \le \nu$ stages. If $A, \tilde{A}$ and $c, \tilde{c}$ extend each other, then the pair defines an embedded Runge–Kutta method with Butcher tableaux

c	A
	$\tilde{b}^T$
	b^T

.

7.2.4. *Local extrapolation.* $\tilde{y}_{n+1}$ is built from y_n instead of the less accurate $\tilde{y}_n$.

7.2.5. *Fehlberg's trick.* One sets coefficients such that the first f-evaluation of the $\tilde{\nu}$-stage ERK is the same as the last one of the ν-stage ERK (FSAL formula):

$$\begin{aligned} f(t_n + c_\nu h, \xi_\nu) &= f(t_n + c_\nu h, y_n + h\sum_{j=1}^{\nu-1} a_{\nu j} f(t_n + c_j h, \xi_j)) \\ &\overset{!}{=} f(t_{n+1}, \tilde{y}_{n+1}) = f(t_n + h, y_n + h\sum_{j=1}^{\tilde{\nu}} \tilde{b}_j f(t_n + c_j h, \tilde{\xi}_j)) \end{aligned}$$

is achieved with $c_\nu = 1$ and $A[\nu, 1 : \tilde{\nu} - 1] = \tilde{b}$.

7.2.6. *ode23 and ode45.* The Bogacki-Shampine pair embeds a 3rd order 3-stage ERK into a 2nd order 4-stage ERK. The Dormand-Prince pair embeds a 5th order 6-stage ERK into a 4th order 7-stage ERK.

Literature: [I, 3.2–3.4]

7.3. **Stability (26.1.)**

7.3.1. *The exponential function.* Let $\lambda \in \mathbb{C}$. The solution of

$$y'(t) = \lambda y(t), \qquad y(0) = 1$$

is $y(t) = \mathrm{e}^{\lambda t}$. If $\mathrm{Re}(\lambda) < 0$, then $\lim_{t\to\infty} y(t) = 0$.

```
t(1) = 0; y(1) = 1; z(1) = 1; h = 1;
for n=1:5/h, t(n+1)=t(n)+h; y(n+1)=fsolve(@(x) x-y(n)+3*h*x,y(n));
z(n+1)=z(n)-h*3*z(n); end,
subplot(1,2,1), plot(t,y,'b*',t,exp(-3*t),'g-')
subplot(1,2,2), plot(t,z,'r*-')
```

7.3.2. *A-stability.* We consider for a method $(y_n)_{n\geq 0}$ with timestep $h > 0$ the linear stability domain

$$\mathcal{D} = \{h\lambda \in \mathbb{C} : y_n \text{ generated for } y' = \lambda y, y(0) = 1 \text{ satisfies } \lim_{n\to\infty} y_n = 0\}$$

The method is called A-stable, if $\{z \in \mathbb{C} : \mathrm{Re}(z) < 0\} \subseteq \mathcal{D}$.

7.3.3. *Forward Euler.* We have $y_n = (1 + h\lambda)y_{n-1} = (1 + h\lambda)^n$. The stability domain is $\mathcal{D} = \{z \in \mathbb{C} : |z + 1| < 1\}$. The method is not A-stable.

7.3.4. *Backward Euler.* We have $y_n = y_{n-1} + h\lambda y_n$ and $y_n = (1 - h\lambda)^{-1}y_{n-1} = (1 - h\lambda)^{-n}$. Hence, $\mathcal{D} = \{z \in \mathbb{C} : |z - 1| > 1\}$ and A-stability.

7.3.5. *Trapezoidal rule.* We have $y_n = y_{n-1} + \frac{1}{2}h\lambda(y_{n-1} + y_n)$ and $y_n = (1 + \frac{1}{2}h\lambda)/(1 - \frac{1}{2}h\lambda)y_{n-1} = (1 + \frac{1}{2}h\lambda)^n/(1 - \frac{1}{2}h\lambda)^n$. Hence, $\mathcal{D} = \{z \in \mathbb{C} : \mathrm{Re}(z) < 0\}$ and A-stability.

7.3.6. *Runge–Kutta methods.* We write $\xi_j = y_n + h\lambda \sum_{k=1}^{\nu} a_{jk}\xi_k$ as $\xi = y_n\mathbf{1} + h\lambda A\xi = y_n(\mathrm{Id} - h\lambda A)^{-1}\mathbf{1}$ and $y_{n+1} = y_n + h\lambda \sum_{j=1}^{\nu} b_j\xi_j = y_n + h\lambda b^T\xi$. Hence, $y_{n+1} = (1 + h\lambda b^T(\mathrm{Id} - h\lambda A)^{-1}\mathbf{1})y_n$ and $y_n = r(h\lambda)^n$ with

$$r(z) = 1 + zb^T(\mathrm{Id} - zA)^{-1}\mathbf{1}.$$

Hence, $\mathcal{D} = \{z \in \mathbb{C} : |r(z)| < 1\}$.

