Numerical Linear Algebra

MATH 1080

Theory and Solutions for Exercises

Contents

List of Figures

CHAPTER 1

A brief summary on Linear Algebra

1. Matrices

DEFINITION 1.1.

- *Let $m,n \in \mathbb{N}$. We call a* **matrix** *having m rows and n columns $A \in \mathscr{M}_{m,n}(K)$, with elements in a field $K = \mathbb{R}, C$ a set of $m \times n$ scalars $A = (a_{i,j})_{i=1:m,j=1:n}$, $a_{ij} \in K$ represented in the following rectangle array:*

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

  When $K = \mathbb{R}$ or $\mathbb{C}$ we explicitly write $A \in \mathbb{R}^{m\times n}$ or $\mathbb{C}^{m\times n}$.
- *The vector $(a_{11}, a_{22}, \cdots, a_{ss})$ is the main diagonal, where $s := \min\{m,n\}$.*
- *If $m = n$ we say A is a square matrix.*
- *The matrix A is symmetric if*

$$a_{ij} = a_{ji} \qquad \forall i,j \qquad \text{(symmetric)}$$

  and denote it $A = A^T$.
- *The matrix A is hermitian if*

$$\overline{a_{ij}} = a_{ji} \qquad \forall i,j \qquad \text{(hermitian)}$$

  and denote it $A = \overline{A^T} = A^H = A^*$.

1.1. Operation with matrices.

Let $A = (a_{ij}), B = (b_{ij}) \in \mathscr{M}_{m\times n}(\mathbb{R})$.

DEFINITION 1.2. *We say $A = B$ 'A equals B' if $a_{ij} = b_{ij} \quad \forall i = 1:m, j = 1:n$.*
Moreover, we define the operations:
matrix sum: $A + B = (a_{ij} + b_{ij})$.
The neutral element in a matrix sum is the null matrix, still denoted 0. (Matlab: `zeros(m,n)`*.)*
matrix multiplication by a scalar: $\lambda A = (\lambda_{ij}), \quad \lambda \in K$.
matrix product: $A \in \mathscr{M}(m,p), B \in \mathscr{M}(p,n) \Longrightarrow A \cdot B = C \in \mathscr{M}(m,n)$ *with entries*

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}, \qquad i = 1:m, j = 1:n.$$

The matrix product is

- *associative:* $(A \cdot B) \cdot C = A \cdot (B \cdot C)$
- *distributive wrt matrix sum:* $A \cdot (B + C) = A \cdot B + A \cdot C$

REMARK 1.1. *In general the matrix product is* **not** *commutative:* $A \cdot B \neq B \cdot A$.

COUNTEREXAMPLE 1.1.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}; \quad B = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$
$$A \cdot B = \begin{bmatrix} 1 & 3 \\ 3 & 7 \end{bmatrix} \neq B \cdot A = \begin{bmatrix} 4 & 6 \\ 3 & 4 \end{bmatrix}.$$

DEFINITION 1.3. *The square matrices for which* $A \cdot B = B \cdot A$ *holds are called commutative.*

EXAMPLE 1.1. *Diagonal matrices (and in particular the identity matrix)*

$$D = \begin{bmatrix} d_{11} & 0 & \cdots & 0 \\ 0 & d_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_{nn} \end{bmatrix} \qquad \text{(diagonal matrix)}$$

or 2×2 `rotation matrices`.

DEFINITION 1.4. *In the case of square matrices, the neutral element in the matrix product is a square matrix of order n called unit matrix or identity matrix*

$$I_n = (\delta_{ij}) = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \qquad \text{(identity matrix)}$$

which is by definition the only matrix s.t.

$$AI_n = I_n A = A, \qquad \forall A \in \mathscr{M}_{n,n}.$$

Matlab: $I_n =$ `eye`(n), *or* `eye`(n,n).

DEFINITION 1.5. *A diagonal matrix is a matrix with non-zero elements on the main diagonal.*

```
>> diag([11,22,33])
ans =
    11     0     0
     0    22     0
     0     0    33
>> A = [101 102; 201 202];
>> diag(A)
ans =
   101
   202
```

DEFINITION 1.6. *If* $A \in \mathscr{M}(n,n)$, $p \in \mathbb{N}$, *we define*

$$A^p = A * A * \cdots * A \qquad (p\text{-times}),$$
$$A^0 = I_n.$$

1.2. Inverse of a matrix.

DEFINITION 1.7. *A* **square** *matrix $A \in \mathscr{M}(n,n)$ is called* **invertible** *(or* **regular** *or* **nonsingular***) of there exists $\exists$ a square matrix $B \in \mathscr{M}(n,n)$ such that*

$$A * B = B * A = I_{n\times n}.$$

The matrix B is called the **inverse** *of A and is denoted by A^{-1}, or $\mathtt{inv}(A)$.*
A matrix which is not invertible is called **singular**.

REMARK 1.2.

- *The set of singular matrices is of 'measure' 0, but non-empty!*
  Examples:

$$A_1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}.$$

```
>> A1 = [1 2 3 ; 4 5 6; 7 8 9];
>> inv(A1)
Warning: Matrix is close to singular or badly
scaled. Results may be inaccurate. RCOND =
1.541976e-18.

ans =
   1.0e+16 *
   -0.4504    0.9007   -0.4504
    0.9007   -1.8014    0.9007
   -0.4504    0.9007   -0.4504
>> det(A1)
ans =
   6.6613e-16
```

- *If A is invertible, its inverse is also invertible, with $(A^{-1})^{-1} = A$.*

```
>> inv(inv(A1))
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND =  1.541976e-18.

Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND =  9.251859e-18.
ans =
    0.0000    0.5000    1.0000
    1.0000    0.7500    0.5000
    2.0000    1.0000         0

>> A1*inv(A1)
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND =  1.541976e-18.
ans =
```

```
   4     0     2
  12    -8     4
  16   -16    16
```

- *If A,B are invertible, $(A*B)$ is also invertible, and $(A*B)^{-1} = B^{-1}A^{-1}$.*

DEFINITION 1.8. *We call the* **transpose** *of a matrix $A \in \mathbb{R}^{m\times n}$ the matrix $B := A^T \in \mathbb{R}^{n\times m}$ obtained by exchanging the rows of A with columns of A^T:*

$$b_{ij} = a_{ji}.$$

Example:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \in \mathscr{M}(2,3), \qquad A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \in \mathscr{M}(3,2).$$

```
>> A = [1 2 3 ; 4 5 6];
>> A'
ans =
     1     4
     2     5
     3     6
```

REMARK 1.3.

- *Clearly*

$$(A^T)^T = A, \qquad (A+B)^T = A^T + B^T; \qquad (A*B)^T = B^T * A^T, \qquad (\alpha A)^T = \alpha A^T,\ \forall \alpha \in \mathbb{R}.$$

- *If A is invertible (a square matrix), then*

$$(A^T)^{-1} = (A^{-1})^T = A^{-T}.$$

DEFINITION 1.9. *Let $A \in \mathbb{C}^{m\times n}$. The matrix $B := A^H \in \mathbb{C}^{n\times m}$ is the* **conjugate transpose** *of A if*

$$b_{i,j} = \overline{a_{ji}}. \qquad \text{(conjugate transpose)}$$

DEFINITION 1.10. *The matrix $A \in \mathbb{R}^{n\times n}$ is called*

- **symmetric** *if*

$$A = A^T, \qquad \text{(symmetric)}$$

- **antisymmetric** *or* **skew-symmetric** *if*

$$A = -A^T, \qquad \text{(skew-symmetric)}$$

- **orthogonal** *if*

$$AA^T = A^T A = I_{n\times n}, \quad \textit{i.e.,} \quad A^{-1} = A^T. \qquad \text{(orthogonal)}$$

The matrix $U \in \mathbb{C}^{n\times n}$ is called

- **unitary** *if*

$$U^*U = I_{n\times n} = UU^*. \qquad \text{(unitary)}$$

Matrices A,B are

- *similar matrices if there exists a matrix P nonsingular such that*

$$B = P^{-1}AP. \tag{similar}$$

- *unitarily similar matrices if there exists a unitary matrix U such that*

$$B = U^{-1}AU. \tag{unitarily similar}$$

REMARK 1.4. *The product of two symmetric matrices is* **NOT** *a symmetric matrix:*

$$\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 5 \\ 14 & 10 \end{bmatrix} \neq \begin{bmatrix} 7 & 14 \\ 5 & 10 \end{bmatrix}.$$

1.3. Trace and determinant of a matrix.

DEFINITION 1.11. *Let $A \in \mathscr{M}(n,n)$.*

- *The* **trace** *of A is*

$$tr(A) = a_{11} + a_2 2 + \cdots + a_{nn}. \tag{trace}$$

- *The* **determinant** *of A is defined (in the Leibniz form) as*

$$\det(A) = \sum_{\pi \in S_n} \mathtt{sign}(\pi)\, a_{1\pi_1} a_{2\pi_2} \cdots a_{n\pi_n}. \tag{determinant}$$

```
>> A = [1 2 ; 3 4];
>> trace(A)
ans =
     5
>> det(A)
ans =
    -2
```

PROPOSITION 1.1.

$$det(A) = \begin{cases} a_{11}, & \text{if} \quad n = 1 \\ \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \Delta_{ij}, & \text{for} \quad n > 1 \end{cases} \tag{Laplace expansion}$$

where minors Δ_{ij} are

$$\Delta_{ij} = \det(A_{ij}), \tag{minors}$$

and A_{ij} is the matrix obtained by eliminating the i-th row and j-th column from A.

PROPOSITION 1.2. *If A is invertible, then*

$$A^{-1} = \frac{1}{\det(A)} C, \qquad \text{with}$$
$$C_{ij} = (-1)^{i+j} \Delta_{ji}, \qquad i,j = 1:n, \tag{adjugate matrix}$$

i.e., C is the transpose of the matrix of cofactors $\left((-1)^{i+j}\Delta_{ij}\right)^T$.

1.4. Special matrices. Let $A \in \mathscr{M}(n,n)$, $L,U \in \mathscr{M}(n,n)$.

$$L = \begin{bmatrix} \ell_{11} & 0 & 0 & 0 \\ \ell_{21} & \ell_{22} & 0 & 0 \\ \vdots & \cdots & \ddots & \\ \ell_{n1} & \ell_{n2} & \cdots & \ell_{nn} \end{bmatrix}, \qquad \text{(lower triangular)}$$

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \cdots & \ddots & \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix}. \qquad \text{(upper triangular)}$$

PROPOSITION 1.3.

- $det(L) = \ell_{11} \cdots \ell_{nn}, det(U) = u_{11} \cdots u_{nn}$.
- $inv(L_1) = L_2$, $inv(U_1) = U_2$.
- $L_1 * L_2 = L_3$, $U_1 * U_2 = U_3$.

DEFINITION 1.12. **Unit triangular matrix** *is any (upper- or lower-) triangular matrix which has only 1 as diagonal entries.*

1.5. Eigenvalues and eigenvectors.

DEFINITION 1.13. *Let $A \in \mathscr{M}_{n\times n}(\mathbb{R}$ or $\mathbb{C})$.*
The number $\lambda \in \mathbb{C}$ is called an **eigenvalue** *of A if there exists a non-null vector $\mathbf{0} \neq \boldsymbol{x} \in \mathbb{C}^n$ such that*

$$A\boldsymbol{x} = \lambda \boldsymbol{x}.$$

The vector $\boldsymbol{x} \in \mathbb{C}^n$ is the **eigenvector** *associated with the eigenvalue λ.*
$(\lambda, \boldsymbol{x}) \in \mathbb{C}^{n+1}$ is called an eigenpair of A.
The set of eigenvalues of A is called the **spectrum** *of A, denoted $\sigma(A)$*

$$\sigma(A) = \{\lambda \in \mathbb{C}; \quad \lambda \text{ is an eigenvalue of } A\} \subset \mathbb{C}. \qquad \text{(spectrum)}$$

REMARK 1.5.

- *The eigenvalue λ corresponding to the eigenvector $\boldsymbol{x}$ can be computed by the Rayleigh quotient*

$$\lambda = \frac{\boldsymbol{x}^H A \boldsymbol{x}}{\boldsymbol{x}^H \boldsymbol{x}}. \qquad \text{(Rayleigh quotient)}$$

- *The eigenvalue λ is the the solution of the characteristic equation*

$$p_A(\lambda) = 0, \qquad \text{(characteristic equation)}$$

where

$$p_A(r) := \det(A - rI) \qquad \text{(characteristic polynomial)}$$

is the characteristic polynomial, hence having n roots (complex numbers), i.e., the matrix $A \in \mathscr{M}(n,n)$ has n eigenvalues, not necessarily distinct!

PROPOSITION 1.4.

$$\det(A) = \prod_{i=1}^{n} \lambda_i, \qquad \operatorname{trace}(A) = \sum_{i=1}^{n} \lambda_i. \tag{1.1}$$

PROOF. This is a consequence of the Faddeev–LeVerrier algorithm,

$$0 = (-1)^n(\lambda-\lambda_1)\cdots(\lambda-\lambda_n) = \det(A-\lambda I) = p_A(\lambda)$$
$$= (-1)^n\lambda^n + (-1)^{n-1}\lambda^{n-1}\mathrm{trace}(A) + \cdots + \det(A), \quad \text{(Faddeev–LeVerrier algorithm)}$$

which calculates the coefficients of the (characteristic polynomial). □

REMARK 1.6. *From the first relation in (1.1) we see that the matrix A is* **singular** *(non-invertible) if and only if there exists an eigenvalue $\lambda_i = 0$, since*

$$p_A(0) = \det(A) = 0.$$

DEFINITION 1.14. *The maximum module of the eigenvalues of A is called the spectral radius of A is:*

$$\rho(A) = \max_{\lambda\in\sigma(A)} |\lambda|. \quad \text{(spectral radius)}$$

EXAMPLE 1.2.

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}; \quad p_A(\lambda) = \det(A-\lambda I) = \det\begin{bmatrix} 1-\lambda & 2 \\ 0 & 3-\lambda \end{bmatrix} = (1-\lambda)(3-\lambda),$$
$$\lambda_1 = 1, \lambda_2 = 3, \qquad \sigma(A) = \{1,3\}, \qquad \text{(spectrum of } A\text{)}$$
$$\det(A) = 1\cdot 3 = 3, \qquad \mathrm{trace}(A) = 1+3 = 4, \qquad \rho(A) = 3.$$

Calculate $\boldsymbol{x}_1$ (corresponding to $\lambda_1 = 1$):

$$(A-\lambda_1 I)\boldsymbol{x}_1 = 0, \qquad \begin{bmatrix} 1-1 & 2 \\ 0 & 3-1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$
$$\begin{cases} 0\cdot x + 2\cdot y = 0 \\ 0\cdot x + 2\cdot y = 0 \end{cases}, \qquad y = 0, x = 1, \qquad \boldsymbol{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Calculate $\boldsymbol{x}_2$ (corresponding to $\lambda_2 = 3$):

$$(A-\lambda_2 I)\boldsymbol{x}_2 = 0, \qquad \begin{bmatrix} 1-3 & 2 \\ 0 & 3-3 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$
$$\begin{cases} -2\cdot x + 2\cdot y = 0 \\ 0\cdot x + 0\cdot y = 0 \end{cases}, \qquad x = 1, y = 1 \qquad \boldsymbol{x}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

2. Scalar product and norms in Vector Spaces

Let $V = \mathbb{R}^n$ be a vector space.

DEFINITION 2.1. *A scalar product on a vector space V is a map $\langle\cdot,\cdot\rangle : V\times V \to \mathbb{R}$ (or $\mathbb{C}$) such that*

(1) is linear w.r.t. vectors in V:

$$\langle a\boldsymbol{x} + b\boldsymbol{y}, \boldsymbol{z}\rangle = a\langle \boldsymbol{x},\boldsymbol{z}\rangle + b\langle \boldsymbol{y},\boldsymbol{z}\rangle$$

(2) is hermitian (commutative):

$$\langle \boldsymbol{x},\boldsymbol{y}\rangle = \overline{\langle \boldsymbol{y},\boldsymbol{x}\rangle}$$

(3) positive definite:

$$\langle \boldsymbol{x},\boldsymbol{x}\rangle > 0, \qquad \forall \boldsymbol{x} \neq \boldsymbol{0}.$$

DEFINITION 2.2. *The map $\|\cdot\| : V \to \mathbb{R}_+$ is a (vector) norm if*

(1)

$$\|\boldsymbol{v}\| \geq 0 \quad \forall \boldsymbol{v} \qquad \text{(non-negativity)}$$

and

$$\|\boldsymbol{v}\| = 0 \Leftrightarrow \boldsymbol{v} = \boldsymbol{0}. \qquad \text{(positive-definiteness)}$$

(2)

$$\|\alpha \boldsymbol{v}\| \leq |\alpha| \cdot \|\boldsymbol{v}\| \qquad \text{(homogeneity)}$$

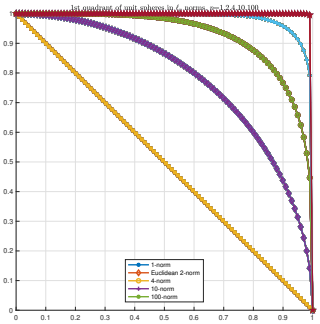
(3)

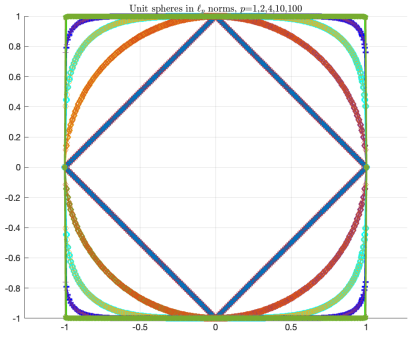
$$\|\boldsymbol{v} + \boldsymbol{w}\| \leq \|\boldsymbol{v}\| + \|\boldsymbol{w}\| \qquad \text{(triangle inequality)}$$

EXAMPLE 2.1.

- *Hölder norm:*

$$\|\boldsymbol{x}\|_p = \left(\sum x_i^p\right)^{\frac{1}{p}}, \qquad \forall 1 \leq p < \infty \qquad (\ell_p \text{ norm})$$






```
x=linspace(0,1,100);
set(groot,'defaulttextinterpreter','latex');
y1 = 1-x; y2 = (1-x.^2).^(1/2); y4 =  (1-x.^4).^(1/4);
y10 =  (1-x.^10).^(1/10); y100 =  (1-x.^100).^(1/100);
grid on; hold on
plot(x,y1,'*-',x,y2,'d-',x,y4,'o-',x,y10,'+-',...
    x,y100,'pentagram-','linewidth',2)
legend('1-norm','Euclidean 2-norm','4-norm','10-norm',...
    '100-norm','location','best')
title(['1st quadrant of unit spheres in $\ell_p$ norms, ' ...
    '$p$=1,2,4,10,100'],'interpreter','latex')
```

- *Euclidian norm ($p = 2$):* $\|\boldsymbol{x}\|_2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle = \left(\sum x_i^2\right)^{\frac{1}{2}}$.
- $\|\boldsymbol{x}\|_\infty = \max |x_i|, \qquad \|\boldsymbol{x}\|_1 = \sum |x_i|.$

$$\begin{aligned} \boldsymbol{x} &= [1;2;3;4], \\ \|\boldsymbol{x}\|_2 &= \texttt{norm}(\boldsymbol{x}) = 5.4772, \\ \|\boldsymbol{x}\|_1 &= \texttt{norm}(\boldsymbol{x}, 1) = 10, \\ \|\boldsymbol{x}\|_\infty &= \texttt{norm}(\boldsymbol{x}, \texttt{inf}) = 4, \end{aligned}$$

$$\|\boldsymbol{x}\|_4 = \texttt{norm}(\boldsymbol{x},4) = 4.3376.$$

Matlab computes vector norms:

```
>> x=[1;2;3;4]

x =

     1
     2
     3
     4

>> norm(x,2)

ans =

   5.477225575051661

>> norm(x,1)

ans =

    10

>> norm(x,inf)

ans =

     4

>> a = norm(x,4)

a =

   4.337613136533361
```

PROPOSITION 2.1 (Cauchy-Schwarz and Hölder inequalities).

$$\langle \boldsymbol{x},\boldsymbol{y}\rangle \le \|\boldsymbol{x}\|\cdot\|\boldsymbol{y}\|, \qquad \text{(Cauchy-Schwarz inequality)}$$

$$\langle \boldsymbol{x},\boldsymbol{y}\rangle \le \|\boldsymbol{x}\|_p\cdot\|\boldsymbol{y}\|_q, \qquad \forall p,q\in[1,\infty] \quad \textit{such that} \quad \frac{1}{p}+\frac{1}{q}=1. \quad \text{(Hölder inequality)}$$

```
>> x=[1;2;3;4];y=[11;12;13;14];
>> x' * y

ans =

   130
>> norm(x,2) * norm(y,2)
```

```
ans =

    137.4772708486752
>> p=4;q=4/3;
>> norm(x,p) * norm(y,q)

ans =

    153.5624642407844
```

PROPOSITION 2.2. *All norms are equivalent in finite dimensions.*

For example:

$$\|\boldsymbol{x}\|_\infty \le \|\boldsymbol{x}\|_2 \le \sqrt{n}\|\boldsymbol{x}\|_\infty, \qquad (\text{since} : \max|x_i| \le \left(\sum |x_i|^2\right)^{\frac{1}{2}} \le \sqrt{n}\max|x_i|),$$

$$\|\boldsymbol{x}\|_2 \le \|\boldsymbol{x}\|_1 \le \sqrt{n}\|\boldsymbol{x}\|_2. \qquad (\text{use the (Cauchy-Schwarz inequality) for last inequality})$$

REMARK 2.1. *The equivalence between ℓ_p norms is a consequence of the following inequality between* GENERALIZED MEANS*:*

$$M_p \le M_q, \qquad \forall p \le,$$

where

$$M_p(x_1,\cdots,x_n) := \left(\frac{1}{n}\sum_{i=1}^{n} x_i^p\right)^{\frac{1}{p}}, \qquad p \in \mathbb{R}, p \neq 0,$$

$$M_0(x_1,\cdots,x_n) := \left(\Pi_{i=1}^{n} x_i\right)^{\frac{1}{p}},$$

From Example 2.1 we see that indeed:

$$\underbrace{\|\boldsymbol{x}\|_\infty}_{4} \le \underbrace{\|\boldsymbol{x}\|_2}_{5.4772} \le \underbrace{\sqrt{2}\cdot\|\boldsymbol{x}\|_\infty}_{8},$$

$$\underbrace{\|\boldsymbol{x}\|_2}_{5.4772} \le \underbrace{\|\boldsymbol{x}\|_1}_{10} \le \underbrace{2\cdot\|\boldsymbol{x}\|_\infty}_{10.9345},$$

DEFINITION 2.3. *$\boldsymbol{x},\boldsymbol{y} \in V$ are orthogonal if*

$$\langle \boldsymbol{x},\boldsymbol{y}\rangle = 0. \qquad (\text{orthogonality})$$

EXAMPLE 2.2.

$$\boldsymbol{x} = [0;1], \quad \boldsymbol{y} = [1;0], \qquad \boldsymbol{x}^T\boldsymbol{y} = [0\ 1]\begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0$$

In $\mathbb{R}^2$ the scalar product has a geometric interpretation:

$$\langle \boldsymbol{x},\boldsymbol{y}\rangle = \|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2\cos(\theta).$$

For example, when

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by

$$\cos(\theta) = \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2} = \frac{\sqrt{2}}{2} = 0.7071,$$

hence $\theta = \pi/4$.

2.1. Matrix norms.

DEFINITION 2.4. *A matrix norm $\|\cdot\| : \mathbb{R}^{m\times n} \to \mathbb{R}$ if satisfies*

(1) $\|A\| \geq 0 \; \forall A$, *and* $\|A\| = 0 \Leftrightarrow A = 0$
(2) $\|\alpha A\| = |\alpha| \cdot \|A\|$
(3) $\|A + B\| \leq \|A\| + \|B\|$

DEFINITION 2.5. *$\|\cdot\|$ is compatible or consistent with a vector norm $\|\cdot\|$ if*

$$\|A\boldsymbol{x}\| \leq \|A\| \cdot \|\boldsymbol{x}\| \qquad \forall \boldsymbol{x} \in \mathbb{R}^n. \qquad \text{(consistent matrix norm)}$$

DEFINITION 2.6. *A matrix norm is submultiplicative if $\forall A \in \mathbb{R}^{n\times m}, B \in \mathbb{R}^{m\times q}$ we have*

$$\|AB\| \leq \|A\| \cdot \|B\|. \qquad \text{(submultiplicative)}$$

REMARK 2.2 (Counterexample). *The submultiplicative condition is NOT satisfied by all matrix norms!*

COUNTEREXAMPLE 2.1. *Let*

$$\|A\|_\triangle = \max |a_{ij}|,$$

and let

$$A = B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

then

$$AB = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$$

and therefore

$$2 = \|AB\|_\triangle > \|A\|_\triangle \|B\|_\triangle = 1.$$

EXAMPLE 2.3. *The Frobenius (Euclidian) norm: in $\mathbb{C}^{n\times n}$, compatible with the Euclidean (vector) norm $\|\cdot\|_2$:*

$$\|A\|_F = \sqrt{\sum_{i,j=1}^{n} |a_{ij}|^2} = \mathtt{tr}(AA^H). \qquad \text{(Frobenius norm)}$$

Indeed, using the Cauchy-Schwarz inequality we have

$$\|Ax\|_2^2 = \left\| \begin{pmatrix} \sum_{j=1}^n a_{1j}x_j \\ \vdots \\ \sum_{j=1}^n a_{nj}x_j \end{pmatrix} \right\|_2^2 = \sum_{i=1}^{n} \left| \sum_{j=1}^{n} a_{ij}x_j \right|^2 \leq \sum_{i=1}^{n} \left(\left(\sum_{j=1}^{n} a_{ij}^2 \right) \left(\sum_{j=1}^{n} x_j^2 \right) \right) = \sum_{j=1}^{n} x_j^2 \sum_{i,j=1}^{n} a_{ij}^2 = \|A\|_F^2 \|x\|_2^2.$$

Note that $\|I_n\|_F = \sqrt{n}$.

THEOREM 2.1. *Let $\|\cdot\|$ be a vector norm. The function*

$$\|A\| := \sup_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \qquad \text{(induced matrix norm)}$$

is a matrix norm called induced matrix norm or natural matrix norm.

REMARK 2.3. *Notice that the definition* (induced matrix norm) *is equivalent to*

$$\|A\| = \sup_{\|x\|=1} \frac{\|Ax\|}{\|x\|},$$

since

$$\sup_{\|x\|=1} \frac{\|Ax\|}{\|x\|} = \sup_{\boldsymbol{y}\neq\boldsymbol{0}} \left\| A\left(\frac{1}{\|\boldsymbol{y}\|}\boldsymbol{y}\right)\right\|, \qquad \text{as } \left\|\frac{1}{\|\boldsymbol{y}\|}\boldsymbol{y}\right\| = 1.$$

PROOF. *of Theorem 2.1.* This is a direct consequence of Definition 2.4. Indeed, we have

(1) $\|A\| = \sup_{\|x\|=1} \frac{\|Ax\|}{\|x\|}$ since $\|Ax\| \geq 0$. Also $\|A\| = 0 \Rightarrow Ax = 0\ \forall x \neq 0 \Rightarrow A = 0$.
(2) $\|\alpha A\| = \sup_{\|x\|=1} \|\alpha Ax\| = |\alpha| \sup_{\|x\|=1} \|Ax\| = |\alpha|\|A\|$.
(3) The triangle inequality follows similarly,

which concludes the argument. □

A special discussion is deserved by the 2-norm or spectral norm.

THEOREM 2.2. *Let $\rho(A)$ be the spectral radius of A. Then*

$$\|A\|_2 = \sqrt{\rho(A^H A)} = \rho(A). \tag{2.1}$$

PROOF. Let $(\lambda, \boldsymbol{v})$ be an eigenpair of A.

($\leq$) Then by the (homogeneity) of the vector norm, and the definition of the (induced matrix norm), we have

$$\|A\boldsymbol{v}\|_2 = |\lambda|\|\boldsymbol{v}\|_2 \quad \Longrightarrow \quad \|A\|_2 := \sup_{\boldsymbol{v}\neq\boldsymbol{0}} \frac{\|A\boldsymbol{v}\|_2}{\|\boldsymbol{v}\|_2} \leq |\lambda| \leq \rho(A).$$

($\geq$) Conversely,

$$\|A\|_2 := \sup_{\boldsymbol{v}\neq\boldsymbol{0}} \frac{\|A\boldsymbol{v}\|_2}{\|\boldsymbol{v}\|_2} \geq \frac{\|A\boldsymbol{v}\|_2}{\|\boldsymbol{v}\|_2} = \frac{|\lambda|\|\boldsymbol{v}\|_2}{\|\boldsymbol{v}\|_2} = |\lambda|, \qquad \forall(\lambda, \boldsymbol{v}) \text{ eigenpair of } A.$$

Taking the maximum over all eigenvalues λ in the spectrum $\sigma(A)$ concludes the argument. □

PROPOSITION 2.3.

$$\|A\|_1 = \max_{j=1:n} \sum_{i=1}^{m} |a_{ij}| \qquad \text{(maximum of column sum norm)}$$

$$\|A\|_\infty = \max_{i=1:m} \sum_{j=1}^{n} |a_{ij}| \qquad \text{(maximum of row sum norm)}$$

Relations between Norms and the spectral radius of a matrix

THEOREM 2.3. *Let $\|\cdot\|$ be a consistent matrix norm. Then*

$$\rho(A) \leq \|A\|, \qquad \forall A \in \mathbb{C}^{n\times n}.$$

PROOF. Let $\lambda \in \sigma(A)$ and $\boldsymbol{v} \neq \mathbf{0}$ an associated eigenvector. Since $\|\cdot\|$ is a consistent matrix norm, we have

$$|\lambda|\|\boldsymbol{v}\| = \|\lambda \boldsymbol{v}\| = \|A\boldsymbol{v}\| \leq \|A\|\|\boldsymbol{v}\| \quad \Rightarrow \quad |\lambda| \leq \|A\|,$$

which concludes the argument. □

2.2. Positive definite, diagonally dominant and M-matrices.

DEFINITION 2.7. *The matrix $A \in \mathbb{C}^{n\times n}$ is positive (semi-) definite if*

$$\langle Ax, x\rangle (\geq) > 0, \qquad \forall x \in \mathbb{C}^n. \qquad \text{(positive definite)}$$

REMARK 2.4. *Matrices which are positive definite are not necessarily symmetric, for example*

$$A = \begin{bmatrix} 2 & \alpha \\ -2-\alpha & 2 \end{bmatrix}, \qquad \alpha \neq -1.$$

Indeed,

$$\langle A\boldsymbol{x}, \boldsymbol{x}\rangle = \left\langle \begin{bmatrix} 2 & \alpha \\ -2-\alpha & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right\rangle = \begin{bmatrix} 2x_1+\alpha x_2 \\ (-2-\alpha)x_1 + 2x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= 2x_1^2 + \cancel{\alpha x_1 x_2} - 2x_1x_2 - \cancel{\alpha x_1 x_2} + 2x_2^2 = 2(x_1^2 - x_1x_2 + x_2^2) > 0, \qquad \forall \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \neq \mathbf{0}.$$

DEFINITION 2.8. *Let $A \in \mathbb{R}^n$. Then the symmetric part and skew-symmetric part of A are defined as*

$$A_s = \frac{1}{2}(A + A^T), \qquad \text{(symmetric part)}$$

$$A_{ss} = \frac{1}{2}(A - A^T). \qquad \text{(skew-symmetric part)}$$

Obviously, $A = A_s + A_{ss}$.
If $A \in \mathbb{C}^n$, the symmetric and skew-symmetric part of A are

$$A_s = \frac{1}{2}(A + A^H), \qquad \text{(symmetric part)}$$

$$A_{ss} = \frac{1}{2}(A - A^H). \qquad \text{(skew-symmetric part)}$$

EXAMPLE 2.4. *For the matrix A in Remark 2.4 we have*

$$A_s = \frac{1}{2}\begin{bmatrix} 2 & \alpha \\ -2-\alpha & 2 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 2 & -2-\alpha \\ \alpha & 2 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix},$$

$$A_{ss} = \frac{1}{2}\begin{bmatrix} 2 & \alpha \\ -2-\alpha & 2 \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 2 & -2-\alpha \\ \alpha & 2 \end{bmatrix} = \begin{bmatrix} 0 & 1+\alpha \\ -1-\alpha & 0 \end{bmatrix}.$$

DEFINITION 2.9. *A matrix $A \in \mathbb{R}^{n\times n}$ is (strictly) diagonally dominant by rows if*

$$|a_{ii}|(>) \geq \sum_{j=1, j\neq i}^{n} |a_{ij}|, \quad \forall i = 1:n, \qquad \begin{pmatrix} & \ddots & \vdots & & \\ a_{i1} & \cdots & a_{ii} & \cdots & a_{in} \\ & & \vdots & \ddots & \end{pmatrix}$$

(diagonally dominant by rows)

$$|a_{ii}|(>) \geq \sum_{j=1, j\neq i}^{n} |a_{ji}|, \quad \forall i = 1:n. \qquad \begin{pmatrix} \cdots & a_{1i} & \cdots \\ \ddots & \vdots & \\ \cdots & a_{ii} & \cdots \\ & \vdots & \ddots \\ \cdots & a_{ni} & \cdots \end{pmatrix}$$

(diagonally dominant by columns)

PROPOSITION 2.4. *A strictly diagonally dominant matrix that is symmetric with positive diagonal entries is also positive definite.*

PROOF. Indeed,

$$\langle Ax, x\rangle = \sum_{i=1}^{n}\left(\sum_{j=1}^{n} a_{ij}x_j\right)x_i = \sum_{i=1}^{n} a_{ii}x_i^2 + 2\sum_{j\neq i}^{n} a_{ij}x_ix_j \geq \sum_{i=1}^{n}\left(\sum_{j\neq i}^{n}|a_{ij}|\right)x_i^2 + 2\sum_{j\neq i}^{n} a_{ij}x_ix_j$$

$$= \sum_{i\neq j}^{n}|a_{ij}|(x_i^2+x_j^2) + 2\sum_{j\neq i}^{n} a_{ij}x_ix_j \geq \sum_{i\neq j}^{n}|a_{ij}|(x_i^2+x_j^2) - 2\sum_{j\neq i}^{n}|a_{ij}||x_i||x_j| = \sum_{i\neq j}^{n}|a_{ij}|(|x_i|-|x_j|)^2,$$

which proves the positive definiteness. □

EXAMPLE 2.5. *An example (the minus second difference matrix):*

$$A_1 = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix},$$

and a counterexample

$$A_2 = \begin{bmatrix} 1 & 3/4 & 0 \\ 3/4 & 1 & 3/4 \\ 0 & 3/4 & 1 \end{bmatrix}.$$

The matrix A_2 is symmetric, but not diagonally dominant, also not positive definite, as $x^TAx \approx -0.2426 < 0$ for $x = [1, -\sqrt{2}, 1]^T$. The spectra

$$\sigma(A_1) = \{0.5858, 2.0000, 3.4142\}, \qquad \sigma(A_2) = \{-0.0607, 1.0000, 2.0607\}.$$

For more examples of symmetric positive definite matrices, see the Hilbert matrix, the Pascal matrix, and the Wilson matrix.

DEFINITION 2.10. *[15] A nonsingular matrix $A \in \mathbb{R}^{n\times n}$ is an M-matrix if $a_{ij} \leq 0$ for $i \neq j$ and if all the entries of its inverse are nonnegative.*

M-matrices enjoy the so-called *discrete maximum principle*: if A is an M-matrix and $A\boldsymbol{x} \leq \mathbf{0}$, then $\boldsymbol{x} \leq \mathbf{0}$ (where the inequalities are meant componentwise).

PROPOSITION 2.5. *[M-criterion] Let a matrix A satisfy $a_{ij} \leq 0$ for $i \neq j$. Then A is an M-matrix if and only if there exists a vector $\boldsymbol{w} > 0$ such that $A\boldsymbol{w} > \mathbf{0}$.*

M-matrices are related to strictly diagonally dominant matrices by the following property.

PROPOSITION 2.6. *A matrix $A \in \mathbb{R}^{n\times n}$ that is strictly diagonally dominant by rows and whose entries satisfy the relations $a_{ij} \leq 0$ for $i \neq j$ and $a_{ii} > 0$, is an M-matrix.*

See also [1] and [18].

3. Direct methods for the solution of linear systems

$$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}, \qquad \boldsymbol{A}\in\mathscr{M}_{m\times n}, \quad \boldsymbol{x}\in\mathbb{C}^n, \quad \boldsymbol{b}\in\mathbb{C}^m,$$

$$\boldsymbol{A}=(a_{ij})=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n}\\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ a_{i1} & a_{i2} & a_{i3} & \cdots & a_{in}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn}\end{pmatrix}, \quad \boldsymbol{x}=\begin{pmatrix} x_1\\ x_2\\ x_3\\ \vdots\\ x_i\\ \vdots\\ x_n\end{pmatrix}, \quad \boldsymbol{b}=\begin{pmatrix} b_1\\ b_2\\ \vdots\\ b_i\\ \vdots\\ b_m\end{pmatrix}, \qquad i=1:m, j=1:n.$$

The i^{th} equation writes:

$$\sum_{j=1}^{n} a_{ij}x_j=b_i, \qquad i=1:m,$$

hence this is a system of m equations, with n unknowns $x_1,\cdots,x_n$.

DEFINITION 3.1. *We call solution an n-tuple of $\boldsymbol{x}=(x_1,\cdots,x_n)^T$ such that $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$.*

REMARK 3.1. *We will deal only with (in this chapter) real valued square matrices $\boldsymbol{A}\in\mathbb{R}^{n\times n}, \boldsymbol{b}\in\mathbb{R}^n$. In such cases, the existence and uniqueness of the solution is ensured if one of the following assumptions holds:*

(1) $\boldsymbol{A}$ is invertible
(2) the homogeneous system $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{0}$ admits the only solution $\boldsymbol{x}=\boldsymbol{0}$.

DEFINITION 3.2. *(Cramer's rule)*
The solution to $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}, \boldsymbol{A}\in\mathbb{R}^{n\times n}$ is formally provided by

$$x_j=\frac{\Delta_j}{\det(A)}, \qquad j=1:n, \qquad \text{(Cramer's rule)}$$

where Δ_j is the determinant of a matrix obtained from $\boldsymbol{A}$ by substituting its j^{th} column by the RHS $\boldsymbol{b}$.

EXAMPLE 3.1.

$$\boldsymbol{A}=\begin{bmatrix} 1 & 2\\ 3 & 4\end{bmatrix}, \quad \boldsymbol{x}=\begin{bmatrix} 1\\ 2\end{bmatrix}, \quad \boldsymbol{b}=\begin{bmatrix} 5\\ 11\end{bmatrix},$$

$$\det(\boldsymbol{A})=\begin{vmatrix} 1 & 2\\ 3 & 4\end{vmatrix}=-2, \quad \Delta_1=\begin{vmatrix} 5 & 2\\ 11 & 4\end{vmatrix}=20-22=-2, \quad \Delta_2=\begin{vmatrix} 1 & 5\\ 3 & 11\end{vmatrix}=-4,$$

$$x_1=\frac{-1}{-1}=1, \qquad x_2=\frac{-4}{-2}=2.$$

REMARK 3.2. *Cramer's rule is of little practical use! Indeed, if $\det(\boldsymbol{A})$ is evaluated by Laplace expansion*

$$\det(A)=\begin{cases} a_{11}, & \text{if } n=1,\\ \sum_{j=1}^{n}\Delta_{ij}a_{ij}, & \text{for } n>1,\end{cases} \qquad \text{(Laplace expansion)}$$

then the computational effort for Cramer's rule is of order $(n+1)!$ flops.

```
>> n=100; factorial(n+1)

ans =

  9.4259e+159
>> n=169; factorial(n+1)

ans =

  7.2574e+306
>> n=170;factorial(n+1)

ans =

   Inf
```

EXAMPLE 3.2. *For the matrix A is Example 3.1 the computational effort is*

$$\begin{aligned} \det(A): &\quad 2*, \quad 1+ \\ \Delta_{11}: &\quad 2*, \quad 1+ \\ \Delta_{22}: &\quad 2*, \quad 1+ \end{aligned}$$

hence $6 = 3!$ *multiplications and* 3 *additions, plus 2 more divisions to compute* x_1, x_2.

EXAMPLE 3.3. *The computational cost is "*`unacceptable`*" even for 'small' dimensions of A. Let's assume we have a computer with* 10^9 *floating point operations per second* (≈ 1 *giga flop/sec). Using Cramer's rule to solve a linear system of only 50 equations it would take* $9.6 * 10^{147}$ *years*

$$51! \quad \Rightarrow \quad \frac{1.5e+66 \text{ operations}}{10^9 \text{ operations}/\text{ second}} = 1.5e+57 \text{ seconds} \times \frac{1 \text{ year}}{31449600 \text{ seconds}} \approx 4.7532E+49 \text{ years}$$

In 2019:

$$\begin{aligned} &\text{Mac Pro} \equiv 102 \text{ gigaflops} \\ &\text{Los Alamos} \approx 12,142 \text{ faster} \end{aligned}$$

In comparison with Cramer' rule, the Gaussian Elimination Method (GEM):

$$\frac{2}{3}50^3 \frac{\text{operations}}{10^9 \text{operations/second}} \approx 8.3333 \times 10^{-5} \text{seconds}.$$

Numerical methods as alternatives to Cramer's rule have been developed:

(i) direct methods: yield the (exact) solution in a finite number of steps.
(ii) iterative methods: they require (theoretically) an infinite number of steps;

$$\texttt{at every (iteration) step } \kappa\texttt{, evaluate } \boldsymbol{x}^{(\kappa)} \texttt{ such that } \|\boldsymbol{x}^{(\kappa)} - \boldsymbol{x}\|_{\text{vector norm}} \overset{\kappa\to\infty}{\longrightarrow} 0.$$

Iterative methods give a sequence of approximate solutions $\boldsymbol{x}^{(\kappa)} \to \boldsymbol{x}$ converging (*in what 'vector norm?'*) when the number of steps $\kappa \to \infty$ tends to infinity.
$\{\boldsymbol{x}^{(\kappa)}\}$ may give useful results with fewer arithmetic operations than direct methods, but

this is true only for systems with special properties.

REMARK 3.3. *For systems $A\boldsymbol{x} = \boldsymbol{b}$ where A is a*

- *full / (dense) matrix (most elements nonzero), direct (GEM/LU factorization) methods are almost always the most efficient.*
- *sparse (large proportion of elements are zero), iterative methods offer certain advantages and for some very large systems, they are indispensable.*

Stability Analysis for Linear Systems

Forward (a priori) Analysis: sensitivity of the solution $\boldsymbol{x}$ to $A\boldsymbol{x} = \boldsymbol{b}$ to changes in the data $A, \boldsymbol{b}$.

DEFINITION 3.3. *The condition number of a matrix A is:*

$$\kappa(A) \equiv \texttt{cond(A)} = \|A\|\|A^{-1}\|, \qquad \text{(condition number)}$$

where $\|\cdot\|$ is an induced matrix norm.
The condition number of a singular matrix is set to ∞.
(In general, $\kappa(A)$ depends on the choice of $\|\cdot\|$.)

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> cond(A,2)
ans =
   5.0523e+16
>> cond(A,inf)
ans =
   8.6469e+17
>> det(A)
ans =
   6.6613e-16
```

REMARK 3.4.

(i) *An increase in the condition number (of the matrix) produces a higher sensitivity of the solution to the changes in the data.*
A large condition number indicates a **'nearly' singular matrix**.

(ii) *For any consistent matrix norm: $\kappa(A) \geq 1$, since $1 = \|I\| = \|AA^{-1}\| \leq \|A\|\|A^{-1}\| = \kappa(A)$.*

(iii) *$\kappa(A) = \kappa(A^{-1})$, and $\forall \alpha \in \mathbb{C}, \alpha \neq 0$: $\kappa(\alpha A) = \kappa(A)$.*

(iv) *If A is orthogonal matrix, i.e.,*

$$AA^T = A^T A = I, \qquad \text{(orthogonal matrix)}$$

then $\kappa(A) = 1$, since $\|A\|_2 = \sqrt{\rho(A^T A)} = \sqrt{\rho(I)} = 1$, $A^{-1} = A^T$.

(v) *For $p = 2$, $\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 = \frac{|\lambda_{\max}(A)|}{|\lambda_{\min}(A)|}$.* *(see (2.1))*

Consider the equation

$$A\boldsymbol{x} = \boldsymbol{b}$$

with the exact solution $\boldsymbol{x}$, and an 'approximate/computed' solution $\widetilde{\boldsymbol{x}}$:

$$A\widetilde{\boldsymbol{x}} \approx \boldsymbol{b}.$$

Let us define the residual as:

$$\widetilde{\boldsymbol{r}} = A\widetilde{\boldsymbol{x}} - \boldsymbol{b}, \qquad \boldsymbol{r} = A\boldsymbol{x} - \boldsymbol{b}. \tag{residual}$$

We note that since

$$\boldsymbol{r} = \boldsymbol{0} \quad \Leftrightarrow \quad \boldsymbol{x} = A^{-1}\boldsymbol{b} \quad \text{is the exact solution,}$$

it is natural to expect that if

$$\widetilde{\boldsymbol{r}} \ll 1 \quad \Rightarrow \quad \widetilde{\boldsymbol{x}} \quad \text{to be an accurate solution.}$$

Now let's look at this is by actually considering the error (in the 'approximate solution' $\widetilde{\boldsymbol{x}}$):

$$\boldsymbol{e} := \boldsymbol{x} - \widetilde{\boldsymbol{x}} = A^{-1}\boldsymbol{b} - A^{-1}A\widetilde{\boldsymbol{x}} = -A^{-1}\big(A\widetilde{\boldsymbol{x}} - \boldsymbol{b}\big) = -A^{-1}\widetilde{\boldsymbol{r}}. \tag{error}$$

We see that the error could be large, even for small residuals $\|\widetilde{\boldsymbol{r}}\| \ll 1$, if the matrix A is 'close to' being singular (for large size $\|A^{-1}\|$). See e.g., Exercise 5 (1.1):

$$\|\widetilde{\boldsymbol{r}}\|_2 = 2.1 \times 10^{-3}, \quad \|\widehat{\boldsymbol{r}}\|_2 = 1.0 \times 10^{-6},$$
$$\|\widetilde{\boldsymbol{e}}\|_2 = 1.4 \times 10^{-3}, \quad \|\widehat{\boldsymbol{e}}\|_2 = 1.126.$$

COUNTEREXAMPLE 3.1 (From section 5.5.1 in [4], an example constructed by W. Kahan). *Let*

$$A = \begin{bmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 0.8642 \\ 0.1440 \end{bmatrix},$$

with the exact solution

$$\boldsymbol{x} = \begin{bmatrix} 2 \\ -2 \end{bmatrix},$$

We note that

$$\begin{aligned} &\det(A) \approx 1.0000e-8 && (= 9.999999998544971e-09) \\ &\texttt{norm}(A,2) \approx 1.5803, && \texttt{norm}(\texttt{inv}(A)) \approx 1.5803e+08, \\ &\texttt{cond}(A) \approx 2.4973\ e+08. && \text{(ill-conditioned)} \end{aligned}$$

Assume that

$$\widetilde{\boldsymbol{x}} = \begin{bmatrix} 0.9911 \\ -0.4870 \end{bmatrix},$$

which gives a residual

$$\widetilde{\boldsymbol{r}} = \boldsymbol{b} - A\boldsymbol{x} = \begin{bmatrix} 10^{-8} \\ -10^{-8} \end{bmatrix},$$
$$\|\widetilde{\boldsymbol{r}}\|_2 \approx 1.4142e-8, \quad \|\widetilde{\boldsymbol{r}}\|_\infty \approx 1.e-8, \quad \|\widetilde{\boldsymbol{r}}\|_1 \approx 2.e-8.$$

But not a single digit in $\widetilde{\boldsymbol{x}}$ is meaningful!
The relative errors component-wise are:

$$\begin{bmatrix} \frac{2-0.9911}{2} \\ \frac{-2+0.4870}{-2} \end{bmatrix} = \begin{bmatrix} 0.5045 \\ 0.7565 \end{bmatrix},$$

therefore a 50% *and* 70% *error in each component of the solution, respectively. In order to better understand the way in which the ill-conditioning manifests, let us try to compute* $\widetilde{\boldsymbol{x}}$ *by Gaussian elimination:*

$$\left[\begin{array}{cc|c} 1.2969 & 0.8648 & 0.8642 \\ 0.2161 & 0.1441 & 0.1440 \end{array}\right], \qquad (* \left[\begin{array}{c} \frac{0.2161}{1.2969} \\ \odot \end{array}\right])$$

$$\left[\begin{array}{cc|c} 1.2969 & 0.8648 & 0.8642 \\ 0 & 7.7907e-9 & -1.5421e-8 \end{array}\right],$$

hence

$$a_{22} = 7.710694749363256e-09 \approx 7.7907 \times 10^{-9}$$
$$b_2 = -1.542138947097094e-08 \approx -1.5421 \times 10^{-8},$$

with a_{22} *being very small, giving*

$$\widetilde{\boldsymbol{x}}_2 = \frac{\boldsymbol{b}_2}{a_{22}} = \frac{-1.542138947097094e-08}{7.710694749363256e-09} = -1.9999999996400379.$$

Also, remark that in single precision

$$\widetilde{\boldsymbol{x}} = \texttt{single}(A)\backslash\texttt{single}(\boldsymbol{b}) = \left[\begin{array}{c} 1.5443 \\ -1.3166 \end{array}\right],$$

while

$$\widetilde{\boldsymbol{x}} = A\backslash\boldsymbol{b} = \left[\begin{array}{c} 2.000000001091036 \\ -2.000000001636175 \end{array}\right]. \qquad \text{(MatLab's "backslash" operation)}$$

Even if the size of the residual vector gives no direct indication of the error in $\widetilde{\boldsymbol{x}}$*, it is possible to use accurately computed residuals to estimate the error, or even to correct the approximate solution.*

A large condition number indicates a **'nearly' singular matrix**.

DEFINITION 3.4. *The relative distance of* $A \in \mathbb{C}^{n\times n}$ *from the set of singular matrices w.r.t the p-norm:*

$$\texttt{dist}_p(A) = \min_{\delta A}\left\{\frac{\|\delta A\|_p}{\|A\|_p};\ \text{such that } A+\delta A \text{ is singular}\right\}.$$

PROPOSITION 3.1. *It can be shown that*

$$\texttt{dist}_p(A) = \frac{1}{\kappa_p(A)} \equiv \frac{1}{\texttt{cond}_p(A)}.$$

Forward Error Analysis

Consider the linear system and its perturbation

$$A\boldsymbol{x} = \boldsymbol{b},$$
$$(A+\delta A)(\boldsymbol{x}+\delta\boldsymbol{x}) = \boldsymbol{b}+\delta\boldsymbol{b}.$$

THEOREM 3.1. *Let* $A \in \mathbb{R}^{n\times n}$ *be non-singular,* $\delta A \in \mathbb{R}^{n\times n}$ *such that*

$$\|A\|_p < 1, \qquad \|\delta A\|_p < 1.$$

Then

$$\frac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le \frac{\kappa(A)}{1-\kappa(A)\frac{\|\delta A\|}{\|A\|}}\left(\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}+\frac{\|\delta A\|}{\|A\|}\right).$$

In particular, if $\delta A = 0$*, then*

$$\frac{1}{\kappa(A)}\frac{\|\delta \boldsymbol{b}\|}{\|\boldsymbol{b}\|} \leq \frac{\|\delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \kappa(A)\frac{\|\delta \boldsymbol{b}\|}{\|\boldsymbol{b}\|}.$$

EXAMPLE 3.4 (Small changes in data ($\boldsymbol{b}$) $\Rightarrow$ large changes in the solution($\boldsymbol{x}$)). *Let*

$$A = \begin{bmatrix} 1001 & 1000 \\ 1000 & 1001 \end{bmatrix}$$

which is a symmetric, positive definite matrix, with

$$\texttt{eig}(A) = \{1, 2001\}, \qquad \texttt{cond(A,2)} \equiv \rho(A) = 2001.$$

Also let

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 2001 \\ 2001 \end{bmatrix}, \qquad \delta \boldsymbol{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

The relative changes in the data $\mathbf{b}$ *are small*

$$\begin{bmatrix} \frac{1}{2001} \\ \frac{0}{2001} \end{bmatrix} = \begin{bmatrix} 4.9975e-04 \\ 0 \end{bmatrix}, \qquad \text{(componentwise)}$$

$$\frac{\|\delta \boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2} = \frac{1}{2001\sqrt{2}} = 3.5338e-04 \approx 0.03\%, \qquad \text{(2-norm)}$$

compared to the changes in the solution

$$\delta \boldsymbol{x} = \begin{bmatrix} 0.5002 \\ -0.4998 \end{bmatrix}, \qquad (\delta \boldsymbol{x} = A \backslash \delta \boldsymbol{b})$$

which are relatively large:

$$\frac{\|\delta \boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = \frac{0.7071}{1.4142} = 0.5,$$

i.e., about 50% *errors in the solution, due to very small changes* 0.03% *in the data.*

EXAMPLE 3.5 (Large changes in data ($\boldsymbol{b}$) $\Rightarrow$ small changes in the solution($\boldsymbol{x}$)). *Let the matrix be the same as in Example 3.4:*

$$A = \begin{bmatrix} 1001 & 1000 \\ 1000 & 1001 \end{bmatrix}$$

which is a symmetric, positive definite matrix, with

$$\texttt{eig}(A) = \{1, 2001\}, \qquad \texttt{cond(A)} = 2001.$$

Now consider

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \delta \boldsymbol{b} = \begin{bmatrix} -2001 \\ -2001 \end{bmatrix}.$$

This large data perturbation

$$\frac{\|\delta \boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2} = 2001$$

yields the solution

$$\delta \boldsymbol{x} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \qquad (A \backslash \boldsymbol{b})$$

which represents a small relatively change in the solution

$$\frac{\|\delta \boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = 1$$

compared to 2001*, the change in the data.*

CHAPTER 2

Direct Methods for Systems of linear equations

1. Naive Gaussian Elimination

Assume $A \in \mathbb{R}^{n\times n}$ is nonsingular and

$$a_{11} \neq 0. \qquad \text{(pivot)}$$

Let us denote $A^{(1)} = A$ and consider solving the linear system

$$A^{(1)}x = b^{(1)} \approx \quad [A^{(1)}|b^{(1)}] \quad \approx$$

$$\approx \left[\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right] \qquad \left(\left[\begin{array}{c} \odot \\ m_{21} \\ \vdots \\ m_{n1} \end{array}\right]\right)$$

where we wrote it in an 'extended matrix form'.
We introduce the **multipliers**

$$m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, \qquad i = 2:n \qquad \text{(multipliers)}$$

and compute (Gaussian Elimination Method)

$$i^{\text{th}}\ \text{row}^{(2)} \Longleftarrow i^{\text{th}}\ \text{row}^{(1)} - m_{i1} \times 1^{\text{st}}\ \text{row}^{(1)}, \qquad \text{(GEM)}$$

$$\begin{cases} a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)}, & i = 2:n, j = 2:n \\ b_i^{(2)} = b_i^{(1)} - m_{i1}b_1^{(1)}, & i = 2:n, \end{cases}$$

obtaining the equivalent system

$$A^{(2)}x = b^{(2)}, \qquad [A^{(2)}|b^{(2)}] = \left[\begin{array}{cccc|c} a_{11}^{(1)} & * & \cdots & * & b_1^{(1)} \\ 0 & \square & \cdots & \square & \square \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \square & \cdots & \square & \square \end{array}\right],$$

or

$$\left[\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{array}\right].$$

Here is how this is implemented in the MATLAB function NaiveGauss(A), which gives back a new matrix A, having (the LU factorization, i.e.,) in the left lower triangular part the multipliers, and the right upper triangular part the matrix above.

```
% Forward elimination part of the Naive
% Gaussian Elimination (only update of 'A', i.e. the LU factorization)

function [A]  = NaiveGauss(A)

% Given as input:
%   a square coefficient matrix 'A'
% we will have as output:
%   the 'A=LU' factorization

[m,n] = size(A);
if abs(m-n)>0
   error('The Matrix A is not square');
end

for k=1:n-1
    for i=k+1:n
        xmult=A(i,k)/A(k,k);
        A(i,k) = xmult;
        for j = k+1:n
            A(i,j) = A(i,j) - xmult*A(k,j);
        end
    end
end
```

The solution of the system $Ax = b$ by Naive Gaussian Elimination, i.e., the Forward Elimination part (the LU factorization of A and 'update' of the RHS b) and the Backward Substitution (computation of x_is), is implemented in the MATLAB function NaiveSolve(A,b)

```
% Forward elimination part of Naive Gaussian Elimination "update of 'b'"
% and the backward substitution

function [x] = NaiveSolve(A,b)
%Given input:
%   coefficient matrix 'A' and RHS 'b'
% will get output the solution of "Ax=b"
%    'x'
[A] = NaiveGauss(A);
% This call for the function Gauss gives 'A=LU' and the index vector 'l'
[n,n] = size(A);
p = length(b);

if abs(p-n)>0
   error('The RHS b has the wrong dimension')
end

% Forward Elimination update of 'b'
    for k=1:n-1
    for i=k+1:n
```

```
        b(i)=b(i) - A(i,k)*b(k);
    end
    end
% Backward Substitution
x(n) = b(n)/A(n,n);
for i=n-1:-1:1
    sum=b(i);
    for j=i+1:n
        sum=sum - A(i,j) * x(j);
    end
    x(i)=sum/A(i,i);
end
```

PSEUDOCODE 1.1.

integer i,j,l; **real array** $(a_{ij})_{1:n,1:n},(b_i)_{1:n}$
for $k=1$ **to** $n-1$ **do**
  for $i=k+1$ **to** n **do**
    for $j=k$ **to** n **do**
      $a_{ij} \longleftarrow a_{ij}-\frac{a_{ik}}{a_{kk}}a_{kj}$
    end for
    $b_i \longleftarrow b_i - -\frac{a_{ik}}{a_{kk}}b_k$
  end for
end for

A special case of the Gaussian Elimination Method (GEM) is the **computation of A^{-1}, the inverse of A matrix** (see Remark 1.4).
Operation count in naive/complete GE:

$$(n-1)+(n-2)+\cdots+1 \qquad \text{divisions} \qquad (\text{multipliers } m_{ij})$$
$$2\big((n-1)^2+(n-2)^2+\cdots+1^2\big) \qquad \text{one '}\times\text{', and one '+'} \qquad (a_{ij})$$
$$2\big((n-1)+(n-2)+\cdots+1\big) \qquad \text{one '}\times\text{', and one '+'} \qquad (b_{ij})$$

giving a total of

$$3(1+\cdots+(n-1))+2(1+\cdots+(n-1)^2)=3\frac{n(n-1)}{2}+2\sum_{k=1}^{n-1}k^2=\frac{3}{2}n^2-\frac{3}{2}n+2\frac{(n-1)n(2n-1)}{2\cdot 3}$$
$$=\frac{3}{2}n^2-\frac{3}{2}n+\frac{2n^3}{3}-n^2+\frac{n}{3}=\frac{2n^3}{3}+\frac{n^2}{2}-\frac{7}{6}n\approx\mathcal{O}\Big(\frac{2}{3}n^3\Big).$$

Actually, just for the LU factorization $A=LU$:

$$\frac{n(n-1)}{2}+2\frac{(n-1)n(n+1)}{2\cdot 3}=\frac{2}{3}n^3-\frac{n^2}{2}. \qquad (\text{cost of } LU \text{ factorization})$$

There are also n^2 flops for a backward solve from the triangular system $Ux=b^{(n)}$, hence

$$\frac{2}{3}n^3+2n^2\approx\mathcal{O}\Big(\frac{2}{3}n^3\Big) \text{ to solve } Ax=b.$$

REMARK 1.1.

- *The Gaussian Elimination Method terminates safely if $A_{kk}^{(k)}\neq 0$ for $k=1:n-1$.*
- *Non-zero diagonal entries in $A \not\Rightarrow$ (does not imply) nonzero pivots !!*

COUNTEREXAMPLE 1.1. *Consider the matrix:*

$$A^{(1)} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix} \qquad \left(\begin{bmatrix} \odot \\ 2 \\ 7 \end{bmatrix} \right)$$

$$A^{(2)} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & -1 \\ 0 & -6 & -12 \end{bmatrix}$$

where the leading principal minors are

$$d_1 = 1, \qquad d_2 = \mathtt{det}\left(\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \right) = 0 \text{ !!}$$

If the original 2^{nd} *and* 3^{rd} *rows are switched:*

$$C^{(1)} = \begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \\ 2 & 4 & 5 \end{bmatrix} \qquad \left(\begin{bmatrix} \odot \\ 7 \\ 2 \end{bmatrix} \right)$$

$$C^{(2)} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -6 & -12 \\ 0 & 0 & -1 \end{bmatrix}.$$

REMARK 1.2. *The GEM can be safely employed on matrices which are*

- *diagonally dominant by rows or diagonally dominant by columns (see Theorem 4.1),*
- *SPD (symmetric:* $A^T = A$*, and positive definite:* $x^T Ax > 0 \quad \forall x \neq 0$*) (see Theorem 4.2).*

1.1. Solution of Triangular Systems.

Forward Substitution

For example, consider the nonsingular 3×3 Lower triangular system

$$\begin{bmatrix} \ell_{11} & 0 & 0 \\ \ell_{21} & \ell_{22} & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}, \qquad \ell_{ii} \neq 0, \quad i = 1:n \equiv 3,$$

$$x_1 = b_1/\ell_{11}, \quad x_2 = (b_2 - \ell_{21}x_1)/\ell_{22}, \quad x_3 = (b_3 - \ell_{31}x_1 - \ell_{32}x_2)/\ell_{33},$$

$$\begin{cases} x_1 = b_1/\ell_{11} \\ x_i = \dfrac{1}{\ell_{ii}}\left(b_i - \displaystyle\sum_{j=1}^{i-1} \ell_{ij}x_j\right) \quad i = 2:n. \end{cases} \qquad \text{(forward substitution)}$$

Operation count:

'$\times, \backslash$' : $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$

'$+, -$' : $1 + 2 + \cdots + (n-1) = \frac{(n-1)n}{2}$

hence n^2 flops.

Backward Substitution

Similarly, consider the nonsingular 3×3 Upper triangular system

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \qquad \text{with } u_{ii} \neq 0, \quad i = 1:n,$$

$$\begin{cases} x_n = b_n/u_{nn} \\ x_i = \dfrac{1}{u_{ii}}\Big(b_i - \displaystyle\sum_{j=i+1}^{n} u_{ij}x_j\Big) \quad i = n-1:1. \end{cases} \qquad \text{(backward substitution)}$$

Operation count:

'$\times, \backslash$' : $1+2+\cdots+n = \frac{n(n+1)}{2}$

'$+,-$' : $1+2+\cdots+(n-1) = \frac{(n-1)n}{2}$

hence n^2 flops.

REMARK 1.3. *If $A = LU$, then solving $Ax = b$ is equivalent to solving two triangular systems*

$$\begin{cases} Ly = b, \\ Ux = y, \end{cases}$$

i.e., $\mathcal{O}(2n^2)$ flops.

REMARK 1.4. **[Computation of A^{-1}, the inverse of A matrix]**
Denote by X the inverse of the **nonsingular** *matrix $A \in \mathbb{R}^{n\times n}$:*

$$AX = XA = \mathbb{I}_{n\times n},$$

with columns $\{x_{\star i}\}$, i.e.,

$$X = [x_{\star 1}, x_{\star 2}, \cdots, x_{\star n}], \quad x_{\star i} = \begin{bmatrix} x_{1i} \\ x_{2i} \\ \vdots \\ x_{ni} \end{bmatrix}.$$

Then, in order to find the inverse $A^{-1} = X$, we have to solve n linear systems

$$Ax_{\star i} = e_i, \qquad i = 1:n,$$

where e_i are the unit vectors of the cartesian coordinates system in $\mathbb{R}^n$:

$$e_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ i \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (i^{\text{th}} \text{ position}).$$

Now recall that the cost of LU factorization of A is $\mathcal{O}(\frac{2}{3}n^3)$. Therefore solving $e_i = Ax_{\star i} \equiv LUx_{\star i}$ costs $2 \times n^2$ operations, for each $i = 1 : n$. Finally this means that the total cost of finding the inverse A^{-1} is $\frac{8}{3}n^3$.

1.2. Exercises.

Exercise 1. *Show that the system of equations*

$$\begin{cases} x_1 + 4x_2 + \alpha x_3 = 6 \\ 2x_1 - x_2 + 2\alpha x_3 = 3 \\ \alpha x_1 + 3x_2 + x_3 = 5 \end{cases}$$

possesses a unique solution when $\alpha = 0$, no solution when $\alpha = -1$, and infinitely many solutions when $\alpha = 1$.
Also, investigate the corresponding situation when the right-hand side is replaced by 0's.
Solution:

$$\left[\begin{array}{ccc|c} 1 & 4 & \alpha & 6 \\ 2 & -1 & 2\alpha & 3 \\ \alpha & 3 & 1 & 5 \end{array}\right], \quad \left[\begin{array}{ccc|c} 1 & 4 & \alpha & 6 \\ 0 & -9 & 0 & -9 \\ 0 & 3-4\alpha & 1-\alpha^2 & 5-6\alpha \end{array}\right] \quad \begin{array}{l} x_1 = 6-4-\alpha x_3 \equiv 2-\alpha x_3, \\ \Rightarrow \ x_2 = 1, \\ (1-\alpha^2)x_3 = 2(1-\alpha). \end{array}$$

Hence

$$\begin{array}{ll} x_1 = 2, \quad x_2 = 1, \quad x_3 = 2 & (\alpha = 0) \\ 0 \cdot x_3 = 4 \quad \Rightarrow \text{ NO solutions} & (\alpha = -1) \\ 0 \cdot x_3 = 0 \quad \Rightarrow \infty \text{ number of solutions} & (\alpha = 1) \end{array}$$

In the homogeneous case

$$\left[\begin{array}{ccc|c} 1 & 4 & \alpha & 0 \\ 2 & -1 & 2\alpha & 0 \\ \alpha & 3 & 1 & 0 \end{array}\right], \quad \left[\begin{array}{ccc|c} 1 & 4 & \alpha & 0 \\ 0 & -9 & 0 & 0 \\ 0 & 3-4\alpha & 1-\alpha^2 & 0 \end{array}\right] \quad \begin{array}{l} x_1 = -\alpha x_3, \\ \Rightarrow \ x_2 = 0, \\ (1-\alpha^2)x_3 = 0. \end{array}$$

we then have

$$\begin{array}{ll} x_1 = 0, \quad x_2 = 0, \quad x_3 = 0 & (\alpha = 0) \\ 0 \cdot x_3 = 0, \quad x_1 = x_3 \quad \Rightarrow \infty \text{ number of solutions} & (\alpha = -1) \\ 0 \cdot x_3 = 0,, \quad x_1 = -x_3 \quad \Rightarrow \infty \text{ number of solutions} & (\alpha = 1) \end{array}$$

Exercise 2. *For what values of α does naive Gaussian elimination produce erroneous answers for this system?*

$$\begin{cases} x_1 + x_2 = 2 \\ \alpha x_1 + x_2 = 2 + \alpha \end{cases}$$

Explain what happens in the computer.
Solution:

$$\left[\begin{array}{cc|c} 1 & 1 & 2 \\ \alpha & 1 & 2+\alpha \end{array}\right] \quad \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 1-\alpha & 2-\alpha \end{array}\right] \quad \Longrightarrow \quad x_2 = \frac{2-\alpha}{1-\alpha}, \quad x_1 = -\frac{\alpha}{1-\alpha},$$

which for $\alpha \approx 1$ gives erroneous results.
Exercise 3. *Apply naive Gaussian elimination to these examples and account for the failures.*
Solve the systems by other means if possible.

(a)

$$\begin{cases} 3x_1 + 2x_2 = 4 \\ -x_1 - \frac{2}{3}x_2 = 1 \end{cases}$$

(b)

$$\begin{cases} 6x_1 - 3x_2 = 6 \\ -2x_1 + x_2 = -2 \end{cases}$$

(c)

$$\begin{cases} 0x_1 + 2x_2 = 4 \\ x_1 - x_2 = 5 \end{cases}$$

(d)

$$\begin{cases} x_1 + x_2 + 2x_3 = 4 \\ x_1 + x_2 + 0x_3 = 2 \\ 0x_1 + x_2 + x_3 = 0 \end{cases}$$

Solution:

(a)

$$\left[\begin{array}{cc|c} 3 & 2 & 4 \\ -1 & -\frac{2}{3} & 1 \end{array}\right], \quad \left[\begin{array}{cc|c} 3 & 2 & 4 \\ 0 & 0 & \frac{7}{3} \end{array}\right],$$

The LHS matrix is singular. No solution: the equations represent two parallel lines in $\mathbb{R}^2$.

(b)

$$\left[\begin{array}{cc|c} 6 & -3 & 6 \\ -2 & 1 & -2 \end{array}\right], \quad \left[\begin{array}{cc|c} 6 & -3 & 6 \\ 0 & 0 & 0 \end{array}\right],$$

The LHS matrix is singular. Infinite number of solutions: the equations represent one line in $\mathbb{R}^2$.

(c) The pivot is zero

$$\left[\begin{array}{cc|c} 0 & 2 & 4 \\ 1 & -1 & 5 \end{array}\right],$$

so Naive Gauss Elimination cannot be performed.
Switching the equations

$$\left[\begin{array}{cc|c} 1 & -1 & 5 \\ 0 & 2 & 4 \end{array}\right], \qquad \Rightarrow x_2 = 2,\ x_1 = 7.$$

(d)

$$\left[\begin{array}{ccc|c} 1 & 1 & 2 & 4 \\ 1 & 1 & 0 & 2 \\ 0 & 1 & 1 & 0 \end{array}\right], \qquad \left[\begin{array}{ccc|c} 1 & 1 & 2 & 4 \\ 0 & 0 & -2 & -2 \\ 0 & 1 & 1 & 0 \end{array}\right],$$

which has a zero pivot, hence Naive Gauss elimination cannot be performed.
Nonetheless, the system has a unique solution $x = [3; -1; 1]$.

```
>> A = [1 1 1; 1 1 0; 0 1 1];b=[4;2;0];
>> Anew=NaiveGauss(A)

Anew =

     1     1     2
```

```
      1     0    -2
      0   Inf   Inf
>> x = NaiveSolve(A,b)

x =

   NaN   NaN   NaN
>> x=Solve(A,b)

x =

     3    -1     1
>> x=A\b

x =

     3
    -1
     1
```

Exercise 4. *Solve the following system of equations, retaining only four significant figures in each step of the calculation, and compare your answer with the solution obtained when eight significant figures are retained.*
Be consistent by either always rounding to the number of significant figures that are being carried or always chopping.

$$\begin{cases} 0.1036x_1 + 0.2122x_2 = 0.7381 \\ 0.2081x_1 + 0.4247x_2 = 0.9327 \end{cases}$$

Solution: With **four** significant digits:

$$\begin{cases} 0.1036x_1 + 0.2122x_2 = 0.7381 \\ 0.2081x_1 + 0.4247x_2 = 0.9327 \end{cases} \qquad (\times \tfrac{0.2081}{0.1036} \approx 2.009)$$

$$\begin{cases} 0.1036x_1 + 0.2122x_2 = 0.7381 \\ \quad - 0.00\mathbf{1610}x_2 = -.5501 \end{cases} \Rightarrow x_2 = 341.7,\ x_1 = -697.3.$$

With **eight** significant digits:

$$\begin{cases} 0.1036x_1 + 0.2122x_2 = 0.7381 \\ 0.2081x_1 + 0.4247x_2 = 0.9327 \end{cases} \qquad (\times \tfrac{0.2081}{0.1036} \approx 2.0086873)$$

$$\begin{cases} 0.1036x_1 + 0.2122x_2 = 0.7381 \\ \quad - 0.00\mathbf{15434451}x_2 = -0.54991210 \end{cases} \Rightarrow x_2 = 356.28873,\ x_1 = -722.64834.$$

```
>> A = [0.1036 0.2122 ; 0.2081 0.4247]; b= [0.7381;0.9327];
>> [x] = NaiveSolve(A,b)
x =
   1.0e+02 *
  -7.226524702939380   3.562907442151365
>> y = A\b
ans =
   1.0e+02 *
```

```
  -7.226524702939361
   3.562907442151356
>> cond(A)
ans =
     1.747576038160110e+03
```

Exercise 5. *Consider*

$$\boldsymbol{A} = \begin{bmatrix} 0.780 & 0.563 \\ 0.913 & 0.659 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 0.217 \\ 0.254 \end{bmatrix}, \tag{1.1}$$

$$\widetilde{\boldsymbol{x}} = \begin{bmatrix} 0.999 \\ -1.001 \end{bmatrix}, \qquad \widehat{\boldsymbol{x}} = \begin{bmatrix} 0.341 \\ -0.087 \end{bmatrix}. \tag{1.2}$$

Compute residual vectors $\widetilde{\boldsymbol{r}} = \boldsymbol{A}\widetilde{\boldsymbol{x}} - \boldsymbol{b}$ *and* $\widehat{\boldsymbol{r}} = \boldsymbol{A}\widehat{\boldsymbol{x}} - \boldsymbol{b}$ *and decide which of* $\widetilde{\boldsymbol{x}}$ *and* $\widehat{\boldsymbol{x}}$ *is the better solution vector. Now compute the error vectors* $\boldsymbol{e} = \boldsymbol{x} - \widetilde{\boldsymbol{x}}$ *and* $\widehat{\boldsymbol{e}} = \boldsymbol{x} - \widehat{\boldsymbol{x}}$*, where* $\boldsymbol{x} = [1, -1]^T$ *is the exact solution.*
Discuss the implications of this example.
Solution: The residuals are

$$\widetilde{\boldsymbol{r}} = \boldsymbol{A}\widetilde{\boldsymbol{x}} - \boldsymbol{b} = \begin{bmatrix} 0.780 & 0.563 \\ 0.913 & 0.659 \end{bmatrix} \begin{bmatrix} 0.999 \\ 1.001 \end{bmatrix} - \begin{bmatrix} 0.217 \\ 0.254 \end{bmatrix} \equiv 10^{-3} * \begin{bmatrix} -1.343 \\ -1.572 \end{bmatrix}, \quad \text{('large')}$$

$$\widehat{\boldsymbol{r}} = \boldsymbol{A}\widehat{\boldsymbol{x}} - \boldsymbol{b} = \begin{bmatrix} 0.780 & 0.563 \\ 0.913 & 0.659 \end{bmatrix} \begin{bmatrix} 0.341 \\ -0.087 \end{bmatrix} - \begin{bmatrix} 0.217 \\ 0.254 \end{bmatrix} \equiv 10^{-6} * \begin{bmatrix} -0.999999999945489 \\ 0 \end{bmatrix}, \quad \text{('small')}$$

while the errors are

$$\widetilde{\boldsymbol{e}} = \boldsymbol{x} - \widetilde{\boldsymbol{x}} \equiv \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0.999 \\ -1.001 \end{bmatrix} = 10^{-3} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \text{('small')}$$

$$\widehat{\boldsymbol{e}} = \boldsymbol{x} - \widehat{\boldsymbol{x}} \equiv \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0.341 \\ -0.087 \end{bmatrix} = 10^{-1} \begin{bmatrix} 6.59 \\ 9.13 \end{bmatrix}. \quad \text{('large')}$$

Although $\widehat{\boldsymbol{x}}$ yields a smaller $\mathcal{O}(10^{-6})$ residual than $\widetilde{\boldsymbol{x}}$ (namely $\mathcal{O}(10^{-3})$), it has larger errors: $\mathcal{O}(1)$ versus $\mathcal{O}(10^{-3})$:

$$\|\widetilde{\boldsymbol{r}}\|_2 = 2.1 \times 10^{-3}, \quad \|\widehat{\boldsymbol{r}}\|_2 = 1.0 \times 10^{-6},$$
$$\|\widetilde{\boldsymbol{e}}\|_2 = 1.4 \times 10^{-3}, \quad \|\widehat{\boldsymbol{e}}\|_2 = 1.126.$$

The problem is ill-conditioned: $cond(A) = 2.193 * 10^6$.

Exercise 6. *Consider the system*

$$\begin{cases} 10^{-4}x_1 + x_2 = b_1 \\ x_1 + x_2 = b_2 \end{cases}$$

where $b_1 \neq 0$ *and* $b_2 \neq 0$*. Its exact solution is*

$$x_1 = \frac{-b_1 + b_2}{1 - 10^{-4}}, \qquad x_2 = \frac{b_1 - 10^{-4} b_2}{1 - 10^{-4}}.$$

(a) Let $b_1 = 1$ *and* $b_2 = 2$*. Solve this system using naive Gaussian elimination with* **three-digit (rounded) arithmetic** *and compare with the exact solution* $x_1 = 1.000100010001000$ *and* $x_2 = 0.999899989999000$*.*
(b) Repeat the preceding part after interchanging the order of the two equations.
(c) Find values of b_1 *and* b_2 *in the original system so that naive Gaussian elimination does not give poor answers.*

Solution:

$$A = \begin{bmatrix} 10^{-4} & 1 \\ 1 & 1 \end{bmatrix}, \qquad \text{cond}(A) = 2.618.$$

(a)

$$\left[\begin{array}{cc|c} 10^{-4} & 1 & 1 \\ 1 & 1 & 2 \end{array}\right], \quad \left[\begin{array}{cc|c} 10^{-4} & 1 & 1 \\ 0 & 1-10^4 & 2-10^4 \end{array}\right] \equiv \left[\begin{array}{cc|c} 10^{-4} & 1 & 1 \\ 0 & -9999 & -9998 \end{array}\right],$$

$x_2 = 9998/9999 = 0.999899989999000 \approx 1$

$x_1 = (1-1) * 10^4 = 0,$

which have order 1 errors.

```
>> A = [10^(-4) 1 ; 1 1 ]; b = [1 ; 2];
>> format long
>> Anew = NaiveGauss(A)

Anew =

   1.0e+04 *

   0.000000010000000   0.000100000000000
   1.000000000000000  -0.999900000000000
>> x = NaiveSolve(A,b)

x =

   1.000100010001281   0.999899989999000
```

Note that even with 16 significant digits, the first component has last three digits wrong.

(b) Similarly

$$\left[\begin{array}{cc|c} 1 & 1 & 2 \\ 10^{-4} & 1 & 1 \end{array}\right], \quad \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 1-10^4 & 1-2*10^4 \end{array}\right] \equiv \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & -9999 & -9998 \end{array}\right],$$

$x_2 = 9998/9999 = 0.999899989999000 \approx 1$

$x_1 = 2-1 = 1$

which are very close to the exact solution.

```
>> B = [1 1 ; 10^(-4) 1 ]; b = [2 ; 1];
>> Bnew = NaiveGauss(B)

Bnew =

   1.000000000000000   1.000000000000000
   0.000100000000000   0.999900000000000

>> y = NaiveSolve(B,b)

y =
```

```
1.000100010001000    0.999899989999000
```

Note that with 16 significant digits, the exact solution is recovered.

(c) In general

$$\left[\begin{array}{cc|c} 10^{-4} & 1 & b_1 \\ 1 & 1 & b_2 \end{array}\right], \quad \left[\begin{array}{cc|c} 10^{-4} & 1 & b_1 \\ 0 & 1-10^4 & b_2-10^4 b_1 \end{array}\right].$$

For $b_1 = b_2 = 1$ we get $x_2 = 1, x_1 = 0$.

Exercise 7. *Solve each of the following systems using naive Gaussian elimination - that is, forward elimination and back substitution.*
Carry **four significant** *figures.*

(a)

$$\begin{cases} 3x_1 + 4x_2 + 3x_3 = 10 \\ x_1 + 5x_2 - x_3 = 7 \\ 6x_1 + 3x_2 + 7x_3 = 15 \end{cases}$$

(b)

$$\begin{cases} 3x_1 + 2x_2 - 5x_3 = 0 \\ 2x_1 - 3x_2 + x_3 = 0 \\ x_1 + 4x_2 - x_3 = 4 \end{cases}$$

(c)

$$\begin{bmatrix} 1 & -1 & 2 & 1 \\ 3 & 2 & 1 & 4 \\ 5 & 8 & 6 & 3 \\ 4 & 2 & 5 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

(d)

$$\begin{cases} 3x_1 + 2x_2 - x_3 = 7 \\ 5x_1 + 3x_2 + 2x_3 = 4 \\ -x_1 + x_2 - 3x_3 = -1 \end{cases}$$

(e)

$$\begin{cases} x_1 + 3x_2 + 2x_3 + x_4 = -2 \\ 4x_1 + 2x_2 + x_3 + 2x_4 = 2 \\ 2x_1 + x_2 + 2x_3 + 3x_4 = 1 \\ x_1 + 2x_2 + 4x_3 + x_4 = -1 \end{cases}$$

Solution:

(a) Forward elimination gives

$$\left[\begin{array}{ccc|c} 3 & 4 & 3 & 10 \\ 1 & 5 & -1 & 7 \\ 6 & 3 & 7 & 15 \end{array}\right], \quad \left[\begin{array}{ccc|c} 3 & 4 & 3 & 10 \\ 0 & \frac{11}{3} & -2 & \frac{11}{3} \\ 0 & -5 & 1 & -5 \end{array}\right], \quad \left[\begin{array}{ccc|c} 3 & 4 & 3 & 10 \\ 0 & \frac{11}{3} & -2 & \frac{11}{3} \\ 0 & 0 & -\frac{19}{11} & 0 \end{array}\right],$$

and then backward substitution yields

$$x_3 = 0, \quad x_2 = 1, \quad x_1 = 2.$$

(b) Forward elimination gives

$$\left[\begin{array}{ccc|c} 3 & 2 & -5 & 0 \\ 2 & -3 & 1 & 0 \\ 1 & 4 & -1 & 4 \end{array}\right], \quad \left[\begin{array}{ccc|c} 3 & 2 & -5 & 0 \\ 0 & -\frac{13}{3} & \frac{13}{3} & 0 \\ 0 & \frac{10}{3} & \frac{2}{3} & 4 \end{array}\right], \quad \left[\begin{array}{ccc|c} 3 & 2 & -5 & 0 \\ 0 & -\frac{13}{3} & \frac{13}{3} & 0 \\ 0 & 0 & 4 & 4 \end{array}\right],$$

and then backward substitution yields

$$x_3 = 1, \quad x_2 = 1, \quad x_1 = 1.$$

(c)

(d) Forward elimination gives

$$\left[\begin{array}{ccc|c} 3 & 2 & -1 & 7 \\ 5 & 3 & 2 & 4 \\ -1 & 1 & -3 & -1 \end{array}\right], \qquad \left(\begin{bmatrix} \odot \\ \frac{5}{3} \\ -\frac{1}{3} \end{bmatrix}\right)$$

$$\left[\begin{array}{ccc|c} 3 & 2 & -1 & 7 \\ 0 & 3-2\frac{5}{3}=-\frac{1}{3} & 2+\frac{5}{3}=\frac{11}{3} & 4-\frac{5}{3}=-\frac{23}{3} \\ 0 & 1+2\frac{1}{3}=\frac{5}{3} & -3-\frac{1}{3}=-\frac{10}{3} & -1+\frac{7}{3}=\frac{4}{3} \end{array}\right] \qquad \left(\begin{bmatrix} \odot \\ \odot \\ -5 \end{bmatrix}\right)$$

$$\left[\begin{array}{ccc|c} 3 & 2 & -1 & 7 \\ 0 & -\frac{1}{3} & \frac{11}{3} & -\frac{23}{3} \\ 0 & 0 & 15 & -37 \end{array}\right],$$

and then backward substitution yields

$$x_3 = -\frac{37}{15}, \quad x_2 = -\frac{62}{15}, \quad x_1 = \frac{64}{15}.$$

The LU factorization of the matrix is

$$\begin{bmatrix} 3 & 2 & -1 \\ 5 & 3 & 2 \\ -1 & 1 & -3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{5}{3} & 1 & 0 \\ -\frac{1}{3} & -5 & 1 \end{bmatrix} \begin{bmatrix} 3 & 2 & -1 \\ 0 & -\frac{1}{3} & \frac{11}{3} \\ 0 & 0 & 15 \end{bmatrix}.$$

(e)

2. Gaussian Elimination with Scaled Partial Pivoting

In order to avoid division by zero when defining the (multipliers) for the (GEM), the matrix A can be (totally or) partially pivoted, i.e., we can interchange the (columns and the rows, or only) rows of A, b.
This can be done according to an order given by the size of the rations of the elements of the first column with the 'scales' of each row.
pseudocode:

- $\ell =$ index vector
- $s =$ scales (DO NOT CHANGE!) $\approx$ the maximum value of the elements on each row
- – $r =$ ratios: elements on the 1st column, divided by the scales s;
  note: the 1st largest ratio gives the **pivot row**
  – update the index vector ℓ
  – compute the multipliers, and proceed with the Gaussian elimination method

The MATLAB function '[A,l] = Gauss(A)' performs the Forward Elimination with Scaled Partial Pivoting, taking the matrix A as input, outputting the 'factorized matrix' and the pivoting index ℓ:

```
%The Scaled Partial Pivoting Gaussian part
% (only update of 'A', i.e. the LU factorization)
```

```
function [A,l]  = Gauss(A)

% Given as input:
%   a square coefficient matrix 'A'
% we will have as output:
%   the 'A=LU' factorization and the pivoting index vector 'l'

[m,n] = size(A);
if abs(m-n)>0
   error('The Matrix A is not square');
end
%fprintf('This is the partial scaled Gaussian elimination part: \n\n')
%
% start with index vector 'l' being l=[1,2,...,n]
% and generate a scale vector 's'
for i=1:n
    l(i) = i;
    smax =0;
    for j=1:n
        smax = max(smax,abs(A(i,j)));
    end
    s(i)=smax;
end
%fprintf('The (global) scale vector is %1.f \n')
%s
%fprintf('The initial vector ell of indexes is %1.f \n')
%l
%pause
% This is the scaled partial pivoting forward Gaussian elimination
for k=1:n-1
    rmax=0;
    for i=k:n
        r=abs(A(l(i),k)/s(l(i)));
        if (r>rmax)
            rmax = r;
            j=i;
        end
    end
    alpha = l(j);
    beta = l(k);
    l(j) = beta;
    l(k) = alpha;
%fprintf('\n The iteration(column no.) in partial scaled Gauss elimination is %1.f\n',k)
%fprintf('the vector ell of indexes in the partial scaled Gaussian  elimination is %1.f \n ')
%l
    for i=k+1:n
        xmult=A(l(i),k)/A(l(k),k);
        xmultipliers(l(i),k)=xmult;
```

```
        A(l(i),k) = xmult;
        for j = k+1:n
            A(l(i),j) = A(l(i),j) - xmult*A(l(k),j);
        end
    end
    %fprintf('the multipliers for column %1.f \n',k)
    %xmultipliers
% pause
end
```

while the MATLAB function 'x=Solve(A,b) ' solves the linear system using the Scaled Partial Pivoting procedure and the Backward Substitution.

```
% Forward elimination part of Scaled Partial pivoting "update of 'b'"
% and the backward substitution

function [x] = Solve(A,b)
%Given input:
%   coefficient matrix 'A' and RHS 'b'
% will get output the solution of "Ax=b"
%    'x'
[A,l] = Gauss(A);
% This call for the function Gauss gives 'A=LU' and the index vector 'l'
[n,n] = size(A);
p = length(b);

if abs(p-n)>0
   error('The RHS b has the wrong dimension')
end

%fprintf('\n This is the backward solve part of algorithm: \n\n')

% Forward Elimination update of 'b'
    for k=1:n-1
        for i=k+1:n
            b(l(i))=b(l(i)) - A(l(i),k)*b(l(k));
        end
    end
% Backward Substitution
x(n) = b(l(n))/A(l(n),n);
for i=n-1:-1:1
    sum=b(l(i));
    for j=i+1:n
        sum=sum - A(l(i),j) * x(j);
    end
    x(i)=sum/A(l(i),i);
end
%fprintf('The solution, using the partial scaled pivoting, is')
```

Matlab's own function

$$[L,U,P] = \ell u(A)$$

gives the LU-factorization of the matrix A with its rows interchanged according to the index ℓ, equivalently the LU-factorization of the matrix A multiplied to the left by the permutation matrix P, i.e.,

$$PA = LU.$$

2.1. Exercises.

Exercise 1. *Show how Gaussian elimination with scaled partial pivoting works on the following matrix* $\boldsymbol{A}$*:*

$$A = \begin{bmatrix} 2 & 3 & -4 & 1 \\ 1 & -1 & 0 & -2 \\ 3 & 3 & 4 & 3 \\ 4 & 1 & 0 & 4 \end{bmatrix}.$$

Solution:

```
>> A =[2 3 -4 1 ; 1 -1 0 -2; 3 3 4 3; 4 1 0 4];
>> [B,l]  = Gauss(A)

B =
    0.5000   -2.0000   -1.0000  -12.4000
    0.2500   -1.2500         0   -3.0000
    0.7500   -1.8000    4.0000   -5.4000
    4.0000    1.0000         0    4.0000
l =
     4     2     3     1
```

$$\begin{bmatrix} 2 & 3 & -4 & 1 \\ 1 & -1 & 0 & -2 \\ 3 & 3 & 4 & 3 \\ 4 & 1 & 0 & 4 \end{bmatrix}, \quad \ell = (1,2,3,4), \quad \text{scales: } \begin{bmatrix} 4 \\ 2 \\ 4 \\ 4 \end{bmatrix}, \quad \text{ratios: } \begin{bmatrix} \frac{2}{4} = 0.5 \\ \frac{1}{2} = 0.5 \\ \frac{3}{4} = 0.75 \\ \frac{4}{4} = 1\text{: pivot line} \end{bmatrix}, \quad \ell_1 = (4,2,3,1),$$

$$\begin{bmatrix} 2 & 3 & -4 & 1 \\ 1 & -1 & 0 & -2 \\ 3 & 3 & 4 & 3 \\ 4 & 1 & 0 & 4 \end{bmatrix} \quad \text{multipliers: } \begin{bmatrix} 1/2 \\ 1/4 \\ 3/4 \\ \odot \end{bmatrix}$$

$$\Longrightarrow \begin{bmatrix} 0 & 5/2 & -4 & -1 \\ 0 & -5/4 & 0 & -3 \\ 0 & 9/4 & 4 & 0 \\ 4 & 1 & 0 & 4 \end{bmatrix}, \quad \text{ratios: } \begin{bmatrix} \frac{5}{8} \\ \frac{5}{8} \approx 0.6\text{: pivot line} \\ \frac{9}{16} \\ \odot \end{bmatrix}, \quad \ell_2 = (4,2,3,1), \quad \text{multipliers: } \begin{bmatrix} -2 \\ \odot \\ -9/5 \\ \odot \end{bmatrix}$$

$$\Longrightarrow \begin{bmatrix} 0 & 0 & -4 & -7 \\ 0 & -5/4 & 0 & -3 \\ 0 & 0 & 4 & -27/5 \\ 4 & 1 & 0 & 4 \end{bmatrix}, \text{ratios: } \begin{bmatrix} 1 \\ \odot \\ 1\text{: pivot line} \\ \odot \end{bmatrix}, \quad \ell_3 = (4,2,3,1), \quad \text{multipliers: } \begin{bmatrix} -1 \\ \odot \\ \odot \\ \odot \end{bmatrix}$$

$$\Longrightarrow \begin{bmatrix} 0 & 0 & 0 & -62/5 \\ 0 & -5/4 & 0 & -3 \\ 0 & 0 & 4 & -27/5 \\ 4 & 1 & 0 & 4 \end{bmatrix}.$$

Then the multipliers (giving the L matrix in Gaussian the LU factorization) are

$$\begin{bmatrix} 1/2 & -2 & -1 & \odot \\ 1/4 & \odot & \odot & \odot \\ 3/4 & -9/5 & \odot & \odot \\ \odot & \odot & \odot & \odot \end{bmatrix},$$

the U matrix

$$\begin{bmatrix} 0 & 0 & 0 & -62/5 \\ 0 & -5/4 & 0 & -3 \\ 0 & 0 & 4 & -27/5 \\ 4 & 1 & 0 & 4 \end{bmatrix},$$

and the labels

$$\ell_3 = (4,2,3,1), \qquad P = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Note that when concatenating the above matrices L and U, one obtains the matrix B generated by the algorithm $Gauss(A)$

$$\begin{bmatrix} 1/2 & -2 & -1 & -62/5 \\ 1/4 & -5/4 & 0 & -3 \\ 3/4 & -9/5 & 4 & -27/5 \\ 4 & 1 & 0 & 4 \end{bmatrix},$$

which gives

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/4 & 1 & 0 & 0 \\ 3/4 & -9/5 & 1 & 0 \\ 1/2 & -2 & -1 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 4 & 1 & 0 & 4 \\ 0 & -5/4 & 0 & -3 \\ 0 & 0 & 4 & -27/5 \\ 0 & 0 & 0 & -62/5 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix},$$

so that

$$PA = L * U.$$

```
>> A =[2 3 -4 1 ; 1 -1 0 -2; 3 3 4 3; 4 1 0 4];
>> L = [1 0 0 0; 1/4 1 0 0 ; 3/4 -9/5 1 0; 1/2 -2 -1 1];
>> U = [4 1 0 4; 0 -5/4 0 -3; 0 0 4 -27/5 ; 0 0 0 -62/5];
>> P = [ 0 0 0 1; 0 1 0 0 ; 0 0 1 0; 1 0 0 0];
>> P*A - L*U

ans =

     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

```
>> A =[2 3 -4 1 ; 1 -1 0 -2; 3 3 4 3; 4 1 0 4];
>> [LL,UU,PP]= lu(A)

LL =

    1.0000         0         0         0
    0.5000    1.0000         0         0
    0.7500    0.9000    1.0000         0
    0.2500   -0.5000   -0.2632    1.0000


UU =

    4.0000    1.0000         0    4.0000
         0    2.5000   -4.0000   -1.0000
         0         0    7.6000    0.9000
         0         0         0   -3.2632


PP =

     0     0     0     1
     1     0     0     0
     0     0     1     0
     0     1     0     0

>> PP*A - LL*UU

ans =

   1.0e-15 *

         0         0         0         0
         0         0         0         0
         0         0    0.4441         0
         0         0   -0.2220         0
```

Exercise 2. *Solve the following system using Gaussian elimination with scaled partial pivoting:*

$$\begin{bmatrix} 1 & -1 & 2 \\ -2 & 1 & -1 \\ 4 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \\ -1 \end{bmatrix}$$

Show intermediate matrices at each step.
Solution:

$2/2$

$-2 \quad 1 \quad -1 \quad 2$

Exercise 3. *Carry out Gaussian elimination with scaled partial pivoting on the matrix*

$$\begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 1 & 3 & -1 \\ 3 & -3 & 0 & 6 \\ 0 & 2 & 4 & -6 \end{bmatrix}$$

Show intermediate matrices.

Solution:

```
>> A = [1 0 3 0 ; 0 1 3 -1; 3 -3 0 6; 0 2 4 -6];
>> [B,l]=Gauss(A)
B =
    0.3333    1.0000         0   -1.0000
         0    1.0000    3.0000   -1.0000
    3.0000   -3.0000         0    6.0000
         0    2.0000   -2.0000   -4.0000
l =
     3     2     4     1
```

$$\begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 1 & 3 & -1 \\ 3 & -3 & 0 & 6 \\ 0 & 2 & 4 & -6 \end{bmatrix}, \quad \ell = (1,2,3,4),$$

$$\text{scales: } \begin{bmatrix} 3 \\ 3 \\ 6 \\ 6 \end{bmatrix}, \quad \text{ratios: } \begin{bmatrix} \frac{1}{3} = 0.(3) \\ \frac{0}{3} = 0 \\ \frac{3}{6} = 0.5\text{: pivot line} \\ \frac{0}{6} = 0 \end{bmatrix}, \quad \ell_1 = (3,2,1,4),$$

$$\begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 1 & 3 & -1 \\ 3 & -3 & 0 & 6 \\ 0 & 2 & 4 & -6 \end{bmatrix}, \quad \text{multipliers: } \begin{bmatrix} \frac{1}{3} \\ 0 \\ \odot \\ 0 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 0 & 1 & 3 & -2 \\ 0 & 1 & 3 & -1 \\ 3 & -3 & 0 & 6 \\ 0 & 2 & 4 & -6 \end{bmatrix}, \quad \text{ratios: } \begin{bmatrix} 1/3 \\ 1/3 = 0.(3)\text{ : pivot line} \\ \odot \\ 2/6 = 0.(3) \end{bmatrix}, \quad \ell_2 = (3,2,1,4),$$

$$\begin{bmatrix} 0 & 1 & 3 & -2 \\ 0 & 1 & 3 & -1 \\ 3 & -3 & 0 & 6 \\ 0 & 2 & 4 & -6 \end{bmatrix}, \quad \text{multipliers: } \begin{bmatrix} 1 \\ \odot \\ \odot \\ 2 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 1 & 3 & -1 \\ 3 & -3 & 0 & 6 \\ 0 & 0 & -2 & -4 \end{bmatrix},$$

$$\text{ratios: } \begin{bmatrix} 0 \\ \odot \\ \odot \\ 1/3 : \text{pivot line} \end{bmatrix}, \quad \ell_3 = (3,2,4,1).$$

Note that concatenating the L and U matrices

$$L = \begin{bmatrix} 1/3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}, \qquad U = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 1 & 3 & -1 \\ 3 & -3 & 0 & 6 \\ 0 & 0 & -2 & -4 \end{bmatrix},$$

one obtains the matrix B generated by the algorithm $\texttt{Gauss}(A)$

$$B = \begin{bmatrix} 1/3 & 1 & 0 & -1 \\ 0 & 1 & 3 & -1 \\ 3 & -3 & 0 & 6 \\ 0 & 2 & -2 & -4 \end{bmatrix},$$

Exercise 4. *Consider the matrix*

$$\begin{bmatrix} -0.0013 & 56.4972 & 123.4567 & 987.6543 \\ 0.0000 & -0.0145 & 8.8990 & 833.3333 \\ 0.0000 & 102.7513 & -7.6543 & 69.6869 \\ 0.0000 & -1.3131 & -9876.5432 & 100.0001 \end{bmatrix}$$

Identify the entry that will be used as the next pivot element of naive Gaussian elimination, of Gaussian elimination with partial pivoting (the scale vector is $[1,1,1,1]$*), and of Gaussian elimination with scaled partial pivoting (the scale vector is* $[987.6543, 46.79, 256.29, 1.096]$*).*

Solution:

$$\begin{bmatrix} -0.0013 & 56.4972 & 123.4567 & 987.6543 \\ 0.0000 & -0.0145 & 8.8990 & 833.3333 \\ 0.0000 & 102.7513 & -7.6543 & 69.6869 \\ 0.0000 & -1.3131 & -9876.5432 & 100.0001 \end{bmatrix} \quad \text{(Naive Gauss elimination)}$$

$$\begin{bmatrix} -0.0013 & 56.4972 & 123.4567 & 987.6543 \\ 0.0000 & -0.0145 & 8.8990 & 833.3333 \\ 0.0000 & 102.7513 & -7.6543 & 69.6869 \\ 0.0000 & -1.3131 & -9876.5432 & 100.0001 \end{bmatrix}, \quad \text{ratios: } \begin{bmatrix} 0.0145/1 \\ 102.7513/1 \\ 1.3/1 \end{bmatrix},$$

(partial pivoting)

$$\begin{bmatrix} -0.0013 & 56.4972 & 123.4567 & 987.6543 \\ 0.0000 & -0.0145 & 8.8990 & 833.3333 \\ 0.0000 & 102.7513 & -7.6543 & 69.6869 \\ 0.0000 & -1.3131 & -9876.5432 & 100.0001 \end{bmatrix}, \quad \text{ratios: } \begin{bmatrix} \\ \frac{0.0145}{46.79} \\ \frac{102.7513}{256.29} \\ \frac{1.3131}{1.096} \approx 1.2 \end{bmatrix}.$$

(scaled partial pivoting)

Exercise 5. *Without using the computer, determine the final contents of the array* (a_{ij}) *after procedure* `Gauss` *has processed the following array. Indicate the multipliers by underlining them*

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 4 & 2 & 1 & 2 \\ 2 & 1 & 2 & 3 \\ 1 & 2 & 4 & 1 \end{bmatrix}$$

Solution:

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 4 & 2 & 1 & 2 \\ 2 & 1 & 2 & 3 \\ 1 & 2 & 4 & 1 \end{bmatrix}, \quad \ell = (1,2,3,4), \text{scales: } \begin{bmatrix} 3 \\ 4 \\ 3 \\ 4 \end{bmatrix}, \text{ratios: } \begin{bmatrix} \frac{1}{3} \\ 1 : \text{pivot line} \\ 1 \\ \frac{1}{4} \end{bmatrix}, \quad \ell_1 = (2,1,3,4),$$

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 4 & 2 & 1 & 2 \\ 2 & 1 & 2 & 3 \\ 1 & 2 & 4 & 1 \end{bmatrix}, \quad \text{multipliers: } \begin{bmatrix} \frac{1}{4} \\ \odot \\ \frac{1}{2} \\ \frac{1}{4} \end{bmatrix}, \quad \begin{bmatrix} 0 & 5/2 & 7/4 & 1/2 \\ 4 & 2 & 1 & 2 \\ 0 & 0 & 3/2 & 2 \\ 0 & 3/2 & 15/4 & 1/2 \end{bmatrix}, \quad \text{ratios: } \begin{bmatrix} \frac{5}{6} : \text{pivot line} \\ \\ \\ \frac{3}{8} \end{bmatrix}, \quad \ell_2 = (2,1,3,4)$$

$$\begin{bmatrix} 0 & 5/2 & 7/4 & 1/2 \\ 4 & 2 & 1 & 2 \\ 0 & 0 & 3/2 & 2 \\ 0 & 3/2 & 15/4 & 1/2 \end{bmatrix}, \quad \text{multipliers: } \begin{bmatrix} \odot \\ \odot \\ \odot \\ \frac{3}{5} \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 5/2 & 7/4 & 1/2 \\ 4 & 2 & 1 & 2 \\ 0 & 0 & 3/2 & 2 \\ 0 & 0 & 27/10 & 1/5 \end{bmatrix},$$

$$\text{ratios: } \begin{bmatrix} \\ 0.5 \\ \frac{27}{40} \approx 0.67 : \text{pivot line} \end{bmatrix}, \quad \ell_3 = (2,1,4,3),$$

$$\begin{bmatrix} 0 & 5/2 & 7/4 & 1/2 \\ 4 & 2 & 1 & 2 \\ 0 & 0 & 3/2 & 2 \\ 0 & 0 & 27/10 & 1/5 \end{bmatrix}, \quad \text{multipliers: } \begin{bmatrix} \odot \\ \odot \\ \frac{5}{9} \\ \odot \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 5/2 & 7/4 & 1/2 \\ 4 & 2 & 1 & 2 \\ 0 & 0 & 0 & 17/9 \\ 0 & 0 & 27/10 & 1/5 \end{bmatrix}.$$

Putting back together the LU factorization (filling the zeros with the multipliers):

$$\begin{bmatrix} \frac{1}{4} & \odot & \odot & \odot \\ \odot & \odot & \odot & \odot \\ \frac{1}{2} & \odot & \frac{5}{9} & \odot \\ \frac{1}{4} & \frac{3}{5} & \odot & \odot \end{bmatrix},$$

gives

$$\begin{bmatrix} 1/4 & 5/2 & 7/4 & 1/2 \\ 4 & 2 & 1 & 2 \\ 1/2 & 0 & 5/9 & 17/9 \\ 1/4 & 3/5 & 27/10 & 1/5 \end{bmatrix}.$$

```
>> [A,l]  = Gauss(A)
A =
    0.2500    2.5000    1.7500    0.5000
    4.0000    2.0000    1.0000    2.0000
    0.5000         0    0.5556    1.8889
    0.2500    0.6000    2.7000    0.2000
```

```
l =
     2     1     4     3
```

Exercise 6. *If the Gaussian elimination algorithm with scaled partial pivoting is used on the matrix shown, what is the scale vector? What is the second pivot row?*

$$\begin{bmatrix} 4 & 7 & 3 \\ 1 & 3 & 2 \\ 2 & -4 & -1 \end{bmatrix}$$

Solution:

```
>> A = [4 7 3 ; 1 3 2 ; 2 -4 -1];
>> [B,l]=Gauss(A)
B =
    4.0000    7.0000    3.0000
    0.2500   -0.1667    0.8333
    0.5000   -7.5000   -2.5000
l =
     1     3     2
>> [L,U,p]=lu(A,'vector')
L =
    1.0000         0         0
    0.5000    1.0000         0
    0.2500   -0.1667    1.0000
U =
    4.0000    7.0000    3.0000
         0   -7.5000   -2.5000
         0         0    0.8333
p =
     1     3     2
```

$\begin{bmatrix} 4 & 7 & 3 \\ 1 & 3 & 2 \\ 2 & -4 & -1 \end{bmatrix}$, $\ell = (1,2,3)$, scales: $\begin{bmatrix} 7 \\ 3 \\ 4 \end{bmatrix}$ ratios: $\begin{bmatrix} 4/7: \text{ pivot line} \\ 1/3 \\ 1/2 \end{bmatrix}$, $\ell_1 = (1,2,3)$,

$\begin{bmatrix} 4 & 7 & 3 \\ 1 & 3 & 2 \\ 2 & -4 & -1 \end{bmatrix}$, multipliers: $\begin{bmatrix} \odot \\ 1/4 \\ 1/2 \end{bmatrix}$ $\Rightarrow \begin{bmatrix} 4 & 7 & 3 \\ 0 & 5/4 & 5/4 \\ 0 & -15/2 & -5/2 \end{bmatrix}$,

ratios: $\begin{bmatrix} \odot \\ 5/12 \\ 15/8 \approx 1.8: \text{ pivot line} \end{bmatrix}$, $\ell_2 = (1,3,2)$, $\begin{bmatrix} 4 & 7 & 3 \\ 0 & 5/4 & 5/4 \\ 0 & -15/2 & -5/2 \end{bmatrix}$, multipliers: $\begin{bmatrix} \odot \\ -1/6 \\ \odot \end{bmatrix}$

$\Rightarrow \begin{bmatrix} 4 & 7 & 3 \\ 0 & 0 & 5/6 \\ 0 & -15/2 & -5/2 \end{bmatrix}$.

The second pivot is the third row.

The LU factorization of the matrix is

$$\begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/4 & -1/6 & 1 \end{bmatrix} \star \begin{bmatrix} 4 & 7 & 3 \\ 0 & -15/2 & -5/2 \\ 0 & 0 & -5/6 \end{bmatrix} = \begin{bmatrix} 4 & 7 & 3 \\ 2 & -4 & -1 \\ 1 & 3 & 2 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 4 & 7 & 3 \\ 1 & 3 & 2 \\ 2 & -4 & -1 \end{bmatrix},$$

where the matrices in the LHS where rearranged according to the index $\ell = (1,3,2)$, and therefore the matrix in the RHS is the initial matrix, but with the last two rows switched.

Exercise 7. *If the Gaussian elimination algorithm with scaled partial pivoting is used on the example shown, which row will be selected as the third pivot row?*

$$\begin{bmatrix} 8 & -1 & 4 & 9 & 2 \\ 1 & 0 & 3 & 9 & 7 \\ -5 & 0 & 1 & 3 & 5 \\ 4 & 3 & 2 & 2 & 7 \\ 3 & 0 & 0 & 0 & 9 \end{bmatrix}$$

Solution:

```
>> A = [8 -1 4 9 2 ; 1 0 3 9 7; -5 0 1 3 5; 4 3 2 2 7; 3 0 0 0 9]
>> cond(A)
ans =
   35.9803
>> [B,l]=Gauss(A)
B =
   -1.6000   -0.3333    6.5333   15.2667   13.6667
   -0.2000         0    0.4898    2.1224    1.3061
   -5.0000         0    1.0000    3.0000    5.0000
   -0.8000    3.0000    2.8000    4.4000   11.0000
   -0.6000         0    0.0918    0.1875   10.5000
l =
     3     4     1     2     5
```

$$\begin{bmatrix} 8 & -1 & 4 & 9 & 2 \\ 1 & 0 & 3 & 9 & 7 \\ -5 & 0 & 1 & 3 & 5 \\ 4 & 3 & 2 & 2 & 7 \\ 3 & 0 & 0 & 0 & 9 \end{bmatrix}, \quad \ell = (1,2,3,4,5), \quad \text{scales: } \begin{bmatrix} 9 \\ 9 \\ 5 \\ 7 \\ 9 \end{bmatrix}, \text{ratios: } \begin{bmatrix} 8/9 \\ 1/9 \\ 1: \text{ pivot line} \\ 4/7 \\ 1/3 \end{bmatrix},$$

$$\ell_1 = (3,2,1,4,5), \quad \text{multipliers: } \begin{bmatrix} -8/5 \\ -1/5 \\ \odot \\ -4/5 \\ -3/5 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & -1 & 28/5 & 69/5 & 10 \\ 0 & 0 & 16/5 & 48/5 & 8 \\ -5 & 0 & 1 & 3 & 5 \\ 0 & 3 & 14/5 & 22/5 & 11 \\ 0 & 0 & 3/5 & 9/5 & 12 \end{bmatrix},$$

$$\text{ratios: } \begin{bmatrix} 1/9 \\ 0 \\ \odot \\ 3/7: \text{ pivot line} \\ 0 \end{bmatrix}, \quad \ell_2 = (3,4,1,2,5), \quad \text{multipliers: } \begin{bmatrix} -1/3 \\ 0 \\ \odot \\ \odot \\ 0 \end{bmatrix},$$

$$\Longrightarrow \begin{bmatrix} 0 & 0 & 98/15 & 229/15 & 41/3 \\ 0 & 0 & 16/5 & 48/5 & 8 \\ -5 & 0 & 1 & 3 & 5 \\ 0 & 3 & 14/5 & 22/5 & 11 \\ 0 & 0 & 3/5 & 9/5 & 12 \end{bmatrix}, \quad \text{ratios: } \begin{bmatrix} 98/135: \text{ pivot line} \\ 16/45 \\ \odot \\ \odot \\ 3/45 \end{bmatrix}, \quad \ell_3 = (3,4,1,2,5).$$

The third pivot is the first row.

Exercise 8. *Solve the system*

$$\begin{cases} 2x_1 + 4x_2 - 2x_3 = 6 \\ x_1 + 3x_2 + 4x_3 = -1 \\ 5x_1 + 2x_2 = 2 \end{cases}$$

using Gaussian elimination with scaled partial pivoting.
Show intermediate results at each step; in particular, display the scale and index vectors.
Solution:

$$[A|b] = \left[\begin{array}{ccc|c} 2 & 4 & -2 & 6 \\ 1 & 3 & 4 & -1 \\ 5 & 2 & 0 & 2 \end{array}\right], \ell = (1,2,3) \quad \text{scales: } \begin{bmatrix} 4 \\ 4 \\ 5 \end{bmatrix}, \quad \text{ratios: } \begin{bmatrix} 2/4 \\ 1/4 \\ 1 \end{bmatrix}$$

$$\ell_1 = (3,2,1), \quad \text{multipliers: } \begin{bmatrix} 2/5 \\ 1/5 \\ \odot \end{bmatrix} \quad \left[\begin{array}{ccc|c} 0 & 16/5 & -2 & 26/5 \\ 0 & 13/5 & 4 & -7/5 \\ 5 & 2 & 0 & 2 \end{array}\right],$$

$$\text{ratios: } \begin{bmatrix} 4/5 \\ 13/20 \\ \odot \end{bmatrix}, \quad \ell_2 = (3,1,2), \quad \text{multipliers: } \begin{bmatrix} \odot \\ 13/16 \\ \odot \end{bmatrix} \quad \left[\begin{array}{ccc|c} 0 & 16/5 & -2 & 26/5 \\ 0 & 0 & 45/8 & -45/8 \\ 5 & 2 & 0 & 2 \end{array}\right],$$

hence, by a backward solve (according to the indexed rows $\ell_2 = (3,1,2)$) we have:

$$x_3 = -1, \quad x_2 = 1, \quad x_1 = 0.$$

```
>> A = [2 4 -2 ; 1 3 4; 5 2 0]; b=[6; -1; 2];
>> x=A\b
x =
     0
     1
    -1
>> y = NaiveSolve(A,b)
y =
     0     1    -1
>> z=Solve(A,b)
l =
     3     1     2
z =
     0     1    -1
```

Exercise 9. *Consider the linear system*

$$\begin{cases} 2x_1 + 3x_2 = 8 \\ -x_1 + 2x_2 - x_3 = 0 \\ 3x_1 + 2x_3 = 9 \end{cases}$$

Solve for x_1, x_2, and x_3 using Gaussian elimination with scaled partial pivoting.
Show intermediate matrices and vectors.
Solution:

Exercise 10. *Consider the linear system of equations*

$$\begin{cases} -x_1 + x_2 - 3x_4 = 6 \\ -x_1 + 3x_3 + x_4 = 0 \\ x_2 - x_3 - x_4 = 3 \\ 3x_1 + x_3 + 2x_4 = 1 \end{cases}$$

Solve this system using Gaussian elimination with scaled partial pivoting. Show all intermediate steps, and write down the index vector at each step.
Solution:

```
>> A = [-1 1 0 -3 ; -1 0 3 1 ; 0 1 -1 -1 ; 3 0 1 2]; b= [6 ; 0 ; 3 ;1];
>> cond(A)
ans =
    7.5874
>> x = NaiveSolve(A,b)
x =
    1.1000    2.3000    0.9000   -1.6000
>> y = A\b
y =
    1.1000
    2.3000
    0.9000
   -1.6000
>> [B,l]=Gauss(A)
B =
   -0.3333    1.0000    0.4000   -2.0000
   -0.3333         0    3.3333    1.6667
         0    1.0000   -1.0000   -1.0000
    3.0000         0    1.0000    2.0000
l =
     4     3     2     1
>> z=Solve(A,b)
l =
     4     3     2     1
z =
    1.1000    2.3000    0.9000   -1.6000
```

$$[A|b] = \left[\begin{array}{cccc|c} -1 & 1 & 0 & -3 & 6 \\ -1 & 0 & 3 & 1 & 0 \\ 0 & 1 & -1 & -1 & 3 \\ 3 & 0 & 1 & 2 & 1 \end{array}\right], \quad \ell = (1,2,3,4), \quad \text{scales: } \begin{bmatrix} 3 \\ 3 \\ 1 \\ 3 \end{bmatrix}, \quad \text{ratios: } \begin{bmatrix} 1/3 \\ 1/3 \\ 0 \\ 1 \end{bmatrix}, \quad \ell_1 = (4,2,3,1),$$

$$\text{multipliers: } \begin{bmatrix} -1/3 \\ -1/3 \\ 0 \\ \odot \end{bmatrix} \Longrightarrow \left[\begin{array}{cccc|c} 0 & 1 & 1/3 & -7/3 & 19/3 \\ 0 & 0 & 10/3 & 5/3 & 1/3 \\ 0 & 1 & -1 & -1 & 3 \\ 3 & 0 & 1 & 2 & 1 \end{array}\right], \quad \text{ratios: } \begin{bmatrix} 1/3 \\ 0 \\ 1 \\ \odot \end{bmatrix}, \quad \ell_2 = (4,3,2,1),$$

$$\text{multipliers: } \begin{bmatrix} 1 \\ 0 \\ \odot \\ \odot \end{bmatrix} \Longrightarrow \left[\begin{array}{cccc|c} 0 & 0 & 4/3 & -4/3 & 10/3 \\ 0 & 0 & 10/3 & 5/3 & 1/3 \\ 0 & 1 & -1 & -1 & 3 \\ 3 & 0 & 1 & 2 & 1 \end{array}\right], \quad \text{ratios: } \begin{bmatrix} 4/9 \\ 5/9 \\ \odot \\ \odot \end{bmatrix}, \quad \ell_3 = (4,3,2,1),$$

$$\text{multipliers: } \begin{bmatrix} 2/5 \\ \odot \\ \odot \\ \odot \end{bmatrix} \Longrightarrow \left[\begin{array}{cccc|c} 0 & 0 & 0 & -2 & 16/5 \\ 0 & 0 & 10/3 & 5/3 & 1/3 \\ 0 & 1 & -1 & -1 & 3 \\ 3 & 0 & 1 & 2 & 1 \end{array}\right],$$

which gives

$$x_4 = -\frac{8}{5}, \quad x_3 = \frac{9}{10}, \quad x_2 = \frac{23}{10}, \quad x_1 = \frac{11}{10}.$$

Exercise 13. *Solve each of the following systems using Gaussian elimination with scaled partial pivoting. Carry four significant figures. What are the contents of the index array at each step?*

(a)

$$\begin{cases} 3x_1 + 4x_2 + 3x_3 = 10 \\ x_1 + 5x_2 - x_3 = 7 \\ 6x_1 + 3x_2 + 7x_3 = 15 \end{cases}$$

(b)

$$\begin{cases} 3x_1 + 2x_2 - 5x_3 = 0 \\ 2x_1 - 3x_2 + x_3 = 0 \\ x_1 + 4x_2 - x_3 = 4 \end{cases}$$

(c)

$$\begin{bmatrix} 1 & -1 & 2 & 1 \\ 3 & 2 & 1 & 4 \\ 5 & 8 & 6 & 3 \\ 4 & 2 & 5 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

(d)

$$\begin{cases} 3x_1 + 2x_2 - x_3 = 7 \\ 5x_1 + 3x_2 + 2x_3 = 4 \\ -x_1 + x_2 - 3x_3 = -1 \end{cases}$$

(e)

$$\begin{cases} x_1 + 3x_2 + 2x_3 + x_4 = -2 \\ 4x_1 + 2x_x + x_3 + 2x_4 = 2 \\ 2x_1 + x_2 + 2x_3 + 3x_4 = 1 \\ x_1 + 2x_2 + 4x_3 + x_4 = -1 \end{cases}$$

Solution:

(a)
```
>> A = [3 4 3 ; 1 5 -1 ; 6 3 7] ;  b=[10;7;15];
>> cond(A)
ans =
   34.9233
>> x = Solve(A,b)
The (global) scale vector is
s =
     4     5     7
l =
     3     2     1
x =
     2     1     0
>> z=A\b
z =
    2.0000
```

```
    1.0000
   -0.0000
>> y=NaiveSolve(A,b)
y =
     2     1     0
```

$$[A|b] = \left[\begin{array}{ccc|c} 3 & 4 & 3 & 10 \\ 1 & 5 & -1 & 7 \\ 6 & 3 & 7 & 15 \end{array}\right], \quad \ell = (1,2,3), \quad \text{scales: } \begin{bmatrix} 4 \\ 5 \\ 7 \end{bmatrix}, \quad \text{ratios: } \begin{bmatrix} 3/4 \\ 1/5 \\ \frac{6}{7} \end{bmatrix}, \quad \ell_1 = (3,2,1),$$

$$\text{multipliers: } \begin{bmatrix} 1/2 \\ 1/6 \\ \odot \end{bmatrix} \Longrightarrow \left[\begin{array}{ccc|c} 0 & 5/2 & -1/2 & 5/2 \\ 0 & 9/2 & -13/6 & 9/2 \\ 6 & 3 & 7 & 15 \end{array}\right], \quad \text{ratios: } \begin{bmatrix} 9/10 \\ 5/8 \\ \odot \end{bmatrix}, \quad \ell_2 = (3,2,1),$$

$$\text{multipliers: } \begin{bmatrix} 5/9 \\ \odot \\ \odot \end{bmatrix} \Longrightarrow \left[\begin{array}{ccc|c} 0 & 0 & 19/27 & 0 \\ 0 & 9/2 & -13/6 & 9/2 \\ 6 & 3 & 7 & 15 \end{array}\right], \quad x_3 = 0, \quad x_2 = 1, \quad x_1 = 2.$$

(b)

$$[A|b] = \left[\begin{array}{ccc|c} 3 & 2 & -5 & 0 \\ 2 & -3 & 1 & 0 \\ 1 & 4 & -1 & 4 \end{array}\right], \quad \ell = (1,2,3), \quad \text{scales: } \begin{bmatrix} 5 \\ 3 \\ 4 \end{bmatrix}, \quad \text{ratios: } \begin{bmatrix} \frac{3}{5} \\ \frac{2}{3} \\ \frac{1}{4} \end{bmatrix}, \quad \ell_1 = (2,1,3),$$

$$\text{multipliers: } \begin{bmatrix} 3/2 \\ \odot \\ 1/2 \end{bmatrix} \Longrightarrow \left[\begin{array}{ccc|c} 0 & 13/2 & -13/2 & 0 \\ 2 & -3 & 1 & 0 \\ 0 & 11/2 & -3/2 & 4 \end{array}\right], \quad \text{ratios: } \begin{bmatrix} 13/10 \\ \odot \\ 11/8 \approx 1.375 \end{bmatrix}, \quad \ell_2 = (2,3,1),$$

$$\text{multipliers: } \begin{bmatrix} 13/11 \\ \odot \\ \odot \end{bmatrix} \Longrightarrow \left[\begin{array}{ccc|c} 0 & 0 & -52/11 & -52/11 \\ 2 & -3 & 1 & 0 \\ 0 & 11/2 & -3/2 & 4 \end{array}\right], \quad x_3 = 1, \quad x_2 = 1, \quad x_1 = 1.$$

(c)

(d)

$$[A|b] = \left[\begin{array}{ccc|c} 3 & 2 & -1 & 7 \\ 5 & 3 & 2 & 4 \\ -1 & 1 & -3 & -1 \end{array}\right], \quad \ell = (1,2,3), \quad \text{scales: } \begin{bmatrix} 3 \\ 5 \\ 3 \end{bmatrix}, \quad \text{ratios: } \begin{bmatrix} 1 \\ 1 \\ \frac{1}{3} \end{bmatrix}, \quad \ell_1 = (1,2,3),$$

$$\text{multipliers: } \begin{bmatrix} \odot \\ 5/3 \\ -1/3 \end{bmatrix} \Longrightarrow \left[\begin{array}{ccc|c} 3 & 2 & -1 & 7 \\ 0 & -1/3 & 11/3 & -23/3 \\ 0 & 5/3 & -10/3 & 4/3 \end{array}\right], \quad \text{ratios: } \begin{bmatrix} \odot \\ 2/15 \\ 5 \end{bmatrix}, \quad \ell_2 = (1,3,2),$$

$$\text{multipliers: } \begin{bmatrix} \odot \\ -1/5 \\ \odot \end{bmatrix} \Longrightarrow \left[\begin{array}{ccc|c} 3 & 2 & -1 & 7 \\ 0 & 0 & -3 & -\frac{111}{15} \\ 0 & 5/3 & -10/3 & 4/3 \end{array}\right],$$

$$x_3 = -\frac{37}{15} \approx -2.4667, \quad x_2 = -\frac{62}{15} \approx -4.1333, \quad x_1 = \frac{64}{15} \approx 4.2667.$$

```
>> A = [3 2 -1 ; 5 3 2; -1 1 -3]; b=[7;4;-1];
>> A\b
ans =
    4.2667
```

```
   -4.1333
   -2.4667
>> [AA,ell] = Gauss(A)
AA =
    3.0000    2.0000   -1.0000
    1.6667   -0.2000    3.0000
   -0.3333    1.6667   -3.3333
ell =
     1     3     2
>> x = Solve(A,b)
This is the backward solve part of algorithm:
The solution, using the partial scaled pivoting, is
x =
    4.2667   -4.1333   -2.4667
```

Exercise 15. *Derive the formula*

$$\sum_{k=1}^{n} k = \frac{1}{2}n(n+1).$$

Solution:

Exercise 16. *Derive the formula*

$$\sum_{k=1}^{n} k^2 = \frac{1}{6}n(n+1)(2n+1).$$

Solution:

Exercise 18. *Count the number of divisions in procedure* `Gauss`. *Count the number of multiplications. Count the number of additions or subtractions. Using execution times in microseconds (multiplication 1, division 2.9, addition 0.4, subtraction 0.4), write a function of n that represents the time used in these arithmetic operations.*

Solution:

Exercise 19. *Considering long operations only and assuming 1-microsecond execution time for all long operations, give the approximate execution times and costs for procedure* `Gauss` *when* $n = 10, 10^2, 10^3, 10^4$. *Use only the dominant term in the operation count. Estimate costs at $500 per hour.*

Solution:

Exercise 20. *(Continuation) How much time would be used on the computer to solve 2000 equations using Gaussian elimination with scaled partial pivoting? How much would it cost? Give a rough estimate based on operation times.*

Solution:

3. Tridiagonal and Banded Matrices

MOTIVATION 3.1 (the Poisson equation). *Consider solving the 1D Poisson equation, a one-space dimension elliptic (equation) Boundary Value Problem:*

$$-\Delta u(x) = f(x), \qquad \forall x \in (0,L), \qquad \text{(1D Poisson equation)}$$

$$u(0) = \widetilde{u_0}, \quad u(L) = \widetilde{u_L}, \qquad \text{(boundary conditions)}$$

where u_0, u_L (boundary conditions) and $f(x), \forall x \in (0,L)$ are given, such that they are compatible. (The Laplace operator in 1 spatial dimension is $\Delta u(x) \equiv u_{xx}$.)
In order to approximate the solution $u(x)$, we consider a uniform mesh: $\{x_i = x_0 + ih, h = \frac{L}{N}\}_{i=0:N}$ ($x_0 = 0, x_N = L$) and seek the approximate values

$$u_i \approx u(x_i), \qquad i = 0 : N.$$

First of all, the boundary conditions yield the end-values:

$$u_0 = \widetilde{u_0}, \quad u_L = \widetilde{u_L},$$

leaving us with the task of finding $N-1$ values $u_1, \cdots, u_{N-1}$.
Using the Second-Order Central finite difference formula

$$u''(x) = \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} + \mathscr{O}(h^2)$$

to approximate the Laplace operator (at each node $x_i, i = 1 : N-1$)

$$-\frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2} = f(x_i)$$

we obtain the following equations

$$\begin{array}{rccccccccccccc}
i=1: & -u_0 & + & 2u_1 & - & u_2 & & & & & & & = & h^2 f_1 \\
i=2: & & - & u_1 & + & 2u_2 & - & u_3 & & & & & = & h^2 f_2 \\
i=3: & & & & - & u_2 & + & 2u_3 & - & u_4 & & & = & h^2 f_3 \\
\vdots & & & & & & & & \ddots & & & & & \vdots \\
i=N-2: & & & & & & - & u_{N-3} & + & 2u_{N-2} & - & u_{N-1} & = & h^2 f_{N-2} \\
i=N-1: & & & & & & & & - & u_{N-2} & + & 2u_{N-1} & - \; u_N = & h^2 f_{N-1}
\end{array}$$

Therefore to find the unknowns $u_1, u_2, \cdots u_{N-1}$ amounts to solving the linear system

$$\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} h^2 f_1 + u_0 \\ h^2 f_2 \\ \vdots \\ h^2 f_{N-2} \\ h^2 f_{N-1} + u_N \end{bmatrix}.$$

In particular, with $f(x) = \frac{d^2}{dx^2}\left(x^2(L-x)^2\right)$ and $u(0) = u(L) = 0$, the BVP above has the exact solution

$$u(x) = x^2(L-x)^2.$$

run PDE_Poisson1D.m

$L = 5, N = 5, 10, 20, 40 (h = 1, 0.5, 0.25, 0.125)$, `rms` $= 4.1633, 1.0879, 0.2784, 0.0704 = \mathscr{O}(h^2)$

REMARK 3.1. *Similarly, the slightly more general (elliptic) BVP*

$$\begin{aligned} -\Delta u(x) + cu(x) &= f(x), \qquad \forall x \in (0,L),, \quad c > 0 \qquad (3.1) \\ u(0) = \widetilde{u_0}, \quad u(L) &= \widetilde{u_L}, \qquad \text{(boundary conditions)} \end{aligned}$$

can be numerically approximated by solving a linear system with the following matrix

$$\begin{bmatrix} 2+ch^2 & -1 & & & \\ -1 & 2+ch^2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2+ch^2 & -1 \\ & & & -1 & 2+ch^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} h^2 f_1 + u_0 \\ h^2 f_2 \\ \vdots \\ h^2 f_{N-2} \\ h^2 f_{N-1} + u_N \end{bmatrix}.$$

Consider the particular case of a linear system with a non-singular tridiagonal matrix A:

$$A = \begin{bmatrix} d_1 & c_1 & & & & \\ a_1 & d_2 & c_2 & & & \\ & a_2 & d_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{n-2} & d_{n-1} & c_{n-1} \\ & & & & a_{n-1} & d_n \end{bmatrix}$$

```
function [x] = Tri(a,d,c,b)
% Tridiagonal system solve
% Input: a = (n-1) vector, sub-diagonal;
%        d = n vector, diagonal;
%        c = (n-1) vector, super-diagonal;
%        b = n vector, RHS;
%Output: x = n vector, the solution

n=length(d);

for i = 2:n
    xmult = a(i-1)/d(i-1);
    d(i)=d(i) - xmult*c(i-1);
    b(i) = b(i) - xmult*b(i-1);
end

x(n)=b(n)/d(n);
for i=n-1:-1:1
    x(i) = (b(i)-c(i)*x(i+1))/d(i);
end
```

In such a case, the matrices L, U of the LU factorization are bidiagonal matrices

$$L = \begin{bmatrix} 1 & 0 & & & \\ \beta_2 & 1 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-1} & 1 & 0 \\ & & & \beta_n & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} \alpha_1 & c_1 & & & \\ 0_2 & \alpha_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & \alpha_{n-1} & c_{n-1} \\ & & & 0 & \alpha_n \end{bmatrix},$$

(Doolittle method)

where the coefficients α_i, β_i can be easily computed ($LU = A$)

$$\begin{bmatrix} 1 & 0 & & & \\ \beta_2 & 1 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-1} & 1 & 0 \\ & & & \beta_n & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 & c_1 & & & \\ 0 & \alpha_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & \alpha_{n-1} & c_{n-1} \\ & & & 0 & \alpha_n \end{bmatrix} =$$

$$= \begin{bmatrix} \alpha_1 & c_1 & & & \\ \beta_2\alpha_1 & \beta_2 c_1+\alpha_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & \beta_{n-1}c_{n-2}+\alpha_{n-1} & c_{n-1} \\ & & & \beta_n\alpha_{n-1} & \beta_n c_{n-1}+\alpha_n \end{bmatrix} = \begin{bmatrix} d_1 & c_1 & & & \\ a_1 & d_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-2} & d_{n-1} & c_{n-1} \\ & & & a_{n-1} & d_n \end{bmatrix}$$

which is the Thomas algorithm:

$$\begin{aligned} &\alpha_1 = d_1, \quad \alpha_i = d_i - \beta_i c_{i-1}, \quad i = 2:n, \\ &\qquad\qquad \beta_i = a_{i-1}/\alpha_{i-1}, \quad i = 2:n. \end{aligned} \tag{3.2}$$

Therefore solving the tridiagonal system $Ax = b \Leftrightarrow LUx = b$ by

$$\begin{cases} Ly = b \\ Ux = y \end{cases}$$

is equivalent to the **forward solve**:

$$\begin{bmatrix} 1 & 0 & & & \\ \beta_2 & 1 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-1} & 1 & 0 \\ & & & \beta_n & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix} \iff y_1 = f_1, \quad y_i = b_i - \beta_i y_{i-1}, \quad i = 2:n$$

and a **backward solve**:

$$\begin{bmatrix} \alpha_1 & c_1 & & & \\ 0 & \alpha_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & \alpha_{n-1} & c_{n-1} \\ & & & 0 & \alpha_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} \iff x_n = \frac{y_n}{a_n}, \quad x_i = \frac{y_i - c_i x_{i+1}}{\alpha_i}, \quad i = n-1:1.$$

Operation count:
'$\times, \backslash$' : $2(n-1)$ in (factorization), $n-1$ in (forward solve) and $1+2(n-1)$ in (backward solve)
'$+,-$' : $n-1$ in (factorization), $n-1$ in (forward solve) and $n-1$ in (backward solve)
hence $8n-7$ flops.