7.3.7. *The rational function r.* We write

$$(\mathrm{Id} - zA)^{-1} = \mathrm{adj}(\mathrm{Id} - zA)/\det(\mathrm{Id} - zA),$$

where $\mathrm{adj}(C) \in \mathbb{C}^{\nu\times\nu}$ of a matrix $C \in \mathbb{C}^{\nu\times\nu}$ has entries $\mathrm{adj}(C)_{ij} = (-1)^{i+j}\det(C_{ji})$. Each Element of $\mathrm{adj}(\mathrm{Id}-zA)$ is a polynomial of degree $\leq \nu-1$, so that r is a rational function with $r(0) = 1$.

7.3.8. *Explicit Runge–Kutta methods.* A strictly lower triangular matrix A has $\det(\mathrm{Id} - zA) = 1$, and r is a polynomial of degree $\leq \nu$ with $r(0) = 1$. Hence, explicit Runge–Kutta methods are never A-stable.

Literature: [I, 4.2–4.3]

7.4. **Multistep methods (28.1.)**

7.4.1. *Multistep methods.* An s-step method $(y_n)_{n\geq 0}$ for solving $y'(t) = f(t, y(t))$, $y(t_0) = y_0$ on a time grid $t_n = t_0 + nh$, $n \geq 0$, is defined by

$$\sum_{m=0}^{s} a_m y_{n+m} = h \sum_{m=0}^{s} b_m f(t_{n+m}, y_{n+m})$$

with $a, b \in \mathbb{R}^{s+1}$ such that $a_s = 1$. The method is explicit, if $b_s = 0$.

7.4.2. *One-step methods.*

$$\begin{aligned}
&\text{FE}: && y_{n+1} = y_n + hf(t_n, y_n), && a = (-1, 1),\ b = (1, 0),\\
&\text{BE}: && y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}), && a = (-1, 1),\ b = (0, 1),\\
&\text{TR}: && y_{n+1} = y_n + \tfrac{h}{2}(f(t_n y_n) + f(t_{n+1}, y_{n+1})), && a = (-1, 1),\ b = (\tfrac{1}{2}, \tfrac{1}{2}).
\end{aligned}$$

7.4.3. *Adams methods.* An Adams method sets

$$y_{n+s} = y_{n+s-1} + \int_{t_{n+s-1}}^{t_{n+s}} p(t)dt = y_{n+s-1} + h \sum_{m=0}^{s} b_m f(t_{n+m}, y_{n+m})$$

where p is the polynomial of degree $\leq s$ interpolating on $f(t_n, y_n), \ldots, f(t_{n+s}, y_{n+s})$. Adams methods are s-step methods with $a = (0, \ldots, 0, -1, 1)$. Started with $y_m = y(t_m) + O(h^{s+1})$ for $m < s$, they are of order s. In the explicit case they are called Adams–Bashforth methods, otherwise Adams–Moulton methods.

7.4.4. *Stability.* A multistep method for $y' = \lambda y$, $y(0) = 1$ yields

$$\sum_{m=0}^{s} (a_m - h\lambda b_m) y_{n+m} = 0.$$

Let $z \in \mathbb{C}$ and $w_1(z), \ldots, w_s(z)$ be the roots of the so-called stability polynomial

$$w \mapsto \sum_{m=0}^{s} (a_m - z b_m) w^m.$$

Then, y_n is a linear combination of $w_1(h\lambda)^n, \ldots, w_s(h\lambda)^n$. The linear stability domain is $\mathcal{D} = \{z \in \mathbb{C} : |w_1(z)|, \ldots, |w_s(z)| < 1\}$.

7.4.5. *The second Dahlquist barrier.* The highest order of an A-stable multistep method is two. Explicit multistep methods are never A-stable.

7.4.6. *Backward differentiation.* An implicit multistep method

$$\sum_{m=0}^{s} a_m y_{n+m} = h b_s f(t_{n+s}, y_{n+s}),$$

where $a_0, \ldots, a_{s-1}$ and b_s are chosen to achieve order s is called backward differentiation formula. These coefficients satisfy

$$p'(t_{n+s}) = \tfrac{1}{b_s h} \left(a_0 p(t_n) + \cdots + a_{s-1} p(t_{n+s-1}) + p(t_{n+s})\right)$$

for all polynomials p of degree $\leq s$. BDF1 and BDF2 are A-stable.

7.4.7. *ode15s.* ode15s uses BDFSs with $1 \leq s \leq 5$.

Literature: [I, 2.1–2.3, 4.4]