REMARK 3.2. *The inverse matrix A^{-1} of a banded matrix* **generally is a full matrix**! *Therefore one should* **never** *try to compute the inverse of a band matrix explicitly.*

EXAMPLE 3.1. *If A is tridiagonal and $n = 10^4$, A^{-1} has 10^8 non-zero elements, L and U together have some 3×10^4.*

EXAMPLE 3.2. *Let*

$$A = \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}$$

which has 13 non-zero elements. Then

$$U = L^T = \begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & 1 & -1 & \\ & & & 1 & -1 \\ & & & & 1 \end{bmatrix}$$

We note that A is both diagonally dominant and positive definite, with the inverse a full matrix (25 nonzeros):

```
A = diag([1,2,2,2,2]) + diag(ones(4,1)*(-1),-1) + diag(ones(4,1)*(-1),1)
>> eig(A)
ans =
    0.0810
    0.6903
    1.7154
    2.8308
    3.6825
 >> inv(A)
ans =
     5     4     3     2     1
     4     4     3     2     1
     3     3     3     2     1
     2     2     2     2     1
     1     1     1     1     1
```

3.1. Exercises.

Exercise 5. *What is the appearance of a matrix A if its elements satisfy $a_{ij} = 0$ when:*

(a) $j < i-2$
(b) $j > i+1$

Solution:

Exercise 6. *Consider a strictly diagonally dominant matrix A whose elements satisfy $a_{ij} = 0$ when $i > j+1$. Does Gaussian elimination without pivoting preserve the strictly diagonal dominance? Why or why not?*
Solution:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \cdots & a_{1,n-1} & a_{1n} \\ a_{21} & a_{22} & a_{23} & a_{24} & \cdots & a_{2,n-1} & a_{2n} \\ 0 & a_{32} & a_{33} & a_{34} & \cdots & a_{3,n-1} & a_{3n} \\ 0 & 0 & a_{43} & a_{44} & \cdots & a_{4,n-1} & a_{4n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & 0 & 0 & \cdots & a_{n-1,n} & a_{nn} \end{bmatrix}.$$

Exercise 7. *Let A be a matrix of form*

$$\begin{bmatrix} d_1 & c_1 & & & \\ a_1 & d_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-2} & d_{n-1} & c_{n-1} \\ & & & a_{n-1} & d_n \end{bmatrix}$$

such that $a_i c_i > 0$ for $1 \le i \le n-1$.
Find the general form of the diagonal matrix $D = \mathtt{diag}(\alpha_i)$ with $\alpha_i \neq 0$ such that $D^{-1}AD$ is symmetric.
What is the general form of $D^{-1}AD$?
Solution:

Computer problem 3. *Write and test a special procedure to solve the tridiagonal system in which $a_i = c_i = 1$ for all i.*
Solution:

Computer problem 4. *Use procedure* `Tri` *to solve the following system of 100 equations. Compare the numerical solution to the obvious exact solution.*

$$\begin{cases} x_1 + 0.5x_2 = 1.5 \\ \vdots \quad \ddots \\ 0.5x_{i-1} + x_i + 0.5x_{i+1} = 2 \qquad (2 \le i \le 99) \\ \qquad\qquad \ddots \quad \vdots \\ 0.5x_{99} + x_{100} = 1.5 \end{cases}$$

Solution:

```
>> n = 100;
>>  A = diag(ones(n,1)) + diag(ones(n-1,1)*(0.5),-1) + diag(ones(n-1,1)*(0.5),1)
>> b = 2*ones(n,1); b(1)=1.5; b(n)=1.5;
>> x = A\b;
>> rms(x-ones(n,1))
ans =
   9.9920e-16
>> y = Tri(0.5*ones(n-1,1),ones(n,1),0.5*ones(n-1,1),b);
>> rms(y'-ones(n,1))
ans =
   1.0137e-14
```

or

```
function [x] = Tri_Ex4()
n=5;
d = ones(n,1);
a = 0.5 * ones(n-1,1);
c = 0.5 * ones(n-1,1);
b=2*ones(n,1);b(1)=1.5;b(n)=1.5;
[x] = Tri(a,d,c,b)
```

```
end
>> Tri_Ex4
>> x =
     1     1     1     1     1
```

Computer problem 5. *Solve the system*

$$\begin{cases} 4x_1 - x_2 = -20 \\ \ddots \\ x_{j-1} - 4x_j + x_{j+1} = 40 \qquad (2 \le j \le n-1) \\ \ddots \\ - x_{n-1} + 4x_n = -20 \end{cases}$$

using procedure `Tri` *with* $n = 100$.

Solution:

```
function [x] = Tri_Ex5()
n=7;
d = -4*ones(n,1); d(1)=4;d(n)=4;
a = ones(n-1,1);a(n-1) = -1;
c = ones(n-1,1);c(1)=-1;
b=40*ones(n,1);b(1)=-20;b(n)=-20;
[x] = Tri(a,d,c,b);
>> Tri_Ex5
ans =
   -9.2784  -17.1134  -19.1753  -19.5876  -19.1753  -17.1134   -9.2784
```

Computer problem 6. *Let A be the* 50×50 *tridiagonal matrix*

$$\begin{bmatrix} 5 & -1 & & & & & \\ -1 & 5 & -1 & & & & \\ & -1 & 5 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 5 & -1 & \\ & & & & -1 & 5 & \end{bmatrix}$$

Consider the problem $Ax = b$ *for 50 different vectors b of the form*

$$[1,2,...,49,50]^T \qquad [2,3,...,50,1]^T \qquad [3,4,...,50,1,2]^T \qquad \cdots$$

Write and test an efficient code for solving this problem.
*(***Hint***: Rewrite procedure* `Tri`*.)*

Solution:

Computer problem 18. *An upper* **Hessenberg** *matrix is of the form*

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ & a_{32} & a_{33} & \cdots & a_{3n} \\ & & \ddots & \ddots & \vdots \\ & & & a_{n,n-1} & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}.$$

Write a procedure for solving such a system, and test it on a system having 10 or more equations.

Solution: Hint: Change the Pseudocode 1.1 of the Naive Gaussian elimination to make only one zero below the main diagonal, i.e., modify one line in the code:

PSEUDOCODE 3.1.

integer i, j, l; **real array** $(a_{ij})_{1:n,1:n}, (b_i)_{1:n}$
for $k = 1$ **to** $n-1$ **do**
  for $i = k+1$ **do**
    for $j = k$ **to** n **do**
      $a_{ij} \longleftarrow a_{ij} - \frac{a_{ik}}{a_{kk}} a_{kj}$
    end for
    $b_i \longleftarrow b_i - -\frac{a_{ik}}{a_{kk}} b_k$
  end for
end for

Note that MATLAB has a "**hess**(A)" function

```
>> H = hess(A)
```

Description:
$H = hess(A)$ finds H, the Hessenberg form of matrix A.

```
>> hess(randn(4))
ans =
   -0.1241   -0.8448    0.0794   -0.2374
   -2.4926    0.4965    0.3280    1.0667
         0    1.3877   -0.9144   -0.5067
         0         0    1.7526   -1.2097
```

4. Matrix Factorization

4.1. LU factorization.

EXAMPLE 4.1. *Let us consider the Hilbert matrix*

$$\texttt{hilb}(3) = A^{(1)} \equiv \begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix}, \qquad i.e., \quad h_{ij} = \frac{1}{i+j-1}, \quad i, j = 1:3.$$

Note that the condition number for the Hilbert matrix in general grows with the dimension

n	cond(hilb(n),*inf*)
3	*5.24 E + 2*
6	*1.50 E + 7*
9	*4.93 E + 11*

```
>> n=3;A = hilb(n);x=ones(n,1);b=A*x; xNS=NaiveSolve(A,b);
>> norm(x-xNS,inf)

ans =
```

```
   2.6645e-14
>> n=6;A = hilb(n);x=ones(n,1);b=A*x; xNS=NaiveSolve(A,b);
>> norm(x-xNS,inf)

ans =

   1.1494e-09

>> n=9;A = hilb(n);x=ones(n,1);b=A*x; xNS=NaiveSolve(A,b);
>> norm(x-xNS,inf)

ans =

   5.3756e-05
>> norm(x-A\b,inf)

ans =

   1.9578e-05
```

Using Naive Gaussian Elimination we get

$$\textit{multipliers}: m_{i1} = \begin{bmatrix} \odot \\ 1/2 \\ 1/3 \end{bmatrix}, \qquad i = 2:3$$

$$\begin{bmatrix} 1 & 1/2 & 1/3 \\ 0 & 1/12 & 1/12 \\ 0 & 1/12 & 4/45 \end{bmatrix} \equiv A^{(2)}, \quad \textit{multipliers}: m_{j2} = \begin{bmatrix} \odot \\ \odot \\ 1 \end{bmatrix}, \qquad j = 3:3$$

$$\begin{bmatrix} 1 & 1/2 & 1/3 \\ 0 & 1/12 & 1/12 \\ 0 & 0 & 1/180 \end{bmatrix} \equiv A^{(3)} \equiv U.$$

Arranging the multipliers as columns vectors we obtain the lower triangular matrix, with unit diagonal entries, denoted L:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/3 & 1 & 1 \end{bmatrix}$$

and the $A^{(3)}$ matrix obtained above is the upper triangular matrix U:

$$U = \begin{bmatrix} 1 & 1/2 & 1/3 \\ 0 & 1/12 & 1/12 \\ 0 & 0 & 1/180 \end{bmatrix},$$

such that

$$A = LU.$$

```
>> A=NaiveGauss(hilb(3))
>> A =
```

```
    1.0000    0.5000    0.3333
    0.5000    0.0833    0.0833
    0.3333    1.0000    0.0056
>> [A,l]=Gauss(hilb(3))

A =

    1.0000    0.5000    0.3333
    0.5000    1.0000   -0.0056
    0.3333    0.0833    0.0889


l =

     1     3     2


 >> help lu
 LU     LU factorization.
    [L,U] = LU(A) returns an upper triangular matrix in U and a permuted
    lower triangular matrix in L, such that A = L*U. The input matrix A can
    be full or sparse.

    [L,U,P] = LU(A) returns unit lower triangular matrix L, upper
    triangular matrix U, and permutation matrix P such that P*A = L*U.


>> [L,U,P]=lu(hilb(3))
L =
    1.0000         0         0
    0.3333    1.0000         0
    0.5000    1.0000    1.0000
U =
    1.0000    0.5000    0.3333
         0    0.0833    0.0889
         0         0   -0.0056
P =
     1     0     0
     0     0     1
     0     1     0
>> P*hilb(3)- L*U

ans =

     0     0     0
     0     0     0
     0     0     0
```

We also remark that denoting

$$M_1 := \begin{bmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ -1/3 & 0 & 1 \end{bmatrix}, \qquad M_2 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix},$$

we have that

$$M_1 A^{(1)} = A^{(2)}, \qquad M_2 A^{(2)} = U,$$

and therefore

$$M_2 \underbrace{M_1 A^{(1)}}_{A^{(2)}} = U \qquad \Leftrightarrow \quad A \equiv A^{(1)} \equiv (M_2 M_1)^{-1} U = \underbrace{M_1^{-1} M_2^{-1}}_{L} U, \quad \textit{i.e.,} \quad LU = A,$$

since M_1, M_2 being lower triangular, their product and its inverse are also lower triangular matrices.

THEOREM 4.1. *(Theorem 3.4 in [15, page 76]) Let $A \in \mathbb{R}^{n\times n}$. The LU factorization of A with $\ell_{ii} = 1$ for $i = 1, \cdots, n$ exists and is unique* **if and only if** *the principal submatrices A_i of A of order $i = 1, \cdots, n-1$ are nonsingular.*

Recall the Example 1.1:

$$A^{(1)} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix}, \quad det \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} = 0, \qquad A^{(2)} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & -1 \\ 0 & -6 & -12 \end{bmatrix}.$$

"For certain classes of matrices it is not necessary to pivot. It is important to identify such classes because pivoting usually degrades performance. To illustrate the kind of analysis required to prove that pivoting can be safely avoided, we consider the case of diagonally dominant matrices."

THEOREM 4.2. *(Theorem 3.4.3 in [7, page 120]) If A^T is strictly diagonally dominant, then A has an LU factorization and $\ell_{ij} < 1$.*

Compact forms of factorization

Component-wise $A = LU$ writes

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & & & & & & \vdots \\ a_{i1} & a_{i2} & \cdots & \cdots & \boldsymbol{a_{ij}} & \cdots & a_{in} \\ \vdots & & & & & & \vdots \\ a_{n1} & a_{n2} & \cdots & \cdots & a_{nj} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} \ell_{11} & 0 & \cdots & 0 & \cdots & 0 \\ \ell_{21} & \ell_{22} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & & & & \vdots \\ \boldsymbol{\ell_{i1}} & \boldsymbol{\ell_{i2}} & \cdots & \boldsymbol{\ell_{ii}} & \cdots & 0 \\ \vdots & & & & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \cdots & \ell_{nj} & \cdots & \ell_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1j} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2j} & \cdots & u_{2n} \\ \vdots & & & & & \vdots \\ 0 & 0 & \cdots & \boldsymbol{u_{ij}} & \cdots & u_{in} \\ \vdots & & & & & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & a_{nn} \end{bmatrix}$$

where the generic term $\boldsymbol{a_{ij}}$ is computed as

$$\boldsymbol{a_{ij}} = \boldsymbol{\ell_{i1}} u_{1j} + \boldsymbol{\ell_{i2}} u_{2j} + \cdots + \boldsymbol{\ell_{ir}} u_{rj} = \sum_{k=1}^{r} \boldsymbol{\ell_{ik}} u_{kj}, \quad \text{with } r = \min\{j, i\}, \quad \forall i, j = 1 : n.$$

(compact factorization)

To determine the decomposition $A = LU$ we need to find the coefficients of L, U, namely

$$2(1 + 2 + \cdots + n) = n^2 + n$$

unknowns, with only n^2 compact factorization equations.
To fully determine the LU factorization (n^2 equations and n^2 unknowns), there are several choices, e.g.:

$$\ell_{ii} = 1, \qquad \forall i = 1:n, \qquad \text{(Doolittle factorization)}$$
$$u_{ii} = 1, \qquad \forall i = 1:n. \qquad \text{(Crôut factorization)}$$

4.2. LDL^T factorization.

Other types of factorization: for $A \in \mathbb{R}^{n\times n}$ and `symmetric`

$A = LDM^T$ where $D =$ diagonal matrix, $L =$ unit triangular matrix, $M^T =$ upper triangular.

```
>> help ldl
 LDL   Block LDL' factorization for Hermitian indefinite matrices.
    Assuming that A is a Hermitian matrix (that is, A == A'),

    [L,D,P] = LDL(A) returns unit lower triangular matrix L, block diagonal D,
    and permutation matrix P so that P'*A*P = L*D*L'.  This is equivalent
    to [L,D,P] = LDL(A,'matrix').
```

THEOREM 4.3. *If all principal minors of $A \in \mathbb{R}^{n\times n}$ are nonzero, then there exists a unique diagonal matrix D, a unique lower triangular matrix L, and a unique upper triangular matrix M^T such that $A = LDM^T$.*

PROOF. ($A = LDD^{-1}U$, and set $D^{-1}U = M^T$.)
From Theorem 4.1 we have that there exists a unique LU decomposition, with $\ell_{ii} = 1$. Then we set $D = \mathtt{diag}(u_{ii})$, which are non-zero, since U is nonsingular. This implies that $A = LU = LD(\underbrace{D^{-1}U}_{:=M^T})$, with $M^T = D^{-1}U$ a unit upper triangular matrix. □

COROLLARY 4.1. *If A is symmetric, then $M = L$ and therefore $A = LDL^T$. The computational cost for the LDL^T factorization is $\mathcal{O}(\frac{n^3}{3})$.*

EXAMPLE 4.2. *Consider the 3×3 Hilbert matrix*

$$\mathtt{hilb}(3) = \begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix}.$$

```
>> [L,D,P]=ldl(hilb(3)) % P' * A * P = L * D * L'

L =

    1.0000         0         0
    0.5000    1.0000         0
    0.3333    1.0000    1.0000
```

```
D =

    1.0000         0         0
         0    0.0833         0
         0         0    0.0056


P =

     1     0     0
     0     1     0
     0     0     1
>> [L,D,P]=ldl(hilb(6));norm(P'*hilb(6)*P-L*D*L',inf)

ans =

   4.1633e-17
>> [L,D,P]=ldl(hilb(9));norm(P'*hilb(9)*P-L*D*L',inf)

ans =

   6.2450e-17
```

4.3. Cholesky factorization.

Other types of factorization: for $A \in \mathbb{R}^{n\times n}$ SPD: `symmetric` and `positive definite`

If A is symmetric and positive definite, then the diagonal matrix D in Corollary 4.1 has positive entries. Moreover, the following result holds.

THEOREM 4.4 (Cholesky factorization). *Let $A \in \mathbb{R}^{n\times n}$ be an* **SPD** *matrix. Then there exists an* **upper triangular matrix, with positive diagonal entries** *H such that*

$$A = H^T H, \qquad \text{(Cholesky factorization)}$$

i.e., $H^T := \{h_{ij}\}_{i,j=1:n}$ is a lower triangular matrix

$$H^T := \begin{bmatrix} h_{11} & 0 & 0 & \\ h_{12} & h_{22} & 0 & \\ \vdots & \vdots & \ddots & \\ h_{n1} & h_{n2} & h_{n3} & h_{nn} \end{bmatrix},$$

where $\{h_{ij}\}_{i,j=1:n}$ are computed as

$$\begin{cases} h_{11} = \sqrt{a_{11}}, \\ i = 2:n \begin{cases} h_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} h_{ik}h_{jk}\right)/h_{jj}, & j = 1,\dots,i-1, \\ h_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} h_{ik}^2\right)^{\frac{1}{2}}. \end{cases} \end{cases} \qquad \text{(Cholesky)}$$

The computational cost for the Cholesky factorization is $\mathcal{O}(\frac{n^3}{3})$.

EXAMPLE 4.3. *Find the* (Cholesky) *factorization of the* 3×3 *Hilbert matrix*

$$\texttt{hilb}(3) = \begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix}.$$

Using the Cholesky formulae we have

$$h_{11} = \sqrt{a_{11}} = 1, \qquad (i=1)$$

$$h_{21} = \Big(a_{21} - \sum_{k=1}^{0} h_{2k}h_{1k}\Big)/h_{11} \equiv a_{21}/h_{11} = (1/2)/1 = 1/2 \qquad (i=2, j=1)$$

$$h_{22} = \Big(a_{22} - \sum_{k=1}^{1} h_{2k}^2\Big)^{1/2} = \Big(1/3 - 1/4\Big)^{1/2} \equiv 1/\sqrt{12} \qquad (i=2, j=2)$$

$$h_{31} = \Big(a_{31} - \sum_{k=1}^{0} h_{3k}h_{1k}\Big)/h_{11} = (1/3)/1 \equiv 1/3 \qquad (i=3, j=1)$$

$$h_{32} = \Big(a_{32} - \sum_{k=1}^{1} h_{3k}h_{2k}\Big)/h_{22} = \Big(1/4 - h_{31}h_{21}\Big)/h_{22} = \Big(1/4 - 1/6\Big)/(1/\sqrt{12}) \equiv 1/\sqrt{12} \qquad (i=3, j=2)$$

$$h_{33} = \Big(a_{33} - \sum_{k=1}^{2} h_{3k}^2\Big)^{1/2} = \Big(1/5 - 1/9 - 1/16\Big)^{1/2} = 1/\sqrt{180} \qquad (i=3, j=3)$$

i.e.,

$$H^T = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1/\sqrt{12} & 0 \\ 1/3 & 1/\sqrt{12} & 1/\sqrt{180} \end{bmatrix}.$$

Check your hand-computed solution against Matlab's solution:

```
>> help chol
 chol   Cholesky factorization.
    R = chol(A) calculates the Cholesky factor of full or sparse A using
    the diagonal and upper triangle of A, such that A = R'*R. A must be
    positive definite, and the lower triangle is assumed to be the (complex
    conjugate) transpose of the upper triangle.
>> R = chol(hilb(3)); R'
ans =
    1.0000         0         0
    0.5000    0.2887         0
    0.3333    0.2887    0.0745

>> n=3;R=chol(hilb(n));norm(hilb(n)-R'*R,inf)

ans =

     0
>> n=14;R=chol(hilb(n));norm(hilb(n)-R'*R,inf)
```

```
ans =

   8.3267e-17

>> n=15;R=chol(hilb(n));norm(hilb(n)-R'*R,inf)
Error using chol
Matrix must be positive definite.
>> cond(hilb(15))

ans =

   2.4960e+17
```

4.4. Exercises.

Exercise 1. *Using naive Gaussian elimination, factor the following matrices in the form $A = LU$, where L is a unit lower triangular matrix and U is an upper triangular matrix.*

(a) $A = \begin{bmatrix} 3 & 0 & 3 \\ 0 & -1 & 3 \\ 1 & 3 & 0 \end{bmatrix}$

(b) $A = \begin{bmatrix} 1 & 0 & \frac{1}{3} & 0 \\ 0 & 1 & 3 & 1 \\ 3 & -3 & 0 & 6 \\ 0 & 2 & 4 & -6 \end{bmatrix}$

(c) $A = \begin{bmatrix} -20 & -15 & -10 & -5 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

Solution:

(a)

$$A^{(1)} = \begin{bmatrix} 3 & 0 & 3 \\ 0 & -1 & 3 \\ 1 & 3 & 0 \end{bmatrix}, \quad \begin{bmatrix} \odot \\ 0 \\ 1/3 \end{bmatrix}, \quad A^{(2)} = \begin{bmatrix} 3 & 0 & 3 \\ 0 & -1 & 3 \\ 0 & 3 & -1 \end{bmatrix}, \quad \begin{bmatrix} \odot \\ \odot \\ -3 \end{bmatrix}, \quad A^{(3)} = \begin{bmatrix} 3 & 0 & 3 \\ 0 & -1 & 3 \\ 0 & 0 & 8 \end{bmatrix},$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/3 & -3 & 1 \end{bmatrix}, \qquad U = A^{(3)}.$$

```
>> A = [3 0 3; 0 -1 3; 1 3 0];
>> NaiveGauss(A)
ans =
    3.0000         0    3.0000
         0   -1.0000    3.0000
    0.3333   -3.0000    8.0000
>> [B,l]=Gauss(A)

B =
```

```
    3.0000         0    3.0000
         0   -0.3333    2.6667
    0.3333    3.0000   -1.0000


l =

     1     3     2
```

i.e.,

$$\widetilde{L} = \begin{bmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 0 & -1/3 & 1 \end{bmatrix}, \qquad \widetilde{U} = \begin{bmatrix} 3 & 0 & 3 \\ 0 & 3 & -1 \\ 0 & 0 & 8/6 \end{bmatrix},$$

which means

$$P * A = \widetilde{L} * \widetilde{U}.$$

```
>> [L,U,P]=lu(A)

L =

    1.0000         0         0
    0.3333    1.0000         0
         0   -0.3333    1.0000


U =

    3.0000         0    3.0000
         0    3.0000   -1.0000
         0         0    2.6667


P =

     1     0     0
     0     0     1
     0     1     0
```

(b)

$$A^{(1)} = \begin{bmatrix} 1 & 0 & \frac{1}{3} & 0 \\ 0 & 1 & 3 & 1 \\ 3 & -3 & 0 & 6 \\ 0 & 2 & 4 & -6 \end{bmatrix} \quad \begin{bmatrix} \odot \\ 0 \\ 3 \\ 0 \end{bmatrix}, \quad A^{(2)} = \begin{bmatrix} 1 & 0 & \frac{1}{3} & 0 \\ 0 & 1 & 3 & 1 \\ 0 & -3 & -1 & 6 \\ 0 & 2 & 4 & -6 \end{bmatrix} \quad \begin{bmatrix} \odot \\ \odot \\ -3 \\ 2 \end{bmatrix},$$

$$A^{(3)} = \begin{bmatrix} 1 & 0 & \frac{1}{3} & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 8 & 9 \\ 0 & 0 & -2 & -8 \end{bmatrix} \quad \begin{bmatrix} \odot \\ \odot \\ \odot \\ -1/4 \end{bmatrix}, \quad A^{(4)} = \begin{bmatrix} 1 & 0 & \frac{1}{3} & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 8 & 9 \\ 0 & 0 & 0 & -23/4 \end{bmatrix},$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 3 & -3 & 1 & 0 \\ 0 & 2 & -1/4 & 1 \end{bmatrix}, \quad U = A^{(4)}.$$

```
>> A = [1 0 1/3 0; 0 1 3 1 ; 3 -3 0 6; 0 2 4 -6];
>> NaiveGauss(A)
ans =
    1.0000         0    0.3333         0
         0    1.0000    3.0000    1.0000
    3.0000   -3.0000    8.0000    9.0000
         0    2.0000   -0.2500   -5.7500
```

(c) $A = \begin{bmatrix} -20 & -15 & -10 & -5 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

```
>> A = [1 0 0 2 ; 0 3 0 0 ; 0 9 4 0; 5 0 8 10];
>> NaiveGauss(A)
ans =
     1     0     0     2
     0     3     0     0
     0     3     4     0
     5     0     2     0
```

Exercise 2. *Consider the matrix*

$$A = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 3 & 0 & 0 \\ 0 & 9 & 4 & 0 \\ 5 & 0 & 8 & 10 \end{bmatrix}$$

(a) Determine a unit lower triangular matrix M and an upper triangular matrix U such that $MA = U$.

(b) Determine a unit lower triangular matrix L and an upper triangular matrix U such that $A = LU$. Show that $ML = I$ so that $L = M^{-1}$.

Solution: The lower triangular matrix L such that

$$A = LU, \qquad L^{-1}A = U,$$

containing the multipliers from the naive Gaussian elimination procedure is

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 3 & 1 & 0 \\ 5 & 0 & 2 & 1 \end{bmatrix},$$

hence

$$M = L^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -3 & 1 & 0 \\ -5 & 0 & -2 & 1 \end{bmatrix}.$$

Exercise 3. *Consider the matrix*

$$A = \begin{bmatrix} 25 & 0 & 0 & 0 & 1 \\ 0 & 27 & 4 & 3 & 2 \\ 0 & 54 & 58 & 0 & 0 \\ 0 & 108 & 116 & 0 & 0 \\ 100 & 0 & 0 & 0 & 24 \end{bmatrix}$$

(a) Determine a unit lower triangular matrix M and an upper triangular matrix U such that $MA = U$.
(b) Determine $M^{-1} = L$ such that $A = LU$.

Solution:

Exercise 4. *Consider the matrix*

$$A = \begin{bmatrix} 2 & 2 & 1 \\ 1 & 1 & 1 \\ 3 & 2 & 1 \end{bmatrix}$$

(a) Show that A **cannot** *be factored into the product of a unit lower triangular matrix and an upper triangular matrix.*
(b) Interchange the rows of A so that this can be done.

Solution:

Exercise 9. *Consider*

$$A = \begin{bmatrix} 2 & -1 & 2 \\ 2 & -3 & 3 \\ 6 & -1 & 8 \end{bmatrix}$$

(a) Find the matrix factorization $A = LDU'$, where L is unit lower triangular, D is diagonal, and U' is unit upper triangular.
(b) Use this decomposition of A to solve $Ax = b$, where $b = [-2, -5, 0]^T$.

Solution:

Exercise 10. *Consider*

$$A = \begin{bmatrix} -2 & 1 & -2 \\ -4 & 3 & -3 \\ 2 & 2 & 4 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 4 \\ 4 \end{bmatrix},$$

(a) Find the matrix factorization $A = LDM^T$, where L is unit lower triangular, D is diagonal, and M^T is unit upper triangular.
(b) Use this decomposition of A to solve $Ax = b$.

Solution:

(a)

$$[A^{(1)}|b^{(1)}] = \left[\begin{array}{ccc|c} -2 & 1 & -2 & 1 \\ -4 & 3 & -3 & 4 \\ 2 & 2 & 4 & 4 \end{array}\right], \quad \left[\begin{array}{c} \odot \\ 2 \\ -1 \end{array}\right], \quad [A^{(2)}|b^{(2)}] = \left[\begin{array}{ccc|c} -2 & 1 & -2 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 3 & 2 & 5 \end{array}\right], \quad \left[\begin{array}{c} \odot \\ \odot \\ 3 \end{array}\right],$$

$$[A^{(3)}|b^{(3)}] = \left[\begin{array}{ccc|c} -2 & 1 & -2 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & -1 & -1 \end{array}\right], \qquad x_3 = 1,\ x_2 = 1,\ x_1 = -1,$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 3 & 1 \end{bmatrix}, \quad D = \mathtt{diag}(A^{(3)}) = \begin{bmatrix} -2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \quad M^T = D^{-1} * A^{(3)} \begin{bmatrix} 1 & -1/2 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

```
>> A = [-2 1 -2 ; -4 3 -3; 2 2 4]; b=[1;4;4];
>> NaiveGauss(A)
ans =
    -2     1    -2
     2     1     1
    -1     3    -1
```

which gives

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 3 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} -2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \quad M^T = D^{-1} * \begin{bmatrix} -2 & 1 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & -1 \end{bmatrix} \equiv \begin{bmatrix} 1 & -1/2 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

(b)
```
>> x = NaiveSolve(A,b)
x=
    -1     1     1
>> y = A\b
y =
   -1.0000
    1.0000
    1.0000
```

Exercise 14. *Consider the matrix* $A =$*Tridiagonal*$(a_{i,i-1}, a_{ii}, a_{i,i+1})$, *where* $a_{i,i} \neq 0$.

(a) *Establish the algorithm for the determining the elements of a lower bidiagonal matrix* $L = (\ell_{ij})$ *and a* **unit** *upper bidiagonal matrix* $U = (u_{ij})$ *such that* $A = LU$.

```
function [L,U]  = Exercise8_1_14a(A)
[m,n] = size(A);
if abs(m-n)>0
   error('The Matrix A is not square');
end
U=eye(n);
L(1,1) = A(1,1);
for i=2:n
    L(i,i-1)=A(i,i-1);
    U(i-1,i)=A(i-1,i)/L(i-1,i-1);
    L(i,i) = A(i,i) - L(i,i-1)*U(i-1,i);
end
```

(b) Establish the algorithm for the determining the elements of a **unit** *lower bidiagonal matrix* $LL = (\ell_{ij})$ *and an upper bidiagonal matrix* $UU = (u_{ij})$ *such that* $A = LL * UU$.

```
function [L,U]  = Exercise8_1_14b(A)
[m,n] = size(A);
if abs(m-n)>0
   error('The Matrix A is not square');
end
L=eye(n);
U(1,1) = A(1,1);
for i=2:n
    U(i-1,i) = A(i-1,i);
    L(i,i-1) = A(i,i-1)/U(i-1,i-1);
    U(i,i) = A(i,i) - L(i,i-1)*U(i-1,i);
end
```

Solution:

```
(a) >> A = [11 12 0 0; 21 22 23 0; 0 32 33 34; 0 0 43 44];
    >> [L,U]  = Exercise8_1_1ab(A)
    >> L
    L =
       11.0000         0         0         0
       21.0000   -0.9091         0         0
            0   32.0000  842.6000         0
            0         0   43.0000   42.2649
    >> U
    U =
        1.0000    1.0909         0         0
             0    1.0000  -25.3000         0
             0         0    1.0000    0.0404
             0         0         0    1.0000
(b) >> A = [11 12 0 0; 21 22 23 0; 0 32 33 34; 0 0 43 44];
    >> [L,U]  = Exercise8_1_14b(A)
    >> L
    L =
        1.0000         0         0         0
        1.9091    1.0000         0         0
             0  -35.2000    1.0000         0
             0         0    0.0510    1.0000
    >> U
    U =
       11.0000   12.0000         0         0
             0   -0.9091   23.0000         0
             0         0  842.6000   34.0000
             0         0         0   42.2649
```

Exercise 18. *Find the LU factorization of this matrix:*

$$A = \begin{bmatrix} 2 & 2 & 1 \\ 4 & 7 & 2 \\ 2 & 11 & 5 \end{bmatrix}.$$

Solution:

```
>> A=[2 2 1; 4 7 2 ; 2 11 5];
>> [L,U,P] = lu(A)   % L*U = P*A
L =

    1.0000         0         0
    0.5000    1.0000         0
    0.5000   -0.2000    1.0000
U =
    4.0000    7.0000    2.0000
         0    7.5000    4.0000
         0         0    0.8000
P =
     0     1     0
     0     0     1
     1     0     0

>> [LL,UU]=lu(A)   % LL*UU = A
LL =
    0.5000   -0.2000    1.0000
    1.0000         0         0
    0.5000    1.0000         0
UU =
    4.0000    7.0000    2.0000
         0    7.5000    4.0000
         0         0    0.8000
>> NaiveGauss(A)
ans =
     2     2     1
     2     3     0
     1     3     4
>> [B,l]=Gauss(A)
B =
    2.0000    2.0000    1.0000
    2.0000    0.3333   -1.3333
    1.0000    9.0000    4.0000
l =
     1     3     2
```

The Naive Gaussian Elimination gives:

$$A^{(1)} = \begin{bmatrix} 2 & 2 & 1 \\ 4 & 7 & 2 \\ 2 & 11 & 5 \end{bmatrix}, \quad \begin{bmatrix} \odot \\ 2 \\ 1 \end{bmatrix} \implies A^{(2)} = \begin{bmatrix} 2 & 2 & 1 \\ 0 & 3 & 0 \\ 0 & 9 & 4 \end{bmatrix}, \quad \begin{bmatrix} \odot \\ \odot \\ 3 \end{bmatrix} \implies A^{(3)} = \begin{bmatrix} 2 & 2 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix},$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 3 & 1 \end{bmatrix}, \qquad U = A^{(3)} = \begin{bmatrix} 2 & 2 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix}.$$

Exercise 19.

(a) Prove that the product of two lower triangular matrices is lower triangular.
(b) Prove that the product of two unit lower triangular matrices is unit lower triangular.
(c) Prove that the inverse of a unit lower triangular matrix is unit lower triangular.
(d) By using the transpose operation, prove that all of the preceding results are true for upper triangular matrices.

Solution:

Exercise 21. *Determine the LDL^T factorization for the following matrix:*

$$A = \begin{bmatrix} 1 & 2 & -1 & 1 \\ 2 & 3 & -4 & 3 \\ -1 & -4 & -1 & 3 \\ 1 & 3 & 3 & 0 \end{bmatrix}.$$

Solution:

```
>> A=[1 2 -1 1;2 3 -4 3; -1 -4 -1 3; 1 3 3 0];
>> [L,D,p] = ldl(A,'vector') % P' * A * P = L* D * L'
L =
    1.0000         0         0         0
   -1.3333    1.0000         0         0
    1.0000   -1.1053    1.0000         0
    0.6667   -0.2632    0.1778    1.0000
D =
    3.0000         0         0         0
         0   -6.3333         0         0
         0         0    4.7368         0
         0         0         0   -0.0444
p =
     2     3     4     1
>> eig(A)
ans =
   -6.2525
   -0.0372
    2.5525
    6.7372
```

The Naive Gaussian Elimination gives:

$$A^{(1)} = \begin{bmatrix} 1 & 2 & -1 & 1 \\ 2 & 3 & -4 & 3 \\ -1 & -4 & -1 & 3 \\ 1 & 3 & 3 & 0 \end{bmatrix}, \quad \begin{bmatrix} \odot \\ 2 \\ -1 \\ 1 \end{bmatrix} \implies A^{(2)} = \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & -1 & -2 & 1 \\ 0 & -2 & -2 & 4 \\ 0 & 1 & 4 & -1 \end{bmatrix} \quad \begin{bmatrix} \odot \\ \odot \\ 2 \\ -1 \end{bmatrix}$$

$$\implies A^{(2)} = \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & -1 & -2 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 0 \end{bmatrix} \begin{bmatrix} \odot \\ \odot \\ \odot \\ 1 \end{bmatrix} \implies A^{(3)} = \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & -1 & -2 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & -2 \end{bmatrix},$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ -1 & 2 & 1 & 0 \\ 1 & -1 & 1 & 1 \end{bmatrix}, \qquad D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}.$$

Exercise 22*. *Find the* (Cholesky) *factorization of the* 3×3 *Hilbert matrix*

Solution: See Example 4.3.

Exercise 22. *Find the* (Cholesky) *factorization of*

$$A = \begin{bmatrix} 4 & 6 & 10 \\ 6 & 25 & 19 \\ 10 & 19 & 62 \end{bmatrix}.$$

Solution: Check your hand-computed solution against Matlab's solution:

```
>> R= chol(A)
R =
     2     3     5
     0     4     1
     0     0     6
>> R'*R
ans =
     4     6    10
     6    25    19
    10    19    62
```

Exercise 24. *Determine the* LDL^T *factorization of the matrix*

$$A = \begin{bmatrix} 3 & 35 & -20 & 65 \\ 35 & 244 & -143 & 461 \\ -20 & -143 & 73 & -232 \\ 65 & 461 & -232 & 856 \end{bmatrix}.$$

Can you find the Cholesky factorization?

Solution:

```
>> A=[3 35 -20 65; 35 244 -143 461; -20 -143 73 -232; 65 461 -232 856];
>> [L,D,p] = ldl(A).
% returns unit lower triangular matrix L, block diagonal D,
% and permutation matrix P so that P'*A*P = L*D*L'
L =
    1.0000         0         0         0
    0.5386    1.0000         0         0
   -0.2710         0    1.0000         0
    0.0759    0.1167   -0.0273    1.0000
D =
  856.0000         0         0         0
```

```
         0   -4.2722  -18.0561         0
         0  -18.0561   10.1215         0
         0         0         0   -2.0001
p =
     0     0     0     1
     0     1     0     0
     0     0     1     0
     1     0     0     0
>> eig(A)
ans =
   1.0e+03 *
   -0.0154
   -0.0020
    0.0177
    1.1757
>> R = chol(A,'lower')
Error using chol
Matrix must be positive definite.
```

The Naive Gaussian Elimination gives

$$A^{(1)} = \begin{bmatrix} 3 & 35 & -20 & 65 \\ 35 & 244 & -143 & 461 \\ -20 & -143 & 73 & -232 \\ 65 & 461 & -232 & 856 \end{bmatrix}, \quad \begin{bmatrix} \odot \\ 35/3 \\ -20/3 \\ 65/3 \end{bmatrix}$$

$$\Longrightarrow \quad A^{(2)} = \begin{bmatrix} 3 & 35 & -20 & 65 \\ 0 & -493/3 & 271/3 & -892/3 \\ 0 & 271/3 & -181/3 & 604/3 \\ 0 & -892/3 & 604/3 & -1657/3 \end{bmatrix}, \quad \begin{bmatrix} \odot \\ \odot \\ -271/493 \\ 892/493 \end{bmatrix}$$

$$\Longrightarrow \quad A^{(3)} = \begin{bmatrix} 3 & 35 & -20 & 65 \\ 0 & -493/3 & 271/3 & -892/3 \\ 0 & 0 & -5264/493 & 18680/493 \\ 0 & 0 & 18680/493 & -7079/493 \end{bmatrix}, \quad \begin{bmatrix} \odot \\ \odot \\ \odot \\ -18680/5264 \end{bmatrix}$$

$$\Longrightarrow \quad A^{(4)} = \begin{bmatrix} 3 & 35 & -20 & 65 \\ 0 & -493/3 & 271/3 & -892/3 \\ 0 & 0 & -5264/493 & 77242/493 \\ 0 & 0 & 0 & 632208/5264 \end{bmatrix},$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 35/3 & 1 & 0 & 0 \\ -20/3 & -271/493 & 1 & 0 \\ 65/3 & 892/493 & -18680/5264 & 1 \end{bmatrix},$$

$$D = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & -493/3 & 0 & 0 \\ 0 & 0 & -5264/493 & 0 \\ 0 & 0 & 0 & 632208/5264 \end{bmatrix}.$$

```
>> L =[1 0 0 0; 35/3 1 0 0 ; -20/3 -271/493 1 0 ; 65/3 892/493 -18680/5264 1];
>> D = [3 0 0 0 ; 0 -493/3 0 0 ; 0 0 -5264/493 0; 0 0 0 632208/5264];
>> L*D*L'
```

```
ans =
    3.0000   35.0000  -20.0000   65.0000
   35.0000  244.0000 -143.0000  461.0000
  -20.0000 -143.0000   73.0000 -232.0000
   65.0000  461.0000 -232.0000  856.0000
>> NaiveGauss(A)
ans =
    3.0000   35.0000  -20.0000   65.0000
   11.6667 -164.3333   90.3333 -297.3333
   -6.6667   -0.5497  -10.6775   37.8905
   21.6667    1.8093   -3.5486  120.1003
```

Computer problem 4. *Write and test a procedure for determining A^{-1} for a given square matrix A of order n. Your procedure should use procedures* `Gauss` *and* `Solve`.
Solution:

CHAPTER 3

ITERATIVE METHODS FOR SOLVING LINEAR SYSTEMS

1. Iterative methods: consistency and convergence

Idea: in order to solve $A\boldsymbol{x} = \boldsymbol{b}$, construct a sequence of vectors $\{\boldsymbol{x}^{(\kappa)}\}_{\kappa \geq 0}$, starting from a given ("initial guess") $\boldsymbol{x}^{(0)}$ such that

$$\lim_{k \to \infty} \boldsymbol{x}^{(\kappa)} = \boldsymbol{x} \equiv \text{'}A \backslash b\text{'},$$

where the convergence is meant in a 'vector norm' sense, i.e.,

$$\lim_{k \to \infty} \|\boldsymbol{x}^{(\kappa)} - \boldsymbol{x}\| = 0.$$

REMARK 1.1. **At each iteration** κ, *the vector* $\boldsymbol{x}^{(\kappa)}$ *allows to compute a* **residual**

$$\boldsymbol{r}^{(\kappa)} = \boldsymbol{b} - A\boldsymbol{x}^{(\kappa)},$$

which costs (in the case of a full matrix A) order of $\mathcal{O}(2n^2)$, *more exactly* $\nu(n) = 2n^2$ *operations.*

Recall: the cost of solving $Ax = b$ with **direct methods** is $\mathcal{O}(\frac{2}{3}n^3)$ operations.
Conclusion: iterative methods are competitive in comparison with direct methods provided the number of iterates κ required for convergence (within a prescribed toelrance)

$$\|\boldsymbol{x} - \boldsymbol{x}^{(\kappa)}\| < \texttt{tol}$$

is independent of n, or it scales sublinearly with n.

REMARK 1.2. *The residual* $\boldsymbol{r}^{(\kappa)}$ *is known (computable) at each iteration, while the error* $\boldsymbol{e}^{\kappa} = \boldsymbol{x} - \boldsymbol{x}^{(\kappa)}$ *has to be "***guesstimated***"!*

REMARK 1.3.
- **Direct methods** *are good for* **dense systems**, *of* **small or modest size**, *such that the entire matrix could be stored (GEM).*
- **Iterative methods** *are good for* **large sparse**, *and* **large dense systems**.

```
>> help spy
 spy Visualize sparsity pattern.
    spy(S) plots the sparsity pattern of the matrix S.
>> n=101;A=hilb(n);[L,U,P]=lu(A);spy (L)
>> spy(P)
```

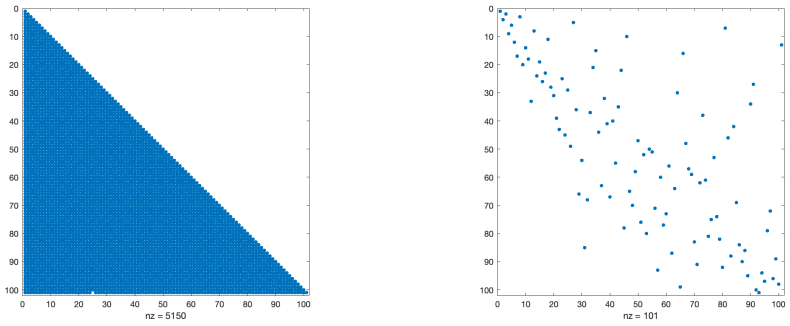


FIGURE 1. Matlab's 'sparse' function for L and P: `spy(L)` and `spy(P)`, respectively.

EXAMPLE 1.1. *The discretization of boundary value problems (BVP) for partial differential equations (PDEs) yields large sparse systems, with patterns for sparsity, see e.g. the Poisson equation* (3.1) *and its approximation.*

REMARK 1.4. *Let's compare the operation costs necessary for a direct method to obtain the solution $\boldsymbol{x} = A^{-1}b$, versus an iterative method, to obtain an (approximation) iteration $\boldsymbol{x}^{(\kappa)}$ within a given tolerance $\varepsilon > \|\boldsymbol{x} - \boldsymbol{x}^{(\kappa)}\|$.*
The respective costs are $\frac{2}{3}n^3$ versus $\kappa \times 2n^2$.
For example, if at each iteration it holds

$$\|\boldsymbol{x} - \boldsymbol{x}^{(\kappa)}\| \le C\|\boldsymbol{x} - \boldsymbol{x}^{(k-1)}\| \le \cdots \le C^{\kappa}\|\boldsymbol{x} - \boldsymbol{x}^{(0)}\|,$$

where $C < 1$, then in order to have

$$\|\boldsymbol{x} - \boldsymbol{x}^{(\kappa)}\| \le C^{\kappa}\|x - x^{(0)}\| \le \varepsilon\|\boldsymbol{x} - \boldsymbol{x}^{(0)}\|,$$

it suffices to have $C^{\kappa} \le \varepsilon$, or $\kappa \underbrace{\log C}_{\le 0} \le \log \varepsilon$, i.e.,

$$\kappa \ge \frac{\log \varepsilon}{\log C}.$$

Finally, for an iterative method to be more efficient than the direct method it would mean

$$\kappa \cdot 2n^2 < \frac{2n^3}{3} \quad \Longrightarrow \quad n \ge 3\kappa \ge \frac{\log \varepsilon}{\log C},$$

which would hold for large enough systems.

$$\frac{\log(1.e-3)}{\log(.5)} \approx 9.97, \qquad \frac{\log(1.e-7)}{\log(.5)} \approx 23.25, \qquad \frac{\log(1.e-15)}{\log(.5)} \approx 49.83.$$

1.1. Convergence of iterative methods.

Idea: construct $\{\boldsymbol{x}^{(\kappa)}\}$ such that $\boldsymbol{x}^{(\kappa)} \xrightarrow[\kappa \nearrow \infty]{\|\cdot\|} \boldsymbol{x} = A^{-1}b$.
Recall that $\boldsymbol{x}$ is not available!
An ideal stopping criterion for the iterative method should be

$$\|\boldsymbol{x} - \boldsymbol{x}^{(\kappa)}\| \le \varepsilon := \text{tolerance.} \qquad \text{(ideal stopping criterion)}$$

Goal: given an iterative method of type

$$\boldsymbol{x}^{(\kappa+1)} = M\boldsymbol{x}^{(\kappa)} + f,$$

find a stopping criterion (independent of the solution $\boldsymbol{x}$).
Note that above we assumed that

$$\boldsymbol{x} = M\boldsymbol{x} + f, \quad \text{i.e.,} \quad f = (I-M)\boldsymbol{x} = (I-M)A^{-1}b$$

hence

$$(I-M)A^{-1}b = f.$$

Subtracting $\boldsymbol{x}^{(\kappa+1)} = M\boldsymbol{x}^{(\kappa)} + f$ from $\boldsymbol{x} = M\boldsymbol{x} + f$ gives

$$\boldsymbol{x} - \boldsymbol{x}^{(\kappa+1)} = M(\boldsymbol{x} - \boldsymbol{x}^{(\kappa)}),$$

hence the errors in the κ iterations, i.e., $\boldsymbol{x} - \boldsymbol{x}^{(\kappa)}$, satisfy (in a vector norm) the following 'recurrent' inequality

$$\begin{aligned} \|\boldsymbol{x} - \boldsymbol{x}^{(\kappa+1)}\| = \|M(\boldsymbol{x} - \boldsymbol{x}^{(\kappa)})\| &\le \|M\|\|(\boldsymbol{x} - \boldsymbol{x}^{(\kappa)})\| \qquad (1.1)\\ &\le \|M\|^2\|(\boldsymbol{x} - \boldsymbol{x}^{(\kappa-1)})\| \le \|M\|^3\|(\boldsymbol{x} - \boldsymbol{x}^{(\kappa-2)})\| \le \cdots \le \|M\|^{k+1}\|(\boldsymbol{x} - \boldsymbol{x}^{(0)})\|. \end{aligned}$$

This suggests that, in order to have a convergence iterative method, we would need

$$\|M\| < 1.$$

On the other hand, subtracting $\boldsymbol{x}^{(\kappa)} = M\boldsymbol{x}^{(\kappa-1)} + f$ from $\boldsymbol{x}^{(\kappa+1)} = M\boldsymbol{x}^{(\kappa)} + f$ yields

$$\boldsymbol{x}^{(\kappa+1)} - \boldsymbol{x}^{(\kappa)} = M(\boldsymbol{x}^{(\kappa)} - \boldsymbol{x}^{(\kappa-1)}),$$

and therefore

$$\frac{\|\boldsymbol{x}^{(\kappa+1)} - \boldsymbol{x}^{(\kappa)}\|}{\|\boldsymbol{x}^{(\kappa)} - \boldsymbol{x}^{(\kappa-1)}\|} \le \|M\|.$$

Using the inverse triangle inequality and (1.1) we have

$$\|\boldsymbol{x}^{(\kappa+1)} - \boldsymbol{x}^{(\kappa)}\| \ge \|\boldsymbol{x}^{(\kappa)} - \boldsymbol{x}\| - \|\boldsymbol{x}^{(\kappa+1)} - \boldsymbol{x}\| \ge \frac{1}{\|M\|}\|\boldsymbol{x}^{(\kappa+1)} - \boldsymbol{x}\| - \|\boldsymbol{x}^{(\kappa+1)} - \boldsymbol{x}\| \qquad \text{(useful estimate)}$$

$$= \frac{1-\|M\|}{\|M\|}\|\boldsymbol{x}^{(\kappa+1)} - \boldsymbol{x}\|.$$

Finally, putting together (ideal stopping criterion) and (useful estimate) we obtain our (useful) stopping criterion

$$\|\boldsymbol{x}^{(\kappa+1)} - \boldsymbol{x}\| \le \frac{\|M\|}{1-\|M\|}\|\boldsymbol{x}^{(\kappa+1)} - \boldsymbol{x}^{(\kappa)}\| \le \varepsilon, \qquad \text{(stopping criterion)}$$

in the sense that the term in the red color are known (computed) quantities.

DEFINITION 1.1. *We consider (stationary)* **iterative methods** *of the form:*

given $\boldsymbol{x}^{(0)}$, *construct* $\{\boldsymbol{x}^{(\kappa)}\}_{\kappa\ge 0}$ *by:*

$$\boldsymbol{x}^{(\kappa+1)} = B\boldsymbol{x}^{(\kappa)} + f, \quad \text{(stationary iterative method)}$$

where $B \in \mathscr{M}_{n\times n}$ *is the* **iteration matrix**, *and* f *is a vector obtained from the RHS* $b = Ax$.

DEFINITION 1.2. *An iterative method* $\boldsymbol{x}^{(\kappa+1)} = B\boldsymbol{x}^{(\kappa)} + f$ *is* **consistent** *with the equation* $A\boldsymbol{x} = b$ *if* f *and the iteration matrix* B *satisfy*

$$\boldsymbol{x} = B\boldsymbol{x} + f, \qquad \text{(consistent iterative method)}$$

equivalently

$$f = (I - B)\boldsymbol{x} = (I - B)A^{-1}b.$$

DEFINITION 1.3. *The error in the iteration* κ*:*

$$\boldsymbol{e}^{(\kappa)} = \boldsymbol{x} - \boldsymbol{x}^{(\kappa)}.$$

REMARK 1.5. *The convergence condition*

$$\boldsymbol{x} = \lim_{\kappa \to \infty} \boldsymbol{x}^{(\kappa)}$$

writes the

$$0 = \boldsymbol{x} - \boldsymbol{x} = \boldsymbol{x} - \lim_{\kappa \to \infty} \boldsymbol{x}^{(\kappa)} = \lim_{\kappa \to \infty} (\boldsymbol{x} - \boldsymbol{x}^{(\kappa)}) = \lim_{\kappa \to \infty} \boldsymbol{e}^{(\kappa)},$$

for any choice of initial datum (initial guess) $\boldsymbol{x}^{(0)}$.

REMARK 1.6. **Consistency** *alone does not suffice to ensure convergence of the iterative method* $\boldsymbol{x}^{\kappa+1} = B\boldsymbol{x}^{(\kappa)} + f$.

EXAMPLE 1.2. *To solve* $2I\boldsymbol{x} = \boldsymbol{b}$*, consider the iterative method* $\boldsymbol{x}^{(\kappa+1)} = -\boldsymbol{x}^{(\kappa)} + \boldsymbol{b}$*. Hence in this case*

$$A = 2I, \quad \boldsymbol{x} = \frac{1}{2}\boldsymbol{b},$$
$$B = -I, f = \boldsymbol{b}$$

the method is obviously consistent:

$$\boldsymbol{x} = B\boldsymbol{x} + f,$$

since

$$\frac{1}{2}\boldsymbol{b} = -I\frac{1}{2}\boldsymbol{b} + \boldsymbol{b}.$$

Nonetheless, the method is **not convergent** *for all choices of the initial guess:*

$$\textit{If} \quad \boldsymbol{x}^{(0)} = \boldsymbol{0}, \quad \textit{then} \quad \boldsymbol{x}^{(2\kappa)} = \boldsymbol{0}, \boldsymbol{x}^{(2\kappa+1)} = \boldsymbol{b}, \quad \kappa = 0, 1, \cdots$$
$$\textit{If} \quad \boldsymbol{x}^{(0)} = \frac{1}{2}\boldsymbol{b}, \quad \textit{the method is convergent} \quad \boldsymbol{x}^{(\kappa)} = \frac{1}{2}\boldsymbol{b}, \quad \forall \kappa.$$

Recall the rootfinding problem for the nonlinear functions

$$F(y) = 0,$$

was analyzed by looking at the fixed-point iteration problem, where

$$\Phi(y) = y. \qquad \text{(in our case } \Phi(y) = By + f\text{)}$$

The fix-point problem was solved by an iterative method:

$$x^{(\kappa+1)} = \Phi x^{(\kappa)},$$

and this was convergent $\xleftrightarrow{\text{if and only if}}$ $|\Phi'(y)| < 1$. If we consider $\Phi(y) = By + f$, then

$$\nabla\Phi(y) = B$$

hence convergence would be guaranteed provided $\|B\| < 1$, which is a consequence of a stronger result, which we should see now, namely convergence is equivalent to the condition that the spectral radius is less than one: $|\texttt{eig}(B)| < 1$.

THEOREM 1.1. *(5.6.1 in [4, pp. 191]) A necessary and sufficient condition for a stationary iterative method $\boldsymbol{x}^{(\kappa+1)} = B\boldsymbol{x}^{(\kappa)} + f$ to converge for an arbitrary initial approximation $\boldsymbol{x}^{(0)}$ is that*

$$\rho(B) = \max_{1\le i\le n} |\lambda_i(B)| < 1,$$

where $\rho(B)$ is called the spectral radius of B.

PROOF. Assume that B has eigenvalues $\lambda_1, \lambda_2, \cdots \lambda_n$, and assume that the corresponding eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n$ are linearly independent. Then we can expand the initial error

$$\boldsymbol{e}^{(0)} = \boldsymbol{x} - \boldsymbol{x}^{(0)} = \alpha_1\boldsymbol{u}_1 + \cdots + \alpha_n\boldsymbol{u}_n,$$

and therefore

$$\boldsymbol{e}^{(\kappa)} = \boldsymbol{x} - \boldsymbol{x}^{(\kappa)} = \alpha_1\lambda_1^{\kappa}\boldsymbol{u}_1 + \cdots + \alpha_n\lambda_n^{\kappa}\boldsymbol{u}_n, \tag{1.2}$$

from which it follows that the iterations converge from an arbitrary starting approximation if and only if we have $|\lambda_i| < 1, \forall i = 1, \cdots, n$. The conclusion holds also without the assumption on the eigenvectors, see e.g., [18]. □

REMARK 1.7. *Let us assume that we want to reduce the amplitude of the error component $\alpha_i u_i$ of the initial error $\boldsymbol{e}^{(0)}$ by a factor of 10^{-m} in κ iterations. From* (1.2) *we immediately see that κ should satisfy*

$$|\lambda_j|^{\kappa} \le 10^{-m} \quad \Leftrightarrow \quad \underbrace{\kappa \log|\lambda_j|}_{<0} \le -m\log 10 \quad \Leftrightarrow \quad \kappa \ge -m\frac{\log 10}{\log|\lambda_j|}.$$

REMARK 1.8. *Since $\rho(A) \le \|A\|$ for all consistent norms (see Theorem 2.3), we have that a sufficient condition for convergence of a stationary iterative method is $\|B\| < 1$.*

REMARK 1.9. *Recall the stopping criterion stopping criterion*

$$\|\boldsymbol{x} - \boldsymbol{x}^{(\kappa)}\| \le \frac{\|B\|}{1-\|B\|}\|\boldsymbol{x}^{(\kappa)} - \boldsymbol{x}^{(\kappa-1)}\| \le \texttt{tolerance}.$$

We note that if $\|B\| < 0.5$, then the iterations can be stopped when

$$\|\boldsymbol{x}^{(\kappa)} - \boldsymbol{x}^{(\kappa-1)}\| \le \texttt{tolerance}. \qquad \text{(useful stopping criterion)}$$

General idea: the most straightforward approach to an iterative solution of a linear system

$$A\boldsymbol{x} = b$$

is to write it as a linear fix-point iterations. One way would be

$$\boldsymbol{x} = (I - A)\boldsymbol{x} + b.$$

This leads to the following

DEFINITION 1.4.

$$x^{(\kappa+1)} = \underbrace{(I-A)}_{=B} x^{(\kappa)} + b. \qquad \text{(Richardson iteration)}$$

1.2. Linear Iterative Methods.

Idea: Technique for consistent linear iterative methods based on additive splitting:

$$A = P - N,$$

where $P =$ (preconditioner) nonsingular and easily invertible matrix (easy to solve $Py = \varphi$), for example P being diagonal or triangular matrix.
Recall: $Ax = b \quad \Leftrightarrow \quad Px = Nx + b$

DEFINITION 1.5 (Linear iterative method). *: Given $x^{(0)}$, compute $x^{(\kappa)}$, $\forall \kappa \geq 1$ by*

$$Px^{(\kappa+1)} = Nx^{(\kappa)} + b \quad \Leftrightarrow \quad x^{(\kappa+1)} = P^{-1}Nx^{(\kappa)} + P^{-1}b, \quad \forall \kappa \geq 0, \qquad \text{(linear iterative method)}$$

i.e, the iteration matrix $B := P^{-1}N$, while $f := P^{-1}b$.

REMARK 1.10. *Alternatively, using the definition of the residual at the iteration κ*

$$r^{(\kappa)} = b - Ax^{(\kappa)} = b - (P-N)x^{(\kappa)},$$

we have that

$$P^{-1}r^{(\kappa)} = P^{-1}b - (I - P^{-1}N)x^{(\kappa)} = P^{-1}b - x^{(\kappa)} + P^{-1}Nx^{(\kappa)},$$
$$P^{-1}b = P^{-1}r^{(\kappa)} + x^{(\kappa)} - P^{-1}Nx^{(\kappa)}.$$

In turn, this allows to express the linear iterative method

$$x^{(\kappa+1)} = P^{-1}Nx^{(\kappa)} + P^{-1}b = \cancel{P^{-1}Nx^{(\kappa)}} + P^{-1}r^{(\kappa)} + x^{(\kappa)} - \cancel{P^{-1}Nx^{(\kappa)}}$$

and therefore one can use

$$\begin{cases} x^{(\kappa+1)} = x^{(\kappa)} + P^{-1}r^{(\kappa)}, \\ r^{(\kappa)} = b - Ax^{(\kappa)}. \end{cases} \qquad \text{(iterative method)}$$

as an alternative definition for the linear iterative method.

REMARK 1.11.
(1) P has to be **easily invertible***, i.e., the equation $Py = \phi$ has to be* **easy to solve** *(e.g., P could be a diagonal or a triangular matrix.)*
(2) If $P \approx A$, i.e., $N \approx 0$, then $Px^{(\kappa+1)} \approx b$ is almost $Ax^{(\kappa+1)} \approx b$, the original problem.
(3) If $P \approx D$, i.e., $A \approx D - N$, then $Dx^{(\kappa+1)} = Nx^{(\kappa)} + f$, which is easy to invert.

THEOREM 1.2 (Householder lemma). *(see [13]) Let A be an SPD matrix, $A = P - N$. The error $e^{(\kappa)} = x - x^{(\kappa)}$ in the linear iterative method satisfies the following identity*

$$(e^{(\kappa)})^T A e^{(\kappa)} - (e^{(\kappa+1)})^T A e^{(\kappa+1)} = \frac{1}{2}\left(x^{(\kappa+1)} - x^{(\kappa)}\right)(P + P^T - A)\left(x^{(\kappa+1)} - x^{(\kappa)}\right)$$

or equivalently

$$\|e^{(\kappa)}\|_A^2 - \|e^{(\kappa+1)}\|_A^2 = \frac{1}{2}\|x^{(\kappa+1)} - x^{(\kappa)}\|_{P+P^T-A}^2.$$

PROOF. Direct calculation. □

COROLLARY 1.1. *Let A be an SPD matrix, $A = P - N$. If $P + P^T - A$ is positive definite, then the iterative method*

$$\begin{cases} \boldsymbol{x}^{(\kappa+1)} = \boldsymbol{x}^{(\kappa)} + P^{-1}\boldsymbol{r}^{(\kappa)}, \\ \boldsymbol{r}^{(\kappa)} = b - A\boldsymbol{x}^{(\kappa)}. \end{cases}$$

*is (***monotonically***) convergent for any choice of the initial guess $\boldsymbol{x}^{(0)}$:*

$$\|\boldsymbol{e}^{(\kappa+1)}\|_A < \|\boldsymbol{e}^{(\kappa)}\|_A.$$

PROOF. This follows directly from Theorem 1.2. □

2. Classical Linear Iterative Methods

2.1. Jacobi, Gauss-Seidel and Relaxation Methods.

Let us decompose the matrix A as follows

$$A = D - E - F, \qquad (D\boldsymbol{x} = (E+F)\boldsymbol{x} + b)$$

where

$$D = \mathtt{diag}(A), \quad E = \begin{cases} -a_{ij}, & i > j \\ 0, & i \le j \end{cases} \text{ (lower } \triangle), \quad F = \begin{cases} -a_{ij}, & i < j \\ 0, & i \ge j \end{cases} \text{ (upper } \triangle).$$

Key remark: if $\mathtt{diag}(A)$ has all nonzero entries, then

$$x_i = \frac{1}{a_{ii}}\Big(b - \sum_{j=1, j\neq i}^{n} a_{ij}x_j\Big) \equiv \frac{1}{a_{ii}}\Big(b - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n-1} a_{ij}x_j\Big) \equiv \sum_{j=1}^{i-1}\Big(-\frac{a_{ij}}{a_{ii}}\Big)x_j + \sum_{j=i+1}^{n-1}\Big(-\frac{a_{ij}}{a_{ii}}\Big)x_j + \frac{b}{a_{ii}},$$

$$\boldsymbol{x} = D^{-1}(E+F)\boldsymbol{x} + D^{-1}b$$

or written explicitly

$$\begin{cases} a_{11}x_1 & = & \square & - & a_{12}x_2 & - & a_{13}x_3 & - & \cdots & - & a_{1,n-1}x_{n-1} & - & a_{1,n}x_n & + & b_1 \\ a_{22}x_2 & = & - & a_{2,1}x_1 & \square & - & a_{23}x_3 & - & \cdots & - & a_{2,n-1}x_{n-1} & - & a_{2,n}x_n & + & b_2 \\ \vdots & & & & & & & & & & & & & & \\ a_{n-1,n-1}x_{n-1} & = & - & a_{n-1,1}x_1 & - & a_{n-1,2}x_2 & - & a_{n-1,3}x_3 & - & \cdots & \square & - & a_{n-1,n}x_n & + & b_{n-1} \\ a_{n,n}x_n & = & - & a_{n,1}x_1 & - & a_{n,2}x_2 & - & a_{n,3}x_3 & - & \cdots & - & a_{n,n-1}x_{n-1} & \square & + & b_n \end{cases}$$

The Jacobi Method: starting with an initial guess $\{x_i^{(0)}\}_{i=1:n}$, construct $\{x_i^{(\kappa)}\}_{\kappa\ge 0}$ by

$$\begin{cases} a_{11}x_1^{(\kappa+1)} & = & \square & - & a_{12}x_2^{(\kappa)} & - & a_{13}x_3^{(\kappa)} & - & \cdots & - & a_{1,n-1}x_{n-1}^{(\kappa)} & - & a_{1,n}x_n^{(\kappa)} & + & b_1 \\ a_{22}x_2^{(\kappa+1)} & = & - & a_{2,1}x_1^{(\kappa)} & \square & - & a_{23}x_3^{(\kappa)} & - & \cdots & - & a_{2,n-1}x_{n-1}^{(\kappa)} & - & a_{2,n}x_n^{(\kappa)} & + & b_2 \\ \vdots & & & & & & & & & & & & & & \\ a_{n-1,n-1}x_{n-1}^{(\kappa+1)} & = & - & a_{n-1,1}x_1^{(\kappa)} & - & a_{n-1,2}x_2^{(\kappa)} & - & a_{n-1,3}x_3^{(\kappa)} & - & \cdots & \square & - & a_{n-1,n}x_n^{(\kappa)} & + & b_{n-1} \\ a_{n,n}x_n^{(\kappa+1)} & = & - & a_{n,1}x_1^{(\kappa)} & - & a_{n,2}x_2^{(\kappa)} & - & a_{n,3}x_3^{(\kappa)} & - & \cdots & - & a_{n,n-1}x_{n-1}^{(\kappa)} & \square & + & b_n \end{cases}$$

or equivalently

$$x_i^{(\kappa+1)} = -\sum_{j=1, j\neq i}^{n} \frac{a_{ij}}{a_{ii}}x_j^{(\kappa)} + \frac{b}{a_{ii}}, \qquad i = 1:n, \qquad \text{(Jacobi method)}$$

$$\boldsymbol{x}^{(\kappa+1)} = D^{-1}(E+F)\boldsymbol{x}^{(k)} + D^{-1}b. \qquad (D\boldsymbol{x}^{(\kappa+1)} = (E+F)\boldsymbol{x}^{(\kappa)} + b)$$

EXAMPLE 2.1. *Let* $A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix}$, $\boldsymbol{b} = \begin{bmatrix} 1 \\ 8 \\ -5 \end{bmatrix}$. *Carry out a number of the Jacobi iterations, starting with the* $\mathbf{0}$ *initial vector. The exact solution is* $\boldsymbol{x} = \begin{bmatrix} 2 \\ 3 \\ -1 \end{bmatrix}$.

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{8}{3} \\ -\frac{5}{2} \end{bmatrix}, \qquad \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(\frac{8}{3}+1) \\ \frac{1}{3}(-2+8) \\ \frac{1}{2}(\frac{8}{3}-5) \end{bmatrix} = \begin{bmatrix} \frac{11}{6} \\ 2 \\ -\frac{7}{6} \end{bmatrix},$$

$$\begin{bmatrix} x_1^{(3)} \\ x_2^{(3)} \\ x_3^{(3)} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(2+1) \\ \frac{1}{3}(\frac{11}{6}-\frac{7}{6}+8) \\ \frac{1}{2}(2-5) \end{bmatrix} = \begin{bmatrix} \frac{3}{2} \\ \frac{26}{9} \\ -\frac{3}{2} \end{bmatrix}.$$

```
>> A = [2 -1 0 ; -1 3 -1 ; 0 -1 2];b=[1;8;-5];
>> [X,xnew,k] = Jacobi(A,b,9,1.e-9)
rJ =
    0.5774
X =
 0.5000    1.8333    1.5000    1.9444    1.8333    1.9815    1.9444    1.9938    1.9815
 2.6667    2.0000    2.8889    2.6667    2.9630    2.8889    2.9877    2.9630    2.9959
-2.5000   -1.1667   -1.5000   -1.0556   -1.1667   -1.0185   -1.0556   -1.0062   -1.0185
xnew =
    1.9815
    2.9959
   -1.0185
k =
     9
```

In order to write the Jacobi method as a linear iterative method ($A\boldsymbol{x} = b$, with $A = P - N$, hence $P\boldsymbol{x} = N\boldsymbol{x} + b$ and the iteration $P\boldsymbol{x}^{(\kappa+1)} = N\boldsymbol{x}^{(\kappa)} + b$, or equivalently $\boldsymbol{x}^{(\kappa+1)} = P^{-1}N\boldsymbol{x}^{(\kappa)} + P^{-1}b$), we define

$$P_J = D, \quad N_J = P_J - A = D - A = E + F,$$

which give (recall the stationary iterative method $\boldsymbol{x}^{(\kappa+1)} = B\boldsymbol{x}^{(\kappa)} + f$)

$$B_J = P_J^{-1}N_J = D^{-1}(E+F) = D^{-1}(D-A) = I - D^{-1}A \qquad \text{(Jacobi iteration matrix)}$$

$$= \begin{bmatrix} 0 & -\frac{a_{12}}{a_{11}} & \cdots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & \cdots & -\frac{a_{2n}}{a_{22}} \\ \vdots & & \cdots & \vdots \\ -\frac{a_{n1}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & \cdots & 0 \end{bmatrix}, \qquad \text{(Jacobi iteration matrix)}$$

$$f_J = D^{-1}b.$$

In order to verify that the Jacobi method ($\boldsymbol{x}^{(\kappa+1)} = B_J\boldsymbol{x}^{(\kappa)} + D^{-1}b$) is a consistent iterative method (i.e, $\boldsymbol{x} = B_J\boldsymbol{x} + D^{-1}b$):

$$\boldsymbol{x} = \underbrace{(I - D^{-1}A)}_{B_J}\boldsymbol{x} + D^{-1}b \quad \Leftrightarrow \quad D^{-1}A\boldsymbol{x} = D^{-1}b.$$

A 'generalization': The **Jacobi Over-Relaxation Method** (JOR)

Let $\omega \in \mathbb{R}$ be a 'relaxation' parameter. Then the new iterate $\boldsymbol{x}_{\text{JOR}}^{(\kappa+1)}$ is defined as an interpolation (weighted average) between the previous iterate $\boldsymbol{x}^{(\kappa)}$ and the Jacobi method iterate $\boldsymbol{x}_{\text{Jacobi}}^{(\kappa+1)}$:

$$\boldsymbol{x}_{\text{JOR}}^{(\kappa+1)} = \omega\,\boldsymbol{x}_{\text{Jacobi}}^{(\kappa+1)} + (1-\omega)\boldsymbol{x}^{(\kappa)}, \tag{JOR}$$

$$x_i^{(\kappa+1)} = \omega\frac{1}{a_{ii}}\Big(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_i^{(\kappa)}\Big) + (1-\omega)x_i^{(\kappa)}, \qquad i = 1:n.$$

The corresponding **iteration matrix** for JOR is then obtained from the Jacobi iteration matrix:

$$B_{\text{JOR}} = \omega B_J + (1-\omega)I = \omega(I - D^{-1}A) + (1-\omega)I = I - \omega D^{-1}A, \tag{JOR iteration matrix}$$

since from (JOR) and (Jacobi iteration matrix) we have

$$\boldsymbol{x}_{\text{JOR}}^{(\kappa+1)} = B_{\text{JOR}}\boldsymbol{x}^{(\kappa)} + f_{\text{JOR}};$$

$$\boldsymbol{x}_{\text{JOR}}^{(\kappa+1)} = \omega\,\boldsymbol{x}_{\text{Jacobi}}^{(\kappa+1)} + (1-\omega)\boldsymbol{x}^{(\kappa)} = \omega\left(B_J\boldsymbol{x}^{(\kappa)} + f_J\right) + (1-\omega)\boldsymbol{x}^{(\kappa)} = \underbrace{\big(\omega B_J + (1-\omega)I\big)}_{B_{\text{JOR}}}\boldsymbol{x}^{(\kappa)} + \underbrace{\omega f_J}_{f_{\text{JOR}}}.$$

Moreover, from (JOR iteration matrix) we have that

$$B_{\text{JOR}} = P_{\text{JOR}}^{-1}N_{\text{JOR}} \equiv= I - \omega D^{-1}A = I - \omega D^{-1}(D - E - F) = (1-\omega)I + \omega D^{-1}(E+F)$$
$$= \omega D^{-1}\Big(\frac{1-\omega}{\omega}D + E + F\Big)$$

hence

$$P_{\text{JOR}} = \frac{1}{\omega}D, \qquad N_{\text{JOR}} = \frac{1-\omega}{\omega}D + E + F.$$

It is easy to verify that (JOR) is a consistent iterative method $\forall\omega$.
Indeed, from (Jacobi iteration matrix) and (JOR iteration matrix) we have

$$\boldsymbol{x} = B_{\text{JOR}}\boldsymbol{x} + f_{\text{JOR}} \equiv (I - \omega D^{-1}A)\boldsymbol{x} + \omega D^{-1}b = \boldsymbol{x} - \cancel{\omega D^{-1}A\boldsymbol{x}} + \cancel{\omega D^{-1}b} = \boldsymbol{x}.$$

REMARK 2.1. *Note that when $\omega = 1$, then JOR $\equiv$ Jacobi method!*

The Gauss-Seidel Method:

$$\begin{cases}
a_{11}x_1^{(\kappa+1)} &= \square &- a_{12}x_2^{(\kappa)} &- a_{13}x_3^{(\kappa)} &- \cdots &- a_{1,n-1}x_{n-1}^{(\kappa)} &- a_{1,n}x_n^{(\kappa)} \\
a_{22}x_2^{(\kappa+1)} &= - a_{2,1}x_1^{(\kappa+1)} &\square &- a_{23}x_3^{(\kappa)} &- \cdots &- a_{2,n-1}x_{n-1}^{(\kappa)} &- a_{2,n}x_n^{(\kappa)} \\
\vdots & & & & & & \\
a_{n-1,n-1}x_{n-1}^{(\kappa+1)} &= - a_{n-1,1}x_1^{(\kappa+1)} &- a_{n-1,2}x_2^{(\kappa+1)} &- a_{n-1,3}x_3^{(\kappa)} &- \cdots &\square &- a_{n-1,n}x_n \\
a_{n,n}x_n^{(\kappa+1)} &= - a_{n,1}x_1^{(\kappa+1)} &- a_{n,2}x_2^{(\kappa+1)} &- a_{n,3}x_3^{(\kappa+1)} &- \cdots &- a_{n,n-1}x_{n-1}^{(\kappa+1)} &\square
\end{cases}$$

or equivalently

$$x_i^{(\kappa+1)} = -\sum_{j=1,(i>j)}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(\kappa+1)} - \sum_{j=i+1,(i<j)}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(\kappa)} + \frac{b}{a_{ii}}, \qquad i = 1:n,$$

(Gauss-Seidel method)

$$\boldsymbol{x}^{(\kappa+1)} = D^{-1}E\boldsymbol{x}^{(\kappa+1)} + D^{-1}F\boldsymbol{x}^{(\kappa)} + D^{-1}b, \qquad (I - D^{-1}E)\boldsymbol{x}^{(\kappa+1)} = D^{-1}F\boldsymbol{x}^{(\kappa)} + D^{-1}b,$$

$$\begin{aligned}\boldsymbol{x}^{(\kappa+1)} &= (I - D^{-1}E)^{-1}D^{-1}F\boldsymbol{x}^{(\kappa)} + (I - D^{-1}E)^{-1}D^{-1}b = \left(D^{-1}(D-E)\right)^{-1}D^{-1}F\boldsymbol{x}^{(\kappa)} + (I - D^{-1}E)^{-1}D^{-1}b \\ &= (D-E)^{-1}F\boldsymbol{x}^{(\kappa)} + (D-E)^{-1}b,\end{aligned}$$

and therefore the **iteration matrix**

$$\boldsymbol{x}_{\mathrm{GS}}^{(\kappa+1)} = B_{\mathrm{GS}}\boldsymbol{x}_{\mathrm{GS}}^{(\kappa)} + f_{\mathrm{GS}}$$

of the Gauss-Seidel method is

$$B_{\mathrm{GS}} = P_{\mathrm{GS}}^{-1}N_{\mathrm{GS}} = (D-E)^{-1}F \equiv (I - D^{-1}E)^{-1}D^{-1}F, \quad \text{(Gauss-Seidel iteration matrix)}$$

$$f_{\mathrm{GS}} = (D-E)^{-1}b, \tag{2.1}$$

and moreover

$$A = P_{\mathrm{GS}} - N_{\mathrm{GS}} = \underbrace{(D-E)}_{P_{\mathrm{GS}}} - \underbrace{F}_{N_{\mathrm{GS}}}$$

$$P_{\mathrm{GS}} = D - E, \qquad N_{\mathrm{GS}} = F.$$

The Gauss-Seidel method

$$\boldsymbol{x}^{(\kappa+1)} = B_{\mathrm{GS}}\boldsymbol{x}^{(\kappa)} + f_{\mathrm{GS}} = (D-E)^{-1}F\boldsymbol{x}^{(\kappa)} + (D-E)^{-1}b$$

is a consistent iterative method since

$$\boldsymbol{x} = (D-E)^{-1}F\boldsymbol{x} + (D-E)^{-1}b \quad \Leftrightarrow \quad (D-E)\boldsymbol{x} = F\boldsymbol{x} + b \quad \Leftrightarrow \quad \underbrace{(D-E-F)}_{A}\boldsymbol{x} = b.$$

EXAMPLE 2.2. *Recall Example 2.1, where* $A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix}$, $\boldsymbol{b} = \begin{bmatrix} 1 \\ 8 \\ -5 \end{bmatrix}$. *Carry out a number of the Gauss-Seidel iterations, starting with the* $\mathbf{0}$ *initial vector. The exact solution is* $\boldsymbol{x} = \begin{bmatrix} 2 \\ 3 \\ -1 \end{bmatrix}$.

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{3}\left(\frac{1}{2}+8\right) \\ \frac{1}{2}\left(\frac{17}{6}-5\right) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{17}{6} \\ -\frac{13}{12} \end{bmatrix}, \qquad \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\left(\frac{17}{6}+1\right) \\ \frac{1}{3}\left(\frac{23}{12}-\frac{13}{12}+8\right) \\ \frac{1}{2}\left(\frac{53}{18}-5\right) \end{bmatrix} = \begin{bmatrix} \frac{23}{12} \\ \frac{53}{18} \\ -\frac{37}{36} \end{bmatrix},$$

$$\begin{bmatrix} x_1^{(3)} \\ x_2^{(3)} \\ x_3^{(3)} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\left(\frac{71}{36}-\frac{37}{36}+8\right) \\ \frac{1}{3}\left(\frac{23}{12}-\frac{13}{12}+8\right) \\ \frac{1}{2}\left(\frac{161}{54}-5\right) \end{bmatrix} = \begin{bmatrix} \frac{71}{36} \\ \frac{161}{54} \\ -\frac{109}{108} \end{bmatrix}.$$

```
>> A = [2 -1 0 ; -1 3 -1 ; 0 -1 2];b=[1;8;-5];
>> [X,k] = Gauss_Seidel(A,b,9,1.e-9)
rGS =
    0.3333
The Gauss-Seidel iterations will converge with tolerance 1.0000e-09
```

```
since the spectral radius of the iteration matrix B_GS is R_GS=0.333333
X =
 0.5000    1.9167    1.9722    1.9907    1.9969    1.9990    1.9997    1.9999    2.0000
 2.8333    2.9444    2.9815    2.9938    2.9979    2.9993    2.9998    2.9999    3.0000
-1.0833   -1.0278   -1.0093   -1.0031   -1.0010   -1.0003   -1.0001   -1.0000   -1.0000
k = 9
```

A 'generalization' of the Gauss-Seidel: The **Successive Over-Relaxation Method** (SOR)
Let $\omega > 0$ be a 'relaxation' parameter.

$$\boldsymbol{x}_{\mathrm{SOR}}^{(\kappa+1)} = \omega \boldsymbol{x}_{\mathrm{GS}}^{(\kappa+1)} + (1-\omega)\boldsymbol{x}^{(\kappa)}$$

$$x_i^{(\kappa+1)} = \omega\Big(- \sum_{j=1,(i>j)}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(\kappa+1)} - \sum_{j=i+1,(i<j)}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(\kappa)} + \frac{b}{a_{ii}} \Big) + (1-\omega)x_i^{\kappa}, \qquad i = 1:n.$$

(SOR method)

The **iteration matrix** of SOR method is

$$B_{\mathrm{SOR}} = (I - \omega D^{-1}E)^{-1}[(1-\omega)I + \omega D^{-1}F] = P_{\mathrm{SOR}}^{-1} N_{\mathrm{SOR}} \qquad \text{(SOR iteration matrix)}$$

$$= \Big(\frac{1}{\omega}D - E\Big)^{-1}\Big(\frac{1-\omega}{\omega}D + F\Big) \equiv (D - \omega E)^{-1}\big((1-\omega)D + \omega F\big),$$

$$P_{\mathrm{SOR}} = \frac{1}{\omega}D - E, \qquad N_{\mathrm{SOR}} = \frac{1-\omega}{\omega}D + F.$$

Note that

$$A = P_{\mathrm{SOR}} - N_{\mathrm{SOR}} = \frac{1}{\omega}D - E - \Big(\frac{1-\omega}{\omega}D + F\Big).$$

2.2. Convergence results for the Jacobi method and the Gauss-Seidel method.
Subtracting the Jacobi method

$$x_i^{(\kappa+1)} = \frac{1}{a_{ii}}\Big(b_i - \sum_{j=1,j\neq i}^{n} a_{ij}x_j^{(\kappa)}\Big), \qquad i = 1:n,$$

from the equation $A\boldsymbol{x} = b$

$$x_i = \frac{1}{a_{ii}}\Big(b_i - \sum_{j=1,j\neq i}^{n} a_{ij}x_j\Big), \qquad i = 1:n,$$

we obtain that the errors satisfy componentwise

$$e_i^{(\kappa+1)} = x_i - x_i^{(\kappa+1)} = -\frac{1}{a_{ii}} \sum_{j=1,j\neq i}^{n} a_{ij}e_j^{(\kappa)} \qquad \text{(errors Jacobi method)}$$

or, in matrix form

$$\boldsymbol{e}^{(\kappa+1)} = B_J \boldsymbol{e}^{(\kappa)} \equiv (I - D^{-1}A)\boldsymbol{e}^{(\kappa)},$$

where B_J is the Jacobi iteration matrix

$$B_J = \begin{bmatrix} 0 & -\frac{a_{12}}{a_{11}} & \cdots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & \cdots & -\frac{a_{2n}}{a_{22}} \\ \vdots & & \cdots & \vdots \\ -\frac{a_{n1}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & \cdots & 0 \end{bmatrix} \qquad \text{(Jacobi iteration matrix)}$$

Similarly to the relation for the errors Jacobi method we obtain the errors Gauss-Seidel method:

$$e_i^{(\kappa+1)} = x_i - x_i^{(\kappa+1)} = -\frac{1}{a_{ii}}\sum_{j=1}^{i-1} a_{ij}e_j^{(\kappa+1)} - \frac{1}{a_{ii}}\sum_{j=i+1}^{n} a_{ij}e_j^{(\kappa)}.$$

(errors Gauss-Seidel method)

Therefore, following from Theorem 1.1 we have the following result.

PROPOSITION 2.1. *The Jacobi method converges* $\forall \boldsymbol{x}^{(0)}$ *if and only if*

$$\rho(B_J) < 1 \quad \Leftrightarrow \quad \boldsymbol{x}^{(\kappa)} \xrightarrow{\kappa\to\infty} \boldsymbol{x} = A^{-1}b.$$

THEOREM 2.1. *If A is strictly* **diagonally dominant**, *then both the Jacobi method and Gauss-Seidel method converge.*

PROOF. • For the Jacobi method:

(1) If A is strictly diagonally dominant by rows, i.e.,

$$|a_{ii}| > \sum_{j=1,j\neq i}^{n} |a_{ij}| \quad \Leftrightarrow \quad \sum_{j=1,j\neq i}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| < 1, \qquad \text{or} \qquad \|B_J\|_\infty < 1 \quad \text{the max (row sum)}$$

(2) If A is strictly diagonally dominant by columns, i.e.,

$$|a_{ii}| > \sum_{j=1,j\neq i}^{n} |a_{ji}| \quad \Leftrightarrow \quad \sum_{j=1,j\neq i}^{n} \left|\frac{a_{ji}}{a_{ii}}\right| < 1, \qquad \text{or} \qquad \|B_J\|_1 < 1 \quad \text{the max (column sum)}$$

• For the Gauss-Seidel method we have that the errors Gauss-Seidel method satisfy

$$\boldsymbol{e}^{(\kappa+1)} = B_{GS}\boldsymbol{e}^{(\kappa)},$$

where B_{GS} is bounded as follows.
Denote

$$\alpha_i = \sum_{j=1}^{i-1}\left|\frac{a_{ij}}{a_{ii}}\right|, \qquad \beta_i = \sum_{j=i+1}^{n}\left|\frac{a_{ij}}{a_{ii}}\right|,$$

with $\alpha_1 = \beta_n = 0$. Then from the (errors Gauss-Seidel method) we see that

$$|e_i^{(\kappa+1)}| \le \alpha_i\|\boldsymbol{e}^{(\kappa+1)}\|_\infty + \beta_i\|\boldsymbol{e}^{(\kappa)}\|_\infty, \qquad \forall i = 1:n,$$

and also denote ℓ the index of the "maximum modulus" component of for error, i.e.,

$$|e_\ell^{(\kappa+1)}| = \|\boldsymbol{e}^{(\kappa+1)}\|_\infty.$$

Therefore, from the above estimate we have that

$$\|e^{(\kappa+1)}\|_\infty \le \frac{\beta_\ell}{1-\alpha_\ell}\|\boldsymbol{e}^{(\kappa)}\|_\infty \le \max_i \frac{\beta_i}{1-\alpha_i}\|\boldsymbol{e}^{(\kappa)}\|_\infty \le \cdots \le \left(\max_i \frac{\beta_i}{1-\alpha_i}\right)^{\kappa+1}\|e^{(0)}\|_\infty.$$

(1) If A is strictly diagonally dominant by rows, then

$$\max_i(\alpha_i+\beta_i) < 1$$

which implies

$$\max_i \frac{\beta_i}{1-\alpha_i} = \max_i\Big(\alpha_i+\beta_i\underbrace{-\frac{\alpha_i(1-\alpha_i-\beta_i)}{1-\alpha_i}}_{<0}\Big) < \underbrace{\max_i(\alpha_i+\beta_i)}_{\|B_J\|_\infty} < 1$$

which gives the (probably faster than Jacobi) convergence of the Gauss-Seidel method

(2) A similar argument holds for the case when A is strictly diagonally dominant by columns,

concluding the proof. □

THEOREM 2.2. *If A and $2D-A$ are SPD matrices, then the Jacobi method is convergent and*

$$\rho(B_J) = \|B_J\|_A = \|B_J\|_D.$$

PROOF. This is a direct consequence of Corollary 1.1, taking $P = D$. □

THEOREM 2.3. *If A is SPD, the JOR is convergent if $0 < \omega < 2/\rho(D^{-1}A)$.*

PROOF. Recall that the (JOR iteration matrix) is $B_{JOR} = I - \omega D^{-1}A$. Then it is easy to see that

$$\rho(B_{JOR}) = \rho(I - \omega D^{-1}A) = \left|1 - \omega\rho(D^{-1}A)\right| < 1 \qquad \Leftrightarrow \quad 0 < \omega\rho(D^{-1}A) < 2,$$

which concludes the argument. □

THEOREM 2.4. *(see [7, pp 512]) If A is SPD, then the Gauss-Seidel method is monotonically convergent in the $\|\cdot\|_A$ norm.*

PROOF. First we note that since A is SPD, then the diagonal matrix $D = \mathtt{diag}(A)$ is also positive definite. Now we use again Corollary 1.1 with $P = D - E$, to obtain that

$$P + P^T - A \equiv D - E + (D-E)^T - (D - E - F) = \cancel{D} - \cancel{E} + D - \underbrace{\cancel{F}}_{E^T} - \cancel{D} + \cancel{E} + \cancel{F} = D = \mathtt{diag}(A)$$

is positive definite, and therefore concluding that the Gauss-Seidel method converges monotonically. (For a different proof, see [7, pp 512].) □

THEOREM 2.5. *If A is* **tridiagonal** *(or block diagonal), it can be shown that*

$$\rho(B_{GS}) = \rho^2(B_J).$$

REMARK 2.2. *When dealing with* **general** *matrices, no a priori conclusions hold on convergence for the Jacobi method or the Gauss-Seidel method.*

EXAMPLE 2.3.

$$A_1 = \begin{bmatrix} 3 & 1.5 & 4 \\ 7 & 4 & 2 \\ -1 & 1 & 2 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -3 & 3 & -6 \\ -4 & 7 & -8 \\ 5 & 7 & -9 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 4 & 1 & 1 \\ 2 & -9 & 0 \\ 0 & -8 & -6 \end{bmatrix}, \quad A_4 = \begin{bmatrix} 7 & 6 & 9 \\ 4 & 5 & -4 \\ -7 & -3 & 8 \end{bmatrix}.$$

Recall that from (Jacobi iteration matrix) *and* (Gauss-Seidel iteration matrix) *we have*

$$A = D - \underbrace{(E+F)}_{N}, \qquad B_J = I - D^{-1}A = P_J^{-1}N_J,$$

$$A = \underbrace{D-E}_{P} - \underbrace{F}_{N}, \qquad B_{GS} = (D-E)^{-1}F = P_{GS}^{-1}N_{GS}.$$

(1) $\rho(B_J) = 1.11635, \rho(B_{GS}) = 0.8546, \rho(B_{SOR}) = 1.4$ *(with* $\omega = 1.1$*).*
(2) $\rho(B_J) = 0.8133, \rho(B_{GS}) = 1.1, \rho(B_{SOR}) = 1.59$
(3) $\rho(B_J) = 0.44$ (slower than) $\rho(B_{GS}) = 0.0185, \rho(B_{SOR}) = 0.168$
(4) $\rho(B_J) = 0.64$ (faster than) $\rho(B_{GS}) = 0.7746, \rho(B_{SOR}) = 0.92$

See in the online folder for the codes for the spectral radii, namely spectralRadiusJacobi.m, spectralRadiusGaussSeidel.m, and spectralRadiusGaussSeidel.m, respectively.
The matrix A_2 yields a convergent sequence using the Jacobi method

```
>> A = [-3 3 -6; -4 7 -8; 5 7 -9];x=ones(length(A),1);b=A*x;
>> [XJ,xJ,k]=Jacobi(A,b,50,1.e-3);
rJ =
   0.813309105469277
Jacobi Convergence at iteration 28
>> xJ
xJ =
   0.999675082273468
   1.000243188578632
   0.999989334136408
```

but not using the Gauss-Seidel method.
The matrix A_3 yields convergent sequences with all three methods:

- *Jacobi, with the null vector as the initial guess $x^{(0)}$, gives*

  $$b = [6; -7; -14]; \qquad x^{(1)} = \left[\frac{3}{2}; \frac{7}{9}; \frac{7}{3}\right]; \qquad x^{(2)} = \left[\frac{13}{12}; \frac{10}{9}; \frac{35}{27}\right];$$

  ```
  >> A=[4 1 1; 2 -9 0; 0 -8 -6];x=ones(3,1);b=A*x;
  >> [XJ,xj,k]=Jacobi(A,b,5,1.e-3)
  rJ =
      0.4438
  XJ =
      1.5000    0.7222    0.8981    1.0525    0.9851
      0.7778    1.1111    0.9383    0.9774    1.0117
      2.3333    1.2963    0.8519    1.0823    1.0302
  xj =
      0.9851
      1.0117
      1.0302
  k =
       5
  ```

  and actually it takes 11 iterations to reach the 10^{-3} tolerance, using the (useful stopping criterion).
- *Gauss Seidel, also with null initial guess gives*

  $$b = [6; -7; -14]; \qquad x^{(1)} = \left[\frac{3}{2}; \frac{10}{9}; \frac{23}{27}\right]; \qquad x^{(2)} = \left[\frac{109}{108}; \frac{487}{486}; \frac{1454}{1458}\right];$$

  and it reaches the 10^{-3} tolerance in 4 iterations, by the (useful stopping criterion).

  ```
  >> [XGS,k]=Gauss_Seidel(A,b,5,1.e-3)
  rGS =
      0.0185
  The Gauss-Seidel iterations will converge with tolerance 0.0010000000000000
  since the spectral radius of the iteration matrix B_GS is R_GS=0.018519
  Gauss-Seidel convergence at iteration k=4
      1.0000
      1.0000
      1.0000
  ```

```
XGS =
    1.5000    1.0093    1.0002    1.0000
    1.1111    1.0021    1.0000    1.0000
    0.8519    0.9973    0.9999    1.0000
k =
     4
```

- *SOR*

```
>> [X] = SOR(A,b,50,1.1,1.e-3)
 The SOR iterations will converge with tolerance 0.0010000000000000
since the spectral radius of the iteration matrix B_sor is R_sor=0.168928
the matrix storing SOR iterates as columns X=
    1.6500    0.9407    0.9931    1.0034    0.9994    1.0000
    1.2589    0.9596    1.0023    1.0006    0.9998    1.0000
    0.7203    1.0872    0.9878    1.0003    1.0003    0.9999
SOR convergence at iteration k=
     6
the actual solution (to four decimal places rounded) is obtained x=
    1.0000
    1.0000
    0.9999
X =
    1.6500    0.9407    0.9931    1.0034    0.9994    1.0000
    1.2589    0.9596    1.0023    1.0006    0.9998    1.0000
    0.7203    1.0872    0.9878    1.0003    1.0003    0.9999
```

2.3. Convergence results for the Relaxation methods.

A necessary condition on ω for the SOR method to converge.

THEOREM 2.6. *$\forall \omega \in \mathbb{R}$ we have*

$$\rho(B_{SOR}(\omega)) \geq |\omega - 1|.$$

Therefore the SOR method fails to converge if $\omega \leq 0$ or $2 \leq \omega$.

PROOF. Recall that the (SOR iteration matrix) is

$$B_{\mathrm{SOR}} = (D - \omega E)^{-1}[(1-\omega)D + \omega F],$$

and also that the characteristic polynomial writes

$$\mathscr{P}_{SOR}(\lambda) := \Pi_{i=1}^{n}(\lambda - \lambda_i) = (-1)^n \det(B_{\mathrm{SOR}} - \lambda I),$$

where $\lambda_i \in \sigma(B_{\mathrm{SOR}})$. Then

$$\left|\Pi_{i=1}^{n}\lambda_i\right| = \det(B_{\mathrm{SOR}}) = \det\big((D-\omega E)^{-1}\big)\det\big((1-\omega)D + \omega F\big) \equiv \Pi\Big(\frac{1}{a_{ii}}\Big)\cdot\Pi\Big((1-\omega)a_{ii}\Big) = (1-\omega)^n.$$

Finally

$$\rho(B_{\mathrm{SOR}}) \geq \left|\Pi_{i=1}^{n}\lambda_i\right|^{1/n} = |1-\omega|,$$

concluding the argument. □

THEOREM 2.7 (Ostrowski-Reich). *If A is SPD, then the SOR method converges (monotonically in the $\|\cdot\|_A$ norm) if and only if $\omega \in (0,2)$.*

PROOF. Recall from the (SOR iteration matrix) that

$$P_{\text{SOR}} = \frac{1}{\omega}D - E, \qquad N_{\text{SOR}} = \frac{1-\omega}{\omega}D + F.$$

Since A is SPD, by Corollary 1.1 it suffices to check that $P_{\text{SOR}}^T + P_{\text{SOR}} - A$ is SPD. Indeed, that is the case, since $E^T = F$

$$P_{\text{SOR}}^T + P_{\text{SOR}} - A = \left(\frac{1}{\omega}D - \cancel{E}\right) + \left(\frac{1}{\omega}D - \cancel{E}\right)^T - (D - \cancel{E} - \cancel{F}) = \frac{2-\omega}{\omega}D,$$

and $D = \texttt{diag}(A)$ is SPD. □

2.4. Sparse matrix computations.

Let $A \in \mathbb{R}^{n\times n}$, $n =$ even number

$$A = \begin{bmatrix} 3 & -1 & & & & 1/2 \\ -1 & 3 & \ddots & & 1/2 & \\ & \ddots & \ddots & -1 & & \\ & & -1 & \ddots & \ddots & \\ & 1/2 & & \ddots & 3 & -1 \\ 1/2 & & & & -1 & 3 \end{bmatrix}, \qquad x = \texttt{ones}(n,1), \qquad b = Ax = \begin{bmatrix} 2.5 \\ 1.5 \\ \vdots \\ -1 \\ -1 \\ \vdots \\ 1.5 \\ 2.5 \end{bmatrix}.$$

See Matlab's function `spdiags.m`.

```
>> sparsesetup(100)
```

The matrix A is sparse (`spy`(A)), with $4n \ll n^2$ non-zero elements.

REMARK 2.3. *When $n = 10^5$, what are the options to solve $Ax = b$?*

- **size**: *treating A as a full matrix $\Rightarrow n^2 = 10^{10}$ entries, which in double precision (8 bytes) would take $8 \times 10^{10} \approx 74.5$ GB. So, depending on the computer, it may be impossible to fit the entire n^2 entries into* `RAM`.
- **time**: *not only memory, but also computational time poses a problem; the number of operations for GEM is $\mathcal{O}(n^3) \approx 10^{15}$.*
  - *on a machine which runs few Ghz (10^9 cycles per second), an upper bound on the number of floating point operations per second is 10^8 (flops/sec) $\Rightarrow$ $10^{15}/10^8 = 10^7$ seconds for* GEM *(1 year $\approx 3 \times 10^7$ seconds).*
  - GEM *is not an* `overnight` *computation!*
  - *on the other hand, one step of an iterative method requires $2 \times 4n \approx 800{,}000 = 8 \times 10^5$ operations.*
  - *hence one could use 100 steps of Jacobi method and still finish with fewer than $100 \times 8 \times 10^5 \approx 10^8$ operations $\approx$ 1 second or less*

```
>> Example2_25JacobiSparse(1000,50)
```

For **example**

n		(Jacobi method)	CG (conjugate gradient)	GE (scaled partial pivoting)
10^2	time	0.1237	0.0049	1.5509
	error	1.12e-6 (44 iter)	7.73-7 (14 iter)	7.28 e-14
10^3	time	0.4049	0.5636	1.4997 e+3
	error	4.0617e-7 (56 iter)	1.9188-7 (15 iter)	9.4230e-13
		3.1402e-16 (166 iter, 0.82 sec)	3.99-15 (33 iter, 3.2 sec)	
10^4	time	180.89	192.23	
	error	4.0617e-7 (56 iter)	6.8376-7 (15 iter)	
		3.1402e-16 (162 iter, 122.05 sec)	8.88-16 (32 iter, 7.49 sec)	

2.5. The Gradient Methods.

Let $A \in \mathbb{R}^{n\times n}$ be a SPD matrix, $A\boldsymbol{x} = b$.

DEFINITION 2.1. *$\phi : \mathbb{R}^n \longrightarrow \mathbb{R}_+$ is the 'energy of the system $A\boldsymbol{x} = b$' and is defined as*

$$\phi(\boldsymbol{y}) = \frac{1}{2}\boldsymbol{y}^T A\boldsymbol{y} - \boldsymbol{y}^T b, \qquad \forall \boldsymbol{y} \in \mathbb{R}^n. \qquad \text{(Quadratic Form)}$$

LEMMA 2.1. *Let A be SPD.*

(i) $\phi(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \underbrace{A\boldsymbol{x}}_{b} - \boldsymbol{x}^T b = \frac{1}{2}\boldsymbol{x}^T b - \boldsymbol{x}^T b = -\frac{1}{2}\boldsymbol{x}^T b.$

(ii) $\nabla\phi(\boldsymbol{y}) = \frac{1}{2}(A^T + A)\boldsymbol{y} - b = A\boldsymbol{y} - b = -r(\boldsymbol{y})$ `[residual]`
$\nabla\phi(\boldsymbol{x}) = A\boldsymbol{x} - b = \boldsymbol{0}.$

(iii) $\phi(\boldsymbol{y}) \geq \phi(\boldsymbol{x}), \quad \forall \boldsymbol{y} \in \mathbb{R}^n$, *since*

$$\begin{aligned}
\phi(\boldsymbol{y}) &= \frac{1}{2}\boldsymbol{y}^T A\boldsymbol{y} - \boldsymbol{y}^T b = \frac{1}{2}\boldsymbol{y}^T A\boldsymbol{y} - \frac{1}{2}b^T\boldsymbol{y} - \frac{1}{2}\boldsymbol{y}^T b + \underbrace{\frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \frac{1}{2}b^T\boldsymbol{x}}_{=0} \\
&= \frac{1}{2}\boldsymbol{y}^T A\boldsymbol{y} - \frac{1}{2}\underbrace{b^T}_{\boldsymbol{x}^T A}\boldsymbol{y} - \frac{1}{2}\boldsymbol{y}^T \underbrace{b}_{A\boldsymbol{x}} + \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \frac{1}{2}b^T\boldsymbol{x} \\
&= \frac{1}{2}(\boldsymbol{y}-\boldsymbol{x})^T A\boldsymbol{y} - \frac{1}{2}(\boldsymbol{y}-\boldsymbol{x})^T A\boldsymbol{x} - \frac{1}{2}b^T\boldsymbol{x} \\
&= \underbrace{\frac{1}{2}(\boldsymbol{y}-\boldsymbol{x})^T A(\boldsymbol{y}-\boldsymbol{x})}_{\geq 0} \underbrace{- \frac{1}{2}b^T\boldsymbol{x}}_{\phi(\boldsymbol{x})} \\
&\geq -\frac{1}{2}b^T\boldsymbol{x} = \phi(\boldsymbol{x}).
\end{aligned}$$

(iv) $\phi(\boldsymbol{y}) - \phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{x}\|_A^2$ *(the $\|\cdot\|_A$ norm or 'energy norm').*

PROPOSITION 2.2. *Finding a minimizer $\boldsymbol{x} \in \mathbb{R}^n$ of ϕ $\quad\Leftrightarrow\quad$ solving A.*

PROOF. "$\Rightarrow$" If $\boldsymbol{x}^*$ is a minimizer, then from (ii) $\nabla\phi(\boldsymbol{x}^*) = 0$, hence the residual $r(\boldsymbol{x}^*) = b - A\boldsymbol{x}^* = 0$, i.e., $\boldsymbol{x}^*$ is a solution.
"$\Leftarrow$" From (iv) we obtain that $\phi(\boldsymbol{x}) \leq \phi(\boldsymbol{y}), \forall y$, and $\nabla\phi(\boldsymbol{x}) = 0$. □

2.5.1. **Gradient descent method.** Given $x^{(0)}$, construct $\{x^{(\kappa)}\}_{\kappa\geq 0}$ by

$$x^{(\kappa+1)} = x^{(\kappa)} + \alpha_\kappa p^{(\kappa)} \qquad \text{(gradient descent method)}$$

where

- α_κ:= length of the step along the direction
- $p^{(\kappa)}$:= descent direction.

DEFINITION 2.2. *A natural choice for the direction is*

- $p^{(\kappa)}$ = *direction of maximum descent along* ϕ *in* $x^{(\kappa)}$, *i.e.,*

$$p^{(\kappa)} := -\nabla\phi(x^{(\kappa)}) \overset{(ii)}{=} r^{(\kappa)} = b - Ax^{(\kappa)},$$

which gives the

$$x^{(\kappa+1)} = x^{(\kappa)} - \alpha_\kappa \nabla\phi(x^{(\kappa)}). \qquad \text{(steepest descent method)}$$

The coefficient α_κ *is computed such that*

$$\begin{aligned} 0 = \nabla\phi(x^{(\kappa+1)}) &\overset{(ii)}{=} Ax^{(\kappa+1)} - b = A\left(x^{(\kappa)} + \alpha_\kappa p^{(\kappa)}\right) - b = Ax^{(\kappa)} + \alpha_\kappa A r^{(\kappa)} - b \\ &= -r^{(\kappa)} + \alpha_\kappa A r^{(\kappa)}, \\ r^{(\kappa)} &= \alpha_\kappa A r^{(\kappa)}, \end{aligned}$$

and therefore

$$\alpha_\kappa = \frac{{r^{(\kappa)}}^T r^{(\kappa)}}{{r^{(\kappa)}}^T A r^{(\kappa)}}.$$

ALGORITHM 2.1 (Steepest (or Gradient) Descent). *Given* $x^{(0)}$,
set $r^{(0)} = b - Ax^{(0)}$,
and for $\kappa = 0, 1, \ldots$ *until "convergence",*
compute

$$\begin{aligned} \alpha_\kappa &= \frac{{r^{(\kappa)}}^T r^{(\kappa)}}{{r^{(\kappa)}}^T A r^{(\kappa)}} \\ x^{(\kappa+1)} &= x^{(\kappa)} + \alpha_\kappa r^{(\kappa)} \\ r^{(\kappa+1)} &= r^{(\kappa)} - \alpha_\kappa A r^{(\kappa)} \end{aligned}$$

PROOF. Indeed, from the definitions of the gradient descent method and the steepest descent method

$$\begin{aligned} &x^{(\kappa+1)} = x^{(\kappa)} + \alpha_\kappa p^{(\kappa)} = x^{(\kappa)} + \alpha_\kappa r^{(\kappa)} = x^{(\kappa)} - \alpha_\kappa \nabla\phi(x^{(\kappa)}) = x^{(\kappa)} + \alpha_\kappa (b - Ax^{(\kappa)}), \\ &Ax^{(\kappa+1)} = Ax^{(\kappa)} + \alpha_\kappa A r^{(\kappa)}, \\ &\underbrace{b - Ax^{(\kappa+1)}}_{r^{(\kappa+1)}} = \underbrace{b - Ax^{(\kappa)}}_{r^{(\kappa)}} - \alpha_\kappa A r^{(\kappa)}, \end{aligned}$$

which is the update relation for the residuals. □

THEOREM 2.8. *Let A be SPD. The* **gradient method (steepest descent)** *is convergent* $\forall x^{(0)}$, *and*

$$\|e^{(\kappa+1)}\|_A \leq \frac{\mathtt{cond}_2(A) - 1}{\mathtt{cond}_2(A) + 1} \|e^{(\kappa)}\|_A, \qquad \kappa = 0, 1, \cdots$$

COROLLARY 2.1. *Let A be SPD.*

$$\|\boldsymbol{x}-\boldsymbol{x}^{(\kappa)}\|_A \leq \left(\frac{\mathtt{cond}_2(A)-1}{\mathtt{cond}_2(A)+1}\right)^{\kappa}\|\boldsymbol{x}-\boldsymbol{x}^{(0)}\|_A, \qquad \kappa = 0,1,\cdots$$

PROPOSITION 2.3. *In order to reach tolerance* `tol` *(for 'convergence'), the steepest descent method needs a minimum of κ iterations, where*

$$\kappa \geq -\frac{\log_{10}(\mathtt{tol})}{2}(\mathtt{cond}_2(A)+1).$$

In particular, in order to gain 1 digit of accuracy ($\mathtt{tol}=10^{-1}$) requires

$$\kappa \geq \frac{\mathtt{cond}_2(A)+1}{2}. \tag{2.2}$$

PROOF. Assuming

$$\left(\frac{\mathtt{cond}_2(A)-1}{\mathtt{cond}_2(A)+1}\right)^{\kappa} \leq \mathtt{tol}$$

then for large conditioning number we have

$$\log_{10}(\mathtt{tol}) \geq \kappa \log_{10}\left(\frac{\mathtt{cond}_2(A)-1}{\mathtt{cond}_2(A)+1}\right) = \kappa \log_{10}\left(1-\frac{2}{\mathtt{cond}_2(A)+1}\right) \approx \kappa\frac{-2}{\mathtt{cond}_2(A)+1},$$

where the approximation is a consequence of a Taylor expansion. □

REMARK 2.4. *In the case of the 1D Poisson equation, A is SPD, and $\lambda_{\max}=\mathscr{O}(1), \lambda_{\min}=\mathscr{O}(h^2)$, hence $\mathtt{cond}_2(A)=\frac{\lambda_{\max}}{\lambda_{\min}}=\mathscr{O}(h^{-2})$.*

EXAMPLE 2.4. *[**13**, p. 193] Let*

$$A=\begin{pmatrix}2&0\\0&50\end{pmatrix}, \quad n=2, \quad \boldsymbol{b}=\begin{pmatrix}2\\0\end{pmatrix}, \quad \boldsymbol{x}=\begin{pmatrix}1\\0\end{pmatrix}, \quad \mathtt{cond}_2(A)=\frac{50}{2}=25.$$

Then the Quadratic Form is

$$\begin{aligned}\phi(\boldsymbol{y}) &= \frac{1}{2}\boldsymbol{y}^T A\boldsymbol{y}-\boldsymbol{y}^T\boldsymbol{b} = \frac{1}{2}(y_1,y_2)\begin{pmatrix}2&0\\0&50\end{pmatrix}\begin{pmatrix}y_1\\y_2\end{pmatrix}-(y_1,y_2)\begin{pmatrix}2\\0\end{pmatrix}\\ &= \frac{1}{2}(y_1,y_2)\begin{pmatrix}2y_1\\50y_2\end{pmatrix}-2y_1 = \frac{1}{2}(2y_1^2+50y_2^2)-2y_1 = y_1^2-2y_1+25y_2^2\\ &= \frac{(y_1-1)^2}{1^2}+\frac{y_2^2}{(1/5)^2}-1,\end{aligned}$$

and the equation $\phi(\boldsymbol{y})=0$ describes an ellipse.
Note from (2.2) *that, in this case, the number of* **gradient descent method** *iterations necessary for gaining a significant digit is*

$$\kappa = \frac{\mathtt{cond}_2(A)+1}{2} = 13.$$

Assume the initial guess is

$$\boldsymbol{x}^{(0)}=\begin{pmatrix}11\\1\end{pmatrix}.$$

Then

$$\boldsymbol{r}^{(0)}=\boldsymbol{b}-A\boldsymbol{x}^{(0)}=\begin{pmatrix}2\\0\end{pmatrix}-\begin{pmatrix}2&0\\0&50\end{pmatrix}\begin{pmatrix}11\\1\end{pmatrix}=\begin{pmatrix}2-22\\-50\end{pmatrix}=\begin{pmatrix}-20\\-50\end{pmatrix},$$

$$\alpha_0 = \frac{\boldsymbol{r}^{(0)^T}\boldsymbol{r}^{(0)}}{\boldsymbol{r}^{(0)^T}A\boldsymbol{r}^{(0)}} = \frac{400+2500}{(-20,-50)\begin{pmatrix}-40\\-2500\end{pmatrix}} = \frac{2900}{800+125000} = \frac{2900}{125800} \approx 0.02305,$$

$$\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} = \alpha_0\boldsymbol{r}^{(0)} = \begin{pmatrix}11\\1\end{pmatrix} + \frac{29}{1258}\begin{pmatrix}-20\\-50\end{pmatrix} = \begin{pmatrix}10.5389\\-0.1526\end{pmatrix},$$

$$\boldsymbol{r}^{(1)} = \boldsymbol{r}^{(0)} - \alpha_0 A\boldsymbol{r}^{(0)} = \begin{pmatrix}-19.0779\\7.6311\end{pmatrix},$$

$$\alpha_1 = \frac{\boldsymbol{r}^{(1)^T}\boldsymbol{r}^{(1)}}{\boldsymbol{r}^{(1)^T}A\boldsymbol{r}^{(1)}} = 0.116,$$

$$\boldsymbol{x}^{(2)} = \boldsymbol{x}^{(1)} + \alpha_1\boldsymbol{r}^{(1)} = \begin{pmatrix}8.3259\\0.73259\end{pmatrix}.$$

The errors in the A norm are:

$$\|\boldsymbol{x}-\boldsymbol{x}^{(0)}\|_A = \left\|\begin{pmatrix}1\\0\end{pmatrix} - \begin{pmatrix}11\\1\end{pmatrix}\right\|_A = \left\|\begin{pmatrix}-10\\-1\end{pmatrix}\right\|_A = \sqrt{(-10,-1)\begin{pmatrix}2&0\\0&50\end{pmatrix}\begin{pmatrix}-10\\-1\end{pmatrix}}$$
$$= \sqrt{(-10,-1)\begin{pmatrix}-20\\-50\end{pmatrix}} = \sqrt{200+50} = 15.8,$$

$$\|\boldsymbol{x}-\boldsymbol{x}^{(1)}\|_A = \left\|\begin{pmatrix}1\\0\end{pmatrix} - \begin{pmatrix}10.5389\\-0.1526\end{pmatrix}\right\|_A = \left\|\begin{pmatrix}-9.5389\\0.1526\end{pmatrix}\right\|_A$$
$$= \sqrt{(-9.5389,0.1526)\begin{pmatrix}2&0\\0&50\end{pmatrix}\begin{pmatrix}-9.5389\\0.1526\end{pmatrix}}$$
$$= \sqrt{(-9.5389,0.1526)\begin{pmatrix}-19.0778\\7.6300\end{pmatrix}} = \sqrt{183.1456} = 13.5331,$$

while the values of the higher index iterations and the corresponding residuals are (using steepestdescent.m):

κ	$x_1^{(\kappa)}$	$x_2^{(\kappa)}$	*residual* $\boldsymbol{r}^{(\kappa)}$
13	*2.4745*	*- 0.0235*	*8.3247*
26	*1.17506*	*0.0175*	*0.4910*
39	*1.0258*	*- 0.00041*	*0.1457*
52	*1.00306*	*0.000306*	*0.00859*
65	*1.00045*	*0.0000072*	*0.00255*
78	*1.000053*	*0.00000053*	*0.0001504*

Also solve this system with the Jacobi and Gauss-Seidel methods.

For another iterative method, used for positive-definite matrices, see the

2.5.2. *Conjugate gradient method.*

ALGORITHM 2.2 (Steepest Descent & Conjugate Gradient (CG)). *Given* $\boldsymbol{x}^{(0)}$, *set* $\boldsymbol{r}^{(0)} = b - A\boldsymbol{x}^{(0)}$, *and also set* $\boldsymbol{p}^{(0)} = \boldsymbol{r}^{(0)}$,

and for $\kappa = 0, 1, \ldots$ until "convergence", compute

$$\alpha_\kappa = \frac{\boldsymbol{p}^{(\kappa)^T}\boldsymbol{r}^{(\kappa)}}{\boldsymbol{r}^{(\kappa)^T}A\boldsymbol{r}^{(\kappa)}}$$

$$\boldsymbol{x}^{(\kappa+1)} = \boldsymbol{x}^{(\kappa)} + \alpha_\kappa \boldsymbol{r}^{(\kappa)} \text{ (i.e., } \nabla\phi(\boldsymbol{x}^{(\kappa+1)}) = 0)$$

$$\boldsymbol{r}^{(\kappa+1)} = \boldsymbol{r}^{(\kappa)} - \alpha_\kappa A\boldsymbol{r}^{(\kappa)}$$

$$\alpha_\kappa = \frac{\boldsymbol{r}^{(\kappa)^T}\boldsymbol{r}^{(\kappa)}}{\boldsymbol{p}^{(\kappa)^T}A\boldsymbol{p}^{(\kappa)}}$$

$$\boldsymbol{x}^{(\kappa+1)} = \boldsymbol{x}^{(\kappa)} + \alpha_\kappa \boldsymbol{p}^{(\kappa)}$$

$$\boldsymbol{r}^{(\kappa+1)} = \boldsymbol{r}^{(\kappa)} - \alpha_\kappa A\boldsymbol{p}^{(\kappa)} \quad (\boldsymbol{r}^{(\kappa+1)} \perp \boldsymbol{p}^{(\kappa)})$$

$$\beta_\kappa = \frac{(A\boldsymbol{p}^{(\kappa)})^T\boldsymbol{r}^{(\kappa+1)}}{(A\boldsymbol{p}^{(\kappa)})^T\boldsymbol{p}^{(\kappa)}}$$

$$\boldsymbol{p}^{(\kappa+1)} = \boldsymbol{r}^{(\kappa+1)} - \beta_\kappa \boldsymbol{p}^{(\kappa)} \quad (\boldsymbol{p}^{(\kappa+1)} \perp_A span\{\boldsymbol{p}^{(\kappa)}, \cdots, \boldsymbol{p}^{(0)}\})$$

REMARK 2.5. *It can be shown that for the Conjugate gradient method*

$$\alpha_\kappa = \frac{\|r^{(\kappa)}\|_2^2}{(p^{(\kappa)})^T A p^{(\kappa)}}, \qquad \beta_\kappa = -\frac{\|r^{(\kappa+1)}\|_2^2}{\|r^{(\kappa)}\|_2^2},$$

PROPOSITION 2.4. *(See Exercise 13 in [15, p.155]) Using the relation*

$$\boldsymbol{r}^{(\kappa+1)} = \boldsymbol{r}^{(\kappa)} - \alpha_\kappa A\boldsymbol{p}^{(\kappa)},$$

the following recursive three-term relation holds for the CG residuals

$$A\boldsymbol{r}^{(\kappa+1)} = -\frac{1}{\alpha_\kappa}\boldsymbol{r}^{(\kappa+1)} + \left(\frac{1}{\alpha_\kappa} - \frac{\beta_{\kappa-1}}{\alpha_{\kappa-1}}\right)r^{(\kappa)} + \frac{\beta_\kappa}{\alpha_{\kappa-1}}r^{(\kappa-1)}.$$

REMARK 2.6. *The new CG residuals $\boldsymbol{r}^{(\kappa+1)}$ are orthogonal to the CG search directions $\boldsymbol{p}^{(\kappa)}$:*

$$\boldsymbol{r}^{(\kappa+1)} \perp \boldsymbol{p}^{(j)}, \qquad \forall j = 0, \cdots, \kappa.$$

The new CG search directions $\boldsymbol{p}^{(\kappa+1)}$ are A-orthogonal to all previous CG search directions $\boldsymbol{p}^{(\kappa)}$:

$$\boldsymbol{p}^{(\kappa+1)} \perp_A \boldsymbol{p}^{(j)}, \qquad \forall j = 0, \cdots, \kappa.$$

THEOREM 2.9. *(See [15, p. 155]) Let A be a symmetric and positive definite matrix. Any method which employs conjugate directions $\boldsymbol{p}^{(\kappa)}$ to solve $Ax = b$ terminates after at most n steps, yielding the exact solution.*

PROOF. Denote $V_\kappa := \text{span}\{\boldsymbol{p}^{(0)}, \cdots, \boldsymbol{p}^{(\kappa)}\}$. Since $\boldsymbol{p}^{(0)} = \boldsymbol{r}^{(0)}$ and $\boldsymbol{p}^{(\kappa+1)} = \boldsymbol{r}^{(\kappa+1)} - \beta_\kappa \boldsymbol{p}^{(\kappa)}$, then the space $V_\kappa := \text{span}\{\boldsymbol{r}^{(0)}, \cdots, \boldsymbol{r}^{(\kappa)}\}$, and therefore by the orthogonality property $\boldsymbol{r}^{(\kappa+1)} \perp \boldsymbol{p}^{(j)}, \quad \forall j = 0, \cdots, \kappa$ we have that $\boldsymbol{r}^{(\kappa+1)} \perp V_k$. This finally gives $\boldsymbol{r}^{(n)} \perp V_{n-1} = \mathbb{R}^n$, and therefore $\boldsymbol{r}^{(n)} = \boldsymbol{0}$, hence $\boldsymbol{x}^{(n)} = \boldsymbol{0}$. □

EXAMPLE 2.5. *(Continuation of Example 2.4)*

$$\boldsymbol{x}^{(0)} = \begin{pmatrix} 11 \\ 1 \end{pmatrix}.$$

$$\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)} = \begin{pmatrix} 2 \\ 0 \end{pmatrix} - \begin{pmatrix} 2 & 0 \\ 0 & 50 \end{pmatrix}\begin{pmatrix} 11 \\ 1 \end{pmatrix} = \begin{pmatrix} 2-22 \\ -50 \end{pmatrix} = \begin{pmatrix} -20 \\ -50 \end{pmatrix},$$

$$\boldsymbol{p}^{(0)} := \boldsymbol{r}^{(0)} = \begin{pmatrix} -20 \\ -50 \end{pmatrix},$$

$$\alpha_0 = \frac{(\boldsymbol{r}^{(0)})^T \boldsymbol{r}^{(0)}}{(\boldsymbol{r}^{(0)})^T A \boldsymbol{r}^{(0)}} = \frac{400+2500}{(-20,-50)\begin{pmatrix} -40 \\ -2500 \end{pmatrix}} = \frac{2900}{800+125000} = \frac{2900}{125800} \approx 0.02305,$$

$$\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + \alpha^{(0)}\boldsymbol{p}^{(0)} = \begin{pmatrix} 11 \\ 1 \end{pmatrix} + \frac{29}{1258}\cdot\begin{pmatrix} -20 \\ -50 \end{pmatrix} = \begin{pmatrix} 11-20*29/1258 \\ 1-50*29/1258 \end{pmatrix} = \begin{pmatrix} 6629/629 \\ -96/629 \end{pmatrix},$$

$$\boldsymbol{r}^{(1)} = \boldsymbol{r}^{(0)} - \alpha^{(0)} A\boldsymbol{p}^{(0)} = \begin{pmatrix} -20 \\ -50 \end{pmatrix} - \frac{29}{1258}\cdot\begin{pmatrix} 2 & 0 \\ 0 & 50 \end{pmatrix}\begin{pmatrix} -20 \\ -50 \end{pmatrix}$$

$$= \begin{pmatrix} -20 \\ -50 \end{pmatrix} + \begin{pmatrix} 40\cdot\frac{29}{1258} \\ 2500\cdot\frac{29}{1258} \end{pmatrix} = \begin{pmatrix} -20 \\ -50 \end{pmatrix} + \begin{pmatrix} \frac{580}{629} \\ \frac{36250}{629} \end{pmatrix} = \begin{pmatrix} -\frac{12000}{629} \\ \frac{4800}{629} \end{pmatrix},$$

$$\beta_0 = \frac{(A\boldsymbol{p}^{(0)})^T \boldsymbol{r}^{(1)}}{(A\boldsymbol{p}^{(0)})^T \boldsymbol{p}^{(0)}} = \frac{(-40,-2500)\begin{pmatrix} -\frac{12000}{629} \\ \frac{4800}{629} \end{pmatrix}}{(-40,-2500)\begin{pmatrix} -20 \\ -50 \end{pmatrix}} = \frac{(480000-2500\cdot 4800)/629}{800+125000} = \frac{-11520000}{629*125800} = -\frac{57600}{395641},$$

$$\boldsymbol{p}^{(1)} = \boldsymbol{r}^{(1)} - \beta_0 \boldsymbol{p}^{(0)} = \begin{pmatrix} -\frac{12000}{629} \\ \frac{4800}{629} \end{pmatrix} + \frac{57600}{395641}\begin{pmatrix} -20 \\ -50 \end{pmatrix} = \begin{pmatrix} -\frac{12000}{629} \\ \frac{4800}{629} \end{pmatrix} - \begin{pmatrix} \frac{11552000}{395641} \\ \frac{19782050}{395641} \end{pmatrix} = \begin{pmatrix} -\frac{1450*6000}{629^2} \\ \frac{139200}{629^2} \end{pmatrix}$$

$$\alpha_1 = \frac{(\boldsymbol{p}^{(1)})^T \boldsymbol{r}^{(1)}}{(\boldsymbol{p}^{(1)})^T A\boldsymbol{p}^{(1)}} = \frac{\frac{8700000}{395641}\cdot\frac{12000}{629} + \frac{139200}{395641}\cdot\frac{4800}{629}}{2*\frac{8700000^2}{395641^2} + 50*\frac{139200^2}{395641^2}} = \frac{\frac{8700000}{395641}\cdot\frac{12000}{629} + \frac{139200}{395641}\cdot\frac{4800}{629}}{152348832000000/629^4} = \frac{629}{1450},$$

$$\boldsymbol{x}^{(2)} = \boldsymbol{x}^{(1)} + \alpha^{(1)}\boldsymbol{p}^{(1)} = \begin{pmatrix} 6629/629 \\ -96/629 \end{pmatrix} + \frac{629}{1450}\begin{pmatrix} -\frac{1450*6000}{629^2} \\ \frac{1450*96}{629^2} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Convergence of the CG method

The Conjugate Gradient method was proposed by Magnus Rudolph Hestenes and Eduard Stiefel in 1952 as a **direct method** for $A\boldsymbol{x} = \boldsymbol{b}$ (in exact arithmetic, the exact solution is obtained in n steps or less, where $A \in \mathbb{R}^{n\times n}$).

THEOREM 2.10. *(Theorem 4.11 in [**15**, p. 155]) Let A be a* **symmetric positive definite** *(SPD) matrix. Then* **CG** *for $A\boldsymbol{x} = \boldsymbol{b}$ terminates* **after at most n steps**, *with the exact solution.*

We note that the 'termination property' after n steps of the CG method is valid only in exact arithmetic, due to cumulating rounding errors which prevent the search directions $\boldsymbol{p}^{(\kappa)}$ from being A-conjugate.

PROOF. The directions $\boldsymbol{p}^{(0)}, \boldsymbol{p}^{(1)}, \cdots, \boldsymbol{p}^{(n-1)}$ form an A-orthogonal basis in $\mathbb{R}^n$. Since

$$(\boldsymbol{p}^{(j)})^T \boldsymbol{r}^{(k)} = 0 \quad \forall j = 0, 1, \cdots k-1 \quad \Rightarrow \quad \boldsymbol{r}^{(k)} \perp V_{k-1} = \text{span}\{\boldsymbol{p}^{(0)}, \boldsymbol{p}^{(1)}, \cdots, \boldsymbol{p}^{(k-1)}\},$$

hence

$$\boldsymbol{r}^{(n-1)} \perp V_{n-1},$$

i.e.,

$$\boldsymbol{r}^{(n)} = \boldsymbol{0} = \boldsymbol{b} - A\boldsymbol{x}^{(n)},$$

so $\boldsymbol{x}^{(n)} = \boldsymbol{x}$. □

REMARK 2.7. *The computational cost for GE is $\mathcal{O}(\frac{2}{3}n^3)$, hence CG is more expensive!! (CG: every iterate costs n^2 operations.)*

THEOREM 2.11. *The error $\boldsymbol{e}^{(k)} = \boldsymbol{x} - \boldsymbol{x}^{(k)}$ in CG, for A SPD, in the k-th iteration satisfies*

$$\|\boldsymbol{e}^{(k)}\|_A \leq \frac{2c^k}{1+c^{2k}}\|\boldsymbol{e}^{(0)}\|_A,$$

with

$$c = \frac{\sqrt{\text{cond}_2(A)}-1}{\sqrt{\text{cond}_2(A)}+1} \approx 1 - \frac{2}{\sqrt{\text{cond}_2(A)}+1}.$$

PROOF. See Theorem 4.12 in [**15**, p. 155]. □

As a direct consequence of Theorem 2.11, we have the following result for the CG (algorithm 2.2) as a counterpart for Proposition 2.3 corresponding to the gradient descent method.

PROPOSITION 2.5. *In order to reach tolerance* `tol` *(for 'convergence'), the Conjugate Algorithm 2.2 needs a minimum of κ iterations, where*

$$\kappa \geq -\frac{\log_{10}(\texttt{tol})}{2}(\sqrt{\texttt{cond}_2(A)}+1).$$

In particular, in order to gain 1 digit of accuracy ($\texttt{tol} = 10^{-1}$) requires

$$\kappa \geq \frac{\sqrt{\texttt{cond}_2(A)}+1}{2}. \tag{2.3}$$

PROOF. Assuming

$$c^\kappa \equiv \left(\frac{\sqrt{\texttt{cond}_2(A)}-1}{\sqrt{\texttt{cond}_2(A)}+1}\right)^\kappa \leq \texttt{tol}$$

then for large conditioning number we have

$$\begin{aligned}\log_{10}(\texttt{tol}) &\geq \kappa \log_{10}\left(\frac{\sqrt{\texttt{cond}_2(A)}-1}{\sqrt{\texttt{cond}_2(A)}+1}\right) \approx \kappa \log_{10}\left(1 - \frac{2}{\sqrt{\texttt{cond}_2(A)}+1}\right)\\ &\approx \kappa \frac{-2}{\sqrt{\texttt{cond}_2(A)}+1},\end{aligned}$$

where the approximation is a consequence of a Taylor expansion. □

REMARK 2.8.

- *CG requires $\mathcal{O}\left(\sqrt{\texttt{cond}_2(A)}/2\right)$ per significant digit,*
  significantly *less than the steepest descent method $\mathcal{O}\left(\texttt{cond}_2(A)/2\right)$.*
- *CG requires* **barely** *more work than the steepest descent method at |bf each step.*

EXAMPLE 2.6. *For a 2D Poisson or Helmholtz equation*

$$\Delta u(x,y) + c(x,y)u(x,y) = 0, \qquad \text{where } c(x,y) > 0, \text{on } \Omega = [0,1]\times[0,1]$$

we have for a 100×100 mesh (i.e., $h = 10^{-2}$), the resulting linear system to solve $A\boldsymbol{u} = \boldsymbol{f}$ has dimension $n = 10^4$, and $\texttt{cond}_2(A) = \mathcal{O}(h^{-2}) = \mathcal{O}(10^4)$.
Therefore one expects:

- *Steepest descent: 30,000 iterations for 6 significant digits:*

$$\kappa \geq \frac{\texttt{cond}_2(A)+1}{2} * 6 \approx 5{,}000 * 6 = 30{,}000 \text{ iterations}.$$

- *CG:*

– *the exact solution is obtained in* $n = 10,000$ *iterations*
– *for 6 significant digits:*

$$\kappa \geq 6 * \frac{\sqrt{\texttt{cond}_2(A)}+1}{2} \approx 6 * \frac{100+1}{2} \approx 300 \text{ iterations}.$$

REMARK 2.9. *Overview of iterative methods (see [17, p. 243]): here is a list of what was considered at the time "very large" computations for dense, direct matrix computations:*

- *1950:* $n = 20$ *(Wilkinson)*
- *1965:* $n = 200$ *(Forsythe & Moler)*
- *1980:* $n = 2,000$ *(LINPACK)*
- *1995:* $n = 20,000$ *(LAPACK)*

2.6. Exercises.

Exercise 1. *Give an alternative solution to Example 2.4.*
Solution:

Exercise 2. *Write the matrix formula for the Gauss-Seidel overrelaxation method.*
Solution:

Exercise 3. (Multiple choice) *In solving a system of equations* $Ax = b$*,it is often convenient to use an iterative method, which generates a sequence of* $x^{(k)}$ *vectors that should converge to a solution. The process is stopped when sufficient accuracy has been attained. A general procedure is to obtain* $x^{(k)}$ *by solving* $Qx^{(k)} = (Q-A)x^{(k-1)} + b$*. Here,* Q *is a certain matrix that is usually connected somehow to* A*. The process is repeated, starting with any available guess,* $x^{(0)}$*. What hypothesis guarantees that the method works, no matter what starting point is selected?*

(a) $\|Q\| < 1$
(b) $\|QA\| < 1$
(c) $\|I - QA\| < 1$
(d) $\|I - Q^{-1}A\| < 1$
(e) None of these.

(Hint: The spectral radius is less than or equal to the norm.)
Solution: Apply Theorem 1.1, where $B = Q^{-1}(Q-A)$.

Exercise 4. (Multiple choice) *From a vector norm, we can create a subordinate* (consistent matrix norm)*.Which relation is satisfied by every subordinate matrix norm?*

(a) $\|Ax\| \geq \|A\|\|x\|$
(b) $\|I\| = 1$
(c) $\|AB\| \geq \|A\|\|B\|$
(d) $\|A+B\| \geq \|A\| + \|B\|$
(e) None of these.

Solution: See the definition of a (consistent matrix norm).

Exercise 5. (Multiple choice) *The condition for diagonal dominance of a matrix A is:*

(a) $|a_{ii}| < \sum_{j=1, j\neq i}^{n} |a_{ij}|$
(b) $|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|$
(c) $|a_{ii}| < \sum_{j=1}^{n} |a_{ij}|$
(d) $|a_{ii}| > \sum_{j=1}^{n} |a_{ij}|$
(e) None of these.

Solution: See the definitions of (diagonally dominant by rows) and (diagonally dominant by columns).

Exercise 6. (Multiple choice) *A necessary and sufficient condition for the standard iteration formula* $x^{(k)} = \mathcal{G}x^{(k-1)} + h$ *to produce a sequence* $x^{(k)}$ *that converges to a solution of the equation* $(I - \mathcal{G})x = h$ *is that:*

(a) The spectral radius of $\mathcal{G}$ *is greater than 1.*
(b) The matrix $\mathcal{G}$ *is diagonally dominant.*
(c) The spectral radius of $\mathcal{G}$ *is less than 1.*
(d) $\mathcal{G}$ *is nonsingular.*
(e) None of these.

Solution: See Theorem 1.1.

Exercise 7. (Multiple choice) *A sufficient condition for the Jacobi method to converge for the linear system* $Ax = b$.

(a) $A - I$ *is diagonally dominant.*
(b) A is diagonally dominant.
(c) $\mathcal{G}$ *is nonsingular.*
(d) The spectral radius of $\mathcal{G}$ *is less than 1.*
(e) None of these.

Solution: See Theorem 2.1.

Exercise 8. (Multiple choice) *A sufficient condition for the Gauss-Seidel method to work on the linear system* $Ax = b$.

(a) A is diagonally dominant.
(b) $A - I$ *is diagonally dominant.*
(c) The spectral radius of A is less than 1.
(d) $\mathcal{G}$ *is nonsingular.*
(e) None of these.

Solution: See Theorem 2.1.

Exercise 9. (Multiple choice) *Necessary and sufficient conditions for the SOR method, where* $0 < \omega < 2$, *to work on the linear system* $Ax = b$.

(a) A is diagonally dominant.
(b) $\rho(A) < 1$.

(c) A is symmetric positive definite.
(d) $x^{(0)} = 0$.
(e) None of these.

Solution: See the Theorem 2.7.

Exercise 10. (Multiple choice) *The Frobenius norm is frequently used because it is so easy to compute. Find the value of this norm for these matrices:*

(a) $\begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 4 \\ 2 & 1 & 3 \end{bmatrix}$

(b) $\begin{bmatrix} 0 & 0 & 1 & 2 \\ 3 & 0 & 5 & 4 \\ 1 & 1 & 1 & 2 \\ 1 & 3 & 2 & 2 \end{bmatrix}$

(c) $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 3 & 0 & 5 & 4 \\ 1 & 1 & 1 & 2 \\ 1 & 3 & 2 & 2 \end{bmatrix}$

Solution:

(a) Let $A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 4 \\ 2 & 1 & 3 \end{bmatrix}$, then

$$\|A\|_F = \sqrt{\sum_{i,j=1}^{3} |a_{i,j}|^2} = \sqrt{69} \approx 8.3066.$$

```
>> A = [1 2 3; 0 5 4; 2 1 3];
>> norm(A,"fro")

ans =

   8.306623862918075
```

(b) $\begin{bmatrix} 0 & 0 & 1 & 2 \\ 3 & 0 & 5 & 4 \\ 1 & 1 & 1 & 2 \\ 1 & 3 & 2 & 2 \end{bmatrix}$

(c) $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 3 & 0 & 5 & 4 \\ 1 & 1 & 1 & 2 \\ 1 & 3 & 2 & 2 \end{bmatrix}$

Exercise 11. *Determine the condition numbers* $\texttt{cond}(A)$ *of these matrices:*

(a) $\begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}$

(b) $\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

(c) $\begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & -2 \end{bmatrix}$

(d) $\begin{bmatrix} -2 & -1 & 2 & -1 \\ 1 & 2 & 1 & -2 \\ 2 & -1 & 2 & 1 \\ 0 & 2 & 0 & 1 \end{bmatrix}$

Solution:

Computer problem 1. *Redo several or all of Examples 1-5 using the linear system involving one of the following coefficient matrix and right-hand side vector pairs:*

(a) $A = \begin{bmatrix} 5 & -1 \\ -1 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} 7 \\ 4 \end{bmatrix}$

(b) $A = \begin{bmatrix} 5 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 7 \\ 4 \\ 5 \end{bmatrix}$

(c) $A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 6 & -2 \\ 4 & -3 & 8 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 3 \\ 9 \end{bmatrix}$

(d) $A = \begin{bmatrix} 7 & 3 & -1 \\ 3 & 8 & 1 \\ -1 & 1 & 4 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ -4 \\ 2 \end{bmatrix}$

Solution:

Computer problem 2. *Using the Jacobi, Gauss-Seidel, and SOR* $(\omega = 1.1)$ *iterative methods, write and execute a computer program to solve the following linear system to four decimal places (rounded) of accuracy:*

$$\begin{bmatrix} 7 & 1 & -1 & 2 \\ 1 & 8 & 0 & -2 \\ -1 & 0 & 4 & -1 \\ 2 & -2 & -1 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 3 \\ -5 \\ 4 \\ -3 \end{bmatrix}$$

Compare the number of iterations needed in each case.
(`Hint`*: The exact solution is* $x = (1,-1,1,-1)^T$*.)*
Solution: Exercise8 2 2 JGSSOR(k)

Computer problem 3. *Using the Jacobi, Gauss-Seidel, and the SOR* $(\omega = 1.4)$ *iterative methods, write and run code to solve the following linear system to four decimal places of accuracy:*

$$\begin{bmatrix} 7 & 3 & 1 & 2 \\ 3 & 8 & 1 & -4 \\ -1 & 1 & 4 & -1 \\ 2 & -4 & -1 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ -3 \\ 1 \end{bmatrix}$$

Compare the number of iterations in each case.
(`Hint`: The exact solution is $x = (-1,1,-1,1)^T$.)
Solution: Exercise8 2 3 JGSSOR(k)

Computer problem 4. *(Continuation) Solve the system using the SOR iterative method with values of $\omega = 1(0.1)2$. Plot the number of iterations for convergence versus the values of ω. Which value of ω results in the fastest convergence?*
Solution: Exercise8 2 4SOR(k)

Computer problem 5. *Program and run the Jacobi, Gauss-Seidel, and SOR methods for the system of Example 1*

(a) using equations involving the splitting matrix Q.
(b) using the equation formulations in Example 4.
(c) using the pseudocode involving matrix-vector multiplication.

Solution:

Computer problem 6. *(Continuation) Select one or more of the systems in Computer Problem 1, and rerun these programs.*
Solution:

Computer problem 7. *Consider the linear system*

$$\begin{bmatrix} 9 & -3 \\ -2 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ -4 \end{bmatrix}$$

Using Maple or Matlab, compare solving it by using the Jacobi method and the Gauss-Seidel method starting with $\boldsymbol{x}^{(0)} = (0,0)^T$.
Solution:

Computer problem 8. *(Continuation)*

(1) Change the $(1,1)$ entry from 9 to 1so that the coefficient matrix is no longer diagonally dominant and see whether the Gauss-Seidel method still works. Explain why or why not.
(2) Then change the $(2,2)$ entry from 8 to 1 as well and test. Again explain the results.

Solution:

Computer problem 9. *Use the conjugate gradient method to solve this linear system:*

$$\begin{bmatrix} 2.0 & -0.3 & -0.2 \\ -0.3 & 2.0 & -0.1 \\ -0.2 & -0.1 & 2.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 5 \\ 3 \end{bmatrix}$$

Using Maple or Matlab, compare solving it by using the Jacobi method and the Gauss-Seidel method starting with $\boldsymbol{x}^{(0)} = (0,0)^T$.
Solution:

Computer problem 10. (Euler-Bernoulli beam) *A simple model for a bending beam under stress involves the Euler-Bernoulli differential equation. A finite difference discretization converts it into a system of linear equations. As the size of the discretization decreases, the linear system becomes larger and more ill-conditioned.*

(1) For a beam pinned at both ends, we obtain the following banded system of linear equations with a bandwidth of five:

$$\begin{bmatrix} 12 & -6 & 4/3 & & & & & & \\ -4 & 6 & -4 & 1 & & & & & \\ 1 & -4 & 6 & -4 & 1 & & & & \\ & 1 & -4 & 6 & -4 & 1 & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & 1 & -4 & 6 & -4 & 1 & \\ & & & & 1 & -4 & 6 & -4 & 1 \\ & & & & & 1 & -4 & 6 & -4 \\ & & & & & & 4/3 & 6 & -12 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_{n-3} \\ y_{n-2} \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \\ b_{n-3} \\ b_{n-2} \\ b_{n-1} \\ b_n \end{bmatrix}$$

The right-hand side represents forces on the beam. Set the right-hand side so that there is a known solution, such as a sag in the middle of the beam. Using an iterative method, repeatedly solve the system by allowing n to increase. Does the error in the solution increase when n increases? Use mathematical software that computes the condition number of the coefficient matrix to explain what is happening.

(2) The linear system of equations for a **cantilever beam** *with a free boundary condition at only one end is*

$$\begin{bmatrix} 12 & -6 & 4/3 & & & & & & \\ -4 & 6 & -4 & 1 & & & & & \\ 1 & -4 & 6 & -4 & 1 & & & & \\ & 1 & -4 & 6 & -4 & 1 & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & 1 & -4 & 6 & -4 & 1 & \\ & & & & 1 & -4 & 6 & -4 & 1 \\ & & & & & 1 & -93/25 & 111/25 & -43/25 \\ & & & & & & 12/25 & 24/25 & 12/25 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_{n-3} \\ y_{n-2} \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \\ b_{n-3} \\ b_{n-2} \\ b_{n-1} \\ b_n \end{bmatrix}$$

Repeat the numerical experiment for this system.

Solution:

(a)
```
>> n=8; A = diag(ones(n,1)*6)+ diag(ones(n-1,1)*(-4),-1) ...
   + diag(ones(n-1,1)*(-4),1)+ diag(ones(n-2,1),-2)+ diag(ones(n-2,1),2);
   >> A(1,1)=12; A(1,2)=-6; A(1,3)=4/3; A(n,n-2)=4/3;A(n,n-1)=6;A(n,n)=-12;
   >> A
   A =
      12.0000   -6.0000    1.3333         0         0         0         0         0
      -4.0000    6.0000   -4.0000    1.0000         0         0         0         0
       1.0000   -4.0000    6.0000   -4.0000    1.0000         0         0         0
            0    1.0000   -4.0000    6.0000   -4.0000    1.0000         0         0
            0         0    1.0000   -4.0000    6.0000   -4.0000    1.0000         0
            0         0         0    1.0000   -4.0000    6.0000   -4.0000    1.0000
            0         0         0         0    1.0000   -4.0000    6.0000   -4.0000
```

```
            0         0         0         0         0    1.3333    6.0000  -12.0000
```

(b)
```
>> n=8; A = diag(ones(n,1)*6)+ diag(ones(n-1,1)*(-4),-1) ...
+ diag(ones(n-1,1)*(-4),1)+ diag(ones(n-2,1),-2)+ diag(ones(n-2,1),2);
>> A(1,1)=12;A(1,2)=-6;A(1,3)=4/3;
>> A(n-1,n-2)=-93/25;A(n-1,n-1)=111/25;A(n-1,n)=-43/25;
>> A(n,n-2)=12/25;A(n,n-1)=24/25;A(n,n)=12/25;
>> A
A =
   12.0000   -6.0000    1.3333         0         0         0         0         0
   -4.0000    6.0000   -4.0000    1.0000         0         0         0         0
    1.0000   -4.0000    6.0000   -4.0000    1.0000         0         0         0
         0    1.0000   -4.0000    6.0000   -4.0000    1.0000         0         0
         0         0    1.0000   -4.0000    6.0000   -4.0000    1.0000         0
         0         0         0    1.0000   -4.0000    6.0000   -4.0000    1.0000
         0         0         0         0    1.0000   -3.7200    4.4400   -1.7200
         0         0         0         0         0    0.4800    0.9600    0.4800
```

CHAPTER 4

EIGENVALUES AND EIGENVECTORS

Let $A \in \mathbb{C}^{n\times n}$ be a square matrix. Recall the Definition 1.13:
The number $\lambda \in \mathbb{C}$ is called an **eigenvalue** of A if there exists a non-null vector $\mathbf{0} \neq \boldsymbol{x} \in \mathbb{C}^n$ such that

$$A\boldsymbol{x} = \lambda \boldsymbol{x}.$$

The vector $\boldsymbol{x} \in \mathbb{C}^n$ is the **eigenvector** associated with the eigenvalue λ.
$(\lambda, \boldsymbol{x}) \in \mathbb{C}^{n+1}$ is called an eigenpair of A.
The set of eigenvalues of A is called the **spectrum** of A, denoted $\sigma(A)$

$$\sigma(A) = \{\lambda \in \mathbb{C}; \quad \lambda \text{ is an eigenvalue of } A\} \subset \mathbb{C}. \qquad \text{(spectrum)}$$

From Remark 1.5 we see that

- the eigenvalue λ corresponding to the eigenvector $\boldsymbol{x}$ can be computed by the Rayleigh quotient

$$\lambda = \frac{\boldsymbol{x}^H A \boldsymbol{x}}{\boldsymbol{x}^H \boldsymbol{x}}. \qquad \text{(Rayleigh quotient)}$$

- The eigenvalue λ is the the solution of the characteristic equation

$$p_A(\lambda) = 0, \qquad \text{(characteristic equation)}$$

where

$$p_A(r) := \det(A - rI) \qquad \text{(characteristic polynomial)}$$

is the characteristic polynomial, hence having n roots (complex numbers), i.e., the matrix $A \in \mathscr{M}(n,n)$ has n eigenvalues, not necessarily distinct!

The methods for finding the eigenvalues are generally thought of being either

- **partial methods:** compute **extremal** eigenvalues of A (with max and min absolute value), for example the *power method* (Section 4) and inverse power method, QR and SVD
- **global methods:** approximate the whole spectrum $\sigma(A)$

MATLAB:

```
[V,D] = eig(A)
```

returns diagonal matrix D of eigenvalues and matrix V whose columns are the corresponding right eigenvectors, so that $A * V = V * D$.

REMARK 0.1. *If $Q \in \mathbb{R}^{n\times n}$ is an orthogonal or $Q \in \mathbb{C}^{n\times n}$ is an unitary matrix, then all the eigenvalues are situated on the unit circle, i.e.,*

$$|\lambda| = 1, \qquad \forall \lambda \in \sigma(Q).$$

PROOF.

$$Q\boldsymbol{x} = \lambda \boldsymbol{x} \Rightarrow \boldsymbol{x}^T Q^T Q \boldsymbol{x} = \lambda^2 |\boldsymbol{x}|^2 \Leftrightarrow |\boldsymbol{x}|^2 = \lambda^2 |\boldsymbol{x}|^2,$$

since $Q^T Q = I$. □

THEOREM 0.1.

(a) $\lambda \in \sigma(A) \Rightarrow \lambda^n \in \sigma(A^n)$, *and, if A is nonsingular, then* $\frac{1}{\lambda} \in \sigma(A^{-1})$.
(b) If A is
- **real** *and symmetric* ($A = A^T$), *then* $\sigma(A) \subset \mathbb{R}$.
- **real** *and skew-symmetric* ($A = -A^T$), *then* $\sigma(A) \subset i\mathbb{R}$.

(c) If A is complex and hermitian, then $\sigma(A) \subset \mathbb{R}$.
(d) If P is nonsingular, then A and $P^{-1}AP$ *are (similar and) have the same characteristic polynomial, hence spectrum.*
(e) If A is a diagonal matrix, a lower triangular or an upper triangular matrix, then

$$\sigma(A) = \text{diag}(A).$$

PROOF.

(a)

$$\lambda \boldsymbol{v} = A\boldsymbol{v} \quad \Rightarrow \quad \lambda^n \boldsymbol{v} = A^n \boldsymbol{v};$$

$$(A^{-1}) \cdot |\ \lambda \boldsymbol{v} = A\boldsymbol{v} \quad \Rightarrow \quad \lambda A^{-1}\boldsymbol{v} = \boldsymbol{v}, \quad A^{-1}\boldsymbol{v} = \frac{1}{\lambda}\boldsymbol{v}.$$

(b)
- $A\boldsymbol{x} = \lambda \boldsymbol{x} \Rightarrow \overline{A}\overline{\boldsymbol{x}} = \overline{\lambda}\overline{\boldsymbol{x}} \Leftrightarrow \overline{\boldsymbol{x}}^T A = \overline{\boldsymbol{x}}^T \overline{\lambda} \Rightarrow \overline{\boldsymbol{x}}^T A\boldsymbol{x} = \overline{\lambda}\overline{\boldsymbol{x}}^T \boldsymbol{x}$
  $\Rightarrow \overline{\boldsymbol{x}}^T A\boldsymbol{x} = \lambda \overline{\boldsymbol{x}}^T \boldsymbol{x}$
  hence $\lambda = \overline{\lambda}$;
- $A\boldsymbol{x} = \lambda \boldsymbol{x} \Rightarrow \overline{A}\overline{\boldsymbol{x}} = \overline{\lambda}\overline{\boldsymbol{x}} \Leftrightarrow -\overline{\boldsymbol{x}}^T A = \overline{\boldsymbol{x}}^T \overline{\lambda} \Rightarrow \overline{\boldsymbol{x}}^T A\boldsymbol{x} = -\overline{\lambda}\overline{\boldsymbol{x}}^T \boldsymbol{x}$
  $\Rightarrow \overline{\boldsymbol{x}}^T A\boldsymbol{x} = \lambda \overline{\boldsymbol{x}}^T \boldsymbol{x}$
  hence $\lambda = -\overline{\lambda}$.

(c) $A\boldsymbol{x} = \lambda \boldsymbol{x} \Rightarrow \overline{A}\overline{\boldsymbol{x}} = \overline{\lambda}\overline{\boldsymbol{x}} \Leftrightarrow \overline{\boldsymbol{x}}^T A = \overline{\boldsymbol{x}}^T \overline{\lambda} \Rightarrow \overline{\boldsymbol{x}}^T A\boldsymbol{x} = \overline{\lambda}\overline{\boldsymbol{x}}^T \boldsymbol{x}$
$\Rightarrow \overline{\boldsymbol{x}}^T A\boldsymbol{x} = \lambda \overline{\boldsymbol{x}}^T \boldsymbol{x}$
hence $\lambda = \overline{\lambda}$;

(d)

$$\begin{aligned} \det(P^{-1}AP - \lambda I) &= \det(P^{-1}(A - \lambda I)P) = \det(P^{-1}) \det(A - \lambda I) \det(P) \\ &= \det(A - \lambda I). \end{aligned}$$

(e) $\det(A - \lambda I) = \prod i = 1^n (a_{ii} - \lambda)$.

□

1. General location of eigenvalues, Gershgorin circles

Since $\lambda_i \in \sigma(A)$ are the roots of the characteristic polynomial $p_\lambda(A) = \det(A - \lambda I)$, **iterative methods** are used to find λ_i, for $n > 5$. Hence, knowledge of eigenvalue location in $\mathbb{C}$ can **accelerate** convergence.

First, we recall some basic estimates.

- $|\lambda| \leq \|A\|$ for any consistent matrix norm $\|\cdot\|$. (See Theorem 2.1.)
- $\det(A) = \prod_{i=1}^{n} \lambda_i, \text{trace}(A) = \sum_{i=1}^{n} \lambda_i$. (See Proposition 1.1.)

THEOREM 1.1 (Gershgorin circles). *Let* $A \in \mathbb{C}^{n\times n}$.

$$\sigma(A) \subset \Big(\cup_{i=1}^{n} \mathscr{R}_i\Big) \cap \Big(\cup_{i=1}^{n} \mathscr{C}_i\Big), \tag{1.1}$$

where

$$\mathscr{R}_i = \Big\{z \in \mathbb{C} : |z - a_{ii}| \le \sum_{j=1,\, j\neq i} |a_{ij}|\Big\}, \qquad \text{(row circles)}$$

$$\mathscr{C}_i = \Big\{z \in \mathbb{C} : |z - a_{jj}| \le \sum_{i=1,\, i\neq j} |a_{ij}|\Big\}. \qquad \text{(column circles)}$$

PROOF. (See [1, p 127].) Let $\lambda \in \sigma(A)$ be an arbitrary eigenvalue, and $\boldsymbol{v}$ the corresponding eigenvector:

$$A\boldsymbol{v} = \lambda \boldsymbol{v} \quad \Leftrightarrow \quad (\lambda - a_{ii})v_i = \sum_{j=1, j\neq i} a_{ij}v_j, \qquad \text{for } i = 1:n.$$

If we denote by $\ell \in 1:n$ the index such that

$$\|\boldsymbol{v}\|_\infty = |v_\ell| \neq 0,$$

then

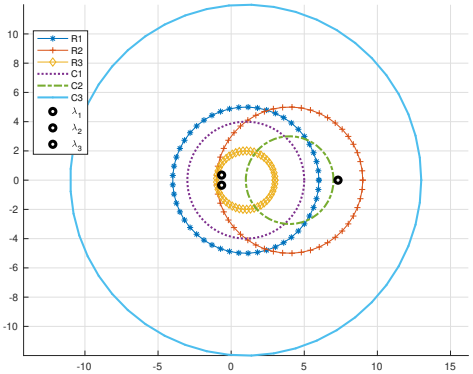
$$|\lambda - v_\ell| = \frac{1}{v_\ell}\Big|\sum_{j=1, j\neq \ell} a_{\ell j}v_j\Big| \le \sum_{j=1, j\neq \ell} |a_{\ell j}|\frac{|v_j|}{|v_\ell|} \le \sum_{j=1, j\neq \ell} |a_{\ell j}|,$$

which proves that the spectrum is inside the union of the row circles $\mathscr{R}_i$.
A similar argument holds for the column $\mathscr{C}_i$. □

EXAMPLE 1.1.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 9 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\mathscr{R}_1 = \{z : |z-1| \le 5\}, \quad \mathscr{R}_2 = \{z : |z-4| \le 12\}, \quad \mathscr{R}_3 = \{z : |z-1| \le 2\},$$
$$\mathscr{C}_1 = \{z : |z-1| \le 4\}, \quad \mathscr{C}_2 = \{z : |z-4| \le 3\}, \quad \mathscr{C}_3 = \{z : |z-1| \le 12\}.$$
$$\sigma(A) = \{7.3067, -0.6533 + 0.3473i, -0.6533 - 0.3473i\}.$$



Note also that

$$\|A\|_1 = \max\{5, 7, 15\} = 13 \qquad \text{(max row sums)}$$
$$\|A\|_2 = 11.0506$$

$$\|A\|_\infty = \max\{6, 16, 3\} = 16, \qquad \text{(max row sums)}$$

hence, by Theorem 2.1 we have

$$|\lambda_i| \leq \min\{13, 11.0506, 16\} = 11.0506.$$

2. Schur factorization and diagonalization

THEOREM 2.1 (Schur canonical form). *Given $A \in \mathbb{C}^{n\times n}$, there exists U unitary such that*

$$U^{-1}AU = U^H AU = \begin{bmatrix} \lambda_1 & b_{12} & \cdots & b_{1n} \\ 0 & \lambda_2 & & b_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix} = T,$$

where λ_i are the eigenvalues of A.

Therefore every A is unitarily similar to an upper triangular matrix, and by Theorem 0.1 (e) we have that $\sigma(A) = \sigma(A) = \text{diag}(T)$.

PROOF. By induction, see [5]. □

Note that the matrices T and U are not necessarily unique [9].

COROLLARY 2.1. *Every square hermitian matrix is unitarily similar to a diagonal matrix with real eigenvalues. Also, every square symmetric matrix is unitarily similar to a diagonal matrix with real eigenvalues.*

PROOF. From the Schur decomposition Theorem 2.1 $\exists U$ unitary $I = U^*U$ such that

$$U^*AU = T \quad \Rightarrow \quad U^* \underbrace{A^*}_{=A} U = T^*,$$

hence $T = T^*$, so $T = \text{diag}\{\lambda_1, \cdots, \lambda_n\}$ and $\lambda_i \in \mathbb{R}$. □

REMARK 2.1. *Note that*

$$A^m = (UTU^*)^m = UT^mU^*.$$

MATLAB

```
[U,T] = schur(A)
```

EXAMPLE 2.1.

```
>> A = hilb(3);
>> [U,T] = schur(A)
U =
   -0.1277    0.5474    0.8270
    0.7137   -0.5283    0.4599
   -0.6887   -0.6490    0.3233
T =
    0.0027         0         0
         0    0.1223         0
         0         0    1.4083
```

Note that

$$U \cdot T \cdot U' = A \quad \textit{and} \quad U' \cdot U = I_3.$$

```
>> U*T*U'
ans =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000
>> U'*U
ans =
    1.0000    0.0000    0.0000
    0.0000    1.0000   -0.0000
    0.0000   -0.0000    1.0000
```

EXAMPLE 2.2.

```
>> A = [ 13 8 8 ; -1 7 -2 ; -1 -2 7];
>> [U,T] = schur(A)
U =
    0.9428   -0.3333         0
   -0.2357   -0.6667   -0.7071
   -0.2357   -0.6667    0.7071
T =
    9.0000  -12.7279   -0.0000
         0    9.0000   -0.0000
         0         0    9.0000
```

3. Singular Value Decomposition (SVD)

The SVD is a very important decomposition which is used for many purposes other than solving least squares problems.

THEOREM 3.1. *Given $A \in \mathbb{C}^{m\times n}$, $\exists$ two unitary matrices $U \in \mathbb{C}^{m\times m}, V \in \mathbb{C}^{n\times n}$ ($U^*U = I_{m\times m}, V^*V = I_{n\times n}$) such that*

$$U^*AV = \Sigma = \operatorname{diag}(\sigma_1, \cdots, \sigma_p) \in \mathbb{R}^{m\times n}, \qquad \text{with } p = \min\{m,n\} \tag{SVD}$$

and

$$\sigma_1 \geq \cdots \geq \sigma_p \geq 0, \tag{singular values}$$

where σ_i = singular values of A.
The first p columns $u_1, \cdots, u_p$ of U are eigenvectors of AA^*, called left singular vectors.
The columns $v_1, \cdots, v_n$ of V are called right singular vectors.

PROOF. By induction, see [5, Theorem 3.2]. □

EXAMPLE 3.1. *Let $m = 4 < 5 = n$ and*

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \end{bmatrix}, \quad AA' = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}, \quad A'A = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 4 \end{bmatrix}.$$

Note that

$$\sqrt{5} = 2.2361, \quad \sqrt{9} = 3, \quad \sqrt{4} = 2.$$

```
>> A = [1 0 0 0 2; 0 0 3 0 0 ; 0 0 0 0 0 ; 0 2 0 0 0 ];
>> [U,S,V]= svd(A);
>> S

S =

    3.0000         0         0         0         0
         0    2.2361         0         0         0
         0         0    2.0000         0         0
         0         0         0         0         0

>> x = [1;2;3;4;5];b = A*x ;
>> y = A\b; z = V/S*U'*b;
>> y
y =
                   0
   2.000000000000000
   3.000000000000000
                   0
   5.500000000000000
>> z
z =
   2.199999999999999
   2.000000000000000
   3.000000000000000
                   0
   4.399999999999999
>> norm(A*y-b)
ans =
     0
>> norm(A*z-b)
ans =
     3.552713678800501e-15
```

THEOREM 3.2.

(a) $\Sigma_i(A) = \sqrt{\lambda_i(A^*A)}$, $i = 1, \cdots, p$.

(b) If A has full rank, the solution of

$$\min_{\boldsymbol{x}} \|A\boldsymbol{x} - \boldsymbol{b}\|_2$$

is

$$\boldsymbol{x} = V\Sigma^{-1}U^T\boldsymbol{b}. \qquad \text{(use the pseudoinverse)}$$

PROOF. (a)

$$A = U\Sigma V^*, \quad A^* = V\Sigma^*U^* \quad \Rightarrow \quad A^*A = V\Sigma^* \underbrace{U^*U}_{=I_{m\times m}} \Sigma V^* = V\Sigma^*\Sigma V^*.$$

Since V is unitary, by Theorem 0.1 part (d) and (SVD) we have that

$$\lambda_i(A^*A) = \lambda_i(\Sigma^*\Sigma) = \big(\sigma_i(A)\big)^2.$$

(b) Since U is unitary, we have

$$\|A\boldsymbol{x}-\boldsymbol{b}\|_2^2 = \|U\Sigma V^T\boldsymbol{x}-\boldsymbol{b}\|_2^2 \qquad \text{(use (SVD))}$$

$$= \left\|U^T(U\Sigma V^T\boldsymbol{x}-\boldsymbol{b})\right\|_2^2 \qquad \text{(use } U \text{ unitary)}$$

$$= \left\|\Sigma V^T\boldsymbol{x}-U^T\boldsymbol{b}\right\|_2^2 = \sum_{i=1}^{p}\left|\sigma_i(V^T\boldsymbol{x})_i-(U^T\boldsymbol{b})_i\right|^2+\sum_{i=p+1}^{m}\left|(U^T\boldsymbol{b})_i\right|^2,$$

which is minimized over $\boldsymbol{x}$ when all the elements in the first sum are zero, i.e.,

$$\boldsymbol{x} = V/\Sigma U^T\boldsymbol{b}.$$

□

EXAMPLE 3.2.

```
>> A = [1 2 3; 4 5 7]; x = [1;2;3]; b= A*x;
>> [U,S,V]=svd(A)
U =
   -0.3636   -0.9315
   -0.9315    0.3636
S =
   10.1815         0         0
         0    0.5811         0
V =
   -0.4017    0.9000    0.1690
   -0.5289   -0.0773   -0.8452
   -0.7476   -0.4289    0.5071
>> y = V/S*U'*b
y =
    1.0000
    2.0000
    3.0000
>> z = A\b
z =
    1.4000
         0
    4.2000
>> norm(y)
ans =
    3.7417
>> norm(z)
ans =
    4.4272
>> norm(A*y-b)
ans =
     1.588821858078255e-14
>> norm(A*z-b)
ans =
     1.776356839400250e-15
```

```
>> norm(A*x-b)
ans =
     0
```

EXAMPLE 3.3.

```
>> B = [1 2; 4 5; 8 9];x = [1 ; 2] ; b = B*x;
>> [U,S,V] = svd(B);
>> y = V/S*U'*b
y =
    1.0000
    2.0000
>> z = B\b
z =
    1.0000
    2.0000
```

DEFINITION 3.1 (E.H. Moore - Roger Penrose: pseudo-inverse matrix). *Suppose* $A \in \mathbb{C}^{n\times n}$ *has rank r, and* $\exists$ *an SVD decomposition* $U^*AV = \Sigma$ *(or* $A = U\Sigma V^*$*). The matrix*

$$A^\dagger = V\Sigma^\dagger U^* \qquad \text{(pseudo-inverse)}$$

is called the Moore-Penrose pseudo-inverse (the generalized inverse of A), where

$$\Sigma^\dagger = \text{diag}\Big(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \cdots, \frac{1}{\sigma_r}, 0, \cdots, 0\Big).$$

Images can be thought of as numeric arrays (although you do have to convert them from the uint8 numeric format used for images to the double format used for numeric arrays.) Therefore, an $m \times n$ image A has an SVD decomposition $A = U * S * V'$.
For any $1 \le r \le \min(m,n)$, a low rank approximation to A is formed by

```
A_r = U(1:m,1:r) * S(1:r,1:r) * V(1:n,1:r)';
```

Properties of the SVD guarantee that A_r is the best possible rank r approximation to the data in A. This means it is often possible to get a good approximation to A using much less data. Regarding a fingerprint image as collection of column vectors, we can apply this technique. Fingerprints can be difficult to compress, since they have a great deal of fine variation and detail.

svd fingerprint.m is a MATLAB code which reads a file containing a fingerprint image and uses the singular value decomposition (SVD) to compute and display a series of low rank approximations to the image (see figure 1).

svd gray test.m is a MATLAB code which calls svd gray(), which reads a file containing a grayscale image and uses the singular value decomposition (SVD) to compute and display a series of low rank approximations to the image (see figure 2).

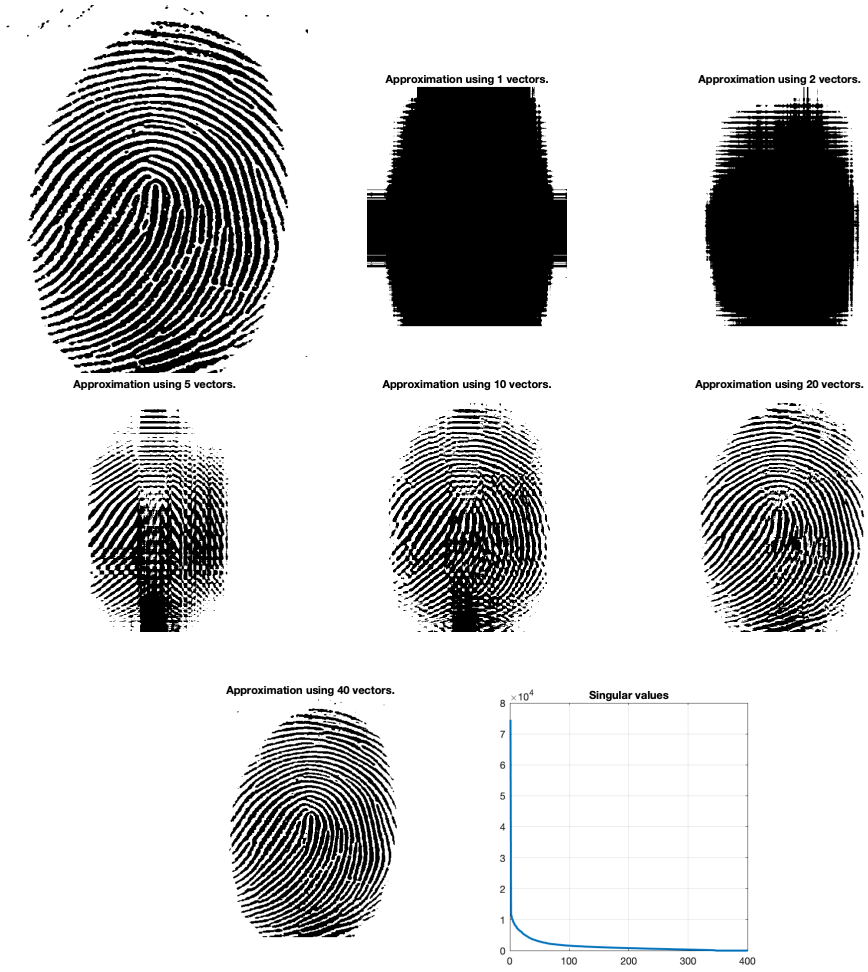


FIGURE 1. SVD of fingerprint: image rows $m = 480$, image columns $n = 400$, image pixels $m * n = 192000$.

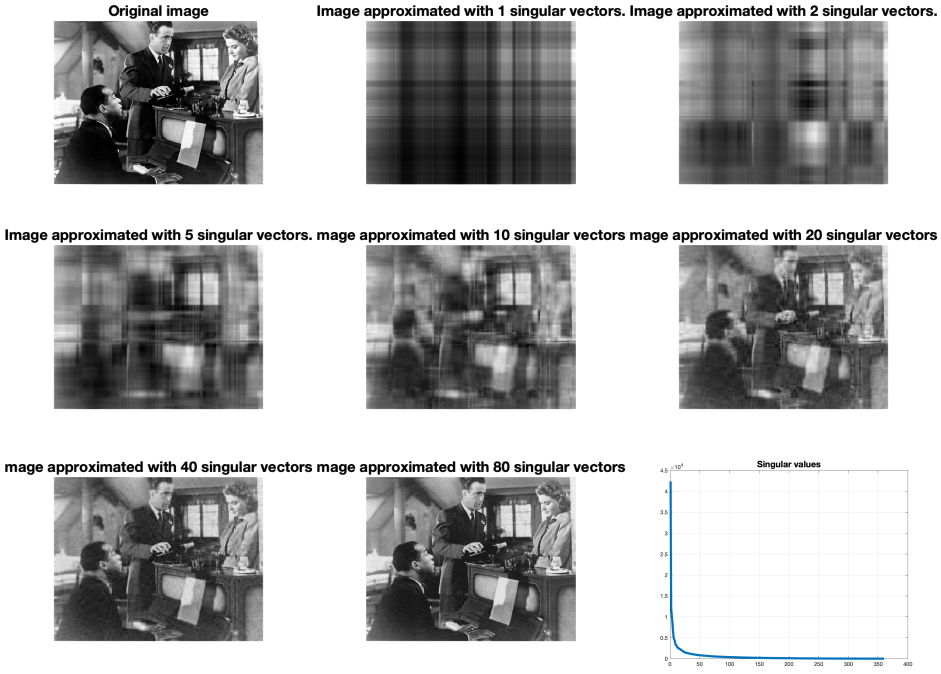


FIGURE 2. SVD of 'Casablanca': image rows $m = 360$, image columns $n = 460$, image pixels $m * n = 165600$.

4. The Power Method

It is very good at approximating the **extremal values** of a matrix (i.e., the **largest** or **smallest** module) λ_1, λ_n and their **associated eigenvalues**.
Applications:

- in geoseismic, structural vibrations, where the computation of λ_n and $\boldsymbol{x}_n$ arises in determination of the **proper frequency** and **fundamental mode** of a physical system
- in numerical analysis, stability analysis of systems of ODEs and PDEs
- search engines

4.1. Approximation of the Eigenvalues of Largest module.

Let $A \in \mathbb{C}^{n\times n}$ be diagonalizable ($\exists P$ invertible such that $P^{-1}AP = \text{diag}\{\lambda_a, \lambda_2, \cdots, \lambda_n\}$), and $C \in \mathbb{C}^{n\times n}$ the matrix of its right eigenvectors $\boldsymbol{x}_i, i = 1:n$.
Suppose the eigenvalues are ordered as

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|, \qquad (\lambda_1 \text{ dominant eigenvalue})$$

and λ_1 has algebraic multiplicity equal to 1, i.e., λ_1 is a simple root of the characteristic polynomial $p(\lambda)$:

$$p'(\lambda_1) \neq 0.$$

Result: *Power Method*
Given $\boldsymbol{x}^{(0)} \in \mathbb{C}^n, \|\boldsymbol{x}^{(0)}\|_2 = 1,$
for $\kappa = 1, 2, \cdots$ *maxiter* **do**

$$\begin{aligned}
&\boldsymbol{z}^{(\kappa)} = A\boldsymbol{x}^{(\kappa-1)} \\
&\boldsymbol{x}^{(\kappa)} = \boldsymbol{z}^{(\kappa)}/\|\boldsymbol{z}^{(\kappa)}\|_2 && (\|\boldsymbol{x}^{(\kappa)}\|_2 = 1) \\
&\lambda^{(\kappa)} = (\boldsymbol{x}^{(\kappa)})^* A\boldsymbol{x}^{(\kappa)}/(\boldsymbol{x}^{(\kappa)})^*\boldsymbol{x}^{(\kappa)} && \text{(Rayleigh quotient)}
\end{aligned}$$

end

THEOREM 4.1. *Assume $A \in \mathbb{C}^{n\times n}$ diagonalizable (the eigenvectors form a basis of $\mathbb{C}^n$), whose eigenvectors satisfy $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$, and $\boldsymbol{x}^{(0)} = \sum_{i=1}^n \alpha_i \boldsymbol{x}_i$, where $(\{\lambda_i, \boldsymbol{x}_i\}_{i=1:n})$ are the eigen-pairs, $\|\boldsymbol{x}_i\|_2 = 1$, with $\alpha_1 =\neq 0$. Then*

$$\lim_{\kappa\to\infty} |\lambda_1 - \lambda^{(\kappa)}| = 0, \quad \textit{and} \quad \|\boldsymbol{x}_1 - \boldsymbol{x}^{(\kappa)}\|_2 \xrightarrow{\kappa\to\infty} 0,$$

with linear conveergence and convergence rate $S = |\lambda_2/\lambda_1| < 1$:

$$\exists C > 0 \quad \textit{such that} \quad \|\boldsymbol{x}_1 - \boldsymbol{x}^{(\kappa)}\|_2 \leq C\left|\frac{\lambda_2}{\lambda_1}\right|^\kappa. \qquad \text{(linear convergence)}$$

PROOF. We shall start by proving using induction that

$$\boldsymbol{x}^{(\kappa)} = \frac{A^\kappa \boldsymbol{x}^{(0)}}{\|A^\kappa \boldsymbol{x}^{(0)}\|_2}, \tag{4.1}$$

so let's verify this for the first few iterations. Indeed,

$$\boldsymbol{z}^{(1)} = A\boldsymbol{x}^{(0)}, \quad \boldsymbol{x}^{(1)} = A\boldsymbol{x}^{(0)}/\|A\boldsymbol{x}^{(0)}\|_2, \qquad (\kappa = 1)$$

$$\boldsymbol{z}^{(2)} = A\boldsymbol{x}^{(1)} = A^2\boldsymbol{x}^{(0)}/\|A\boldsymbol{x}^{(0)}\|_2, \quad \text{hence} \quad \|\boldsymbol{z}^{(2)}\|_2 = \frac{\|A^2\boldsymbol{x}^{(0)}\|_2}{\|A\boldsymbol{x}^{(0)}\|_2}, \quad \text{so}$$

$$\boldsymbol{x}^{(2)} = \frac{\boldsymbol{z}^{(2)}}{\|\boldsymbol{z}^{(2)}\|_2} = \frac{A^2\boldsymbol{x}^{(0)}}{\cancel{\|A\boldsymbol{x}^{(0)}\|_2}} \frac{\cancel{\|A\boldsymbol{x}^{(0)}\|_2}}{\|A^2\boldsymbol{x}^{(0)}\|_2} = \frac{A^2\boldsymbol{x}^{(0)}}{\|A^2\boldsymbol{x}^{(0)}\|_2}, \cdots . \qquad (\kappa = 2)$$

Assume that (4.1) holds for κ, and try to prove it for $\kappa+1$. By the Power Method (1) and the induction hypothesis (4.1) we have

$$\boldsymbol{z}^{(\kappa+1)} = A\boldsymbol{x}^{(\kappa)} = A\frac{A^\kappa\boldsymbol{x}^{(0)}}{\|A^\kappa\boldsymbol{x}^{(0)}\|_2} = \frac{A^{\kappa+1}\boldsymbol{x}^{(0)}}{\|A^\kappa\boldsymbol{x}^{(0)}\|_2},$$

then

$$\|\boldsymbol{z}^{(\kappa+1)}\|_2 = \frac{\|A^{\kappa+1}\boldsymbol{x}^{(0)}\|_2}{\|A^\kappa\boldsymbol{x}^{(0)}\|_2},$$

so again from the algorithm we have

$$\boldsymbol{x}^{(\kappa+1)} = \frac{\boldsymbol{z}^{(\kappa+1)}}{\|\boldsymbol{z}^{(\kappa+1)}\|_2} = \frac{A^{\kappa+1}\boldsymbol{x}^{(0)}}{\cancel{\|A^\kappa\boldsymbol{x}^{(0)}\|_2}} \frac{\cancel{\|A^\kappa\boldsymbol{x}^{(0)}\|_2}}{\|A^{\kappa+1}\boldsymbol{x}^{(0)}\|_2} = \frac{A^{\kappa+1}\boldsymbol{x}^{(0)}}{\|A^{\kappa+1}\boldsymbol{x}^{(0)}\|_2}, \cdots .$$

Since $\boldsymbol{x}^{(0)} = \sum_{i=1}^n \alpha_i \boldsymbol{x}_i$ □

EXAMPLE 4.1 (Power iteration).

$$A = \begin{bmatrix} 1 & 0. & 1 \\ 0 & 3 & -2 \\ 0 & 0 & 2 \end{bmatrix}, \qquad \sigma(A) = \{3,2,1\}; \qquad \left|\frac{\lambda_2}{\lambda_1}\right| = \frac{2}{3} < 1$$

Let $\boldsymbol{x}^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$. *Then*

$$\boldsymbol{z}^{(0)} = A\boldsymbol{x}^{(0)} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 3 & -2 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 2 \end{bmatrix},$$

with

$$\text{norm}(\boldsymbol{z}^{(0)}, 2) = \sqrt{9} = 3,$$

so

$$\boldsymbol{x}^{(1)} = \frac{\boldsymbol{z}^{(0)}}{\|\boldsymbol{z}^{(0)}\|_2} = \begin{bmatrix} 1/3 \\ -2/3 \\ 2/3 \end{bmatrix},$$

$$\lambda^{(1)} = (\boldsymbol{x}^{(1)})^T A\boldsymbol{x}^{(1)} = \begin{bmatrix} 1/3 & -2/3 & 2/3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 3 & -2 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1/3 \\ -2/3 \\ 2/3 \end{bmatrix} = \frac{3+20+8}{9} = \frac{31}{9} \approx 3.4444,$$

$$\boldsymbol{z}^{(1)} = A\boldsymbol{x}^{(1)} = \begin{bmatrix} 1 \\ -10/3 \\ 4/3 \end{bmatrix},$$

then

$$\boldsymbol{x}^{(2)} = \frac{\boldsymbol{z}^{(1)}}{\|\boldsymbol{z}^{(1)}\|_2} = \frac{1}{5\sqrt{5}} \begin{bmatrix} 3 \\ -10 \\ 4 \end{bmatrix},$$

$$\lambda^{(2)} = (\boldsymbol{x}^{(2)})^T A\boldsymbol{x}^{(2)} = 3.4640,$$
$$\boldsymbol{z}^{(2)} = A\boldsymbol{x}^{(2)} = \frac{1}{5\sqrt{5}}\begin{bmatrix} 1 & 0 & 1 \\ 0 & 3 & -2 \\ 0 & 0 & 2 \end{bmatrix}\begin{bmatrix} 13 \\ -10 \\ 4 \end{bmatrix} = \frac{1}{5\sqrt{5}}\begin{bmatrix} 7 \\ -38 \\ 8 \end{bmatrix},$$

also

$$\boldsymbol{x}^{(3)} = \frac{\boldsymbol{z}^{(2)}}{\|\boldsymbol{z}^{(2)}\|_2} = \begin{bmatrix} 0.1774000048296265 \\ 0.963028833608294 \\ 0.202742912338588 \end{bmatrix},$$
$$\lambda^{(3)} = (\boldsymbol{x}^{(3)})^T A\boldsymbol{x}^{(3)} = 3.322414900449583.$$

We have

$$\lambda^{(4)} = 3.212530924572810$$
$$\lambda^{(5)} = 3.139135553593345$$
$$\lambda^{(10)} = 3.017489354617178 \qquad ((\lambda_2/\lambda_1)^{10} = (2/3)^{10} \approx 0.017)$$
$$\lambda^{(50)} = 3.\underbrace{00000000}_{8}1568329 \qquad ((\lambda_2/\lambda_1)^{50} = (2/3)^{50} \approx 1.5e-9)$$
$$\lambda^{(75)} = 3.\underbrace{0000000000000}_{13}62 \qquad ((\lambda_2/\lambda_1)^{75} = (2/3)^{75} \approx 6.2e-14)$$

```
[lambda, x] = powerit(A,x0,maxiter)
```

Recall that the Power Method 1 computes

$$\boldsymbol{x}^{(\kappa)} = \frac{A^\kappa \boldsymbol{x}^{(0)}}{\|A^\kappa \boldsymbol{x}^{(0)}\|_2} \longrightarrow \boldsymbol{x}_1,$$
$$\lambda^{(\kappa)} = (\boldsymbol{x}^{(\kappa)})^T A\boldsymbol{x}^{(\kappa)} \longrightarrow \lambda_1,$$

the eigenvector (corresponding to) and the largest eigenvalue of A.

LEMMA 4.1.

(1) The eigenvalues of A^{-1} are $\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \cdots, \frac{1}{\lambda_n}$, where $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$, with the same eigenvectors of A.

(2) The eigenvalues of the shifted matrix $A - \mu I$ are $\lambda_1 - \mu, \lambda_2 - \mu, \cdots, \lambda_n - \mu$, with the same eigenvectors of A.

PROOF.

(1) Since $A\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i$, multiplying to the left by A^{-1} gives $\boldsymbol{x}_i = \lambda_i A^{-1}\boldsymbol{x}_i$, or $A^{-1}\boldsymbol{x}_i = \frac{1}{\lambda_i}\boldsymbol{x}_i$.

(2) Similarly, from $A\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i$, we have that $(A - \mu I)\boldsymbol{x}_i = (\lambda_i - \mu)\boldsymbol{x}_i$.

□

COROLLARY 4.1.

(3) The largest (magnitude) eigenvalue of $A^{-1} \equiv$ the smallest (magnitude) eigenvalue of A.

(4) The largest (magnitude) eigenvalue of $(A - \mu I)^{-1} \equiv$ the smallest (magnitude) eigenvalue $\lambda_i - \mu$ of $A - \mu I$
$\equiv$ closest eigenvalue λ_i to μ.

Therefore, to compute an eigenvalue 'closest' to a given $\mu \in \mathbb{C}$ ($|\lambda - \mu|$ smallest), we have to apply the Power Iteration Method to $(A-\mu I)^{-1}$, hence obtaining the

4.2. Inverse Power Method.

Let $\mu \in \mathbb{C}, \mu \notin \sigma(A)$ be called a '*shift*'. Assuming $\exists m$ such that

$$|\lambda - \mu| < |\lambda_j - \mu|, \qquad \forall j \neq m, \quad \lambda_j \in \sigma(A),$$

we have the following

Result: *Inverse Power Method*
Given $\boldsymbol{x}^{(0)} \in \mathbb{C}^n, \|\boldsymbol{x}^{(0)}\|_2 = 1$,
for $\kappa = 1, 2, \cdots$ *maxiter* **do**
  Solve $(A-\mu I)\boldsymbol{z}^{(\kappa)} = \boldsymbol{x}^{(\kappa-1)} \Leftrightarrow \boldsymbol{z}^{(\kappa)} = (A-\mu I)^{-1}\boldsymbol{x}^{(\kappa-1)}$
  $\boldsymbol{x}^{(\kappa)} = \boldsymbol{z}^{(\kappa)} / \|\boldsymbol{z}^{(\kappa)}\|_2$ $\qquad (\|\boldsymbol{x}^{(\kappa)}\|_2 = 1)$
  $\lambda^{(\kappa)} = (\boldsymbol{x}^{(\kappa)})^* A \boldsymbol{x}^{(\kappa)} / (\boldsymbol{x}^{(\kappa)})^* \boldsymbol{x}^{(\kappa)}$ $\qquad$ (Rayleigh quotient)
  (since A and $A - \lambda I$ have the same eigenvectors, the Rayleigh quotient is computed on A, not on $A - \mu I$)
  Note that we DO NOT use $(A-\mu I)^{-1}$, but solve a linear system!
end

EXAMPLE 4.2 (Inverse Power iteration). *See also Example 4.1*

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 3 & -2 \\ 0 & 0 & 2 \end{bmatrix}, \qquad \sigma(A) = \{3, 2, 1\}; \qquad \left|\frac{\lambda_2}{\lambda_1}\right| = \frac{2}{3} < 1$$

```
>> A = [1 0 1; 0 3 -2; 0 0 2]; mu=1.6; x0=[0;0;1]; k=100;
>> [lambda,x] = invpowerit(A,x0,mu,k);
>> lambda =
   2.000000000000000
the eigenvalue 2.000000 was reached with tolerance 1e-16 at iteration 87
```

Let $\boldsymbol{x}^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$. *Then*

$$\lambda^{(0)} = (\boldsymbol{x}^{(0)})^T A \boldsymbol{x}^{(0)} = 1.6687$$

and

$$\boldsymbol{z}^{(1)} = (A-\mu I)\backslash \boldsymbol{x}^{(0)} = \begin{bmatrix} 0.5757 \\ 1.9036 \\ 1.0364 \end{bmatrix}, \quad \boldsymbol{x}^{(1)} = \begin{bmatrix} 0.2567 \\ 0.8488 \\ 0.4621 \end{bmatrix}, \quad \lambda^{(1)} = 1.9886, \quad \left|\frac{\lambda^{(1)} - \lambda^{(0)}}{\lambda^{(0)}}\right| \approx 0.191,$$

$$\boldsymbol{z}^{(2)} = (A-\mu I)\backslash \boldsymbol{x}^{(1)} = \begin{bmatrix} 1.4976 \\ 2.2568 \\ 1.1553 \end{bmatrix}, \quad \boldsymbol{x}^{(2)} = \begin{bmatrix} 0.5086 \\ 0.7664 \\ 0.3923 \end{bmatrix}, \quad \lambda^{(2)} = 1.9268, \quad \left|\frac{\lambda^{(2)} - \lambda^{(1)}}{\lambda^{(1)}}\right| \approx 0.031,$$

$$\boldsymbol{z}^{(3)} = (A-\mu I)\backslash \boldsymbol{x}^{(2)} = \begin{bmatrix} 0.7871 \\ 1.9486 \\ 0.9808 \end{bmatrix}, \quad \boldsymbol{x}^{(3)} = \begin{bmatrix} 0.3393 \\ 0.84021 \\ 0.4229 \end{bmatrix}, \quad \lambda^{(3)} = 2.02361, \quad \left|\frac{\lambda^{(3)} - \lambda^{(2)}}{\lambda^{(2)}}\right| \approx 0.05022,$$

$$\boldsymbol{z}^{(4)} = (A-\mu I)\backslash \boldsymbol{x}^{(3)} = \begin{bmatrix} 1.1965 \\ 2.11059 \\ 1.0573 \end{bmatrix}, \quad \boldsymbol{x}^{(4)} = \begin{bmatrix} 0.452 \\ 0.797 \\ 0.399 \end{bmatrix}, \quad \lambda^{(4)} = 1.97500, \quad \left|\frac{\lambda^{(4)} - \lambda^{(3)}}{\lambda^{(3)}}\right| \approx 0.0240.$$

4.3. Exercises.

Exercise 1. *Are* $[i,-1+i]^T$ *and* $[-i,-1-i]^T$ *eigenvectors of the matrix* $A = \begin{bmatrix} 1 & 1 \\ -2 & 3 \end{bmatrix}$ *in Example 2?*
Solution:

Exercise 2. *Prove that if* λ *is an eigenvalue of a real matrix with eigenvector* $\boldsymbol{x}$*, then* $\overline{\lambda}$ *is also an eigenvalue with eigenvector* $\overline{\boldsymbol{x}}$*. (For a complex number* $z = x+iy$*, the conjugate is defined by* $z = x-iy$*.)*
Solution:

Exercise 3. *Let*

$$A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Account for the fact that the matrix A has the effect of rotating vectors counterclockwise through an angle θ *and thus cannot map any vector into a multiple of itself.*
Solution:

Exercise 4. *Let A be an* $m \times n$ *matrix such that* $A = UDV^T$*, where U and V are orthogonal and D is diagonal and nonnegative. Prove that the diagonal elements of D are the singular values of A.*
Solution:

Exercise 5. *Let* A, U, D*, and* V *be as in the singular value decomposition:* $A = UDV^T$*. Let r be as described in the text. Define* U_r *to consist of the first r columns of* U*. Let* V_r *consist of the first r columns of* V*, and let* D_r *be the* $r \times r$ *matrix having the same diagonal as D. Prove that* $A = U_r D_r V_r^T$ *. (This factorization is called the economical version of the singular value decomposition.)*
Solution:

Exercise 6. *A linear map P is a* **projection** *if* $P^2 = P$*. We can use the same terminology for an* $n \times n$ *matrix:* $A^2 = A$ *is the projection property. Use the* **Pierce decomposition**, $I = A + (I - A)$*, to show that every point in* $\mathbb{R}^n$ *is the sum of a vector in the range of A and a vector in the null space of A. What are the eigenvalues of a projection?*
Solution:

Exercise 7. *Find all of the Gershgorin discs for the following matrices. Indicate the smallest region(s) containing all of the eigenvalues:*

(a) $A = \begin{bmatrix} 3 & -1 & 1 \\ 2 & 4 & -2 \\ 3 & -1 & 9 \end{bmatrix}$

(b) $A = \begin{bmatrix} 3 & 1 & 2 \\ -1 & 4 & -1 \\ 1 & -2 & 9 \end{bmatrix}$

(c) $A = \begin{bmatrix} 1-i & 1 & i \\ 0 & 2i & 2 \\ 1 & 0 & 2 \end{bmatrix}$

Solution:

Exercise 8. (Multiple choice) *Let A be an $n \times n$ invertible (nonsingular) matrix. Let $\boldsymbol{x}$ be a nonzero vector. Suppose that $A\boldsymbol{x} = \lambda \boldsymbol{x}$. Which equation does* **not** *follow from these hypotheses?*

(a) $A^k \boldsymbol{x} = \lambda^k \boldsymbol{x}$
(b) $\lambda^{-k} \boldsymbol{x} = (A^{-1})^k \boldsymbol{x}$ *for* $k \geq 0$
(c) $p(A)\boldsymbol{x} = p(\lambda)\boldsymbol{x}$ *for any polynomial* p
(d) $A^k \boldsymbol{x} = (1-\lambda)^k \boldsymbol{x}$
(e) None of these.

Solution:

Exercise 9. (Multiple choice) *For what values of s will the matrix $I - s\boldsymbol{v}\boldsymbol{v}^*$ be unitary, where v is a column vector of unit length?*

(a) $0, 1$
(b) $0, 2$
(c) $1, 2$
(d) $0, \sqrt{2}$
(e) None of these.

Solution:

Exercise 10. (Multiple choice) *Let U and V be unitary $n \times n$ matrices, possibly complex. Which conclusion is* **not** *justified?*

(a) $U + V$ is unitary.
(b) U^* is unitary.
(c) UV is unitary.
(d) $U - \boldsymbol{v}\boldsymbol{v}^*$ is unitary when $\|\boldsymbol{v}\| = \sqrt{2}$ and $\boldsymbol{v}$ is a column vector.
(e) None of these.

Solution:

Exercise 11. (Multiple choice) *Which assertion is true?*

(a) Every $n \times n$ matrix has n distinct (different) eigenvalues.
(b) The eigenvalues of a real matrix are real.
(c) If U is a unitary matrix, then $U^* = U^T$
(d) A square matrix and its transpose have the same eigenvalues.
(e) None of these.

Solution:

Exercise 12. (Multiple choice) *Consider the symmetric matrix*

$$\begin{bmatrix} 1 & 3 & 4 & -1 \\ 3 & 7 & -6 & 1 \\ 4 & -6 & 3 & 0 \\ -1 & 1 & 0 & 5 \end{bmatrix}.$$

What is the smallest interval derived from Gershgorin's Theorem such that all eigenvalues of the matrix A lie in that interval?

(a) $[-7,9]$
(b) $[-7,13]$
(c) $[3,7]$
(d) $[-3,17]$
(e) None of these.

Solution:

Exercise 13. (True or False) *Gershgorin's Theorem asserts that every eigenvalue λ of an $n \times n$ matrix A must satisfy one of these inequalities:*

$$|\lambda - a_{ii}| \leq \sum_{j=1, j\neq i}^{n} |a_{ij}| \qquad \text{for} \quad 1 \leq i \leq n.$$

Solution:

Exercise 14. (True or False) *A consequence of Schur's Theorem is that every square matrix A can be factored as $A = PTP^{-1}$, where P is a nonsingular matrix and T is upper triangular.*
Solution:

Exercise 15. (True or False) *A consequence of Schur's Theorem is that every (real) symmetric matrix A can be factored in the form $A = PDP^{-1}$, where P is unitary and D is upper triangular.*
Solution:

Exercise 16. *Explain why $\|UB\|_2 = \|B\|_2$ for any matrix B when $U^T U = I$.*
Solution:

Exercise 17. *Consider the matrix* $A = \begin{bmatrix} 3 & -\frac{1}{2} & 0 \\ \frac{3}{5} & 5 & -\frac{3}{5} \\ 0 & \frac{1}{2} & 3 \end{bmatrix}$. *Plot the Gershgorin discs in the complex plane for A and A^T as well as indicate the locations of the eigenvalues.*
Solution:

Exercise 18. (Continuation) *Let B be the matrix obtained by changing the negative entries in A to positive numbers. Repeat the process for B.*
Solution:

Exercise 19. (Continuation) *Repeat for* $C = \begin{bmatrix} 4 & 0 & -2 \\ 1 & 2 & 0 \\ 1 & 1 & 9 \end{bmatrix}$.
Solution:

Exercise 20. *Find the Schur decomposition of* $A = \begin{bmatrix} 5 & 7 \\ -2 & -4 \end{bmatrix}$.
Solution:

Computer problem 1. *Use Matlab, Maple, Mathematica, or other computer programs available to you to compute the eigenvalues and eigenvectors of these matrices:*

(a) $A = \begin{bmatrix} 1 & 7 \\ 2 & -5 \end{bmatrix}$.

(b)

Solution: **Computer problem 4, page 381.** *Use the power method, the inverse power method, and their shifted forms to find some or all of the eigenvalues of the following matrices:*

(a) $A = \begin{bmatrix} 5 & 4 & 1 & 1 \\ 4 & 5 & 1 & 1 \\ 1 & 1 & 4 & 2 \\ 1 & 1 & 2 & 4 \end{bmatrix}$.

(b)

Solution:

```
>> A = [5,4,1,1;4,5,1,1;1,1,4,2;1,1,2,4]; x0 = [1;1;1:1]; k = 100;

>> spectrum = eig(A)
spectrum =
 1.0000
    2.0000
    5.0000
   10.0000
>> mu1 = 1.4;
>> [lambda,x]=invpowerit(A,x0,mu1,k)

the eigenvalue 1.000000 was reached with tolerance 1e-16 at iteration 51

lambda =
   1.000000000000000
x =
   0.707106781186547
  -0.707106781186547
  -0.000000003165054
   0.000000003165054
```

$\mu = 1.4 \quad \Rightarrow \lambda = 1 \quad$ in 51 iterations with error $1.e-16$

$\mu = 1.6 \quad \Rightarrow \lambda = 2 \quad$ in 40 iterations with error $1.e-16$

$\mu = 4 \quad \Rightarrow \lambda = 5 \quad$ in 11 iterations with error $1.e-16$

$\mu = 8 \quad \Rightarrow \lambda = 10 \quad$ in 42 iterations with error $1.e-16$

CHAPTER 5

Numerical methods for Ordinary Differential Equations (ODEs)

DEFINITION 0.1 (ODE of order p).

$$\frac{d^p y}{dt^p} + a_{p-1}\frac{d^{p-1}y}{dt^{p-1}} + \cdots + a_1\frac{dy}{dt} + a_0 y = f(t, y, y', \ldots, y^{(p-1)}). \tag{0.1}$$

REMARK 0.1. *An ODE of order p can be written as a system of p first-order equations, with the unknowns:*

$$y_1 = y, \quad , y_2 = y', \quad y_3 = y'', \quad , \ldots, \quad y_{p-1} = y^{(p-2)}, y_p = y^{(p-1)},$$

hence satisfying

$$\begin{cases} y_1' &= y_2 \\ y_2' &= y_3, \\ \vdots & \\ y_{p-1}' &= y_p, \\ y_p' &= f(t, y_1, y_2, \ldots, y_{p-1}) - a_{p-1}y_p - \cdots - a_1 y_2 - a_0 y_1. \end{cases}$$

EXAMPLE 0.1 (Logistic equation).

$$y' = cy\left(1 - \frac{y}{B}\right). \qquad \text{(scalar equation)}$$

See logistic equation.m*, using Matlab's ODE45, ODE23s solvers and the 'forward Euler' method.*

EXAMPLE 0.2 (`Lotka-Volterra`).

$$\begin{cases} y_1' &= y_1(\alpha - \beta y_2) \\ y_2' &= y_2(\delta y_1 - \gamma). \end{cases} \qquad \text{(system of 2 first-order equations)}$$

See Example 6.2 for the Hamiltonian of the Lotka-Volterra model, a conserved quantity $H(y_1, y_2) = \delta y_1 - \gamma \ln y_1 + \beta y_2 - \alpha y_2$, i.e., $\frac{dH(y_1,y_2)}{dt} = 0$.

EXAMPLE 0.3 (Vibrating spring / simple harmonic motion).

$$u''(x) + k(x)u'(x) + m(x)u(x) = f(x). \qquad (2^{\text{nd}} \text{ order scalar equation})$$

See Example 6.1 for the particular case $u'' + u = 0$.

DEFINITION 0.2 (The Cauchy problem, Initial Value Problem (IVP)). *Find $y : I \to \mathbb{R}^n, y \in C^1(\mathbb{R}^n)$ such that*

$$\begin{cases} y' &= f(t, y(t)), \qquad t \in (0, T] \\ y(t_0) &= y_0. \end{cases} \tag{IVP}$$

Here $I=[t_0,T]$, $y_0\in\mathbb{R}^n$ is called 'initial datum', and $y(\cdot)\in C^1(\mathbb{R})$ satisfying (IVP) *is a classical solution. Moreover, if $f(t,y(t))\equiv f(y(t))$, the equation/system* (IVP) *is called 'autonomous'.*

DEFINITION 0.3. *The equation* (IVP) *can be considered in 'integral form':*

$$y(t)=y(t_0)+\int_{t_0}^{t} f(s,y(s))\,ds. \tag{0.2}$$

and then the solution $y(\cdot)\in C(\mathbb{R}^n)$ is called a 'mild solution'.

DEFINITION 0.4. *The function f is globally Lipschitz if*

$$|f(t,y)-f(s,z)|\le L\big(|t-s|+|y-z|\big),\qquad \forall (t,y),(s,z)\in\mathbb{R}\times\mathbb{R}^n. \tag{0.3}$$

THEOREM 0.1 (Existence and uniqueness of IVPs). *If f is Lipschitz continuous, then $\exists!$ there exists a unique solution y to the* (IVP).

1. One-step numerical methods

To simplify the presentation, we shall consider first the constant step size (equidistant mesh) case, i.e.,

$$\Delta t=\frac{T-t_0}{N}, \qquad \text{(constant time step)}$$

where $N\in\mathbb{N}$ is the number of subintervals, and $t_i=t_0+i\Delta t$, for all $i=0:N$.

In order to introduce some of the most classical numerical methods for ODEs, let us consider the Initial Value Problem (IVP) in its integral form (0.2). Or equivalently, let us integrate (IVP) on $[t_n,t_{n+1}]$ to obtain

$$y(t_{n+1})=y(t_n)+\int_{t_n}^{t_{n+1}} f(t,y(t))\,dt, \tag{1.1}$$

and use some of the quadrature (numerical integration) rules to approximate the integral, namely

$$\int_{t_n}^{t_{n+1}} f(t,y(t))\,dt\approx f(t_n,y(t_n))\cdot\Delta t \qquad \text{(rectangle left)}$$

$$\int_{t_n}^{t_{n+1}} f(t,y(t))\,dt\approx f(t_{n+1},y(t_{n+1}))\cdot\Delta t \qquad \text{(rectangle right)}$$

$$\int_{t_n}^{t_{n+1}} f(t,y(t))\,dt\approx f\Big(\frac{t_n+t_{n+1}}{2},y\Big(\frac{t_n+t_{n+1}}{2}\Big)\Big)\cdot\Delta t \qquad \text{(rectangle midpoint rule)}$$

$$\int_{t_n}^{t_{n+1}} f(t,y(t))\,dt\approx \frac{f(t_{n+1},y(t_{n+1}))+f(t_n,y(t_n))}{2}\cdot\Delta t \qquad \text{(trapezoidal rule)}$$

These approximations then yield the following numerical methods for solving (IVP)

$$y_{n+1}=y_n+\Delta t f(t_n,y_n) \qquad \text{(Forward Euler)}$$

$$y_{n+1}=y_n+\Delta t f(t_{n+1},y_{n+1}) \qquad \text{(Backward Euler)}$$

$$y_{n+1}=y_n+\Delta t f\Big(\frac{t_n+t_{n+1}}{2},???\Big)\Big)\approx y_n+\Delta t f\Big(\frac{t_n+t_{n+1}}{2},\frac{y_n+y_{n+1}}{2}\Big)\Big) \qquad \text{(implicit midpoint)}$$

$$y_{n+1}=y_n+\Delta t\frac{f(t_{n+1},y_{n+1})+f(t_n,y_n)}{2} \qquad \text{(trapezoidal)}$$

where $y_n\approx y(t_n),\forall n=1:N$.

NOTATION 1.1. *From now on we are going to denote frequently* $f_n := f(t_n, y_n)$.

DEFINITION 1.1 (One-step numerical method). *Let* Φ *be a functional depending on the unknowns* $\{y_n\}$ *and the values* $f(y_n)$. *Then the relation*

$$y_{n+1} = y_n + \Delta t \Phi(y_{n+1}, y_n, f_n, f_{n+1}) \tag{1.2}$$

is called a one-step numerical method for approximating the Cauchy problem (IVP).

Note that the methods above, namely (Forward Euler) (FE), (Backward Euler) (BE), (implicit midpoint), and (trapezoidal) are all one-step methods.

2. Consistency

DEFINITION 2.1 (Local truncation error). *The residual obtained by substituting the exact solution* $y(t)$ *of* (IVP)*, evaluated at the mesh points* t_n, t_{n+1}, *in the numerical method* (1.2)

$$\Delta t\, \tau_n(\Delta t) := y(t_{n+1}) - y(t_n) - \Delta t \Phi\big(y(t_{n+1}), y(t_n), f(y(t_n)), f(y(t_{n+1}))\big) \tag{LTE}$$

is called the Local Truncation Error (LTE), and is used to evaluate the approximating qualities of each method, order of 'consistency', and serves as an essential tool in time-adaptivity [3].

DEFINITION 2.2 (Order of consistency). *A numerical method is 'consistent with order* p' *if*

$$\Delta t\, \tau_n(\Delta t) = \mathcal{O}(\Delta t^{p+1}),$$

i.e., there exist a positive constant $C > 0$ *such that*

$$|y(t_{n+1}) - y(t_n) - \Delta t \Phi\big(y(t_{n+1}), y(t_n), f(y(t_n)), f(y(t_{n+1}))\big)| \le C \Delta t^{p+1}$$

EXAMPLE 2.1. *The* (Forward Euler) *method has order of consistency* $p = 1$.

PROOF. Indeed, by substituting the exact solution $y(t)$ into the (Forward Euler) formula, or equivalently, using the (LTE) with Φ from (Forward Euler), we obtain

$$\Delta t\, \tau_n^{\mathrm{FE}} := y(t_{n+1}) - y(t_n) - \Delta t f(y(t_n)).$$

Inhere we use the Taylor formula to expand $y(t_{n+1})$ about t_n, namely

$$y(t_{n+1}) = y(t_n) + y'(t_n)\Delta t + y''(\widetilde{t_n})\frac{\Delta t^2}{2},$$

where (of course) this holds provided the exact solution is smooth, meaning it has two continuous derivatives: $y \in C^2[t_0, T]$, and where $\widetilde{t_n} \in (t_n, t_{n+1})$, an arbitrary point. Substituting this into the value of the local truncation above yields

$$\Delta t\, \tau_n^{\mathrm{FE}} := y(t_{n+1}) - y(t_n) - \Delta t f(y(t_n)) = \Big(\cancel{y(t_n)} + y'(t_n)\Delta t + y''(\widetilde{t_n})\frac{\Delta t^2}{2}\Big) - \cancel{y(t_n)} - \Delta t f(y(t_n))$$

$$= y'(t_n)\Delta t + y''(\widetilde{t_n})\frac{\Delta t^2}{2} - \Delta t f(y(t_n)) = \Delta t \underbrace{\big(y'(t_n) - f(y(t_n))\big)}_{=0 \text{ from (IVP)}} + y''(\widetilde{t_n})\frac{\Delta t^2}{2} = \frac{1}{2} y''(\widetilde{t_n})\Delta t^2,$$

which shows that $p = 1$, i.e., (Forward Euler) is first-order consistent. □

EXAMPLE 2.2 (Implicit midpoint / Crank-Nicolson). *The* (implicit midpoint) *method can also be implemented in a refactorized form as*

$$\begin{cases} \dfrac{y^* - y_n}{\Delta t/2} = f(y^*), \\ \dfrac{y^{n+1} - y^*}{\Delta t/2} = f(y^*), \qquad \text{or} \qquad y_{n+1} = 2y^* - y_n \end{cases}, \qquad \text{equivalently} \qquad y_{n+1} = y_n + \Delta t\, f(y^*), \tag{midpoint}$$

where y^* denotes $y_{n+1/2}$. Then it is easily seen that the (implicit midpoint) *method is also second-order consistent*

$$\Delta t\, \tau_n^{\text{midpoint}} := y(t_{n+1}) - y(t_n) - \Delta t\, f(y(t_{n+1/2})) = \frac{1}{24} y'''(t_{n+1/2}) \Delta t^3.$$

PROOF. Indeed, proceeding as above

$$\begin{aligned} \Delta t\, \tau_n^{\text{midpoint}} &= y(t_{n+1}) - y(t_n) - \Delta t\, f(y(t_{n+1/2})) \\ &= \Big(\cancel{y(t_{n+1/2})} + \frac{\Delta t}{2} y'(t_{n+1/2}) + \cancel{\frac{\Delta t^2}{2! \cdot 4} y''(t_{n+1/2})} + \frac{\Delta t^3}{3! \cdot 8} y'''(t_{n+1/2}) + \mathcal{O}(\Delta t^4) \Big) \\ &\quad - \Big(\cancel{y(t_{n+1/2})} - \frac{\Delta t}{2} y'(t_{n+1/2}) + \cancel{\frac{\Delta t^2}{2! \cdot 4} y''(t_{n+1/2})} - \frac{\Delta t^3}{3! \cdot 8} y'''(t_{n+1/2}) + \mathcal{O}(\Delta t^4) \Big) - \Delta t\, f(y(t_{n+1/2})) \\ &= \Delta t \Big(\underbrace{y'(t_{n+1/2}) - f(y(t_{n+1/2}))}_{\equiv 0 \text{ from (IVP)}} + 2 \frac{\Delta t^2}{3! \cdot 8} y'''(t_{n+1/2}) \Big) + \mathcal{O}(\Delta t^4) = \frac{1}{24} y'''(t_{n+1/2}) \Delta t^3 \end{aligned}$$

which implies that $p = 2$. □

EXAMPLE 2.3 (Trapezoidal method). *The* (trapezoidal) *method can also be implemented in a refactorized form as*

$$\begin{cases} \dfrac{\widehat{y} - y_n}{\Delta t/2} = f(y_n), \\ \dfrac{y^{n+1} - \widehat{y}}{\Delta t/2} = f(y_{n+1}), \end{cases}, \qquad \text{equivalently} \qquad y_{n+1} = y_n + \frac{\Delta t}{2} \big(f(y_{n+1} + f(y_n) \big), \tag{trapezoidal}$$

where $\widehat{y}$ denotes $y_{n+1/2}$. Then it is easily seen that the (trapezoidal) *method is also second-order consistent*

$$\Delta t\, \tau_n^{\text{TR}} := y(t_{n+1}) - y(t_n) - \Delta t\, f(y(t_{n+1/2})) = -\frac{1}{12} y'''(t_{n+1/2}) \Delta t^3.$$

Note that the LTE of the (implicit midpoint) *is half the LTE of* (trapezoidal) *method.*

EXAMPLE 2.4 (Explicit midpoint / leapfrog method). *Using the second-order central difference approximation of a first-derivative*

$$y'(t_{n+1}) = \frac{y(t_{n+2}) - y(t_n)}{2\Delta t} + \mathcal{O}(\Delta t^2), \qquad \text{(central difference approximation)}$$

we obtain the following important example of a Linear Multistep Method (LMM) is the leapfrog method:

$$y_{n+2} = y_n + 2\Delta t\, f(y_{n+1}), \tag{LF}$$

which is also second-order consistent.

EXAMPLE 2.5 (BDF2). *Another important LMM is the backward differentiation formula 2 (BDF2) method:*

$$\frac{3y_{n+2}-4y_{n+1}+y_n}{2\Delta t}=f(y_{n+2}), \tag{BDF2}$$

also second-order consistent.

EXAMPLE 2.6. *In their 1952 very famous* `paper` *[10] on the nerve axon,* `Alan Hodgkin` *and* `Andrew Huxley` *proposed the following method*

$$\frac{y_{n+1}-y_n}{\Delta t}=\frac{1}{2}\Big(f(y_{n+1})+f(y_n)-\frac{1}{12}\big(\Delta^2 f_{n+1}+\Delta^2 f_n\big)\Big), \tag{[10]}$$

where $\Delta^2 f_{n+1},\Delta^2 f_n$ *are the second differences:*

$$\Delta^2 f_{n+1}=f(y_{n+1})-2f(y_n)+f(y_{n-1}) \qquad \Delta^2 f_n=f(y_n)-2f(y_{n-1})+f(y_{n-2}),$$

which yields

$$\begin{aligned}\frac{y_{n+1}-y_n}{\Delta t}&=\frac{1}{2}\Big(f(y_{n+1})+f(y_n)-\frac{1}{12}\big(\frac{f_{n+1}-2f_n+f_{n-1}}{2\Delta t^2}+\frac{f_n-2f_{n-1}+f_{n-2}}{2\Delta t^2}\big)\Big)\\&=\frac{1}{2}\Big(f(y_{n+1})+f(y_n)-\frac{1}{12}\big(f_{n+1}-f_n-f_{n-1}+f_{n-2}\big)\Big),\end{aligned}$$

so

$$\frac{y_{n+1}-y_n}{\Delta t}=\frac{1}{2}\Big(f(y_{n+1})+f(y_n)-\frac{1}{12}\big(f(y_{n+1})-f(y_n)-f(y_{n-1})+f(y_{n-2})\big)\Big). \tag{HH}$$

Show that the method (HH) *is third-order consistent.*

$$\begin{aligned}\Delta t\,\tau_n^{(\mathrm{HH})}&=y(t_{n+1})-y(t_n)-\frac{\Delta t}{2}\Big(f(y(t_{n+1}))+f(y(t_n))-\frac{f(y(t_{n+1}))-f(y(t_n))-f(y(t_{n-1}))+f(y(t_{n-2}))}{12}\Big)\\
&=y(t_{n+1})-y(t_n)-\frac{\Delta t}{2}(f(y(t_{n+1}))+f(y(t_n)))+\frac{\Delta t}{24}\Big(f(y(t_{n+1}))-f(y(t_n))-f(y(t_{n-1}))+f(y(t_{n-2}))\Big)\\
&=y(t_{n+1})\\
&-y(t_n)\\
&-\frac{\Delta t}{2}(f(y(t_{n+1}))+f(y(t_n)))\\
&+\frac{\Delta t}{24}\Big(f(y(t_{n+1}))-f(y(t_n))-f(y(t_{n-1}))+f(y(t_{n-2}))\Big)\\
&=\cancel{y_{n+1/2}}+\frac{\Delta t}{2}y'_{n+1/2}+\cancel{\frac{\Delta t^2}{2!\cdot 2^2}y''_{n+1/2}}+\frac{\Delta t^3}{3!\cdot 2^3}y'''_{n+1/2}+\cancel{\frac{\Delta t^4}{4!\cdot 2^4}y^{\mathrm{iv}}_{n+1/2}}+\frac{\Delta t^5}{5!\cdot 2^5}y^{\mathrm{v}}_{n+1/2}\\
&-\Big(\cancel{y_{n+1/2}}-\frac{\Delta t}{2}y'_{n+1/2}+\cancel{\frac{\Delta t^2}{2!\cdot 2^2}y''_{n+1/2}}-\frac{\Delta t^3}{3!\cdot 2^3}y'''_{n+1/2}+\cancel{\frac{\Delta t^4}{4!\cdot 2^4}y^{\mathrm{iv}}_{n+1/2}}-\frac{\Delta t^5}{5!\cdot 2^5}y^{\mathrm{v}}_{n+1/2}\Big)\\
&-\frac{\Delta t}{2}\Big(f_{n+1/2}+\cancel{\frac{\Delta t}{2}f'_{n+1/2}}+\frac{\Delta t^2}{2!\cdot 2^2}f''_{n+1/2}+\cancel{\frac{\Delta t^3}{3!\cdot 2^3}f'''_{n+1/2}}+\frac{\Delta t^4}{4!\cdot 2^4}f^{\mathrm{iv}}_{n+1/2}\\
&\qquad+f_{n+1/2}-\cancel{\frac{\Delta t}{2}f'_{n+1/2}}+\frac{\Delta t^2}{2!\cdot 2^2}f''_{n+1/2}-\cancel{\frac{\Delta t^3}{3!\cdot 2^3}f'''_{n+1/2}}+\frac{\Delta t^4}{4!\cdot 2^4}f^{\mathrm{iv}}_{n+1/2}\Big)\\
&+\frac{\Delta t}{24}\Big(f(y(t_{n+1}))-f(y(t_n))-f(y(t_{n-1}))+f(y(t_{n-2}))\Big)\\
&=\Delta t y'_{n+1/2}+2\frac{\Delta t^3}{3!\cdot 2^3}y'''_{n+1/2}+2\frac{\Delta t^5}{5!\cdot 2^5}y^{\mathrm{v}}_{n+1/2}\end{aligned}$$

$$
\begin{aligned}
&- \Delta t \Big(f_{n+1/2} + \frac{\Delta t^2}{2! \cdot 2^2} f''_{n+1/2} + \frac{\Delta t^4}{4! \cdot 2^4} f^{\mathrm{iv}}_{n+1/2} \Big) \\
&+ \frac{\Delta t}{24} \Big(f(y(t_{n+1})) - f(y(t_n)) - f(y(t_{n-1})) + f(y(t_{n-2})) \Big) \\
&= \cancel{\Delta t y'_{n+1/2}} + 2 \frac{\Delta t^3}{3! \cdot 2^3} y'''_{n+1/2} + 2 \frac{\Delta t^5}{5! \cdot 2^5} y^{\mathrm{v}}_{n+1/2} - \Delta t \Big(\cancel{f_{n+1/2}} + \frac{\Delta t^2}{2! \cdot 2^2} f''_{n+1/2} + \frac{\Delta t^4}{4! \cdot 2^4} f^{\mathrm{iv}}_{n+1/2} \Big) \\
&+ \frac{\Delta t}{24} \Big(f(y(t_{n+1})) - f(y(t_n)) - f(y(t_{n-1})) + f(y(t_{n-2})) \Big) \\
&= \frac{\Delta t^3}{3! \cdot 2^2} y'''_{n+1/2} + \frac{\Delta t^5}{5! \cdot 2^4} y^{\mathrm{v}}_{n+1/2} - \frac{\Delta t^3}{2! \cdot 2^2} f''_{n+1/2} - \frac{\Delta t^5}{4! \cdot 2^4} f^{\mathrm{iv}}_{n+1/2} \\
&+ \frac{\Delta t}{24} \Big(f(y(t_{n+1})) - f(y(t_n)) - f(y(t_{n-1})) + f(y(t_{n-2})) \Big) \\
&= \Delta t^3 \Big(\frac{1}{3! \cdot 2^2} - \frac{1}{2! \cdot 2^2} \Big) y'''_{n+1/2} + \Delta t^5 \Big(\underbrace{\frac{1}{5! \cdot 2^4} - \frac{1}{4! \cdot 2^4}}_{= -\frac{4}{5} \frac{1}{4! \cdot 2^4} = -\frac{1}{5} \frac{1}{4! \cdot 2^2} = -\frac{1}{480}} \Big) y^{\mathrm{v}}_{n+1/2} \\
&\qquad + \frac{\Delta t}{24} \Big(f(y(t_{n+1})) - f(y(t_n)) - f(y(t_{n-1})) + f(y(t_{n-2})) \Big) \\
&= -\frac{1}{12} \Delta t^3 y'''_{n+1/2} - \frac{1}{480} \Delta t^5 y^{\mathrm{v}}_{n+1/2} \\
&\qquad + \frac{\Delta t}{24} \Big(\cancel{f_{n+1/2}} + \frac{\Delta t}{2} f'_{n+1/2} + \cancel{\frac{\Delta t^2}{2! \cdot 2^2} f''_{n+1/2}} + \frac{\Delta t^3}{3! \cdot 2^3} f'''_{n+1/2} + \cancel{\frac{\Delta t^4}{4! \cdot 2^4} f^{\mathrm{iv}}_{n+1/2}} \\
&\qquad\qquad - \cancel{f_{n+1/2}} + \frac{\Delta t}{2} f'_{n+1/2} - \cancel{\frac{\Delta t^2}{2! \cdot 2^2} f''_{n+1/2}} + \frac{\Delta t^3}{3! \cdot 2^3} f'''_{n+1/2} - \cancel{\frac{\Delta t^4}{4! \cdot 2^4} f^{\mathrm{iv}}_{n+1/2}} \\
&\qquad\qquad - f_{n+1/2} + 3 \frac{\Delta t}{2} f'_{n+1/2} - \frac{3^2 \Delta t^2}{2! \cdot 2^2} f''_{n+1/2} + \frac{3^3 \Delta t^3}{3! \cdot 2^3} f'''_{n+1/2} - \frac{3^4 \Delta t^4}{4! \cdot 2^4} f^{\mathrm{iv}}_{n+1/2} \\
&\qquad\qquad + f_{n+1/2} - 5 \frac{\Delta t}{2} f'_{n+1/2} + \frac{5^2 \Delta t^2}{2! \cdot 2^2} f''_{n+1/2} - \frac{5^3 \Delta t^3}{3! \cdot 2^3} f'''_{n+1/2} + \frac{5^4 \Delta t^4}{4! \cdot 2^4} f^{\mathrm{iv}}_{n+1/2} \Big) \\
&= -\frac{1}{12} \Delta t^3 y'''_{n+1/2} - \frac{1}{480} \Delta t^5 y^{\mathrm{v}}_{n+1/2} \\
&\qquad + \frac{\Delta t}{24} \Big(\cancel{\Delta t f'_{n+1/2}} + \frac{\Delta t^3}{24} f'''_{n+1/2} \\
&\qquad\qquad + 3 \cancel{\frac{\Delta t}{2} f'_{n+1/2}} - \frac{3^2 \Delta t^2}{2! \cdot 2^2} f''_{n+1/2} + \frac{3^3 \Delta t^3}{3! \cdot 2^3} f'''_{n+1/2} - \frac{3^4 \Delta t^4}{4! \cdot 2^4} f^{\mathrm{iv}}_{n+1/2} \\
&\qquad\qquad - 5 \cancel{\frac{\Delta t}{2} f'_{n+1/2}} + \frac{5^2 \Delta t^2}{2! \cdot 2^2} f''_{n+1/2} - \frac{5^3 \Delta t^3}{3! \cdot 2^3} f'''_{n+1/2} + \frac{5^4 \Delta t^4}{4! \cdot 2^4} f^{\mathrm{iv}}_{n+1/2} \Big) \\
&= -\frac{1}{12} \Delta t^3 y'''_{n+1/2} - \frac{1}{480} \Delta t^5 y^{\mathrm{v}}_{n+1/2} \\
&\qquad + \frac{\Delta t^2}{24} \Big(\frac{\Delta t^2}{24} f'''_{n+1/2} + \frac{3^3 \Delta t^2}{3! \cdot 2^3} f'''_{n+1/2} - \frac{5^3 \Delta t^2}{3! \cdot 2^3} f'''_{n+1/2} \\
&\qquad\qquad - \frac{3^2 \Delta t}{2! \cdot 2^2} f''_{n+1/2} + \frac{5^2 \Delta t}{2! \cdot 2^2} f''_{n+1/2} \\
&\qquad\qquad - \frac{3^4 \Delta t^3}{4! \cdot 2^4} f^{\mathrm{iv}}_{n+1/2} + \frac{5^4 \Delta t^3}{4! \cdot 2^4} f^{\mathrm{iv}}_{n+1/2} \Big) \\
&= -\frac{1}{12} \Delta t^3 y'''_{n+1/2} - \frac{1}{480} \Delta t^5 y^{\mathrm{v}}_{n+1/2}
\end{aligned}
$$

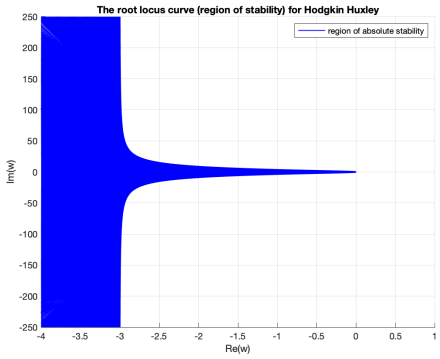


FIGURE 1. Region of absolute stability for the Hodgkin-Huxley method (HH).

$$
\begin{aligned}
&+\frac{\Delta t^2}{24}\Big[\Delta t^2\Big(\underbrace{\frac{1}{24}+\frac{3^3}{3!\cdot 2^3}-\frac{5^3}{3!\cdot 2^3}}_{=-2}\Big)f'''_{n+1/2}\\
&\qquad+\Delta t\Big(\underbrace{-\frac{3^2}{2!\cdot 2^2}+\frac{5^2}{2!\cdot 2^2}}_{=2}\Big)f''_{n+1/2}\\
&\qquad+\Delta t^3\Big(\underbrace{-\frac{3^4}{4!\cdot 2^4}+\frac{5^4}{4!\cdot 2^4}}_{=\frac{32\cdot 17}{16*4*6}=\frac{17}{12}}\Big)f^{\mathrm{iv}}_{n+1/2}\Big]\\
&=-\frac{1}{12}\cancel{\Delta t^3 y'''_{n+1/2}}-\frac{1}{480}\Delta t^5 y^{\mathrm{v}}_{n+1/2}\\
&\quad+\frac{\Delta t^2}{24}\Big[-2\Delta t^2 f'''_{n+1/2}+\cancel{2\Delta t f''_{n+1/2}}+\frac{17}{12}\Delta t^3 f^{\mathrm{iv}}_{n+1/2}\Big]\\
&=-\frac{\Delta t^4}{12}y^{iv}_{n+1/2}
\end{aligned}
$$

(Hint: the local truncation error is $\Delta t\tau_n^{(\mathrm{HH})}=-\frac{1}{12}\Delta t^4 y^{iv}(t_{n+1/2})$.*)*
(Note that the method (HH) is not A-stable - see the region of stability in Figure 1.)

3. Zero-stability

DEFINITION 3.1. *A general 2-step LMM has the form*

$$a_2y_{n+2}+a_1y_{n+1}+a_0y_n=\Delta t\Big(\alpha_2f(y_{n+2})+\alpha_1f(y_{n+1})+\alpha_0f(y_n)\Big). \tag{3.1}$$

The 1st *characteristic polynomial:*

$$\rho(r)=a_2r^2+a_1r+a_0, \tag{3.2}$$

and the 2nd *characteristic polynomial:*

$$\sigma(r)=\alpha_2r^2+\alpha_1r+\alpha_0. \tag{3.3}$$

DEFINITION 3.2 (Stability). *A numerical method is zero-stable if "small" perturbations in the data f, y_0 yield small perturbations in the solutions $\{y_n\}$.*

DEFINITION 3.3 (Root condition). *An equivalent property with the zero-stability of a linear multistep method for ODEs is the root condition, i.e., the roots of the first characteristic polynomial are inside the unit disc, and if they are on the unit circle, they must be simple roots.*

EXAMPLE 3.1 (Zero stability of the Leapfrog method (LF)). *The leapfrog (explicit midpoint) method* (LF)

$$y_{n+2} = y_n + \Delta t \cdot 2f(y_{n+1}), \qquad \text{hence by (3.1) and (3.2):} \quad a_2 = 1, a_1 = 0, a_0 = -1,$$

has the first characteristic polynomial

$$\rho^{\mathrm{LF}}(r) = r^2 - 1,$$

with roots $r_{1,2} = \pm 1$, hence (LF) *is zero-stable (no double roots).*

EXAMPLE 3.2 (Zero stability of the (BDF2) method). *The backward differentiation method* (BDF2)

$$\frac{3}{2}y_{n+2} - 2y_{n+1} + \frac{1}{2}y_n = \Delta t \cdot f(y_{n+2}), \qquad \text{hence by (3.1) and (3.2):} \quad a_2 = \frac{3}{2}, a_1 = -2, a_0 = \frac{1}{2},$$

has the first characteristic polynomial

$$\rho^{\mathrm{BDF2}}(r) = \frac{3}{2}r^2 - 2r + \frac{1}{2},$$

with roots $r_{1,2} = \frac{1}{3}, 1$, hence (BDF2) *is zero-stable.*

EXAMPLE 3.3 (Zero stability of the (Backward Euler) method). *The backward Euler method* (Backward Euler)

$$y_{n+1} - y_n = \Delta t \cdot f(y_{n+1}), \qquad \text{hence by (3.1) and (3.2):} \quad a_2 = 0, a_1 = 1, a_0 = -1,$$

has the first characteristic polynomial

$$\rho^{\mathrm{BE}}(r) = r - 1,$$

with root $r = 1$, hence (Backward Euler) *is zero-stable.*

4. Convergence of a numerical method for ODEs

DEFINITION 4.1 (Convergence). *A numerical method for ODEs is convergent if the sequence of approximating solutions $\{y_n\}_{n\geq 0}$ converges to the exact solution $y(\cdot)$ at the node points $\{t_n\}$, as the mesh size $\Delta t \to 0$, i.e., the error*

$$e_n := y(t_n) - y_n \xrightarrow{\|\cdot\|} 0, \qquad \text{as } \Delta t \to 0. \qquad \text{(global error)}$$

THEOREM 4.1 (Lax-Richtmyer). *A numerical method for ODEs is convergent* ($|y(t_n) - y_n| \xrightarrow{\Delta t \searrow 0} 0$) *if and only if it is consistent and zero-stable (satisfies the root condition).*

COROLLARY 4.1. *The* (Backward Euler), (Forward Euler) *methods are first-order convergent, while* (BDF2), (implicit midpoint) *and* (LF) *leapfrog methods are second-order convergent.*

COUNTEREXAMPLE 4.1 (Dahlquist). *The following method*

$$y_{n+2}+4y_{n+1}-5y_n=\Delta t\big(4f(y_{n+1})+2f(y_n)\big), \tag{4.1}$$

is a famous example of a method which is third-order consistent, but not zero-stable , and therefore not-convergent , according to the Lax-Richtmyer meta-theorem 4.1.

PROOF. We will address separately the zero-stability, convergence and consistency of the method.

- Zero-stability : from the definition of the first characteristic polynomial (3.2) we see that $a_2=1, a_1=4, a_0=-5$, hence $\rho(r)=r^2+4r-5=(r-1)(r+5)$. Since one of the roots $r=-5$ is outside of the unit disc, according to the root-condition (equivalent Definition 3.3), the method (4.1) is not zero-stable.
- Non-convergence : to illustrate the non-convergence of the method (4.1), it suffices to consider the simpler case, namely when $f(y)\equiv 0$, i.e., the ODE is $y'=0$, and (4.1) writes $y_{n+2}+4y_{n+1}-5y_n=0$. By a technique called *difference equations*, it is easy to see that the solutions to $y_{n+2}+4y_{n+1}-5y_n=0$ have the general form (a linear combination of powers of the roots of the first characteristic polynomial, $r_1=1, r_2=-5$):

  $$y_n=A\cdot 1^n+B\cdot(-5)^n.$$

  Here $A,B\in\mathbb{R}$ are constants depending on the initial conditions. With $y_0=1, y_1=1+\Delta t$, the general solution is $(A=1-\frac{\Delta t}{6}, B=-\frac{\Delta t}{6})$:

  $$y_n=1+\frac{1}{6}\Delta t\Big(1-(-5)^n\Big).$$

  For the interval $[t_0,T]\equiv[0,1]$, $\Delta t=1/N$ we have

  $$y_N=1+\frac{1}{6}\frac{1}{N}\Big(1-(-5)^N\Big)\xrightarrow{\Delta t\searrow 0}\pm\infty$$

  when $\Delta t\searrow 0$, equivalently $N\nearrow\infty$. Notice that the exact solution in this case is $y(t)=1$ (since $y'(t)=0$ implies $y(t)=$constant, and since $y_0=1$, the constant is $1=y(t)$). Therefore the method (4.1) is not convergent !!! indeed.
- Consistency : Nonetheless, the method (4.1) is consistent, with the maximum order of consistency for a two step Linear Multistep Method, third-order of consistency. Indeed, we shall see now that the order of consistency is 3 , larger than any of the methods we seen so far.
  Using the definition of the Local Truncation Error, and Taylor expansions of $y(t)$ and $f(y(t))$, evaluated at t_n, t_{n+2} about t_{n+1} we have

$$\begin{aligned}
&\Delta t\,\tau_{n+1}^{\text{Dahlquist}}(\Delta t):=y(t_{n+2})+4y(t_{n+1})-5y(t_n)-\Delta t\big(4f(y(t_{n+1}))+2f(y(t_n))\big)\\
&=\Big(\cancel{y(t_{n+1})}+\Delta t y'(t_{n+1})+\frac{\Delta t^2}{2}y''(t_{n+1})+\frac{\Delta t^3}{3!}y'''(t_{n+1})+\mathcal{O}(\Delta t^4)\Big)\\
&\quad+\cancel{4y(t_{n+1})}\\
&\quad-5\Big(\cancel{y(t_{n+1})}-\Delta t y'(t_{n+1})+\frac{\Delta t^2}{2}y''(t_{n+1})-\frac{\Delta t^3}{3!}y'''(t_{n+1})+\mathcal{O}(\Delta t^4)\Big)\\
&\quad-\Delta t\big(4f(y(t_{n+1}))+2f(y(t_n))\big)\\
&=6\Delta t y'(t_{n+1})-2\Delta t^2y''(t_{n+1})+\Delta t^3y'''(t_{n+1})-\Delta t\big(4f(y(t_{n+1}))+2f(y(t_n))\big)+\mathcal{O}(\Delta t^4)\\
&=6\Delta t y'(t_{n+1})-2\Delta t^2y''(t_{n+1})+\Delta t^3y'''(t_{n+1})
\end{aligned}$$

$$
\begin{aligned}
&- \Delta t \Big[4f(y(t_{n+1})) + 2f\big(\underbrace{y(t_{n+1}) - \Delta t y'(t_{n+1}) + \frac{\Delta t^2}{2} y''(t_{n+1}) + \mathscr{O}(\Delta t^3)}_{= y(t_n) \text{ and use Taylor to expand } f(\cdot) \text{ about } f(y(t_{n+1}))} \big) \Big] + \mathscr{O}(\Delta t^4) \\
&= 6\Delta t y'(t_{n+1}) - 2\Delta t^2 y''(t_{n+1}) + \Delta t^3 y'''(t_{n+1}) \\
&\quad - \Delta t \Big\{ 4f(y(t_{n+1})) + 2\Big[f(y(t_{n+1})) \\
&\qquad + f'(y(t_{n+1}))\big(-\Delta t y'(t_{n+1}) + \frac{\Delta t^2}{2} y''(t_{n+1}) \big) \\
&\qquad + \frac{1}{2!} f''(y(t_{n+1}))\Big| -\Delta t y'(t_{n+1}) + \frac{\Delta t^2}{2} y''(t_{n+1}) \Big|^2 + \mathscr{O}(\Delta t^3) \Big] \Big\} + \mathscr{O}(\Delta t^4) \\
&= 6\Delta t y'(t_{n+1}) - 2\Delta t^2 y''(t_{n+1}) + \Delta t^3 y'''(t_{n+1}) \\
&\quad - 6\Delta t f(y(t_{n+1})) - 2\Delta t f'(y(t_{n+1}))\Big[-\Delta t y'(t_{n+1}) + \frac{\Delta t^2}{2} y''(t_{n+1}) \Big] - \Delta t f''(y(t_{n+1}))\Big| -\Delta t y'(t_{n+1}) + \frac{\Delta t^2}{2} y''(t_{n+1}) \Big|^2 + \mathscr{O}(\Delta t^4) \\
&= 6\Delta t y'(t_{n+1}) - 6\Delta t f(y(t_{n+1})) - 2\Delta t^2 y''(t_{n+1}) + \Delta t^3 y'''(t_{n+1}) \\
&\quad + 2\Delta t^2 f'(y(t_{n+1})) y'(t_{n+1}) - \Delta t^3 f'(y(t_{n+1})) y''(t_{n+1}) - \Delta t^3 f''(y(t_{n+1}))\big| y'(t_{n+1}) \big|^2 + \mathscr{O}(\Delta t^4).
\end{aligned}
$$

Finally, using the exact equation (IVP)

$$y'(t_{n+1}) = f(y(t_{n+1})),$$

also differentiating with respect to t and using the chain rule

$$
\begin{aligned}
y''(t_{n+1}) &= f'(y(t_{n+1})) \cdot y'(t_{n+1}), \\
y'''(t_{n+1}) &= f''(y(t_{n+1})) \cdot |y'(t_{n+1})|^2 + f'(y(t_{n+1})) \cdot y''(t_{n+1}),
\end{aligned}
$$

we obtain that the local truncation error writes as

$$
\begin{aligned}
\Delta t \tau_{n+1}^{\text{Dahlquist}}(\Delta t) &= \cancel{6\Delta t y'(t_{n+1})} - \cancel{6\Delta t f(y(t_{n+1}))} - 2\Delta t^2 \cancel{y''(t_{n+1})} + \Delta t^3 y'''(t_{n+1}) \\
&\quad + 2\Delta t^2 \cancel{f'(y(t_{n+1}))y'(t_{n+1})} - \Delta t^3 f'(y(t_{n+1})) y''(t_{n+1}) - \Delta t^3 f''(y(t_{n+1}))\big| y'(t_{n+1}) \big|^2 + \mathscr{O}(\Delta t^4) \\
&= \Delta t^3 \Big(\underbrace{y'''(t_{n+1}) - f'(y(t_{n+1})) y''(t_{n+1}) - f''(y(t_{n+1}))\big| y'(t_{n+1}) \big|^2}_{\equiv 0} \Big) + \mathscr{O}(\Delta t^4) = \mathscr{O}(\Delta t^4).
\end{aligned}
$$

Therefore $\Delta t \tau_{n+1}^{\text{Dahlquist}}(\Delta t) = \mathscr{O}(\Delta t^4)$ and $p = 3$ giving the third-order of consistency ,

which concludes the argument. □

5. Absolute stability

The previous analysis, concerning the zero-stability, consistency and convergence of a numerical method for ODEs was describing the asymptotic behaviour of a numerical method when the mesh-size (or the time-step) $\Delta t \longrightarrow 0$. Equivalently this can be thought of, on a fixed time interval $[t_0, T]$ and $\Delta t = \frac{T - t_0}{n}$, as the behavior for when the number of time-steps $n \longrightarrow \infty$.

In this paragraph we will consider the behavior of a numerical method for the Initial Value Problem (IVP)

$$
\begin{cases} y' &= f(t, y(t)), \qquad t \in (0, T] \\ y(t_0) &= y_0. \end{cases} \qquad \text{for } T \to \infty
$$

when the length of the interval is growing to ∞, with fixed timestep.

Namely let Δt be fixed , the mesh points defined as before $t_i = t_0 + i\Delta t, t_n = t_0 + n\Delta t \xrightarrow{n \nearrow \infty} \infty$ and study the approximate solution y_n for $n \to \infty$.

The paradigm for this stability notion is the following equation.

DEFINITION 5.1 (Dahlquist test). *Let $\lambda = Re(\lambda) + iIm(\lambda) \in \mathbb{C}$, with $Re(\lambda) \leq 0$. The Dahlquist test is the following linear ODE*

$$u'(t) = \lambda u(t), \quad \text{on } (0,\infty) \qquad \text{(Dahlquist Test)}$$

usually endowed with the initial condition $u(0) = 1$.

We note that (Dahlquist Test) *has the exact solution $u(t) = e^{Re(\lambda)t}\big(\cos(Im(\lambda t)) + i\sin(Im(\lambda t))\big)u(0) \xrightarrow{t \nearrow \infty} 0$, which decays fast to zero as t grows, since $Re(\lambda) < 0$.*

Therefore the goal is to find numerical methods which, when tested on the (Dahlquist Test), yield numerical solutions also decaying to zero.

DEFINITION 5.2 (Region of absolute stability). *The region of absolute stability is the region in the complex plane*

$$\mathscr{C} = \{z = \lambda\Delta t \in \mathbb{C} \quad | \quad \text{such that } u_n(\lambda\Delta t) \to 0 \quad \text{as } n \to \infty\}, \qquad \text{(Region of Absolute Stability)}$$

where $u_n(\lambda\Delta t)$ is the numerical solution to the (Dahlquist Test).

DEFINITION 5.3 (A-stable). *A numerical method is A-stable if the region of absolute stability contains the whole left half plane*

$$\{z \in \mathbb{C} | Re(z) < 0\} \subset \mathscr{C}.$$

EXAMPLE 5.1 (Conditional stability of the (Forward Euler) method).
When the (Forward Euler) *method is applied to the* (Dahlquist Test), *the numerical solu-*

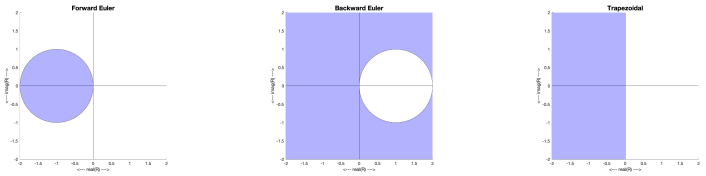


FIGURE 2. Stability region for the (Forward Euler), (Backward Euler) and (implicit midpoint)-(trapezoidal) methods.

tion writes

$$u_{n+1} = u_n + \lambda\Delta t u_n = (1 + \lambda\Delta t)u_n = \cdots = (1 + \lambda\Delta t)^{n+1}u_0$$

and therefore $u_n \xrightarrow{n \nearrow \infty} 0$ if and only $|1 + \lambda\Delta t| < 1$. This implies that the region of stability of the (Forward Euler) *method is the unit disc, centered at $(-1,0)$, contained in the left half of the complex plane:*

$$\mathscr{C}^{\text{FE}} = \{z \in \mathbb{C} | |z+1| < 0\}.$$

This means that the (Forward Euler) *method is not A-stable, only 'conditionally stable', as for large (amplitude) values of λ ("stiff problems"), the time step has to be considerably small. For example, if $\lambda \in \mathbb{R}_-$, the solution $u_n^{\text{FE}} \xrightarrow{n \nearrow \infty} 0$ provided*

$$-2 < \lambda\Delta t < 0,$$

i.e., the time step has to satisfy

$$\Delta t < \frac{2}{-\lambda}.$$

EXAMPLE 5.2 (The (Backward Euler) method is A-stable). *When the* (Backward Euler) *method is applied to the* (Dahlquist Test)*, the numerical solution writes*

$$u_{n+1} = u_n + \lambda \Delta t u_{n+1}$$

equivalently,

$$u_{n+1} = \frac{u_n}{1-\lambda\Delta t} = \cdots = \left(\frac{1}{1-\lambda\Delta t}\right)^{n+1} u_0$$

and therefore

$$u_n \xrightarrow{n \nearrow \infty} 0 \qquad \forall \lambda \Delta t \quad \text{such that } Re(\lambda) < 0, \Delta t > 0.$$

This also implies that the region of stability of the (Backward Euler) *method is the exterior of unit disc centered at* $(1,0)$*, contained in the right half of the complex plane. Therefore*

$$\mathscr{C}^{\text{BE}} = \{z \in \mathbb{C} || z-1| > 0\},$$

contains the whole left half of the complex plane. This means that the (Backward Euler) *method is A-stable, suitable for stiff problems (for large amplitude values of* λ*), for any values of the time step* $\Delta t > 0$*.*

EXAMPLE 5.3 (The (implicit midpoint) method is A-stable). *When the* (implicit midpoint) *or the* (trapezoidal) *method is applied to the* (Dahlquist Test)*, the numerical solution writes*

$$u_{n+1} = u_n + \lambda \Delta t \frac{u_n + u_{n+1}}{2}$$

equivalently,

$$u_{n+1}\left(1 - \frac{\lambda\Delta t}{2}\right) = u_n\left(1 + \frac{\lambda\Delta t}{2}\right)$$

and therefore

$$u_{n+1} = \left(\frac{1+\lambda\Delta t}{1-\lambda\Delta t}\right)^{n+1} u_0 \xrightarrow{n \nearrow \infty} 0 \qquad \forall \lambda \Delta t \quad \text{such that } Re(\lambda) < 0, \Delta t > 0.$$

This also implies that the region of stability of the (implicit midpoint) *method is exactly the left half of the complex plane:*

$$\mathscr{C}^{(\text{implicit midpoint})} = \{z \in \mathbb{C} | Re(z) < 0\},$$

i.e., the (implicit midpoint) *method is A-stable, suitable for stiff problems, for any values of the time step* $\Delta t > 0$*.*

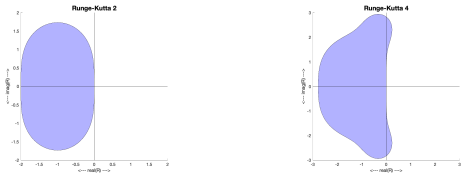


FIGURE 3. Locus curve/stability region for the explicit RK2 and (RK4) methods.

6. Hamiltonian system, conserved quantities

DEFINITION 6.1 (Hamiltonian system). *A Hamiltonian system' is dynamical system' in unknowns y, completely described by a Hamiltonian, a scalar function $H(y)$.*
In particular, consider a system of ODE's with unknowns $(u(t), v(t))$ which is described by the evolution equations

$$\begin{cases} u' = \dfrac{\partial H(u,v)}{\partial v}, \\ v' = -\dfrac{\partial H(u,v)}{\partial u}, \end{cases} \qquad \text{(Hamiltonian system)}$$

with

$$H(u,v) \qquad \text{(Hamiltonian)}$$

being the Hamiltonian.

EXAMPLE 6.1 (Harmonic oscillator, simple pendulum). *Consider the following second-order ODE*

$$u'' + u = 0, \qquad \text{(vibrating spring)}$$

which describes the motion of a vibrating spring, in absence of friction. (Also known as a harmonic oscillator equation). First, as mentioned in Remark 0.1, the vibrating spring equation is also known in system form

$$\begin{cases} u' = v, \\ v' = -u, \end{cases} \qquad \text{(harmonic equation)}$$

or

$$\begin{pmatrix} u \\ v \end{pmatrix}' = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}, \tag{6.1}$$

equivalently

$$\mathbf{u}' = A\mathbf{u}, \quad \textit{where} \quad \mathbf{u} = \begin{pmatrix} u \\ v \end{pmatrix}, \quad A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \tag{6.2}$$

Note that $u(t)$ represents the position (in the case of the simple pendulum, or the displacement in the case of the vibrating spring), and $v(t)$ is the velocity.
Recall that the behavior of equation (6.2) *can be described, in terms of absolute stability by analyzing the* (Dahlquist Test)*, and noticing that the eigenvalues of the (skew-symmetric) matrix A are purely imaginary*

$$\operatorname{eig}(A) = \pm i.$$

In this case, the (Dahlquist Test) *writes*

$$w' = i\omega w, \qquad \text{(Oscillation Equation)}$$

where $\omega \in \mathbb{R}$ represents the frequency of the system, and its exact solution is $w(t) = \cos(\omega t) + i\sin(\omega t)$.

We also note that the exact solution to the (harmonic equation) *is*

$$u(t) = \sin(t), \quad v(t) = \cos(t), \tag{6.3}$$

its Hamiltonian is

$$H(u,v) = \frac{1}{2}u^2 + \frac{1}{2}v^2, \qquad \text{(Hamiltonian of the Harmonic Equation)}$$

and therefore the Hamiltonian represents the total energy of the system, i.e., the sum of its potential energy $\frac{1}{2}u^2(t)$ *and its kinetic energy* $\frac{1}{2}v^2(t)$.
Moreover, since $\sin^2(t)+\cos^2(t)=1$, *the Hamiltonian (the total energy) of the* (harmonic equation)

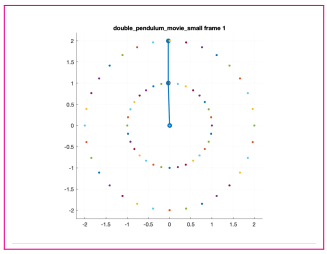


FIGURE 4. The Double pendulum, also a (chaotic) Hamiltonian system. Here is the link for instructions on how to make the movie. See also Figure 5.

is conserved (constant with respect to time):

$$H(u,v)(t)=\frac{1}{2}u^2(t)+\frac{1}{2}v^2(t)\equiv\frac{1}{2}u^2(0)+\frac{1}{2}v^2(0)\equiv\frac{1}{2}. \tag{6.4}$$

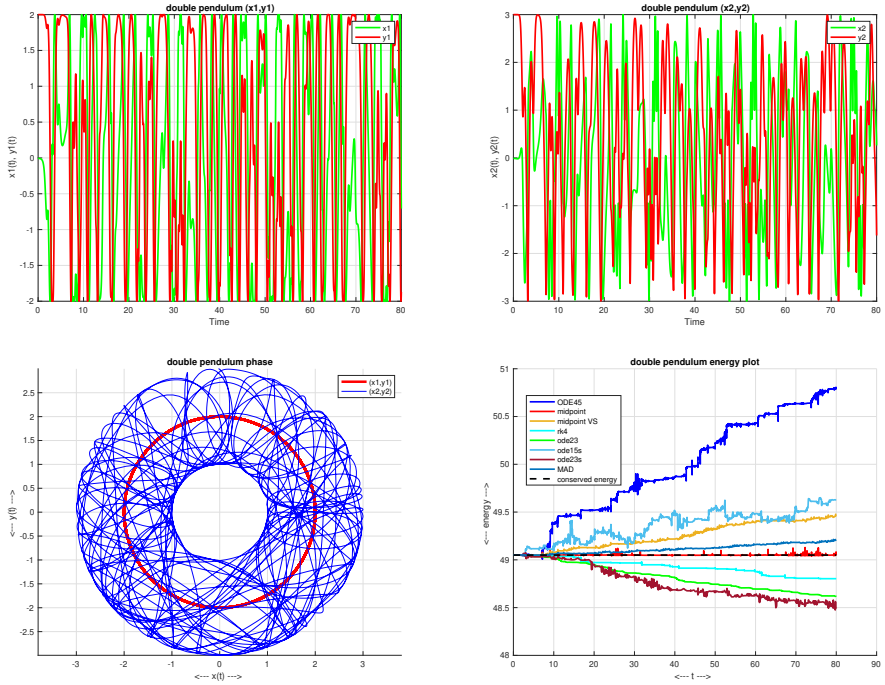


FIGURE 5. The double pendulum: x,y - the positions of the endpoints versus time, the phase plot and the Hamiltonian.

From the (Hamiltonian of the Harmonic Equation)*, since* $\frac{\partial H}{\partial v} = v$, $\frac{\partial H}{\partial u} = u$*, we see that the* (harmonic equation) *is a particular instance of the* (Hamiltonian system)*.*

Multiplying the first equation in (Hamiltonian system) *by* v' *and the second by* $-u'$*, and adding we have*

$$0 = u'v' - v'u' = \frac{\partial H(u,v)}{\partial v}v' + \frac{\partial H(u,v)}{\partial u}u' \equiv \frac{d}{dt}H(u(t),v(t)),$$

which means that $H(u(t),v(t))$ *is constant in time, i.e., the* (Hamiltonian) *is a conserved quantity throughout the evolution of the system.*

EXAMPLE 6.2 (`Lotka-Volterra`). *In the case of the* (Lotka-Volterra) *equations*

$$\begin{cases} u' &= u(\alpha - \beta v) \\ v' &= v(\delta u - \gamma), \end{cases} \qquad \text{(Lotka-Volterra)}$$

the Hamiltonian is (a non-quadratical functional):

$$H(u,v) = \delta u - \gamma \ln(u) + \beta v - \alpha \ln(v). \qquad \text{(Hamiltonian of Lotka-Volterra)}$$

First we note that the (Hamiltonian of Lotka-Volterra) *is a conserved quantity throughout the evolution of the* (Lotka-Volterra) *system:*

$$\frac{d}{dt}H(u,v) = \frac{\partial H}{\partial u}\cdot u' + \frac{\partial H}{\partial v}\cdot v' = \left(\delta - \gamma\frac{1}{u}\right)\cdot u(\alpha - \beta v) + \left(\beta - \alpha\frac{1}{v}\right)\cdot v(\delta u - \gamma)$$
$$= \delta u(\alpha - \beta v) - \gamma(\alpha - \beta v) + \beta v(\delta u - \gamma) - \alpha(\delta u - \gamma) = 0.$$

Secondly, we also note that (Lotka-Volterra) *system can be written in the canonical Hamiltonian form* (Hamiltonian system) *by introducing the canonical coordinates*

$$U = \ln(u), \quad V = \ln(v),$$

and transforming the (Hamiltonian of Lotka-Volterra) *into*

$$\mathscr{H}(V,U) = \beta e^V - \alpha V + \delta e^U - \gamma U. \tag{6.5}$$

Indeed,

$$\begin{cases} \dfrac{dV}{dt} = \dfrac{1}{v}\dfrac{dv}{dt} = \delta u - \gamma \equiv \delta e^U - \gamma = \dfrac{\partial \mathscr{H}}{\partial U}, \\ \dfrac{dU}{dt} = \dfrac{1}{u}\dfrac{du}{dt} = \alpha - \beta v \equiv \alpha - \beta e^V = -\dfrac{\partial \mathscr{H}}{\partial V}. \end{cases}$$

(See predator_midpointVS.m) Figure 6 shows the evolution of u, v *versus time, the phase plot and the Hamiltonian function, for the exact solution - in comparison with the solution given by the* (implicit midpoint)*, adaptive-midpoint methods and Matlab's ode45, ode23s, ode15s solvers.*

The conservation of the Hamiltonian is an important property, which ideally a numerical method for approximating a Hamiltonian system should also preserve.

EXAMPLE 6.3 (The midpoint method for the simple pendulum). *The* (implicit midpoint) *method for the* (harmonic equation) *writes*

$$\begin{cases} u_{n+1} = u_n + \Delta t\dfrac{v_n + v_{n+1}}{2}, \\ v_{n+1} = v_n - \Delta t\dfrac{u_n + u_{n+1}}{2}, \end{cases} \qquad \text{(midpoint method for the simple pendulum)}$$

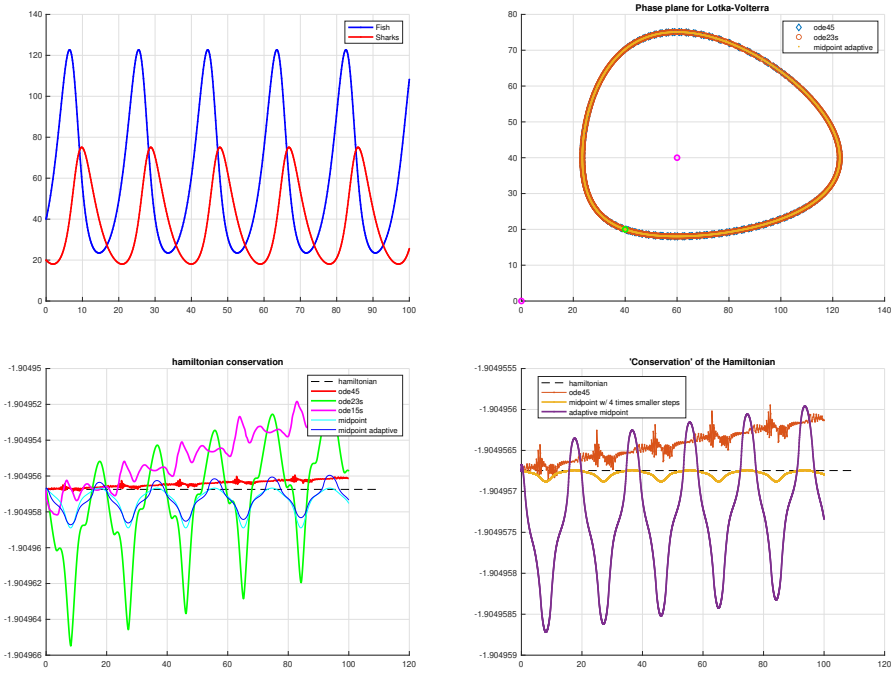


FIGURE 6. Predator-prey: u, v components versus time, the phase plot and the Hamiltonian.

equivalently,

$$\begin{pmatrix} 1 & -\Delta t/2 \\ \Delta t/2 & 1 \end{pmatrix} \begin{pmatrix} u_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} u_n + \Delta t/2 \\ v_n - \Delta t/2 \end{pmatrix},$$

hence the approximating solution at t_{n+1} is obtained by solving a linear system $\mathscr{A}\mathbf{w}_{n+1} = \mathbf{w}_n$, where the matrix

$$\mathscr{A} = \begin{pmatrix} 1 & -\Delta t/2 \\ \Delta t/2 & 1 \end{pmatrix}$$

is skew-symmetric.

Let us now define the (discrete) Hamiltonian function (Hamiltonian of the Harmonic Equation) *at the discrete time levels by*

$$H(u_n, v_n) = \frac{1}{2}u_n^2 + \frac{1}{2}v_n^2.$$

We notice that the (midpoint method for the simple pendulum) *writes as*

$$\begin{cases} u_{n+1} = u_n + \Delta t \dfrac{\partial H}{\partial v}\left(\dfrac{u_n + u_{n+1}}{2}, \dfrac{v_n + v_{n+1}}{2}\right), \\ v_{n+1} = v_n - \Delta t \dfrac{\partial H}{\partial u}\left(\dfrac{u_n + u_{n+1}}{2}, \dfrac{v_n + v_{n+1}}{2}\right). \end{cases}$$

Most importantly, the (discrete) Hamiltonian is also a conserved quantity:

$$H(u_{n+1}, v_{n+1}) = H(u_n, v_n),$$

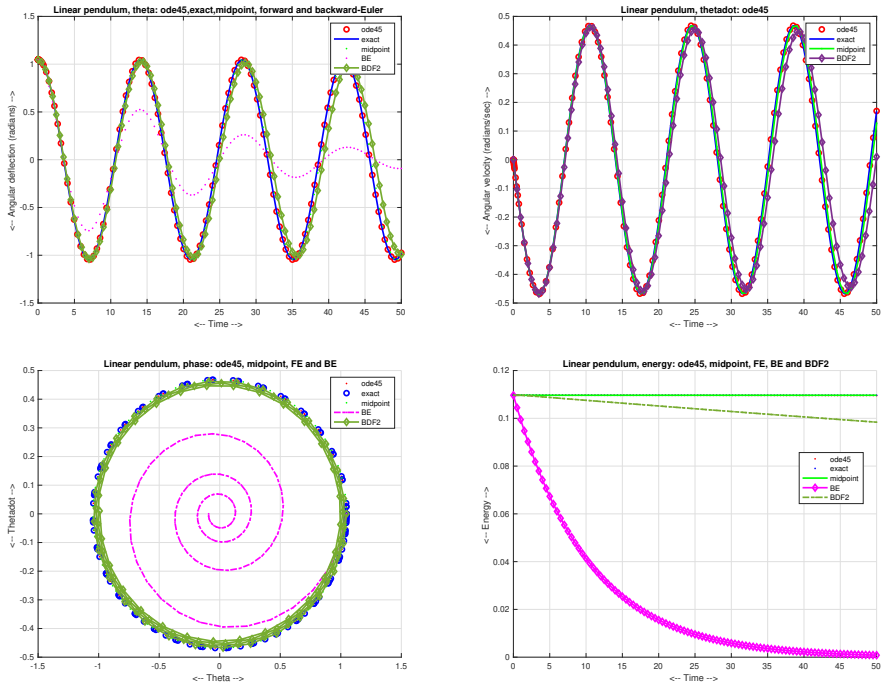


FIGURE 7. Simple pendulum: u, v components versus time, the phase plot and the Hamiltonian.

(similarly to the conservation relation (6.4)*).*
To see this, we multiply the first equation in (midpoint method for the simple pendulum) *by* $\frac{u_n+u_{n+1}}{2}$*, the second equation by* $\frac{v_n+v_{n+1}}{2}$ *and add to obtain*

$$\frac{u_{n+1}^2-u_n^2}{2}+\frac{v_{n+1}^2-v_n^2}{2}=0,$$

which is exactly $H(u_{n+1},v_{n+1})-H(u_n,v_n)=0$.

REMARK 6.1. *Unlike the* (midpoint method for the simple pendulum)*, neither the* (Forward Euler) *method, nor the* (Backward Euler) *method, do not conserve the Hamiltonian function.*
(See pendulum_ode_test.m) Figure 7 shows the evolution of u, v *versus time, the phase plot and the Hamiltonian function, for the exact solution - in comparison with the solution given by the* (implicit midpoint)*,* (Backward Euler)*,* (BDF2) *methods and Matlab's ode45 solver.*

Another remarkable example of a conservative method is the following 'symplectic' method, which is an example of a geometric integration method.

EXAMPLE 6.4 (Symplectic Euler method for the simple pendulum). *The Symplectic Euler method for the* (vibrating spring) *is a sequential application of the* (Forward Euler)

and (Backward Euler) *methods:*

$$\begin{cases} u_{n+1} = u_n + \Delta t v_n \equiv u_n + \Delta t \dfrac{\partial H}{\partial v}(u_{n+1}, v_n), \\ v_{n+1} = v_n - \Delta t u_{n+1} \equiv v_n - \Delta t \dfrac{\partial H}{\partial u}(u_{n+1}, v_n). \end{cases}$$

(Symplectic Euler for the simple pendulum)

Proceeding as in the previous example, let us multiply the first equation by u_{n+1}, the second by v_n and add to obtain

$$u_{n+1}^2 + v_{n+1}v_n = \Big(u_n u_{n+1} + \cancel{\Delta t v_n u_{n+1}}\Big) + \Big(v_n^2 - \cancel{\Delta t u_{n+1} v_n}\Big) = u_n u_{n+1} + v_n^2.$$

Since for the second equation in (Symplectic Euler for the simple pendulum) *we have that $v_n = v_{n+1} + \Delta t u_{n+1}$, substituting this in the left hand side of above equality, and substituting u_{n+1} in the right hand side, we get*

$$u_{n+1}^2 + v_{n+1}\Big(v_{n+1} + \Delta t u_{n+1}\Big) = u_n\Big(u_n + \Delta t v_n\Big) + v_n^2.$$

This means that the symplectic Euler method conserves a 'modified Hamiltonian' quantity

$$\mathscr{H}(u_n, v_n) = \frac{1}{2}\Big(u_n^2 + \Delta t u_n v_n + v_n^2\Big).$$

7. Taylor series method

These methods are mostly used in Stochastic Differential Equations (SDEs).

They are derived from taking various order Taylor expansions of the exaction solution $y(t_{n+1})$ to the Initial Value Problem (IVP)

$$y(t_{n+1}) = y(t_n) + \Delta t y'(t_n) + \frac{\Delta t^2}{2} y''(t_n) + \cdots$$

and use the chain-rule for $\frac{d^m}{dt^m} f(t, y(t))$:

$$y_{n+1} = y_n + \Delta t f(t_n, y_n), \qquad \text{(First-order)}$$

$$y_{n+1} = y_n + \Delta t f(t_n, y_n) + \frac{\Delta t^2}{2}\frac{d}{dt} f(t, y(t))\Big|_{t=t_n}, \qquad \text{(Second order)}$$

$$\vdots$$

$$y_{n+1} = y_n + \Delta t f(t_n, y_n) + \frac{\Delta t^2}{2}\frac{d}{dt} f(t, y(t))\Big|_{t=t_n} + \ldots, \qquad \text{(higher order)}$$

EXAMPLE 7.1 (Exercise 11, page 455). *(a)* $x' = x + e^x, \qquad n = 4.$
(b) $x' = x^2 - \cos(x), \qquad n = 5.$

PROOF. (a)

$$x_{n+1} = x_n + \Delta t x_n' + \frac{\Delta t^2}{2} x_n'' + \frac{\Delta t^3}{3!} x_n''' + \frac{\Delta t^4}{4!} x_n^{(iv)},$$

where

$$x_n'' = \frac{d}{dt}\Big(x(t) + e^{x(t)}\Big)\Big|_{t_n} = \Big(x'(t) + x'(t)e^{x(t)}\Big)\Big|_{t_n} = x_n' + x_n' e^{x_n}$$
$$x_n''' = x_n'' + x_n'' e^{x_n} + (x_n')^2 e^{x_n}$$
$$x_n^{(iv)} = x_n'''(1 + e^{x_n}) + 3x_n' x_n'' e^{x_n} + (x_n')^3 e^{x_n}$$

(b)

$$x_{n+1} = x_n + \Delta t x_n' + \frac{\Delta t^2}{2} x_n'' + \frac{\Delta t^3}{3!} x_n''' + \frac{\Delta t^4}{4!} x_n^{(iv)} + \frac{\Delta t^5}{5!} x_n^{(v)},$$

where

$x'' = 2xx' + x'\sin(x)$

$x''' = (2+\cos(x))(x')^2 + (2x+\sin(x))x''$

$x^{(iv)} = -\sin(x)(x')^3 + 3x'x''(2+\cos(x)) + (2x+\sin(x))x'''$

$x^{(v)} = -\cos(x)(x')^4 - 6(x')^2x''\sin(x) + 3(2+\cos(x))(x'')^2 + 4x'x'''(2+\cos(x)) + (2x+\sin(x))x^{(iv)}.$

□

Exercise 1. *Give the solutions of these differential equations:*

a. $x' = t^3 + 7t^2 - t^{1/2}$
b. $x' = x$
c. $x' = -x$
d. $x'' = -x$
e. $x'' = x$
f. $x'' + 2x' - 2x = 0$ *(Hint: Try* $x = e^{at}$*).*

Solution:

Exercise 2. *Give the solutions of these initial-value problems:*

a. $x' = t^2 + t^{1/3} \quad x(0) = 7$
b. $x' = 2x \quad x(0) = 15$
c. $x'' = -x \quad x(\pi) = 0, x'(\pi) = 3$

Solution:

Exercise 3. *Solve the following differential equations:*

a. $x' = 1 + x^2$ *Hint:* $1 + \frac{\sin^2(t)}{\cos^2(t)} = \frac{1}{\cos^2(t)}$
b. $x' = \sqrt{1-x^2}$ *Hint:* $\sin^2(t) + \cos^2(t) = 1$
c. $x' = \frac{\sin t}{t}$
d. $x' + tx = t^2$ *Hint: Multiply the equation by* $f(t) = e^{t^2/2}$*. The left-hand side becomes* $(xf)'$.

Solution:

a. Equivalently $\frac{x'}{1+x^2} = 1, \quad \big(\arctan(x(t))\big)' = 1, \quad \arctan\big(x(t)\big) = t + C, \quad x(t) = \tan(t + C)$.

b. Equivalently $\frac{x'}{\sqrt{1-x^2}} = 1, \quad \big(\arcsin((x(t))\big)' = 1, \quad \arcsin\big(x(t)\big) = t + C, \quad x(t) = \sin(t + C)$.

Exercise 5. *Determine* x'' *when* $x' = xt^2 + x^3 + e^x t$.

Solution:

Exercise 6. *Find a polynomial* p *with the property* $p - p' = t^3 + t^2 - 2t$.

Solution:

Exercise 8. *Here is an initial-value problem that has two solutions:* $x' = x^{1/3}, x(0) = 0$. *Verify that the two solutions are* $x_1(t) = 0$ *and* $x_2(t) = \left(\frac{2}{3}t\right)^{3/2}$ *for* $t \geq 0$. *If the Taylor*

series method is applied, what happens?
Solution:

Exercise 9. *Consider the more general problem $x' = Lx + \varepsilon$, which has the exact solution*

$$x(t) = e^{Lt}x(0) + \frac{\varepsilon}{L}(e^{Lt} - 1).$$

For $L = 2, t = 10$ the values are $e^{Lt} \approx 4.8 * 10^8$ and $\frac{e^{Lt}-1}{L} \approx 2.42 * 10^8$, while for $L = 2, t = 20$ the values are $e^{Lt} \approx 2.35 * 10^{17}$ and $\frac{e^{Lt}-1}{L} \approx 1.17 * 10^{17}$.

Exercise 10. *If the Taylor series method is used on the initial-value problem $x' = t^2 + x^3$, $x(0) = 0$, and if we intend to use the derivatives of x up to and including $x^{(4)}$, what are the five main equations that must be programmed?*
Solution:

$$x' = t^2 + x^3$$
$$x'' = 2t + 3x^2x'$$
$$x''' = 2 + 6x(x')^2 + 3x^2x''$$
$$x^{(4)} = 6(x')^3 + 18xx'x'' + 3x^2x'''$$
$$x(t + \Delta t) = x + \frac{x'}{1!}\Delta t + \frac{x''}{2!}(\Delta t)^2 + \frac{x'''}{3!}(\Delta t)^3 + \frac{x^{(4)}}{4!}(\Delta t)^4$$

Exercise 12. *Calculate an approximate value for $x(0.1)$ using one step of the Taylor series method of order 3 on the ordinary differential equation*

$$x'' = x^2e^t + x'$$
$$x(0) = 1, \quad x'(0) = 2$$

Solution:

$$x(t + \Delta t) = x + \frac{x'}{1!}\Delta t + \frac{x''}{2!}(\Delta t)^2 + \frac{x'''}{3!}(\Delta t)^3$$

where

$$t = 0, \quad \Delta t = 0.1, \quad x = 1, \quad x' = 2$$

and

$$x'' = x^2e^t + x', \quad x''(0) = 3,$$
$$x''' = 2xx'e^t + x^2e^t + x'', \quad x'''(0) = 4 + 1 + 3 = 8,$$

hence

$$x(0.1) = 1 + \frac{2}{1!}0.1 + \frac{3}{2!}0.01 + \frac{8}{3!}0.001 = 1.2163(3).$$

Exercise 13. *Suppose that a differential equation is solved numerically on an interval $[a, b]$ and that the local truncation error is $c(\Delta t)^p$. Show that if all truncation errors have the same sign (the worst possible case), then the total truncation error is $(b - a)c(\Delta t)^{p-1}$, where $\Delta t = (b - a)/n$.*
Solution:

$$\cancel{x(a + \Delta t)} = x(a) + \sum_{i=1}^{p} \frac{x^{(i)}(a)}{i!}(\Delta t)^i + c(\Delta t)^{p+1}$$

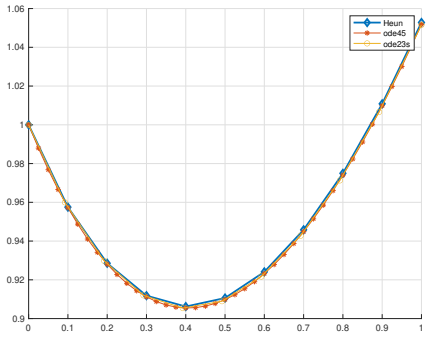


FIGURE 8. The Heun's method solution versus Matlab's and for Exercise 16.

$$x(a+2\Delta t) = \cancel{x(a+\Delta t)} + \sum_{i=1}^{p} \frac{x^{(i)}(a+\Delta t)}{i!}(\Delta t)^i + c(\Delta t)^{p+1}$$

$$\vdots$$

$$\cancel{x(a+(n-1)\Delta t)} = \cancel{x(a+(n-2)\Delta t)} + \sum_{i=1}^{p} \frac{x^{(i)}(a+(n-2)\Delta t)}{i!}(\Delta t)^i + c(\Delta t)^{p+1}$$

$$x(a+n\Delta t) = \cancel{x(a+(n-1)\Delta t)} + \sum_{i=1}^{p} \frac{x^{(i)}(a+(n-1)\Delta t)}{i!}(\Delta t)^i + c(\Delta t)^{p+1},$$

which by adding up gives

$$x(b) = x(a) + \sum\Big(\cdots\Big) + nc(\Delta t)^{p+1} \equiv x(a) + \sum\big(\cdots\big) + n(\Delta t)c(\Delta t)^p \equiv x(a) + \sum\big(\cdots\big) + (b-a)c(\Delta t)^p,$$

hence

$$\Delta t\,\tau_n(\Delta t) := x(b) - x(a) - \sum\big(\cdots\big) = (b-a)c(\Delta t)^p.$$

Exercise 15. *Explain how to use the ODE method that is based on the "trapezoidal rule":*

$$\begin{cases} \widehat{x}(t+h) = x(t) + hf(x,x(t)) \\ x(t+h) = x(t) + h\frac{1}{2}\big(f(t,x(t)) + f(x+h,\widehat{x}(t+h))\big) \end{cases} \qquad \text{(Heun)}$$

This is called the improved Euler's method or Heun's method. Here, $\widehat{x}(t+h)$ is computed by using the Forward Euler method.

Solution:

Exercise 16. (Continuation) *Use Heun's method to solve the following differential equation over the interval $[0,1]$ with step size $h = 0.1$:*

$$\begin{cases} x' = -x + t + \frac{1}{2} \\ x(0) = 1 \end{cases}$$

Solution:

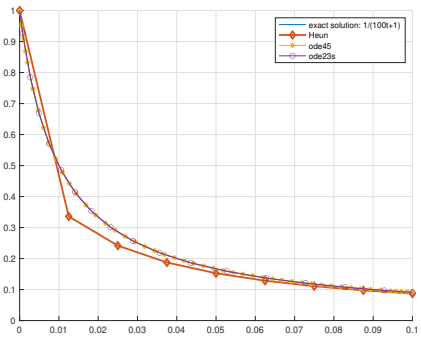


FIGURE 9. The exact solution versus Heun's method (with $\Delta t = 0.0125$) for Exercise 17.

Exercise 17. *Consider the initial-value problem*

$$\begin{cases} x' = -100x^2 \\ x(0) = 1 \end{cases}$$

Use Heun's method to solve the following differential equation over the interval $[0,1]$ *with step size* $h = 0.01$.

Solution: The exact solution can be found as follows

$$\frac{x'}{x^2} = -100, \quad \left(\frac{1}{x}\right)' = 100, \quad x(t) = \frac{1}{100t+1}.$$

On $[0,1]$ *Heun's method blows up with* $\Delta t = 0.1$.

```
%Heun's Method for Solving Initial Value Problems
%Use with ydot.m to evaluate rhs of differential equation
% Input: interval [a,b], initial value y0, step size h
% Output: time steps t, solution y
% Example usage: y=heun([0 1],1,0.01);

function [t,y]=heun(int,y0,h)
t(1)=int(1); y(1)=y0;
s=linspace(int(1),int(2),1000);
n=round((int(2)-int(1))/h);
for i=1:n
  t(i+1)=t(i)+h;
  y(i+1)=heunstep(t(i),y(i),h);
end

[t45,y45] = ode45(@ydot,[int(1),int(2)],y0);
[t23s,y23s] = ode23s(@ydot,[int(1),int(2)],y0);

figure()
hold all
grid on
```

```
plot(s,1./(100*s+1))
plot(t,y,'-d','LineWidth',2)
plot(t45,y45,'-*')
plot(t23s,y23s,'-o')
legend('exact solution: 1/(100t+1)',...
    'Heun',...
     'ode45','ode23s')

function y=heunstep(t,y,h)
%one step of heun's Method
%Input: current time t, current value y,  stepsize h
%Output: approximate solution value at time t+h
y = y + h/2 * ( ydot(t,y)+ydot(t+h,y+h*ydot(t,y)) );
end

function z = ydot(t,y)
%z = - y + t + 0.5;% Exercise 10.1.16
z = - 100*y^2;% Exercise 10.1.17
end

end
```

Computer problem 1. *Write and test a program for applying the Taylor series method to the initial-value problem*

$$\begin{cases} x' = x + x^2 \\ x(1) = \frac{e}{16-e} = 0.20466341728915526943 \end{cases}$$

Generate the solution in the interval $[1, 2.77]$. *Use derivatives to up to* $x^{(5)}$ *in the Taylor series. Use* $h = 1/100$. *Print out for comparison the values of the exact solution* $x(t) = e^t/(16 - e^t)$. *Verify that it is the exact solution.*
Solution:

Computer problem 2. *Write a program to solve each problem on the indicated intervals. Use the Taylor series method with* $\Delta t = 1/100$, *and include terms to* $(\Delta t)^3$. *Account for any difficulties.*

(a) $\begin{cases} x' = t + x^2 \quad on\ [0, 0.9] \\ x(0) = 1 \end{cases}$

(b) $\begin{cases} x' = x - t \quad on\ [1, 1.75] \\ x(1) = 1 \end{cases}$

(c) $\begin{cases} x' = tx + t^2x^2 \quad on\ [2, 5] \\ x(2) = -0.6396625333 \end{cases}$

(d) $\begin{cases} x' = t + x^2 \quad on\ [0, 0.9] \\ x(0) = 1 \end{cases}$

Solution:

Computer problem 3. *Solve the differential equation* $x' = x$ *with initial value* $x(0) = 1$ *by the Taylor series method on the interval* $[0, 10]$. *Compare the result with the exact solution*

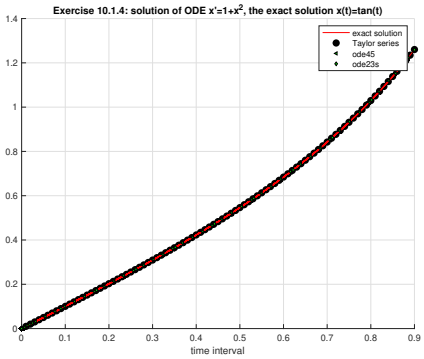


FIGURE 10. Computer problem 10.1.4

$x(t) = e^t$. *Use derivatives up to and including the tenth. Use step size* $h = 1/100$.
Solution:

Computer problem 4. *Solve for* $x(1)$*:*
(a) $x' = 1 + x^2, \quad x(0) = 0$
(b) $x' = \frac{1}{1+t}x, \quad x(0) = 1$
Use the Taylor series method of order 5 with $h = 1/100$*, and compare with the exact solutions, which are* $\tan t$ *and* $1 + t$*, respectively.*
Solution:
(a) $x' = 1 + x^2, \quad x(0) = 0$

$$x'' = 2xx'$$
$$x''' = 2(x')^2 + 2xx''$$
$$x^{(4)} = 6x'x'' + 2xx'''$$
$$x^{(5)} = 6(x'')^2 + 2x'x''' + +2xx^{(4)}$$

```
%Taylor series Method of higher order for Solving Initial Value Problems
% Input: interval [a,b], initial value x0, step size h
% Output: time steps t, solution y
% Computer problem 10.1.r:
%   Solve x'= 1 + x^2, with IC
%           x(0)=0
%   on [0,0.9]
% and compare with the exact solution x(t)=tan(t).
% Use step 1/100 and derivatives up to the 5th.
% Example usage: x=taylor_ex10_1_4([0 0.9],0,0.01);

function [x,y,t] = taylor_ex10_1_4(int,x0,h)
a = int(1); b = int(2);
x = x0;
s = linspace(a,b,200);
```

```
t = a;
n = round((int(2)-int(1))/h);
format long
y = tan(s);   % exact solution
[t45,y45] = ode45(@ydot,[int(1),int(2)],x0);
[t23s,y23s] = ode23s(@ydot,[int(1),int(2)],x0);
for k = 1 : n
    xprime = 1+ x^2;
    xsecond = 2*x*xprime;
    xtertius  = 2*(xprime)^2 + 2*x* xsecond;
    xquatro = 6*xprime*xsecond + 2*x*xtertius;
    xpentium = 6*(xsecond)^2 + 2*xprime*xtertius + 2*x*xquatro;
    x = x + h *(xprime + .5*h*(xsecond + (1/3)*h*(xtertius + .25*h*(xquatro ...
        + 1/5*h*( xpentium )))));
    t = a + k*h;

figure(1)
hold all
grid on
plot(s,y,'-r','LineWidth',1,...
                'MarkerEdgeColor','b',...
                'MarkerFaceColor','r',...
                'MarkerSize',3);
plot(t,x,'ko','LineWidth',5,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',3);
plot(t45,y45,'g<','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',3);
plot(t23s,y23s,'bd','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',3);
legend('exact solution', 'Taylor series','ode45','ode23s')
hold all
end
title('Exercise 10.1.4: solution of ODE x''=1+x^2, the exact solution x(t)=tan(t)')
xlabel('time interval')
end

function z = ydot(t,y)
z = 1 + y^2;% Exercise 10.1.4
end
```

(b) $x' = \frac{1}{1+t}x, \quad x(0) = 1$

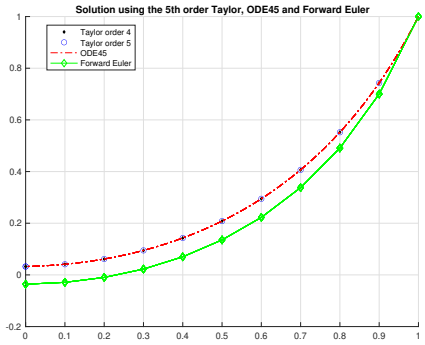


FIGURE 11. Computer problem 5. The solution to the 'final value problem' with timestep $h = -0.01$.

Computer problem 5. *Solve the initial-value problem $x' = t + x + x^2$ on the interval $[0,1]$ with initial condition $x(1) = 1$. Use the Taylor series method of order 5.*
Solution: Note that the condition here is given at the right end of the interval $x(1) = 1$, hence the problem has to be solved 'backward in time', i.e., with a 'negative' time step. See Figure 11.

Computer problem 6. *Solve the initial-value problem $x' = (x+t)^2$ with $x(0) = -1$ on the interval $[0,1]$ using the Taylor series method with derivatives up to and including the fourth. Compare this to Taylor series methods of orders 1, 2, and 3.*
Solution:

Computer problem 7. *Write a program to solve on the interval $[0,1]$ the initial-value problem*

$$\begin{cases} x' = tx \\ x(0) = 1 \end{cases}$$

using the Taylor series method of order 20; that is, include terms in the Taylor series up to and including h^{20}. Observe that a simple recursive formula can be used to obtain $x^{(n)}$ for $n = 1, 2, \cdots, 20$.
Solution:

Computer problem 8. *Write a program to solve the initial-value problem $x' = \sin x + \cos t$, using the Taylor series method. Continue the solution from $t = 2$ to $t = 5$, starting with $x(2) = 0.32$. Include terms up to and including h^3.*
Solution:

8. Runge-Kutta methods

DEFINITION 8.1 (Runge-Kutta methods). *The Runge-Kutta methods for the Cauchy problem* (IVP) *are one-step methods of the general form:*

$$y_{n+1} = y_n + \Delta t F(t_n, y_n, \Delta t; f) \qquad \text{(Runge-Kutta)}$$

where the increment function F is defined as

$$\begin{cases} F(t_n, y_n, \Delta t; f) = \sum_{i=1}^{s} b_i K_i, & \text{with} \\ K_i = f\Big(t_n + c_i \Delta t, y_n + \Delta t \sum_{j=1}^{s} a_{ij} K_j\Big), \quad \forall i = 1:s \end{cases}, \qquad \text{(increment function)}$$

where s denotes the number of stages.
The coefficients $\{a_{ij}\}, \{c_i\}, \{b_j\}$ are collected in the (John Charles) Butcher array :

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \qquad \begin{array}{c|c} c_1 & A \\ \hline & b^T \end{array}. \qquad \text{(Butcher array)}$$

DEFINITION 8.2. *The* (Runge-Kutta) *methods are called explicit if the matrix A in the* (Butcher array) *is lower triangular:*

$$a_{ij} = 0 \qquad \forall j \geq i, \qquad \begin{array}{c|ccccc} c_1 & 0 & 0 & \cdots & 0 & 0 \\ c_2 & a_{21} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ c_{s-1} & a_{s-1,1} & a_{s-1,2} & \cdots & 0 & 0 \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & 0 \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array}$$

(explicit Runge-Kutta)

We recall now the definition 2.1 of the Local Truncation Error we saw in Subsection 2.

DEFINITION 8.3 (Local Truncation Error and Consistency of the (Runge-Kutta) methods). •

The local Truncation Error $\Delta t \tau_{n+1}(\Delta t)$ of the general RK methods is defined through the residual expression

$$\Delta t \tau_{n+1}(\Delta t) := y(t_{n+1}) - y(t_n) - \Delta t F\big(t_n, y(t_n), \Delta t; f\big). \qquad \text{(LTE RK)}$$

- *A* (Runge-Kutta) *method is order p consistent is*

$$\Delta t \tau_{n+1}(\Delta t) = \mathcal{O}(\Delta t)^{p+1}$$

The following conditions on the coefficients in the (Butcher array) *have to be satisfied in order that the* (Runge-Kutta) *methods to be consistent.*

PROPOSITION 8.1. *The* (Runge-Kutta) *method is consistent (at least order 1) if and only if*

$$\begin{cases} c_i = \sum_{j=1}^{s} a_{ij}, & \forall i = 1:s, \\ \sum_{i=1}^{s} b_i = 1. \end{cases} \tag{8.1}$$

For more accurate (higher-order consistency) (Runge-Kutta) *methods, the coefficients in the* (Butcher array) *have to satisfy some extra conditions.*
Although there are a significant number of high-order methods, there are some limitations for (explicit Runge-Kutta) *methods*

THEOREM 8.1 (barriers for the (explicit Runge-Kutta) methods).
- *The order of an s-stage* (explicit Runge-Kutta) *method cannot be greater than s, i.e.,* $p \leq s$.
- *There is no s-stage explicit* (explicit Runge-Kutta) *method with order s if* $s \geq 5$*:*

order	1	2	3	4	5	6	7	8
$s_{\min}$	1	2	3	4	6	7	9	11

EXAMPLE 8.1 (The explicit RK2 methods). *There is a whole family of Runge-Kutta 2 methods are* ($s = 2$)

$$\begin{cases} y_{n+1} = y_n + \Delta t\Big(\big(1 - \frac{1}{2\alpha}\big)K_1 + \frac{1}{2\alpha}K_2\big)\Big) \\ K_1 = f_n \\ K_2 = f(t_n + \alpha\Delta t, y_n + \alpha\Delta t K_1) \end{cases}$$

or (Butcher array)

0	0
α	α
	$\big(1 - \frac{1}{2\alpha}\big)$ $\quad \frac{1}{2\alpha}$

(RK2)

The (Butcher array) *coefficients satisfy the conditions in Proposition 8.1, and more, so the order of consistency of* (RK2) *is* $p = 2$.
Note that
- $\alpha = 1$ *yields Heun's method*
- $\alpha = \frac{2}{3}$ *yields* **Ralston** *method*
- $\alpha = \frac{1}{2}$ *yields the* **explicit midpoint** *method:*

$$y_{n+1} = y_n + \Delta t f\big(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2} f(t_n, y_n)\big).$$

EXAMPLE 8.2 (The implicit RK2 method). *The implicit midpoint rule may be regarded as an implicit Runge–Kutta method [8, page 205] with the* (Butcher array) *tableau given by*

1/2	1/2
	1

which corresponds to

$$K_1 = f\big(t_n + \frac{1}{2}\Delta t, y_n + \frac{1}{2}\Delta t K_1\big),$$
$$y_{n+1} = y_n + \Delta t K_1.$$

To see that this is a reformulation, let's write the implicit midpoint method in a refactorized form (a backward Euler and a forward Euler step):

$$\begin{aligned} y_{n+1/2} &= y_n + \tfrac{\Delta t}{2} f\big(t_n + \tfrac{\Delta t}{2}, y_{n+1/2}\big), \\ y_{n+1} &= y_{n+1/2} + \tfrac{\Delta t}{2} f\big(t_n + \tfrac{\Delta t}{2}, y_{n+1/2}\big). \end{aligned} \tag{refactorized midpoint}$$

Adding these two relations gives the relation above, while subtraction shows that $y_{n+1/2} = \frac{1}{2}(y_{n+1}+y_n)$.

EXAMPLE 8.3 (The RK4 method). *The explicit Runge-Kutta 4 method is* ($s=4$)

$$\begin{cases} y_{n+1} = y_n + \dfrac{\Delta t}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right) \\ K_1 = f_n \\ K_2 = f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2}K_1) \\ K_3 = f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2}K_2) \\ K_4 = f(t_{n+1}, y_n + \Delta t K_3) \end{cases}$$

or (Butcher array)

0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
$\frac{1}{2}$	0	$\frac{1}{2}$	0	0
1	0	0	1	0
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

(RK4)

The (Butcher array) *coefficients satisfy the conditions in Proposition 8.1, and more, so the order of consistency of* (RK4) *is* $p=4$. *See* `RK4Pseudocode.m` *for a pseudocode example implementing the* (RK4) *method. See also Figure 3 for the stability region of the* (RK4) *method.*

9. Stepsize adaptivity with Runge-Kutta methods

Let denote by

$y(t_{n+1})$ *the exact solution*
y_{n+1} *the RK4 solution*
u_{n+1} *the RK5 solution*

hence the (local) truncation errors in the RK4 and RK5 solutions are

$$\begin{aligned} y(t_{n+1}) - y_{n+1} &= C_{n+1}\Delta t_n^5, \qquad \text{(LTE RK4)} \\ y(t_{n+1}) - u_{n+1} &= \mathscr{C}_{n+1}\Delta t_n^6. \end{aligned}$$

We note that for the (Runge-Kutta) *methods, the* (global error) $e_{n+1:=}y(t_{n+1}) - y_{n+1}$ *coincides with the local truncation error* (LTE) $\Delta t\,\tau_{n+1}(\Delta t) := y(t_{n+1}) - y(t_n) - \Delta t F(t_n, y(t_n), \Delta t; f)$, *provided we consider it in only one step, i.e., only on the interval* $[t_n, t_{n+1}]$. *This is the 'localizing assumption' [8, p. 37], namely assuming that*

$$y_n \equiv y(t_n), \qquad \text{(localizing assumption)}$$

which indeed gives

$$\begin{aligned} e_{n+1} := y(t_{n+1}) - y_{n+1} &= y(t_{n+1}) - \Big(y_n - \Delta t F(t_n, y_n, \Delta t; f)\Big) \qquad \text{(using (Runge-Kutta))} \\ &\equiv y(t_{n+1}) - y(t_n) - \Delta t F(t_n, y(t_n), \Delta t; f) := \Delta t\,\tau_{n+1}(\Delta t). \\ &\qquad \text{(using (LTE))} \end{aligned}$$

Since the RK5 solution is 'more' accurate (of a higher-order consistency) than the 'lower-order' solution RK4, we could use the RK5 solution u_{n+1} *instead of the exact (unknown) solution* $y(t_{n+1})$ *to evaluate the local truncation error* (LTE RK4)*:*

$$u_{n+1} - y_{n+1} \approx C_{n+1}(\Delta t_{n,\text{old}})^5. \qquad \text{(RK4 error estimator)}$$

Time-step adaptivity is a technique which adapts the step-size Δt_n in order to keep the local truncation error (LTE RK4) *within a fixed tolerance* `tol`*, so the overall simulation keeps closer to the exact solution (the approximating solution follows the physics):*

$$|u_{n+1} - y_{n+1}| \approx |y(t_{n+1}) - y_{n+1}| \leq \texttt{tol}. \qquad \text{(tolerance check)}$$

Therefore, if the RK4 solution is desired to be more accurate, the user could choose a much smaller tolerance, e.g., $\texttt{tol} = 10^{-13}$*, so the time-step will adapt to satisfy this 'tight' constraint.*

Now we shall briefly describe how an adaptive algorithm chooses the 'new' time-step (in order to satisfy (tolerance check)*. There are two ingredients here:*

(1) First, with a given step-size $\Delta t_{n,\text{old}}$ we compute two solutions (y^{old}_{n+1} and u_{n+1}), using the RK4 and RK5 methods. By estimate (RK4 error estimator) *we express the value of the 'constant' C_{n+1}:*

$$C_{n+1} \approx \frac{u_{n+1} - y^{\text{old}}_{n+1}}{(\Delta t_{n,\text{old}})^5}. \tag{9.1}$$

(2) If the RK4 solution is intended to satisfy the (tolerance check) *with a 'new' timestep $\Delta t_{n,\text{new}}$*

$$|u_{n+1} - y^{\text{new}}_{n+1}| \leq \texttt{tol},$$

then again the (RK4 error estimator) *yields (using also* (9.1)*)*

$$\texttt{tol} \geq \left|u_{n+1} - y^{\text{new}}_{n+1}\right| \approx |C_{n+1}|(\Delta t_{n,\text{new}})^5 \overset{(9.1)}{\approx\approx} \frac{|u_{n+1} - y^{\text{old}}_{n+1}|}{(\Delta t_{n,\text{old}})^5}(\Delta t_{n,\text{new}})^5.$$

Therefore the 'new time-step', which will enforce that the local truncation error is below a set tolerance throughout the simulation, is 'adapted' such that

$$\Delta t_{n,\text{new}} = \left(\frac{\texttt{tol}}{|u_{n+1} - y_{n+1}|}\right)^{\frac{1}{5}} \Delta t_{n,\text{old}}. \tag{9.2}$$

REMARK 9.1. *Matlab' s function* `ODE45` *implements a version of the adaptive RK45Adaptive45.m algorithm described above. On many instances, for relatively short time-interval simulations, the adaptive RK45 is referred to as an 'exact' solution, although it happens that in conservative Hamiltonian systems (see Subsection 6) ODE45 does not perform as well. But in the absence of an exact solution, errors are hard to estimate and RK45 and* `ODE45` *are fairly good tools.*

For stiff problems, Matlab has two other 'solver's': `ODE15s` *and* `ODE23s`.
*See also [**12**] for a family of nonlinear unconditionally stable multistep methods "Time step adaptivity in the method of Dahlquist, Liniger and Nevanlinna" which outperform* `ODE45`, `ODE15s` *and* `ODE23s` *in some stiff problems.*
Note that the algorithm RK45 Adaptive.m provided in the book does not use the procedure described above, namely the estimator (9.2) *to adapt/choose the new time step, but a rather obsolete procedure 'halfing & doubling' [**6**, p.81], [**4**, p. 351].*

EXAMPLE 9.1 (Motion on the surface of a sphere).

As mentioned in the previous Remark 9.1, `ODE45` *does not performed 'ideally', as we are about to see in this example. This is a 'simple' system of 3 ODE's, simulating a (periodic) motion on the surface of a unit* **sphere**, *a closed curve. The system has a 'natural'*

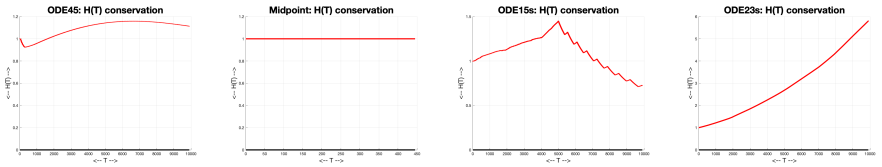


FIGURE 12. Conservation of the Hamiltonian of the `ODE45`, variable-step (implicit midpoint), `ODE15s` and `ODE23s` methods.

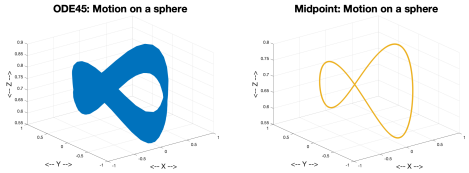


FIGURE 13. The 3D trajectories of the `ODE45` method versus the variable-step (implicit midpoint) method.

Hamiltonian - the sum of the squares of the moving point coordinates, which by Pythagoras' theorem is has to be 1.

We shall see that the time-adaptive higher-order `ODE45` *does not perform well, even when compared with the (also time-adaptive)* (implicit midpoint) *method. The 'better' performance of the second-order* (implicit midpoint) *method over the* `ODE45` *method is mainly due to the fact that* (implicit midpoint) *method is symplectic and conserves all quadratic Hamiltonians.*

Indeed, we obviously see in Figure 12, that the `ODE45` *method has the Hamiltonian deviating from 1, unlike the* (implicit midpoint) *method. (The simulation consists of about 6 dozens of full periodic trajectories.) Moreover, the other two time-adaptive variable-order Matlab functions, the* `ODE15s` *and* `ODE23s` *methods, also do not perform as well as the* (implicit midpoint) *method. This can be interpreted as the* `ODE45` *trajectory initially going 'underground' and then 'flying', instead of just sliding on the surface of the* **sphere**. *Figure 13 shows the two trajectories.*

9.1. Exercises.

Exercise 13. *An important theorem of calculus states that the equation* $f_{tx} = f_{xt}$ *is true, provided that at least one of these two partial derivatives exists and is continuous. Test this equation on some functions, such as* $f(t,x) = xt^2 + x^2t + x^3t^4$, $\log(x-t-1)$, *and* $e^x \sinh(t+x) + \cos(2x-3t)$.

Solution:

Exercise 18. *For the function* $f(x,y) = y^2 - 3\ln x$, *write the first six terms in the Taylor series of* $f(1+h, 0+k)$.

Solution:

Exercise 19. *Using the truncated Taylor series about* $(1,1)$, *give a three-term approximation to* $e^{(1-xy)}$.

Solution: The Taylor series formula in two variables writes

$$f(x+h,y+k)=\sum_{i=0}^{\infty}\frac{1}{i!}\left(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\right)^i f(x,y),$$

$$f(x,y)=\sum_{i=0}^{\infty}\frac{1}{i!}\left((x-a)\frac{\partial}{\partial x}+(y-b)\frac{\partial}{\partial y}\right)^i f(a,b), \qquad \text{(Taylor series in two variables)}$$

which in this case gives (take $a=1,b=1$)

$$f(x,y)=\sum_{i=0}^{\infty}\frac{1}{i!}\left((x-1)\frac{\partial}{\partial x}+(y-1)\frac{\partial}{\partial y}\right)^i f(1,1).$$

Since

$$f(x,y)=e^{(1-xy)},$$
$$\frac{\partial}{\partial x}f(x,y)=-ye^{1-xy}, \qquad \frac{\partial}{\partial y}f(x,y)=-xe^{1-xy}$$

we have that

$$\begin{aligned}f(x,y)&\approx\sum_{i=0}^{1}\frac{1}{i!}\left((x-1)\frac{\partial}{\partial x}+(y-1)\frac{\partial}{\partial y}\right)^i f(1,1)\\&=f(1,1)+(x-1)\frac{\partial}{\partial x}f(1,1)+(y-1)\frac{\partial}{\partial y}f(1,1)\\&=1+(1-x)+(1-y)=3-x-y.\end{aligned}$$

Exercise 20. *The function $f(x,y)=xe^y$ can be approximated by the Taylor series in two variables by $f(x+h,y+k)\approx(Ax+B)e^y$. Determine A and B when terms through the second partial derivatives are used in the series.*
Solution:

Exercise 21. *For $f(x,y)=(y-x)^{-1}$, the Taylor series can be written as*

$$f(x+h,y+k)=Af+Bf^2+Cf^3+\cdots$$

where $f=f(x,y)$. Determine the coefficients A, B, and C.
Solution:

$$\begin{aligned}f(x+h,y+k)&=\sum_{i=0}^{\infty}\frac{1}{i!}\left(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\right)^i f(x,y)\\&\approx\sum_{i=0}^{2}\frac{1}{i!}\left(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\right)^i f(x,y)\\&=f(x,y)+\left(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\right)f(x,y)+\frac{1}{2!}\left(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\right)^2 f(x,y).\end{aligned}$$

Since

$$f(x,y)=\frac{1}{y-x}$$
$$f_x(x,y)=\frac{1}{(y-x)^2}=f^2, \qquad f_y(x,y)=-\frac{1}{(y-x)^2}=-f^2,$$
$$f_{xx}(x,y)=2\frac{1}{(y-x)^3}=2f^3, \qquad f_{yy}(x,y)=2\frac{1}{(y-x)^3}=2f^3, \qquad f_{xy}(x,y)=-2\frac{1}{(y-x)^3}=-2f^3,$$

we have that

$$\begin{aligned} f(x+h,y+k) &= f(x,y)+\Big(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\Big)f(x,y)+\frac{1}{2!}\Big(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\Big)^2 f(x,y) \\ &= f+hf_x+kf_y+\frac{1}{2!}\Big(h^2f_{xx}+2hkf_{xy}+k^2f_{yy}\Big) \\ &= f+hf^2-kf^2+\frac{1}{2!}\Big(h^2 2f^3-4hkf^3+k^2 2f^3\Big) \\ &= f+(h-k)f^2+\big(h^2-2hk+k^2\big)f^3 = f+(h-k)f^2+(h-k)^2f^3. \end{aligned}$$

Exercise 22. *Consider the function e^{x^2+y}. Determine its Taylor series about the point $(0,1)$ through second-partial-derivative terms. Use this result to obtain an approximate value for $f(0.001,0.998)$.*

Solution:

Computer problem 6. *Solve the initial-value problem $x' = x\sqrt{x^2-1}$ with $x(0)=1$ by the Runge-Kutta method on the interval $0 \le t \le 1.6$, and account for any difficulties. Then, using negative h, solve the same differential equation on the same interval with initial value $x(1.6) = 1.0$.*

Solution: Run ($h = 0.016$)

```
[T,x]=RK4Pseudocode([0 1.6],1,100);
```

and then ($h = -0.016$)

```
>> [T,x]=RK4Pseudocode([1.6,0],1,100);
```

```
function [T,x] = RK4Pseudocode(int,x0,n)
% Main Runge-Kutta
%Input: Interval [ta,T], initial condition x0,  and number of steps n
%Output: the approximate solution x, at time T
% Example usage: [T,x]=RK4Pseudocode([1 1.5625],2,72);


ta = int(1);
T = int(2);
h = (T - ta)/n; %%step size h=(T-ta)/n
x = x0;
t = ta;
format long

% The default values for the Relative and Absolute Errors are
%   options = odeset ( 'RelTol', 1.0E-3, 'AbsTol', 1.0E-6 );
%options = odeset ( 'RelTol', 1.0E-10, 'AbsTol', 1.0E-15 );
options = odeset ( 'RelTol', 1.0E-5, 'AbsTol', 1.0E-8 );

[t45,y45] = ode45(@xdot,[int(1),int(2)],x0, options);
[t23s,y23s] = ode23s(@xdot,[int(1),int(2)],x0);
[t15s,y15s] = ode15s(@xdot,[int(1),int(2)],x0);
figure(1)
hold all
grid on
plot(t45,y45,'-*')
```

```
plot(t23s,y23s,'-o')
plot(t15s,y15s,'-d')

for j = 1:n-1
    K1 = h*xdot(t,x);
    K2 = h*xdot(t+h/2,x+K1/2);
    K3 = h*xdot(t+h/2,x+K2/2);
    K4 = h*xdot(t+h,x+K3);
    x = x + (K1 + 2*K2 + 2*K3 + K4)/6;
    t = t + h;
plot(t,x,'.','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10);
legend('ode45','ode23s','ode15s','RK4')
end
```

```
function z = xdot(t,x)  %the RHS of the ODE
%z = 2 + (x - t - 1)^2;
%z = sin(x*t) + atan(t);%Exercise 10.2.10
%z = sin(x) + cos(t);%Exercise 10.2.8
%z = (x+t)^2;%Exercise 10.2.6
z = x*(sqrt(x^2-1));%Computer Exercise 10.2.6
%z = 100*(sin(t) - x);%Computer Exercise 10.2.7
```

Computer problem 7. *The following pathological example has been given by Dahlquist and Björck [4, page 349]. Consider the differential equation*

$$x' = 100(\sin t - x) \qquad \text{(Dahlquist and Björck)}$$

with initial value $x(0) = 0$. Integrate it with the fourth-order Runge-Kutta method on the interval [0, 3], using step sizes h = 0.015, 0.020, 0.025, 0.030. Observe the numerical instability!

Solution:

Δt	0.015	0.020	0.025	0.030
yRK4(3)	0.165812561626782	0.170733281138327	0.175598431702998	4.893749693000771e+11

See Figure 14.

The exact solution is

$$y(t) = (\sin(t) - 0.01 * \cos(t) + 0.01 * e^{-100*t})/1.0001$$
$$y(3) = 0.151004832542617$$
$$y_{ode45}(3) = 0.151004832540368$$
$$y_{ode23s}(3) = 0.151018629630466$$
$$y_{ode15s}(3) = 0.151010875108223$$

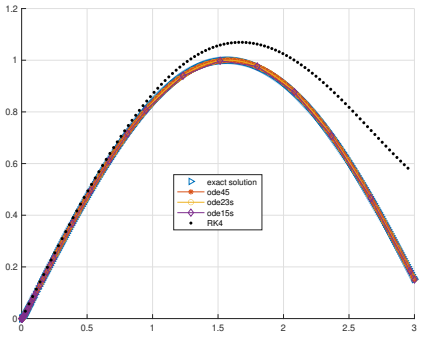


FIGURE 14. Computer problem 10.2.7: Dahlquist and Björck stiff example, $x(0) = 0$ and $\Delta t = 0.0281$.

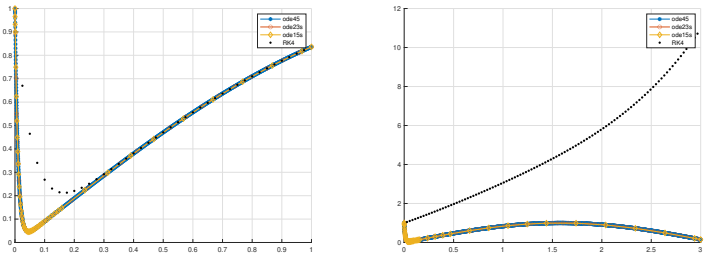


FIGURE 15. Computer problem 10.2.7: Dahlquist and Björck stiff example, $x(0) = 1$ and $\Delta t = 0.025, \Delta t = 0.028$.

where the "RelTol" $= 1.0E-10$, *"AbsTol"* $= 1.0E-15$).
Try also with the initial condition $x(0) = 1$*!!! See Figure 15.*

Computer problem 10. *Write a program to solve* $x' = \sin(xt) + \arctan t$ *on* $1 \le t \le 7$ *with* $x(2) = 4$ *using the Runge-Kutta procedure RK4.*
Solution: Run

```
>> [T,x]=RK4Pseudocode([2 7],4,50);
>> [T,x]=RK4Pseudocode([2 1],4,50);
```

Figure 16.

Exercise 10.3.8 *The initial-value problem* $x' = (1+t^2)x$ *with* $x(0) = 1$ *is to be solved on the interval* $[0,9]$. *How sensitive is* $x(9)$ *to perturbations in the initial value* $x(0)$*?*
Solution: $x(t) = x(0)e^{t+\frac{t^3}{3}}$ *hence* $x(9) = x(0)e^{252} \approx x(0) * 2.7 * 10^{109}$.

Computer problem 10.3.5 *Solve*

$$x' = \frac{3x}{t} + \frac{9}{2}t - 13$$

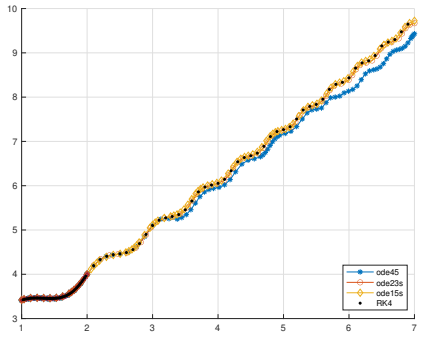


FIGURE 16. Computer problem 10.2.10: $x' = \sin(xt) + \arctan t$ with $x(2) = 4, \Delta t = 0.1$.

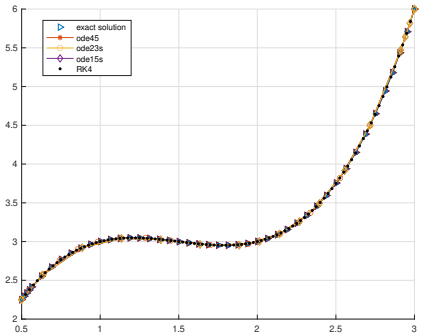


FIGURE 17. Computer problem 10.3.5 with $\Delta t = 0.025$.

$$x(3) = 6$$

at $x(\frac{1}{2})$ *using procedure RK45 Adaptive to obtain the desired solution to nine decimal places. Compare with the true solution:*

$$x = t^3 - \frac{9}{2}t^2 + \frac{13}{2}t.$$

Solution: See Figure 17.

$$x(0.5) = 2.25, \qquad y45(0.5) = 2.249999324675895$$

Computer problem 10.3.6 (Continuation) *Repeat the previous problem for* $x(-\frac{1}{2})$.
Solution: See Figure 18 for the results with AbsTol = 1.E-15, RelTol = 1.E-10.

$$x(0.5) = -4.5, \quad y45(-0.5) = 4.432742126382979 * 10^{11}$$
$$y23s(-0.5) = 1.720162936740793 * 10^{2}$$

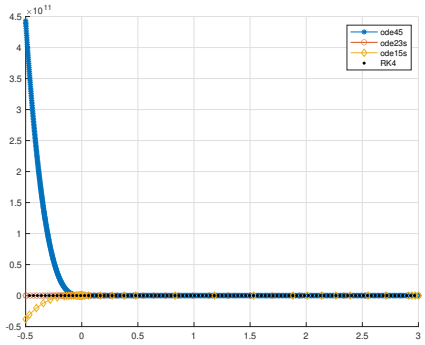


FIGURE 18. Computer problem 10.3.6 with $\Delta t = 0.045$ and AbsTol = 1.E-15, RelTol = 1.E-10.

$$y15s(-0.5) = -3.796060704430997 * 10^{10}$$
$$yRK4(-0.5) = 6.598757851567911 * 10^{4}$$

Note that

$$f_x = \frac{3}{t}$$

which is not defined at $t = 0$, and moreover changes sign when t crosses 0. For $t < 0$, $f_x < 0$, but since the problem is solved backward, the problem becomes unstable, which is conspicuous from Figure 18.

Computer problem 10.3.8 *Compute the numerical solution of*

$$x' = -x,$$
$$x(0) = 1$$

using the **leapfrog method** (LF)

$$x_{n+1} = x_{n-1} + 2\Delta t f(x_n)$$

with $x_0 = 1$ and $x_1 = -h + \sqrt{1+h^2}$.
Are there any difficulties using this method for this problem? Carry out an analysis of the stability of this method. Hint: Consider fixed h and assume $x_n = \lambda_n$.
Solution: See Figure 19 for the results with $\Delta t = 0.05$.
The leapfrog method has the region of the absolute stability on the imaginary axis.
The exact solution is:

$$x(t) = e^{-t}x(0),$$
$$x_0 = 1, \quad x_1 = -h + \sqrt{1+h^2}, \quad x_2 = (h - \sqrt{1+h^2})^2, \cdots$$

Run

```
y=leapfrog([0 1],1,-h+sqrt(1+h^2),0.05);
```

where

```
%Program Leapfrog Method for Solving Initial Value Problems
%Use with ydot.m to evaluate rhs of differential equation
% Input: interval [a,b], initial values y0, y1, step size h
```

```
% Output: time steps t, solution y
% Example usage: y=leapfrog([0 1],1,1,0.1);

function [t,y] = leapfrog(int,y0,y1,h)
t(1)=int(1);
t(2)=t(1)+h;
y(1)=y0;y(2)=y1;
n=round((int(2)-int(1))/h);

% The default values for the Relative and Absolute Errors are
   options = odeset ( 'RelTol', 1.0E-3, 'AbsTol', 1.0E-6 );
%  options = odeset ( 'RelTol', 1.0E-10, 'AbsTol', 1.0E-15 );
[t45,y45] = ode45(@ydot,[int(1),int(2)],y0, options);
[t23s,y23s] = ode23s(@ydot,[int(1),int(2)],y0, options);
[t15s,y15s] = ode15s(@ydot,[int(1),int(2)],y0, options);


figure(1)
clf()
hold all
grid on
plot(t45,exp(-t45),'.-.')
plot(t45,y45,'-*')
plot(t23s,y23s,'-o')
plot(t15s,y15s,'-d')
for i=2:n
  t(i+1)=t(i)+h;
  y(i+1)=leapfrogstep(t(i),y(i-1),y(i),h);
end
plot(t,y,'>-')
legend('exact solution',...
       'ode45','ode23s','ode15s','leapfrog','location','best')

function y=leapfrogstep(t,y0,y1,h)
%one step of Lepfrog Method
%Input: current time t, current value y,  stepsize h
%Output: approximate solution value at time t+h
y=y0+2*h*ydot(t,y1);

function z=ydot(t,y)
z = -y;
```

Computer problem 10.3.20 *Investigate the numerical solution of the initial-value problem*

$$x' = -\sqrt{1-x^2}$$
$$x(0) = 1$$

This problem is ill-conditioned, since $x(t) = \cos t$ *is a solution and* $x(t) = 1$ *is also.*
Solution:

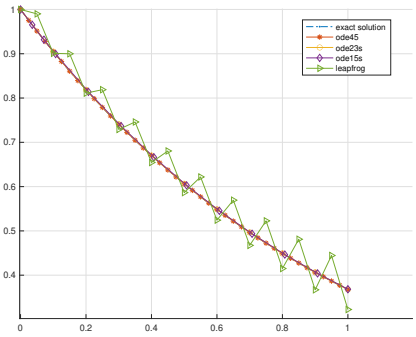


FIGURE 19. Computer problem 10.3.8 with $\Delta t = 0.05$.

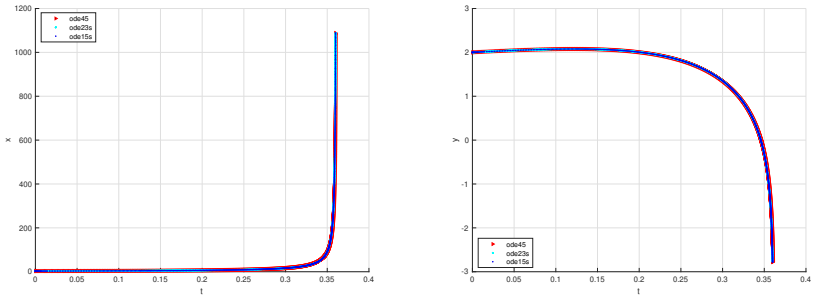


FIGURE 20. Computer problem 11.1.2.

Computer problem 11.1.2 *Solve the initial-value problem*

$$x' = t + x^2 - y$$
$$y' = t^2 - x + y^2$$
$$x(0) = 3, \qquad y(0) = 2$$

on the interval $[0, 0.38]$. *How accurate are the computed function values?*
Solution: Note that $\partial_x f_1 = x, \partial_y f_2 = y$, *which could indicate that the problem is unstable if the solution components are positive. See Figure 20.*

```
%This is Computer Problem 11.1.2 in Cheney and Kincaid, 6th edition, page 475:
% Solve the IVP
% x' = t + x^2 - y
% y' = t^2 - x + y^2
% x(0) = 3, y(0) = 2.
function Exercise11_1_2()

x0 = [3;2];
int = [0 0.36];

% The default values for the Relative and Absolute Errors are
```

```
%  options = odeset ( 'RelTol', 1.0E-3, 'AbsTol', 1.0E-6 );
 options = odeset ( 'RelTol', 1.0E-11, 'AbsTol', 1.0E-15 );
% options = odeset ( 'RelTol', 1.0E-6, 'AbsTol', 1.0E-9 );
[t45,y45] = ode45(@xdot,[int(1),int(2)],x0, options);
[t23s,y23s] = ode23s(@xdot,[int(1),int(2)],x0, options);
[t15s,y15s] = ode15s(@xdot,[int(1),int(2)],x0, options);

figure(1)
clf()
hold all
grid on
plot(t45,y45(:,1), '-o')
plot(t23s,y23s(:,1),'-*')
plot(t15s,y15s(:,1),'-d')
legend('ode45','ode23s','ode15s','location','best')


function z = xdot(t,x)  %the RHS of the system of ODEs
z = [ t + x(1)^2 - x(2) ; t^2 - x(1) + x(2)^2];
```

Computer problem 11.1.3 *Write the Runge-Kutta procedure to solve*

$$\begin{aligned} x_1' &= -3x_2 \\ x_2' &= \frac{1}{3}x_1 \\ x_1(0) &= 0, \qquad x_2(0) = 1 \end{aligned}$$

on the interval $[0,4]$*. Plot the solution.*
Solution:

Computer problem 11.1.6 *Solve the problem*

$$\begin{aligned} x_1' &= 1 \\ x_2' &= -x_3 + \cos(x_1) \\ x_3' &= x_2 + \sin(x_1) \\ x_1(1) &= 1, \qquad x_2(1) = \cos(1), \qquad x_3(1) = \sin(1). \end{aligned}$$

Use the Runge-Kutta method and the interval $[1,4\pi]$.
Solution: The exact solution is

$$x_1(t) = t, \qquad x_2(t) = t\cos t, \qquad x_3(t) = t\sin t,$$

hence

$$|x_2(t)|^2 + |x_3(t)|^2 = t^2$$

which describes a growing spiral. See Figure 21.

```
function [tb,x,y,z] = RK4SystemExercise6at11_1(int,x0,y0,z0,nsteps)
% Runge-Kutta System
%Input: Interval [ta,T], initial condition x0,y0,z0,
% and number of steps n
%Output: the approximate solution x,y,z, at time T
```

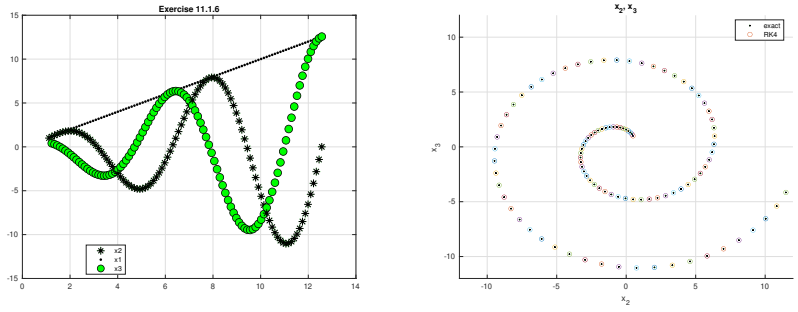


FIGURE 21. Computer problem 11.1.6, the exact and the RK4 solutions, with 100 times-steps, i.e., $\Delta t \approx 0.1157$.

```
% Example usage: [T,x,y,z]=RK4SystemExercise6at11_1([1 2*pi],1,cos(1),sin(1),100);


ta = int(1); tb = int(2);
h = (tb - ta)/nsteps; %%step size h=(T-ta)/n
x = x0;
y = y0;
z = z0;
t = ta;
format long

for j = 1:nsteps
    K11 = h*xdot(t,x,y,z);
    K12 = h*ydot(t,x,y,z);
    K13 = h*zdot(t,x,y,z);

    K21 = h*xdot(t+h/2,x+K11/2,y+K12/2, z+K13/2);
    K22 = h*ydot(t+h/2,x+K11/2,y+K12/2,z+K13/2);
    K23 = h*zdot(t+h/2,x+K11/2,y+K12/2,z+K13/2);

    K31 = h*xdot(t+h/2,x+K21/2,y+K22/2,z+K23/2);
    K32 = h*ydot(t+h/2,x+K21/2,y+K22/2,z+K23/2);
    K33 = h*zdot(t+h/2,x+K21/2,y+K22/2,z+K23/2);

    K41 = h*xdot(t+h,x+K31,y+K32,z+K33);
    K42 = h*ydot(t+h,x+K31,y+K32,z+K33);
    K43 = h*zdot(t+h,x+K31,y+K32,z+K33);

    x = x + (K11 + 2*K21 + 2*K31 + K41)./6;
    y = y + (K12 + 2*K22 + 2*K32 + K42)./6;
    z = z + (K13 + 2*K23 + 2*K33 + K43)./6;

    t = t + h;
figure(1)
```

```
title('Exercise 11.1.6')
grid on
plot(t,x,'.','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10);

plot(t,y,'o','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10);
plot(t,z,'*','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10);

hold all
legend('x2','x1','x3','location','best')
figure(2)
hold all
grid on
plot(t*cos(t),t*sin(t),'*','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',1);
plot(y,z,'o')
xlim([-12,12]); ylim([-12,12])
xlabel('x_2');ylabel('x_3')
legend('exact','RK4','location','best')
title('x_2, x_3')
end



function z1 = xdot(t,x,y,z) %the RHS of the ODE
z1 = 1;

function z2 = ydot(t,x,y,z)
z2 = -z + cos(x);

function z3 = zdot(t,x,y,z)
z3 = y + sin(x);
```

CHAPTER 6

Smoothing of data and the method of Least Squares

1. Linear least squares

Given a set of $m+1$ data points $\{(x_k,y_k)\}_{k=0:m}$, find a line

$$y(x) = ax+b$$

that 'nearly' passes through these points, i.e., minimize the vector of residuals

$$r_k = y(x_k) - y_k = ax_k + b - y_k \tag{1.1}$$

in some vector ℓ_p norm (see the definition of the vector ℓ_p norm):

$$\text{Find } a,b \text{ such that } \varphi_p(a,b) = \|r\|_p^p = \sum_{k=0}^{m} |ax_k+b-y_k|^p \tag{1.2}$$

is minimized

$$\frac{\partial \varphi_p}{\partial a} = 0, \qquad \frac{\partial \varphi_p}{\partial b} = 0. \qquad \text{(necessary conditions)}$$

For example, the (ℓ_1 approximation) problem minimizes

$$\text{Find } a,b \text{ such that } \varphi_1(a,b) = \|r\|_1 = \sum_{k=0}^{m} |ax_k+b-y_k|. \qquad (\ell_1 \text{ approximation})$$

The most common problem is the obtained by minimizing the ℓ_2 norm, called the (Least Squares) problem

$$\text{Find } a,b \text{ such that } \varphi_2(a,b) = \|r\|_2^2 = \sum_{k=0}^{m} (ax_k+b-y_k)^2. \qquad \text{(Least Squares)}$$

Using the (necessary conditions) for the (Least Squares) problem gives the (normal equations for Least Squares) problem

$$\sum_{k=0}^{m} 2x_k(ax_k+b-y_k) = 0, \qquad \sum_{k=0}^{m} 2(ax_k+b-y_k) = 0,$$

(normal equations for Least Squares)

equivalently

$$\Big(\sum_{k=0}^{m} x_k^2\Big)a + \Big(\sum_{k=0}^{m} x_k\Big)b = \sum_{k=0}^{m} x_k y_k, \qquad \Big(\sum_{k=0}^{m} x_k\Big)a + (m+1)b = \sum_{k=0}^{m} y_k,$$

which is a 2×2 *system in* a,b*:*

$$\begin{bmatrix} \sum_{k=0}^{m} x_k^2 & \sum_{k=0}^{m} x_k \\ \sum_{k=0}^{m} x_k & m+1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{k=0}^{m} x_k y_k \\ \sum_{k=0}^{m} y_k \end{bmatrix}. \qquad \text{(linear regression)}$$

EXAMPLE 1.1. *Find the straight line that best fits*

$$(1,0) \quad (2,1) \quad (3,1) \quad (4,2)$$

in the least-squares sense.

PROOF. In this case, the system from (linear regression) is

$$\begin{bmatrix} 30 & 10 \\ 10 & 4 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 13 \\ 4 \end{bmatrix},$$

hence

$$a = 0.6, \qquad b = -0.5.$$

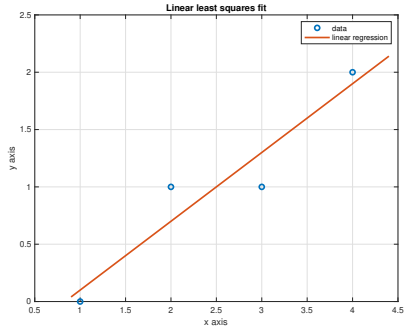


FIGURE 1. The data and the linear polynomial $y = ax+b$ fitting data in the least-squares sense.

```
xdata = [1 2 3 4];
ydata = [ 0 1 1 2];
ls = polyfit(xdata, ydata, 1)
d = ls(1)*xdata + ls(2) - ydata;
phi = sum(d.^2);
x = linspace(xdata(1)*(1.-sign(xdata(1))*0.1),xdata(end)*(1.+sign(xdata(end))*0.1),20);
y = polyval(ls,x);
plot(xdata, ydata, 'o', x, y ,'linewidth',2)
grid on
title('Linear least squares fit')
legend('data', 'linear regression')
xlabel('x axis'), ylabel('y axis')
```

See Figure 1 for a plot of the data and the linear polynomial $y = ax + b$ fitting data in the least-squares sense. □

(See also LinearRegression.m)

2. Non polynomial fit

The method of least squares is not restricted to linear (first-degree) polynomials or any specific function form.

EXAMPLE 2.1. *Suppose we want to fit an equation of the from*

$$y = ae^{x^2} + bx^3$$

to the points

$$(-1,0) \qquad (0,1), \qquad (1,2), \qquad (1.5,2.2)$$

in the lest squares sense.

PROOF. In this case, the 'residual function' (1.2) writes

$$\varphi_2(a,b) = \sum_{k=0}^{3} \left(ae^{x_k^2} + bx_k^3 - y_k\right)^2$$

and the (necessary conditions) become

$$a\sum_{k=0}^{3} e^{2x_k^2} + b\sum_{k=0}^{3} x_k^3 e^{x_k^2} - \sum_{k=0}^{3} y_k e^{x_k^2} = 0$$
$$a\sum_{k=0}^{3} x_k^3 e^{x_k^2} + b\sum_{k=0}^{3} x_k^6 - \sum_{k=0}^{3} y_k x_k^3 = 0.$$

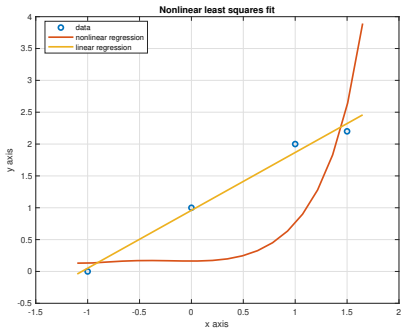


FIGURE 2. The data, the nonlinear function $y = ae^{x^2} + bx^3$, and the linear polynomial $y = mx + n$ fitting data in the least-squares sense.

See Figure 2 for a comparison of the nonlinear and the linear regression for this data.

```
xdata = [-1. 0. 1. 1.5 ];
ydata = [ 0. 1. 2. 2.2];
A = [sum(exp(2*xdata.^2)) sum(xdata.^3.*exp(xdata.^2)) ; sum(xdata.^3.*exp(xdata.^2)) sum(xda
RHS = [sum(ydata.*exp(xdata.^2)) ; sum(ydata.*xdata.^3)];
ls = A\RHS
linearregression = polyfit(xdata, ydata, 1)
d = ls(1)*exp(xdata.^2) + ls(2).*xdata.^3 - ydata;
phi = sum(d.^2);
x = linspace(xdata(1)*(1.-sign(xdata(1))*0.1),xdata(end)*(1.+sign(xdata(end))*0.1),20);
y = ls(1)*exp(x.^2) + ls(2)*x.^3;
yLinearR = polyval(linearregression,x);% linear regression
plot(xdata, ydata, 'o', x, y ,x,yLinearR,'linewidth',2)
grid on
 legend('data', 'nonlinear regression', 'linear regression','Location', 'best')
title('Nonlinear least squares fit')
xlabel('x axis'), ylabel('y axis')
```

□

EXAMPLE 2.2. *Suppose we want to fit a quadratic polynomial*

$$y = ax^2 + bx + c$$

to the points

$$(-3,5) \quad (-2,2), \quad (-1,1), \quad (1,1), \quad (2,2) \quad (3,4.5) \quad (4,3.2)$$

in the lest squares sense.

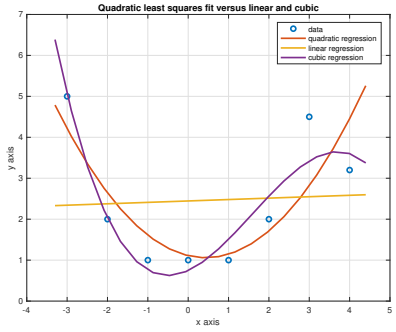


FIGURE 3. The data, the quadratic $y = ax^2 + bx = c$ versus the linear and cubic polynomials, fitting data in the least-squares sense.

```
xdata = [-3 -2 -1 0 1 2 3   4];
ydata = [5. 2  1  1 1 2 4.5 3.2];
qs = polyfit(xdata, ydata, 2)
d = qs(1)*xdata.^2 + qs(2)*xdata + qs(3) - ydata;
phi = sum(d.^2);
ls = polyfit(xdata, ydata, 1);
cs = polyfit(xdata, ydata, 3)
```

```
x = linspace(xdata(1)*(1.-sign(xdata(1))*0.1),xdata(end)*(1.+sign(xdata(end))*0.1),20);
y = polyval(qs,x);
yl = polyval(ls,x);
yc = polyval(cs,x);
plot(xdata, ydata, 'o', x, y , x, yl , x, yc , 'linewidth', 2)
grid on
legend('data', 'quadratic regression', 'linear regression', 'cubic regression','Location', 'b
title('Quadratic least squares fit versus linear and cubic')
xlabel('x axis'), ylabel('y axis')
```

CHAPTER 7

Boundary Value Problems (BVP) for Ordinary Differential Equations

EXAMPLE 0.1 (Two-point BVP). *Let us consider a* 2^{nd} *order scalar ODE (see Definition 0.1)*

$$\begin{cases} y'' = f(x,y,y'), & x \in (a,b) \\ y(a) = \alpha, \quad y(b) = \beta \end{cases} \qquad \text{(two-point BVP)}$$

defined on an interval (a,b)*, and endowed with two Boundary Conditions , i.e., the values of the unknown function* $y(\cdot)$ *on the 'boundary' of the computational interval* (a,b) *are provided (*$y(a), y(b)$ *are given).*

This is a situation somewhat similar to the Cauchy problem / Initial Value Problem (IVP)*, where in order to obtain a solution we needed an n number of 'initial values' (equal to the number of components in the vector y). Since by Remark 0.1 the* p^{th}*-order ODE is equivalent to the n-valued system of ODEs, it is intuitive that in order to solve the BVP we need a number of (boundary values) equal to the order of the equation (i.e., the oder of the highest derivative on y).*

There are several methods of approaching this problem, among them: discretization using finite differences (band matrix), the shooting method, the collocation method, the Galerkin method.

1. Discretization using the finite difference approximations

Similar to the numerical approximation (constant time step) *for* (IVP)*, we start by setting a (space) mesh on the (space domain) interval* $[a,b]$*. Let*

$$h = \frac{b-a}{m} \qquad \text{(space mesh size)}$$

denote the mesh size, and then the mesh points are

$$x_j = a + jh, \qquad \forall j = 0 : m. \qquad \text{(mesh points)}$$

(Note that the end points are $x_0 = a, x_m = b$*.)*

The (two-point BVP) *is the approximated on the mesh points, substituting in the equation the derivates with approximation of the derivatives by standard finite differences. For example, we could use central-difference formulae to approximate the first- and second-order derivatives:*

$$y'(x) \approx \frac{1}{2h}\big(y(x+h) - y(x-h)\big) \qquad \text{(first-order central)}$$

$$y''(x) \approx \frac{1}{h^2}\big(y(x+h) - 2y(x) + y(x-h)\big) \qquad \text{(second-order central)}$$

at all interior nodes $(x_j, j = 1 : m-1)$.
(Recall that both approximations above are second-order accurate, i.e., the error term is $\mathcal{O}(h^3)$*.)*
Let us denote the approximation of the unknown function $y(\cdot)$ *at the mesh points* $\{x_j\}$ *by*

$$y_j \approx y(x_j), \qquad \forall j = 0 : m, \tag{1.1}$$

and in vector form

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{m-2} \\ y_{m-1} \end{bmatrix} \qquad \text{(BVP unknowns)}$$

Then the (two-point BVP) *can be approximated by the following nonlinear system of* $m-1$ *equations, in the* $m-1$ *unknowns* $\{y_j\}_{j=1:m-1}$*:*

$$\begin{cases} y_0 = \alpha, \\ \dfrac{1}{h^2}\big(y_{j+1} - 2y_j + y_{j-1}\big) = f\big(x_j, y_j, \dfrac{1}{2h}(y_{j+1} - y_{j-1})\big) \qquad j = 1 : m-1 \\ y_m = \beta, \end{cases}$$

(BVP system)

In particular, assume that the function in the right hand side of the (BVP system) *depends only on y, i.e.,* (BVP system) *writes:*

$$y'' = f(y), \qquad x \in (a,b).$$

(For the full linear case $f(x,y,y') := u(x) + v(x)y + w(x)y'$*, see Example 1.2.)*
If we also denote

$$f_j \approx f(y(x_j)),$$

then the (BVP system) *writes*

$$A\mathbf{y} = h^2\mathbf{F}(\mathbf{y}) - \mathbf{r}, \tag{1.2}$$

where

$$\mathbf{F}(\mathbf{y}) = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{m-2} \\ f_{m-1} \end{bmatrix}, \qquad \mathbf{r} = \begin{bmatrix} y_0 \\ 0 \\ \vdots \\ 0 \\ y_m \end{bmatrix}.$$

To see this, let us write the (BVP system) *corresponding to the interior nodes* $x_1, x_2, \cdots, x_{m-2}, x_{m-1}$*:*

$$\begin{aligned}
y_2 - 2y_1 \qquad\quad & + y_0 = f_1 && \text{(j=1)} \\
y_3 - 2y_2 \qquad\quad & + y_1 = f_2 && \text{(j=2)} \\
\vdots \qquad\quad & && \\
y_{j+1} - 2y_{mj} \qquad\quad & + y_{j-1} = f_j && \text{(j)} \\
\vdots \qquad\quad & &&
\end{aligned}$$

$$\begin{aligned} y_{m-1} - 2y_{m-2} \quad & + y_{m-3} = f_{m-2} && \text{(j=m-2)} \\ y_m - 2y_{m-1} \quad & + y_{m-2} = f_{m-1} && \text{(j=m-1)} \end{aligned}$$

and move the known values y_0, y_m *to the right hand side, we obtain*

$$\begin{aligned} y_2 - 2y_1 \quad & = f_1 - y_0 && \text{(j=1)} \\ y_3 - 2y_2 \quad & + y_1 = f_2 && \text{(j=2)} \\ & \vdots \\ y_{j+1} - 2y_{mj} \quad & + y_{j-1} = f_j && \text{(j)} \\ & \vdots \\ y_{m-1} - 2y_{m-2} \quad & + y_{m-3} = f_{m-2} && \text{(j=m-2)} \\ - 2y_{m-1} \quad & + y_{m-2} = f_{m-1} - y_m \quad . && \text{(j=m-1)} \end{aligned}$$

Therefore in (1.2) *the matrix* $\mathbf{A}$ *and the vector* $\mathbf{r}$ *are*

$$\mathbf{A} = \begin{bmatrix} -2 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 & 0 & 0 & 0 \\ & & \ddots & \ddots & \ddots & & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & \ddots & \ddots & \ddots & \\ 0 & 0 & 0 & 0 & \cdots & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & -2 \end{bmatrix}, \qquad \mathbf{r} = \begin{bmatrix} f_1 - y_0 \\ f_2 \\ f_3 \\ \vdots \\ \vdots \\ \vdots \\ f_{m-3} \\ f_{m-2} \\ f_{m-1} - y_m \end{bmatrix}.$$

REMARK 1.1. *Notice that the matrix* $\mathbf{A}$ *is a tridiagonal matrix, well studied in the Section dedicated to Linear Systems.*

EXAMPLE 1.1 (Simple harmonic motion). *Let us consider now a forced simple harmonic motion :*

$$\begin{aligned} & y''(x) + y(x) = x \qquad x \in \left(0, \frac{\pi}{2}\right), && \text{(simple harmonic motion)} \\ & y(0) = 1, \quad y\left(\frac{\pi}{2}\right) = \frac{\pi}{2} - 1, && \text{(B.C.)} \end{aligned}$$

which has the exact solution

$$y(x) = \cos(x) - \sin(x) + x. \qquad \text{(exact solution)}$$

The finite difference equation yields

$$\begin{cases} y_0 = 1, \quad y_m = \dfrac{\pi}{2} - 1 \\ y_{j+1} - 2y_j + y_{j-1} + h^2 y_j = h^2 x_j, \qquad j = 1 : m-1. \end{cases} \tag{1.3}$$

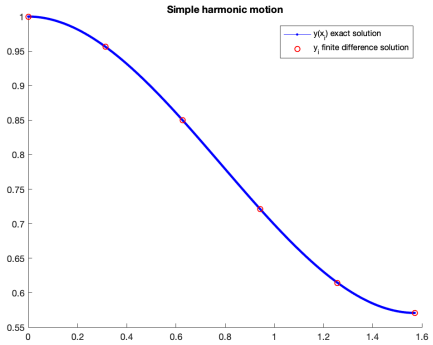


FIGURE 1. The exact and the finite difference solutions to (1.3) on a mesh with $m = 5$.

The linear system arising using the finite difference method writes as

$$\begin{bmatrix} -2+h^2 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 1 & -2+h^2 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & -2+h^2 & 1 & \cdots & 0 & 0 & 0 & 0 \\ & & \ddots & \ddots & \ddots & & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & & \ddots & \ddots & \ddots & & \\ 0 & 0 & 0 & 0 & \cdots & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ \vdots \\ \vdots \\ y_{m-3} \\ y_{m-2} \\ y_{m-1} \end{bmatrix} = h^2 \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ \vdots \\ x_{m-3} \\ x_{m-2} \\ x_{m-1} \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ \\ \\ \frac{\pi}{2} - 1 \end{bmatrix}.$$

This linear system can now be solved by solvers specifically designed for tridiagonal matrices, or any linear solver.

Figure 1 shows the (exact solution) *versus the computed solution with a quite coarse mesh, using only* $m = 5$ *intervals, hence a meshsize* $h = \frac{\pi/2}{5} = 0.3142$. *(The linear solve was the Matlab's 'backslah':* $y = A\backslash\text{RHS}$. *See* `BVPsimpleharmonicmotion.m`*)*
To verify that the numerical solution is **second-order accurate** *with respect to the spatial discretization, namely*

$$\mathscr{E}(h) := \left(\frac{1}{m+1} \sum_{i=0}^{m} \|u(x_i) - u_i\|^2 \right)^{\frac{1}{2}} \approx C h^p, \qquad (p\text{-th order accurate})$$

with $p = 2$ *one can evaluate* p *in terms of computed quantities:*

$$p = \frac{\log(\mathscr{E}(h)/\mathscr{E}(h/2))}{\log(2)}$$

For the example above we have

$$p = \frac{\log(\mathscr{E}(m=6)/\mathscr{E}(m=3))}{\log(2)} = \frac{\log(0.000589306/0.00017094}{\log(2)} = 1.7855,$$

$$p = \frac{\log(\mathscr{E}(m=12)/\mathscr{E}(m=6))}{\log(2)} = \frac{\log(0.00017094/4.43131e-05}{\log(2)} = 1.9477,$$
$$p = \frac{\log(\mathscr{E}(m=24)/\mathscr{E}(m=12))}{\log(2)} = \frac{\log(4.43131e-05/1.1282e-05}{\log(2)} = 1.9737,$$
$$p = \frac{\log(\mathscr{E}(m=48)/\mathscr{E}(m=24))}{\log(2)} = \frac{\log(1.1282e-05/2.84794e-06}{\log(2)} = 1.9860,$$
$$p = \frac{\log(\mathscr{E}(m=96)/\mathscr{E}(m=48))}{\log(2)} = \frac{\log(2.84794e-06/7.15566e-07}{\log(2)} = 1.9928.$$

EXAMPLE 1.2 (The linear case). *Let us consider now the* **linear** *case for the* (BVP system)*, i.e., where the right-hand side has the following form;*

$$f(x,y,y') := u(x) + v(x)y + w(x)y'. \qquad \text{(linear BVP)}$$

Denoting

$$u_j \approx u(x_j), \quad v_j \approx v(x_j) \quad w_j \approx w(x_j),$$

(and using the (first-order central) *and* (second-order central) *formulae) the* (BVP system) *writes*

$$\frac{1}{h^2}\left(y_{j+1} - 2y_j + y_{j-1}\right) = u_j + v_j y_j + w_j \frac{1}{2h}\left(y_{j+1} - y_{j-1}\right).$$

Multiplying by $-h^2$ and regrouping terms this is equivalent to

$$\underbrace{-\left(1+\frac{h}{2}w_j\right)}_{a_j} y_{j-1} + \underbrace{\left(2+h^2 v_j\right)}_{d_j} y_j \underbrace{-\left(1-\frac{h}{2}w_j\right)}_{c_j} y_{j+1} = \underbrace{-h^2 u_j}_{b_j} \qquad (1.4)$$

for all $j = 1 : m-1$, and with the boundary conditions

$$y_0 = \alpha, \qquad y_m = \beta.$$

Therefore the linear Boundary Value Problem

$$\begin{cases} y'' = u(x) + v(x)y + w(x)y', & x \in (a,b) \\ y(a) = \alpha, \quad y(b) = \beta \end{cases}$$

is approximated (using finite differences) by the trilinear linear system

$$\begin{bmatrix} d_1 & c_1 & 0 & 0 & \cdots & & 0 & 0 & 0 \\ a_2 & d_2 & c_2 & 0 & \cdots & & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & & 0 & 0 & 0 \\ & & \ddots & \ddots & \ddots & & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & \ddots & \ddots & \ddots & \\ 0 & 0 & 0 & 0 & \cdots & & a_{m-2} & d_{m-2} & c_{m-2} \\ 0 & 0 & 0 & 0 & \cdots & & 0 & a_{m-1} & d_{m-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ \vdots \\ \vdots \\ y_{m-2} \\ y_{m-1} \end{bmatrix} = \begin{bmatrix} b_1 - a_1 y_0 \\ b_2 \\ b_3 \\ \vdots \\ \vdots \\ \vdots \\ b_{m-2} \\ b_{m-1} - c_{m-1} y_m \end{bmatrix}. \qquad (1.5)$$

EXAMPLE 1.3 (Example in the book, see BVP1.m). *Solve the linear BVP*

$$x'' = e^t - 3\sin(t) + x' - x$$
$$x(1) = 1.097374910854626, \qquad x(2) = 8.637496608572079,$$

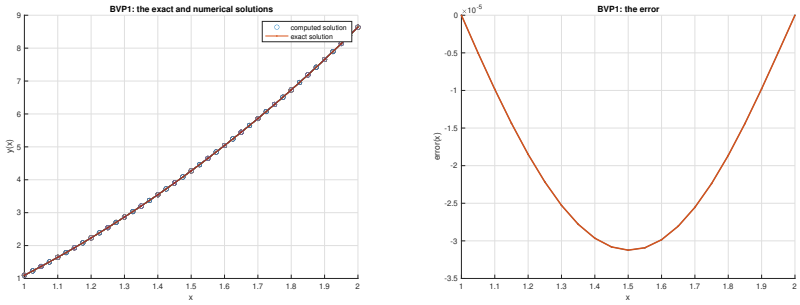


FIGURE 2. The linear BVP example 1.3: the exact and finite-difference solutions ($h = 0.05$), and the error.

which has the exact solution $x(t) = e^t - 3\cos(t)$. *Writing the equation in the linear form* (linear BVP)*, we have that*

$$u(t) = e^t - e\sin t, \qquad v(t) = -1, \qquad w(t) = 1.$$

See http://www.pitt.edu/ trenchea/MATH1080/Chapter14/BVP1.m and Figure 2.

```
>> BVP1([1 2],10)
The root mean square of the error is 8.480505083949440e-05
>> BVP1([1 2],20)
The root mean square of the error is 2.164486089399624e-05
```

Therefore, by the 'error analysis' arguments in (rate of convergence) *gives*

$$p \approx \log\left(\frac{\mathscr{E}^h}{\mathscr{E}^{h/2}}\right) \Big/ \log(2) \equiv \frac{\log(8.480505083949440 * 10^{-5}/2.164486089399624 * 10^{-05})}{\log(2)} \approx 1.9701,$$

confirming the theoretical result that the approximations are second-order accurate.

1.1. Exercises.

EXERCISE 1.1 (This is Exercise 14.2.5 in the textbook). *Show that if* $v_i > 0$ *in the previous Example 1.2, then there exists a meshsize h such that the trilinear matrix in* (1.5) *is diagonally dominant.*
(Hint: consider h small enough so that $|a_i| = -a_i, |c_i| = -c_i$.*)*

Computer problem 14.2.2 *Solve the following two-point boundary value problem numerically. For comparisons, the exact solutions are given.*

(a)

$$\begin{cases} x'' = \dfrac{(1-t)}{(1+t)^2}x + \dfrac{1}{(1+t)^2} \\ x(0) = 1, \qquad x(1) = 0.5 \end{cases} \qquad \text{(solution: } x(t) = \tfrac{1}{1+t}\text{)}$$

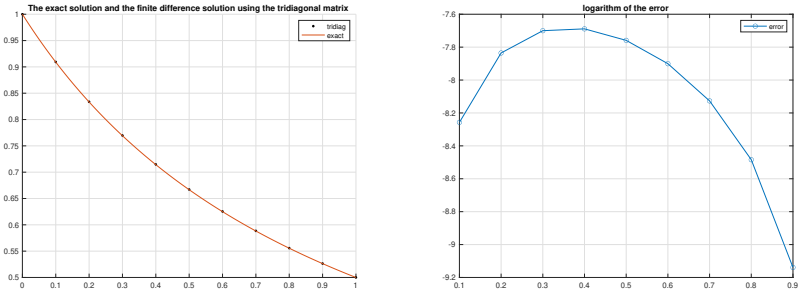


FIGURE 3. Computer exercise 14.2.2(a): the exact and finite-difference solutions ($h = 0.1$), and the logarithm of the error.

(b)

$$\begin{cases} x'' = \dfrac{1}{3}\left((2-t)e^{2x} + \dfrac{1}{1+t}\right) \\ x(0) = 0, \qquad x(1) = -\log(2) \end{cases} \qquad \text{(solution: } x(t) = -\log(1+t)\text{)}$$

Note that this is a nonlinear problem, as the RHS depends nonlinearly on x, namely it involves e^{2x}. This can be solved, for example, by a fix point iteration.

Solution:

(a) Since the equation to solve

$$x'' = \frac{(1-t)}{(1+t)^2}x + \frac{1}{(1+t)^2}$$

is linear, of the form (two-point BVP) *$x'' = f(t,x,x')$, where the linear function $f(t,x,x') = u(t) + v(t)x + w(t)x'$ is of the form* (linear BVP)*, namely with*

$$u(t) = \frac{1}{(1+t)^2}, \qquad v(t) = \frac{(1-t)}{(1+t)^2}, \qquad w(t) = 0,$$

then we can find the solution $\{x_i\}_{i=1:m-1}$ (which approximates the exact solution $\{x(t_i)\}_{i=1:m-1}$) it by solving the linear system (1.5)*, where a_i, b_i, c_i, d_i are defined in* (1.4)*.*
See Figure 3.

```
%function [x,l2error] = BVP1_Exercise2a_14_2(int,alpha,beta,n)
function BVP1_Exercise2a_14_2(int,alpha,beta,n)
%Finite difference scheme for 2nd order BVP
%Input data: interval,
%            the Boundary Values alpha and beta,
%            and the number of intervals defining the mesh
% Example usage: [x,error] = BVP1_Exercise2a_14_2([0 1],1,0.5,100)

ta = int(1);
tb = int(2);
h = (tb - ta)/n;

for i = 1:n-1
    t = ta + i*h;
```

```
    a(i) = -(1+(h/2)*w(t));
    d(i) = 2 + h^2*v(t);
    c(i) = -(1-(h/2)*w(t));
    b(i) = - h^2*u(t);
end

b(1) = b(1) - a(1)*alpha;
b(n-1) = b(n-1) - c(n-1)*beta;

for i=1:n
    dummy = a(i) ;
    a(i+1) = dummy;
end

[x] = Tri(a,d,c,b);
[xx] = [alpha x beta];
tt = [ta:h:tb];
s = [ta:h^2:tb];

errorfunction = 1./(1+tt) - xx ;
l2error = rms(1./(1+tt) - xx);
fprintf ( 1, '  The mean least-squares error is  = %g \n\n',l2error);

figure(1)
clf()
hold all
grid on
plot(tt,xx,'.','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10);

plot(s,1./(1+s),'-','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','r',...
                'MarkerSize',10);
legend('tridiag','exact')
title('The exact solution and the finite difference solution using the tridiagonal matrix'

figure(2)
clf()
plot(tt,log(abs(errorfunction)),'o-')
grid on
legend('error')
title('logarithm of the error')
end
```

```
function z = u(t)
z = 1/(1+t)^2;
end

function z = v(t)
z = (1-t)/(1+t)^2;
end

function z = w(t)
z = 0;
end
```

ERROR ANALYSIS. *We shall check now that the errors are, as theoretically predicted, of second-order (h^p, with $p=2$), provided the mesh size h is small enough (the computations are in the* 'ASYMPTOTIC REGIME'*)*

$$x^{\text{exact}}(t_i) - x_i \approx Ch^p.$$

In order to observe that, we first compute several approximations, for example x^h with mesh sizes h and $x^{h/2}$ with mesh size $h/2$, which should have the corresponding errors approximately $\mathscr{E}^h \approx \text{const} * h^p$ and $\mathscr{E}^{h/2} \approx \text{const} * (h/2)^p$. Then

$$\log\left(\frac{\mathscr{E}^h}{\mathscr{E}^{h/2}}\right) \approx \log\left(\frac{\cancel{\text{const}} * h^p}{\cancel{\text{const}} * (h/2)^p}\right) = p\log(2),$$

and therefore

$$p \approx \log\left(\frac{\mathscr{E}^h}{\mathscr{E}^{h/2}}\right) \Big/ \log(2) \qquad \text{(rate of convergence)}$$

h	error	p
0.2	0.0012	
0.1	3.1618e-04	1.9242
0.05	8.14809e-05	1.9562
0.025	2.06532e-05	1.9801
0.0125	5.19735e-06	1.9905

(b) See Figure 4.
First note that the Right Hand Side of the differential equation is a nonlinear function of the unknown $x(t)$, namely

$$f(t,x) = \frac{1}{3}\left((2-t)e^{2x} + \frac{1}{1+t}\right),$$

and therefore, the (BVP system) *becomes*

$$\begin{cases} x_0 = \cancel{\alpha} \equiv 0, \\ \dfrac{1}{h^2}(x_{j+1} - 2x_j + x_{j-1}) = \cancel{f(t_j, x_j)} \equiv \dfrac{1}{3}\left((2-t_j)e^{2x_j} + \dfrac{1}{1+t_j}\right), \qquad j = 1:m-1 \\ x_m = \cancel{\beta} \equiv -\log(2), \end{cases}$$

which then yields the following form of the nonlinear system (1.2)

$$A\mathbf{x} = h^2\mathbf{F}(t,\mathbf{x}) - \mathbf{r}, \tag{1.6}$$

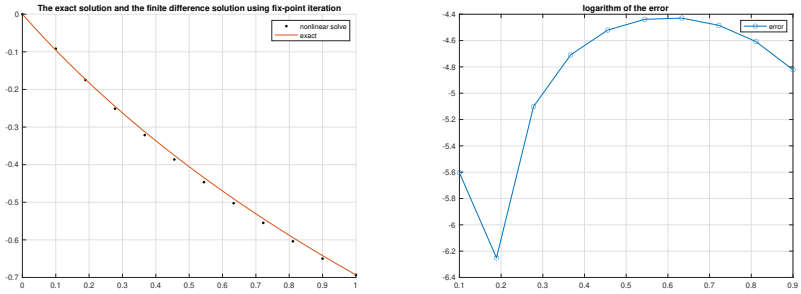


FIGURE 4. Computer exercise 14.2.2(b): the exact and finite-difference solutions ($h = 0.1$, the mean least-squares error is = 0.00830429), and the logarithm of the error. The nonlinear equation (1.6) is solved by fix-point iteration.

where

$$\mathbf{A} = \begin{bmatrix} -2 & 1 & 0 & 0 & \cdots & & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & \cdots & & 0 & 0 & 0 & 0 \\ & & & & \ddots & \ddots & \ddots & & & \\ 0 & 0 & 0 & 0 & \cdots & & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & \cdots & & 0 & 0 & 1 & -2 \end{bmatrix}, \quad \mathbf{F}(t,\mathbf{y}) = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{m-2} \\ f_{m-1} \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} \cancel{\alpha} & 0 \\ & 0 \\ & \vdots \\ & 0 \\ \cancel{\beta} & -\log(2) \end{bmatrix},$$

and

$$f_j := \frac{1}{3}\left((2-t_j)e^{2x_j} + \frac{1}{1+t_j}\right).$$

The nonlinear system $A\mathbf{x} = h^2\mathbf{F}(t,\mathbf{x}) - \mathbf{r}$, can be solved in different ways, e.g., by using Matlab's fsolve.m. In here we simply use a fix point iteration

$$A\mathbf{x}^{(\kappa+1)} = h^2\mathbf{F}(t,\mathbf{x}^{(\kappa)}) - \mathbf{r}.$$

```
function BVP1_Exercise2b_14_2(int,alpha,beta,n)
%Finite difference scheme for 2nd order BVP
%Input data: interval,
%            the Boundary Values alpha and beta,
%            and the number of intervals defining the mesh
% Example usage: [x,error] = BVP1_Exercise2b_14_2([0 1],0,-log(2),10)

ta = int(1);
tb = int(2);
h = (tb - ta)/n
t = linspace(ta+h,tb-h,n);
x0 = ones(n,1);
x = x0;
r = zeros(n,1); r(1) = alpha; r(end) = beta;
% Solve 'Ax = F(x) -r' by fix-point iterations
for i = 1 : 5
```

```
    A = (-2) * diag(ones(n,1)) + diag(ones(n-1,1),1) + diag(ones(n-1,1),-1);
    x = A \ (h^2 /3 *( (2-t').*exp(2*x) + 1./(1+t')) - r);
end
xx = [alpha;x;beta];
c = [t,h,r',n];

tt = [ta t tb]' ;
s = [ta:h^2:tb];

 errorfunction = -log(1+tt) - xx ;
 l2error = rms(-log(1+tt) - xx);

 fprintf ( 1, '  The mean least-squares error is  = %g \n\n',l2error);
 fprintf ( 1, '  The square of the mesh size is h^2 = %g \n\n',h^2);

figure(1)
clf()
hold all
grid on
plot(tt,xx,'.','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10);

plot(s,-log(1+s),'-','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','r',...
                'MarkerSize',10);
legend('nonlinear solve','exact')
title(['The exact solution and the finite difference solution using fix-point iteration'])

figure(2)
clf()
plot(tt,log(abs(errorfunction)),'o-')
grid on
legend('error')
title('logarithm of the error')
end
```

CHAPTER 8

Partial Differential Equations

Partial Differential Equations are equations involving unknowns functions depending on several (dependent) variables, such as times and/or space coordinates, and where the unknown function appears with also partial derivatives.

EXAMPLE 0.1 (*Poisson equation*). *Poisson equation is a typical partial differential equation (of elliptic type) with applications in engineering, physics and chemistry. In two space dimensions x, y, we say that the unknown function $u(x,y)$ satisfies the Poisson equation in the square domain $D = [-1,1]^2$ if*

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial x^2} + 2 = 0, \qquad (x,y) \in \text{interior}(D) = (-1,1)\times(-1,1), \qquad \text{(Poisson equation)}$$

also satisfying the (homogeneous Dirichlet) Boundary Conditions

$$u(x,y) = 0 \qquad \text{on the boundary}(D). \qquad \text{(B.C.)}$$

NOTATION 0.1.

- *For a scalar function $u(x,y,z)$, the gradient is defined as*

$$\nabla u \equiv \operatorname{grad} u := \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial z} \end{pmatrix}. \qquad \text{(gradient)}$$

- *For a vector function $\begin{pmatrix} v_1(x,y,z) \\ v_2(x,y,z) \\ v_3(x,y,z) \end{pmatrix}$, the divergence is defined as*

$$\nabla \cdot \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \equiv \operatorname{div} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y} + \frac{\partial v_3}{\partial z}. \qquad \text{(divergence)}$$

- *For a scalar function $u(x,y,z)$, the Laplace operator is defined as*

$$\Delta u = \nabla \cdot (\nabla u) \equiv \operatorname{div}(\operatorname{grad} u) := \nabla \cdot \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial z} \end{pmatrix} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}. \qquad \text{(Laplacian)}$$

DEFINITION 0.1 (Boundary conditions). *The boundary condition (B.C.) is just one particular case (where the values of the unknown function are prescribed) of the following types.*

- *Dirichlet boundary conditions, where the values of the unknown function are prescribed:*

$$u(x,y,x,\cdot) = g_D(x,y,z,\cdot) \qquad \text{(Dirichlet B.C.)}$$

where $g_D(\cdot)$ is given.

- *Neumann boundary conditions, where the values of the flux are prescribed:*

$$\mathbf{n}\cdot\nabla u(x,y,x,\cdot) = g_N(x,y,z,\cdot) \qquad \text{(Neumann B.C.)}$$

where $g_N(\cdot)$ is given, and $\mathbf{n}$ is the normal vector.

- *Robin (mixed Dirichlet and Neumann) boundary conditions, where the flux depends linearly on the unknown function:*

$$\mathbf{n}\cdot\nabla u(x,y,x) + \alpha u(x,y,x) = g_R(x,y,z) \qquad \text{(Robin B.C.)}$$

where $g_R(\cdot)$ is given, $\alpha \in \mathbb{R}$.

EXAMPLE 0.2 (wave equation). *The* (wave equation) *describing the vibration (displacement) of a 3D elastic body is an evolutionary PDE, on the unknown function $u(x,y,z,t)$, depending on time t and on 3-space dimensions (x,y,z), namely*

$$\frac{\partial^2 u}{\partial t^2} = \Delta u. \qquad \text{(wave equation)}$$

The wave equation is a particular case of a hyperbolic equation, see Section 2.

EXAMPLE 0.3 (heat equation). *The* (heat equation) *describes the evolution (diffusion) of $u(t,x,y,z)$ temperature (heat, concentration) at time t and at space point (x,y,z) in a physical 3D body is an evolutionary PDE, on the unknown function $u(x,y,z,t)$ (temperature), depending on time and on 3-space dimensions, namely*

$$\frac{\partial u}{\partial t} = \kappa\Delta u, \qquad \text{(heat equation)}$$

$$\frac{\partial u}{\partial t} = \nabla\cdot\big(d(x,y,z)\nabla u\big) + \rho, \qquad \text{(diffusion equation)}$$

The heat and diffusion equations are particular cases of parabolic equations, see Section 1.

EXAMPLE 0.4 (Reaction-diffusion equations). *See Section 3.2 for reaction-diffusion equations.*

EXAMPLE 0.5 (Elliptic equations).

$$\Delta u = 0 \qquad \text{(Laplace equation)}$$

(here $u(x,y,z)$ describes the steady distribution of heat, or the steady distribution of electrical charge in a body)

$$-\nabla\cdot\big(d(x,y,z)\nabla u\big) = \rho \qquad \text{(Poisson equation)}$$

$$\Delta u + \kappa^2 u = 0 \qquad \text{(Helmholtz equation)}$$

(this is an eigenvalue problem, with κ is the eigenvalue / wave number)

EXAMPLE 0.6 (Biharmonic equations). *The following biharmonic equation describes the (shearing and elastic) stress of an elastic body:*

$$\frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4} + \frac{\partial^4 u}{\partial z^4} = 0. \qquad \text{(biharmonic equation)}$$

REMARK 0.1. MATLAB *has a* PDE toolbox*:* pdesolve *for solving partial differential equations of the form*

$$m\frac{\partial^2 u}{\partial t^2}+d\frac{\partial^2 u}{\partial t^2}-\nabla\cdot(c\nabla u)+au=f,$$

where m,d,c,a *are given and* f *is a given forcing term (to be specified by the user), in 2D and 3D (two and three space dimensions) of prescribed types:*

$$d\frac{\partial u}{\partial t}-\nabla\cdot(c\nabla u)+au=f \qquad \text{(parabolic)}$$

$$d\frac{\partial^2 u}{\partial t^2}-\nabla\cdot(c\nabla u)+au=f \qquad \text{(hyperbolic)}$$

$$-\nabla\cdot(c\nabla u)+au=f \qquad \text{(elliptic)}$$

1. Parabolic Equations

We consider here the following heat equation in 1D space dimension, where $u(x,t)$ *models the temperature of a rod of length 1 (on the interval* $x\in[0,1]$*), with (Dirichlet B.C.) the temperature being set to zero at the ends of the interval, and with the initial temperature being set as* $\sin(\pi x)$*:*

$$\frac{\partial u}{\partial t}(x,t)-\frac{\partial^2 u}{\partial x^2}(x,t)=0, \qquad t>0,\ x\in(0,1) \qquad \text{(heat equation)}$$

$$u(0,t)=u(1,t)=0, \qquad \forall t\geq 0 \qquad \text{(homogeneous BC)}$$

$$u(x,0)=\sin(\pi x), \qquad x\in[0,1]. \qquad \text{(initial conditions)}$$

The solution to the heat equation with the (homogeneous BC) *and* (initial conditions) *above obviously has to satisfy the following* (compatibility conditions)

$$u(0,0)=0, \qquad u(1,0)=0. \qquad \text{(compatibility conditions)}$$

(check out the heat equation with non-homogeneous Dirichlet, periodic and Neumann B.C.)

THEOREM 1.1 (Maximum Principle). *The solution to the* (heat equation) *satisfies the following maximum principle*

$$\min\{0,\ \inf_{x\in(0,1)} u(x,0)\}\leq u(x,t)\leq \max\{0,\ \sup_{x\in(0,1)} u(x,0)\}, \tag{1.1}$$

for all $(x,t)\in(0,1)\times(0,T)$.

A direct consequence of the maximum principle is that the maximum value of u will not exceed the maximum value that previously occurred.
In particular, since $u(x,0)=\sin(\pi x)\in[0,1] \quad \forall x\in[0,1]$*, by* (1.1) *we obtain that*

$$0\leq u(x,t)\leq 1, \qquad \forall t\geq 0, x\in[0,1]. \qquad \text{(maximum principle)}$$

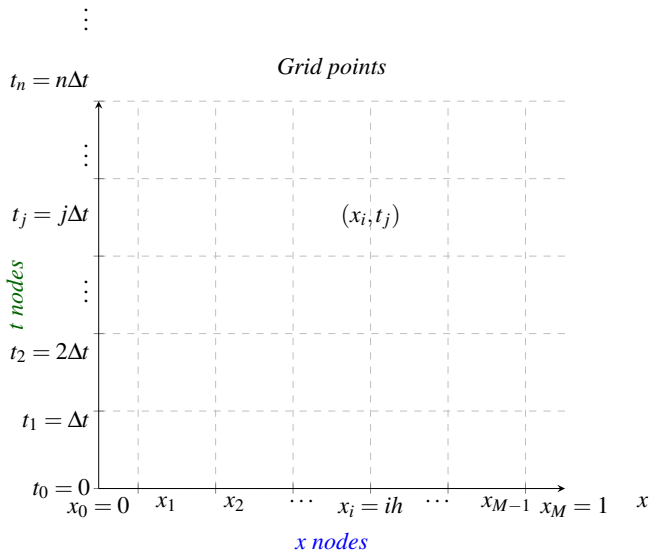
The heat equation above, with the B.C. and I.C. above has the following (exact solution)*:*

$$u(x,t)=e^{-t\pi^2}\sin(\pi x), \qquad \text{(exact solution)}$$

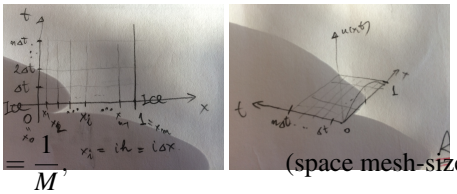
which can be shown by using the Fourier method, or the method of separation of variables, see e.g. [2].

PROOF. Indeed, $u_t = -\pi^2 e^{-t\pi^2}\sin(\pi x)$, $u_x = \pi e^{-t\pi^2}\cos(\pi x)$, $u_{xx} = -\pi^2 e^{-t\pi^2}\sin(\pi x)$. Also the initial conditions are satisfied $u(x,0) = \sin(\pi x)$, as well as the boundary conditions $u(0,t) = u(1,t) = 0$. □

We consider here only the `finite difference` *approximations of the* (heat equation).



To simplify the presentation we restrict ourselves to a space discretization of the interval $[0,1]$ *into equally spaced subintervals, of mesh-size*



$$h \equiv \Delta x = \frac{1}{M}, \qquad \text{(space mesh-size)}$$

with mesh points

$$x_i = ih \qquad i = 0 : M. \qquad \text{(space nodes)}$$

Similarly, let the constant time step be

$$\Delta t = \frac{T}{N} \qquad \text{(time step)}$$

and the time nodes be

$$t_j = j\Delta t, \qquad j = 0 : N, \qquad \text{(time nodes)}$$

i.e., we consider the evolution of the PDE on a generic time-interval $[0,T]$.

First, let us write a few semi-discrete in time approximations, where only the time derivatives are approximated, for all $x \in [0,1]$*:*

$$\frac{u(x,t+\Delta t) - u(x,t)}{\Delta t} - \frac{\partial^2 u}{\partial x^2}(x,t) \approx 0 \qquad \text{(forward-Euler)}$$

$$\frac{u(x,t+\Delta t)-u(x,t)}{\Delta t}-\frac{\partial^2 u}{\partial x^2}(x,t+\Delta t)\approx 0 \qquad \text{(backward-Euler)}$$

$$\frac{u(x,t+\Delta t)-u(x,t)}{\Delta t}-\frac{1}{2}\Big(\frac{\partial^2 u}{\partial x^2}(x,t+\Delta t)+\frac{\partial^2 u}{\partial x^2}(x,t)\Big)\approx 0 \qquad \text{(trapezoidal)}$$

$$\frac{u(x,t+\Delta t)-u(x,t)}{\Delta t}-\frac{\partial^2 u}{\partial x^2}(x,t+\Delta t/2)\approx 0 \qquad \text{(midpoint)}$$

1.1. Forward Euler in time approximations. *Using the* (second-order central) *in the* (forward-Euler) *equation above, we obtain the following fully-discrete time and space approximation:*

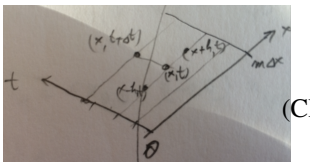
$$u(x,t+\Delta t)-u(x,t)-\frac{\Delta t}{h^2}\Big(u(x-h,t)-2u(x,t)+u(x+h,t)\Big)\approx 0 \tag{1.2}$$

equivalently

$$u(x,t+\Delta t)\approx \frac{\Delta t}{h^2}u(x-h,t)+\Big(1-2\frac{\Delta t}{h^2}\Big)u(x,t)+\frac{\Delta t}{h^2}u(x+h,t). \tag{1.3}$$

We note that if the following Courant-Friedrics-Levy (CFL) condition holds

$$\Delta t\le \frac{h^2}{2} \qquad \text{(CFL)}$$



(or equivalently $1-2\frac{\Delta t}{h^2}\ge 0$*) then all the coefficients in the right-hand side of* (1.3) *are positive and*

$$\begin{aligned} u(x,t+\Delta t) &\approx \frac{\Delta t}{h^2}u(x-h,t)+\Big(1-2\frac{\Delta t}{h^2}\Big)u(x,t)+\frac{\Delta t}{h^2}u(x+h,t)\\ &\le \frac{\Delta t}{h^2}\max_{x\in(0,1)}u(x,t)+\Big(1-2\frac{\Delta t}{h^2}\Big)\max_{x\in(0,1)}u(x,t)+\frac{\Delta t}{h^2}\max_{x\in(0,1)}u(x,t)=\max_{x\in(0,1)}u(x,t), \end{aligned}$$

which represent a discrete version of the (1.1).

NOTATION 1.1. *Let us denote by* u_{ij} *the finite difference approximation solutions*

$$u_{ij}\approx u(x_i,t_j)$$

where the $\{x_i=ih\}_{i=0:M}$ *and* $\{t_j\}_{j=0:N}$ *are the space nodes and time nodes defined in* (space nodes) *and respectively* (time nodes).

The fully-discrete approximation (1.2)-(1.3)*, which uses the* (forward-Euler) *and* (second-order central) *discretization formulae, written for* $x=x_i$ *and* $t=t_j$*, gives:*

$$u_{i,j+1}=\sigma u_{i-1,j}+(1-2\sigma)u_{ij}+\sigma u_{i+1,j}, \qquad \text{(FE time-marching problem)}$$

where

$$\sigma=\frac{\Delta t}{h^2}.$$

We note also that (FE time-marching problem)*, using the in vector notation, writes as*

$$\mathbf{u}_{j+1}=A_{\mathbf{FE}}\mathbf{u}_j,$$

where

$$\mathbf{u}_j = \begin{bmatrix} u_{1j} \\ u_{2j} \\ \vdots \\ u_{n-1,j} \end{bmatrix}, \qquad A_{\mathbf{FE}} = \begin{bmatrix} 1-2\sigma & \sigma & & & \\ \sigma & 1-2\sigma & \sigma & & \\ & \sigma & 1-2\sigma & \sigma & \\ \vdots & & & \ddots & \\ & & & \sigma & 1-2\sigma \end{bmatrix} \tag{1.4}$$

QUESTION*: for what values of* $\sigma = \frac{\Delta t}{h^2}$ *is the tridiagonal matrix* $A_{\mathbf{FE}}$ *above diagonally dominant, with positive diagonal elements (M-matrix - see Proposition 2.6)? See also the* (CFL) *condition.*

PROPOSITION 1.1 (Discrete Maximum Principle). *Provided the* (CFL) *condition holds, i.e., the timestep and the space mesh size satisfy* $\Delta t \leq \frac{h^2}{2}$*, the matrix* $A_{\mathbf{FE}}$ *is an M-matrix, and the discrete version of the* (1.1) *above writes*

$$u_{i,j+1} \leq \max_{\ell=0:n} u_{\ell,j}. \qquad \text{(discrete Maximum Principle)}$$

PROOF. Let us denote first $u_{ij} := \max_{\ell=0:M} u_{\ell,j}$. Then the (FE time-marching problem) gives

$$u_{i,j+1} = \sigma u_{i-1,j} + (1-2\sigma)u_{ij} + \sigma u_{i+1,j} \leq \sigma u_{i,j} + (1-2\sigma)u_{ij} + \sigma u_{i,j} = u_{i,j} = \max_{\ell=0:M} u_{\ell,j},$$

which means that, as time increases, the values of the 'temperature' u at every space point is less than the previous temperature, at any other point in the space domain. □

Result: *FE time-marching Algorithm*
for $j = 0 : N-1$ **do**
  for $i = 1 : M-1$ **do**
    $u_{i,j+1} = \sigma u_{i-1,j} + (1-2\sigma)u_{ij} + \sigma u_{i+1,j}$.
  end
end

EXAMPLE 1.1. *Let us consider the first 'new' values in time, i.e., at* $t_1 = \Delta t$ *(or letting* $j = 0$ *in the* (FE time-marching problem)*), starting from the initial condition* $\{u_{j,0}\}_{j=0:M}$*:*

$$u_{i,1} = \sigma u_{i-1,0} + (1-2\sigma)u_{i0} + \sigma u_{i+1,0} \qquad \forall i = 1 : M-1,$$
$$\textit{where} \quad u_{0,1}, u_{n,1} \quad \textit{are provided by the boundary conditions.}$$

For example, for $i = 1$ *we have*

$$u_{11} = \sigma u_{0,0} + (1-2\sigma)u_{10} + \sigma u_{2,0}.$$

REMARK 1.1. *The* (CFL) *condition, a sufficient for the* (discrete Maximum Principle) *to hold, imposes a quite strict restriction on the space mesh size, making the* (forward-Euler) *time-approximation 'slow'. For example, compared to a spatial mesh size:*

$$h = 10^{-2},$$

the (CFL) *condition requires a 'super-small' time-step:*

$$\Delta t = 5 * 10^{-5}.$$

For this reason, we turn our attention to the implicit methods in Sections 1.2 and 1.3, which do not require a (CFL) condition for stability, nor for a discrete maximum principle.
Run

`'PDE_heat_Forward_Difference(0,1,0,1,50,50)'`

CFL = 50 ≫ 0.5
and

`'PDE_heat_Forward_Difference(0,1,0,1,8,900)'`

CFL = 0.0711111 ≪ 0.5

1.2. Backward Euler in time approximations. *Using the (second-order central) in the (backward-Euler) equation, we obtain similarly to Section 1.1 the following fully-discrete time and space approximation:*

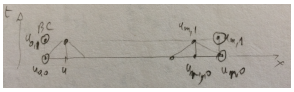
$$u(x,t)-u(x,t-\Delta t)-\frac{\Delta t}{h^2}\Big(u(x-h,t)-2u(x,t)+u(x+h,t)\Big)\approx 0 \tag{1.5}$$

equivalently

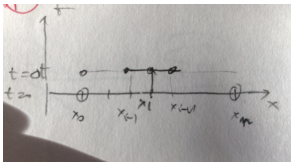
$$-\frac{\Delta t}{h^2}u(x-h,t)+u(x,t)\Big(1+2\frac{\Delta t}{h^2}\Big)-\frac{\Delta t}{h^2}u(x+h,t)\approx u(x,t-\Delta t). \tag{1.6}$$

With the Notations 1.1, and $x_i = ih, t_j = j\Delta t$, the fully-discrete (finite difference) approximations writes

$$-\frac{\Delta t}{h^2}u_{i-1,j}+\Big(1+2\frac{\Delta t}{h^2}\Big)u_{ij}-\frac{\Delta t}{h^2}u_{i+1,j}=u_{i,j-1}. \qquad \text{(BE time-marching heat equation)}$$



Result: *BE time-marching Algorithm*
for $j = 1 : N$ **do**
  for $i = 1 : M-1$ **do**
    $-\sigma u_{i-1,j}+\big(1+2\sigma\big)u_{i,j}-\sigma u_{i+1,j}=u_{i,j-1}.$
  end
end



The algorithm above writes, in matrix form as follows ($\sigma=\frac{\Delta t}{h^2}$):

$$\begin{bmatrix} 1+2\sigma & -\sigma & & & \\ -\sigma & 1+2\sigma & -\sigma & & \\ & -\sigma & 1+2\sigma & -\sigma & \\ \vdots & & & \ddots & \\ & & & -\sigma & 1+2\sigma \end{bmatrix}\begin{bmatrix} u_{1j} \\ u_{2j} \\ u_{3j} \\ \vdots \\ u_{M-1,j} \end{bmatrix}=\begin{bmatrix} u_{1,j-1}+\sigma u_{0j} \\ u_{2,j-1} \\ u_{3,j-1} \\ \vdots \\ u_{M-1,j-1}+\sigma u_{n,j} \end{bmatrix} \tag{1.7}$$

Note that the matrix above

$$A_{\mathbf{BE}} = \begin{bmatrix} 1+2\sigma & -\sigma & & & \\ -\sigma & 1+2\sigma & -\sigma & & \\ & -\sigma & 1+2\sigma & -\sigma & \\ \vdots & & & \ddots & \\ & & & -\sigma & 1+2\sigma \end{bmatrix}$$

is a tridiagonal, strictly diagonally dominant matrix, with positive diagonal entries (hence an M-matrix by Proposition 2.6) for all values of $\sigma = \frac{\Delta t}{h^2}$.

PROPOSITION 1.2 (Discrete Maximum Principle). *For all h and Δt, the following discrete version of the (1.1) holds for the (BE time-marching heat equation). If the initial condition satisfies the bound:*

$$u(x,0) \in [\alpha,\beta] \qquad \textit{i.e.,} \qquad \alpha \le u_{i,0} \le \beta \qquad \forall i = 0:M$$

then the solution at all times satisfies the same bounds

$$\alpha \le u_{i,j} \le \beta \qquad \forall i = 0:M, j = 0:N. \qquad \text{(discrete Maximum Principle)}$$

PROOF. We will prove, by induction, only the right-hand side bound. Assume that $u_{i,j-1} \le \beta, \forall t_{j-1}$ (time nodes) and $\forall x_i$ (space nodes). Denoting

$$u_{i,j} := \max_{\ell} u_{\ell,j}$$

(hence $u_{i-1,j} \le u_{ij}$, $u_{i+1,j} \le u_{ij}$), by (BE time-marching heat equation) we obtain

$$u_{ij} \le -\sigma u_{i-1,j} + \Big(1+2\sigma\Big)u_{ij} - \sigma u_{i+1,j} = u_{i,j-1},$$

which yields by the induction assumption that $u_{ij} \le \beta$. □

Run

`'PDE_heat_Backward_Difference(0,1,0,1,50,50)'`

CFL = 50 ≫ 0.5

and

`'PDE_heat_Backward_Difference(0,1,0,1,8,900)'`

Compare with the results using 'PDE heat Backward Difference'.

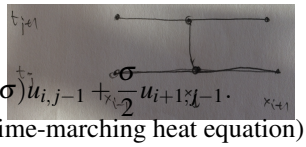
1.3. Midpoint / Crank-Nicolson time approximations. *Using the (second-order central) in the (trapezoidal) equation, we obtain similarly to Section 1.1 the following fully-discrete time and space approximation:*

$$\begin{aligned} &u(x,t) - u(x,t-\Delta t) \qquad (1.8)\\ &\approx \frac{\Delta t}{h^2}\Big(\frac{u(x-h,t)+u(x-h,t-\Delta t)}{2} - 2\frac{u(x,t)+u(x,t-\Delta t)}{2}) + \frac{u(x+h,t)+u(x+h,t-\Delta t)}{2}\Big) \end{aligned}$$

equivalently

$$\begin{aligned} &-\frac{1}{2}\frac{\Delta t}{h^2}u(x-h,t) + u(x,t)\Big(1+\frac{\Delta t}{h^2}\Big) - \frac{1}{2}\frac{\Delta t}{h^2}u(x+h,t) \qquad (1.9)\\ &\quad \approx \frac{1}{2}\frac{\Delta t}{h^2}u(x-h,t-\Delta t) + \Big(1-\frac{\Delta t}{h^2}\Big)u(x,t-\Delta t) + \frac{1}{2}\frac{\Delta t}{h^2}u(x+h,t-\Delta t). \end{aligned}$$

With the Notations 1.1, and $x_i = ih, t_j = j\Delta t$, the fully-discrete (finite difference) approximations writes



$$-\frac{\sigma}{2}u_{i-1,j}+(1+\sigma)u_{ij}-\frac{\sigma}{2}u_{i+1,j}=\frac{\sigma}{2}u_{i-1,j-1}+(1-\sigma)u_{i,j-1}+\frac{\sigma}{2}u_{i+1,j-1}.$$

(CN time-marching heat equation)

Result: *CN time-marching Algorithm*
for $j = 1 : N$ **do**
  for $i = 1 : M-1$ **do**

$$-\frac{\sigma}{2}u_{i-1,j}+(1+\sigma)u_{ij}-\frac{\sigma}{2}u_{i+1,j}=\underbrace{\frac{\sigma}{2}u_{i-1,j-1}+\Big(1-\sigma\Big)u_{i,j-1}+\frac{\sigma}{2}u_{i+1,j-1}}_{b_{ij}}.$$

  end
end

The algorithm above writes, in matrix form as follows ($\sigma = \frac{\Delta t}{h^2}$):

$$\begin{bmatrix} (1+\sigma) & -\frac{\sigma}{2} & & & \\ -\frac{\sigma}{2} & (1+\sigma) & -\frac{\sigma}{2} & & \\ & -\frac{\sigma}{2} & (1+\sigma) & -\frac{\sigma}{2} & \\ \vdots & & & \ddots & \\ & & & -\frac{\sigma}{2} & (1+\sigma) \end{bmatrix} \begin{bmatrix} u_{1j} \\ u_{2j} \\ u_{3j} \\ \vdots \\ u_{M-1,j} \end{bmatrix} = \begin{bmatrix} b_{1j} \\ b_{2j} \\ u_{3j} \\ \vdots \\ b_{M-1,j} \end{bmatrix}. \tag{1.10}$$

Note that the matrix above

$$A_{\mathbf{CN}} = \begin{bmatrix} (1+\sigma) & -\frac{\sigma}{2} & & & \\ -\frac{\sigma}{2} & (1+\sigma) & -\frac{\sigma}{2} & & \\ & -\frac{\sigma}{2} & (1+\sigma) & -\frac{\sigma}{2} & \\ \vdots & & & \ddots & \\ & & & -\frac{\sigma}{2} & (1+\sigma) \end{bmatrix}$$

is a tridiagonal, strictly diagonally dominant matrix, with positive diagonal entries (hence an M-matrix) for all values of $\sigma = \frac{\Delta t}{h^2}$.
See PDE_heat_CrankNicolson.m for a numerical example of heat diffusion.

REMARK 1.2. *The matrix in (1.10) is an M-matrix, strictly diagonally dominant matrix, with positive diagonal entries.*

PROPOSITION 1.3 (Discrete Maximum Principle). *For h and Δt such that $\sigma < 1$, the following discrete version of the (1.1) holds for the (CN time-marching heat equation). If the initial condition satisfies the bound:*

$$u(x,0) \in [\alpha,\beta] \qquad i.e., \qquad \alpha \leq u_{i,0} \leq \beta \qquad \forall i = 0 : M$$

then the solution at all times satisfies the same bounds

$$\alpha \leq u_{i,j} \leq \beta \qquad \forall i = 0 : M, j = 0 : N. \qquad \text{(CN discrete Maximum Principle)}$$

PROOF. The proof, by induction, is similar to the proof of Proposition 1.2. □

REMARK 1.3 (Errors). *The errors are measured (on the whole space-interval $x \in [0,1]$, and in the time interval $t \in [0,T]$) in the 'energy-norm', $L^2(0,1) \times L^2(0,T)$:*

$$\Delta t \sum_{j=1}^{N} h \sum_{i=1}^{M-1} |u(x_i,t_j) - u_{ij}|^2 \approx \int_0^{N\Delta t} \int_{x_0}^{x_M} |u_{\text{exact}} - u_{\text{approx}}^{\text{FD}}|^2 dx\,dt.$$

($L^2((0,1)\times(0,T))$ error)

For a comparison, here are the errors for the heat equation using the (BE time-marching heat equation), (CN time-marching heat equation) *and the* (FE time-marching problem) *methods, corresponding to the following values of the spatial mesh h and time step Δt:*

$$h = 0.01, \qquad \Delta t = 0.01$$

Method	*Root mean Square error*
BE	*0.005356756902686*
CN	*0.000081349744731*
FE	∞ *(the* (CFL) *number is 100 $\gg$ 0.5)*

2. Hyperbolic Equations

We are considering the following 1-dimensional wave equation, modeling the vibration of a string

$$\frac{\partial^2 u}{\partial t^2}(x,t) = \frac{\partial^2 u}{\partial x^2}(x,t) \qquad \forall x \in (0,1), \quad t \geq 0 \qquad \text{(wave equation)}$$

where $u(x,t)$ = the deflection, at time t, of a point which coordinate at rest is x. The (wave equation) *is endowed with the following initial conditions*

$$u(x,0) = f(x), \quad u_t(x,0) = 0, \qquad \forall x \in (0,1), \qquad \text{(I.C.)}$$

and boundary conditions

$$u(0,t) = u(1,t) = 0 \qquad \forall t \geq 0. \qquad \text{(B.C.)}$$

ASSUMPTION 2.1. *Here $f(\cdot) \in C^2(\mathbb{R})$ is a known function satisfying*

- *antisymmetric : $f(-x) = f(x)$*
- *2-periodic : $f(x+2) = f(x)$.*

PROPOSITION 2.1 (Exact solution). *The* (wave equation) *with the* (B.C.) *and* (I.C.) *above has the following* (exact solution)*:*

$$u(x,t) = \frac{1}{2}\Big(f(x+t) + f(x-t)\Big), \qquad \text{(exact solution)}$$

where f satisfies Assumption 2.1.

PROOF. Indeed,

$$u_t = \frac{1}{2}\Big(f'(x+t) - f'(x-t)\Big), \qquad u_{tt} = \frac{1}{2}\Big(f''(x+t) + f''(x-t)\Big),$$

and

$$u_x = \frac{1}{2}\Big(f'(x+t) + f'(x-t)\Big), \qquad u_{xx} = \frac{1}{2}\Big(f''(x+t) + f''(x-t)\Big).$$

Also the initial conditions are satisfied:

$$u(x,0) = \frac{1}{2}\Big(f(x) + f(x)\Big) = f(x),$$

$$u_t(x,0) = \frac{1}{2}\Big(f'(x) - f'(x)\Big) = 0,$$

as well as the boundary conditions

$$u(0,t) = \frac{1}{2}\Big(f(t) + f(-t)\Big) = \frac{1}{2}\Big(f(t) - f(t)\Big) = 0, \qquad \text{(using the antisymmetry of } f\text{)}$$

$$u(1,t) = \frac{1}{2}\Big(f(1+t) + f(1-t)\Big) = \frac{1}{2}\Big(f((t-1)+2) + f(1-t)\Big)$$

$$\equiv \frac{1}{2}\Big(f((t-1)) + f(1-t)\Big) \qquad \text{(using the 2-periodicity of } f\text{)}$$

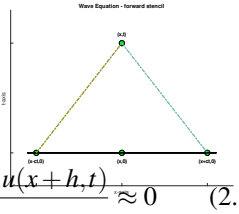
$$\equiv \frac{1}{2}\Big(-f((1-t)) + f(1-t)\Big) = 0. \qquad \text{(using the antisymmetry of } f\text{)}$$

This completes the proof. □

AVAILABLE SOFTWARE*:*

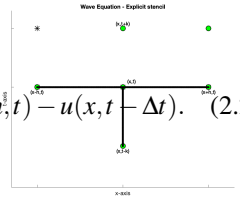
- FEniCS *is a 'finite element' software library for* Solving PDEs in Python *or* C++.
- FreeFEM *is another free and open source 'finite element' software to solve PDEs . (Click here for the* documentation *or an example in* acoustics*.)*
- MATLAB*'s* PDE toolbox *also uses 'finite elements' to solve structural mechanics, heat transfer, and general partial differential equations (PDEs). (Click here for the* documentation *or some* wave equation *examples.)*

2.1. Finite difference approximations. *Using the* (second-order central) *(for the time and space finite difference approximations) in the* (wave equation) *equation, we obtain similarly to Sections 1.1 and 1.2 the following fully-discrete time and space approximation (evaluated at* (x,t)*):*



$$\frac{u(x,t+\Delta t) - 2u(x,t) + u(x,t-\Delta t)}{\Delta t^2} - \frac{u(x-h,t) - 2u(x,t) + u(x+h,t)}{h^2} \approx 0 \tag{2.1}$$

equivalently



$$u(x,t+\Delta t) \approx 2\Big(1 - \frac{\Delta t^2}{h^2}\Big)u(x,t) + \frac{\Delta t^2}{\Delta x^2}u(x-h,t) + \frac{\Delta t^2}{\Delta x^2}u(x+h,t) - u(x,t-\Delta t). \tag{2.2}$$

The above approximation of the wave equation (2.2) *is endowed with the* (B.C.)

$$u(0,t) = u(1,t) = 0. \qquad \text{(B.C. wave eq)}$$

The initial conditions (I.C.) $(u(x,0) = f(x), u_t(x,0) = 0)$ *can be approximated in the following ways.*

- *Using a first-order approximation:*

$$\begin{cases} u(x,0) = f(x), \\ 0 = u_t(x,0) \approx \dfrac{u(x,\Delta t) - u(x,0)}{\Delta t} \quad \Rightarrow \quad u(x,\Delta t) \approx u(x,0) \equiv f(x), \end{cases} \qquad \forall x \in (0,1)$$

(1st order approx I.C. wave eq)

yielding a low accuracy propagating wave.

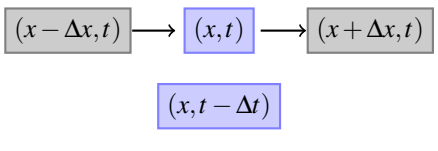


FIGURE 1. Wave-equation explicit stencil.

- *Using a second-order* (central difference approximation) *for the first derivative* $u_t(x,0)$ *we obtain*

$$\begin{cases} u(x,0) = f(x), \\ 0 = u_{x,0} \approx \dfrac{u(x,0+\Delta t) - u(x,0-\Delta t)}{2\Delta t} \quad \Rightarrow \quad u(x,\Delta t) \approx u(x,-\Delta t) \end{cases} \qquad \forall x \in (0,1).$$

(2nd order approx I.C. wave eq)

Notice here that the value $u(x,-\Delta t)$ *is a fictitious value, being evaluated at a time outside the time interval* $[0,T]$*. But this value actually gives important information. Let us now consider the finite difference equation* (2.2) *evaluated at time* $t = 0$*:*

$$u(x,0+\Delta t) \approx 2\Big(1-\frac{\Delta t^2}{h^2}\Big)u(x,0) + \frac{\Delta t^2}{\Delta x^2}u(x-h,0) + \frac{\Delta t^2}{\Delta x^2}u(x+h,0) - u(x,0-\Delta t)$$

(using (2nd order approx I.C. wave eq))

$$= 2\Big(1-\frac{\Delta t^2}{h^2}\Big)f(x) + \frac{\Delta t^2}{\Delta x^2}u(x-h,0) + \frac{\Delta t^2}{\Delta x^2}u(x+h,0) - u(x,\Delta t)$$

which (moving the last term in the RHS to the LHS and multiplying by $\frac{1}{2}$*) can be rearranged as*

$$u(x,\Delta t) \approx \Big(1-\frac{\Delta t^2}{h^2}\Big)f(x) + \frac{1}{2}\frac{\Delta t^2}{\Delta x^2}u(x-h,0) + \frac{1}{2}\frac{\Delta t^2}{\Delta x^2}u(x+h,0), \qquad \text{(1st time level } \Delta t\text{)}$$

for all $x \in (0,1)$*.*
From the second time level on, i.e., for $t = \Delta t, 2\Delta t, \ldots$*, we use* (2.2)

$$u(x,2\Delta t) \approx 2\Big(1-\frac{\Delta t^2}{h^2}\Big)u(x,\Delta t) + \frac{\Delta t^2}{\Delta x^2}u(x-h,\Delta t) + \frac{\Delta t^2}{\Delta x^2}u(x+h,\Delta t) - \underbrace{u(x,0)}_{=f(x)}$$

(2nd time level $2\Delta t$)

$$u(x,3\Delta t) \approx 2\Big(1-\frac{\Delta t^2}{h^2}\Big)u(x,2\Delta t) + \frac{\Delta t^2}{\Delta x^2}u(x-h,2\Delta t) + \frac{\Delta t^2}{\Delta x^2}u(x+h,2\Delta t) - u(x,\Delta t)$$

(3rd time level $3\Delta t$)

$\vdots$

See PDE_hyperbolic.m for a numerical example of the wave equation.

2.2. Transport equation.

REMARK 2.1. *The second-order homogeneous* (wave equation) *can be obtained from the first-order one-dimensional transport equation:*

$$\frac{\partial u}{\partial t}(x,t) + \frac{\partial u}{\partial x}(x,t) = 0 \qquad \forall x \in (0,1), \quad t \geq 0. \qquad \text{(1-D transport equation)}$$

PROOF. Indeed, taking partial derivatives with respect to t, and w.r.t. x, respectively

$$\frac{\partial^2 u}{\partial t^2}(x,t) + \cancel{\frac{\partial^2 u}{\partial t \partial x}(x,t)} = 0,$$
$$\cancel{\frac{\partial^2 u}{\partial x \partial t}(x,t)} + \frac{\partial^2 u}{\partial x^2}(x,t) = 0,$$

and subtracting

$$\frac{\partial^2 u}{\partial t^2}(x,t) - \frac{\partial^2 u}{\partial x^2}(x,t) = 0,$$

we obtain (wave equation). □

REMARK 2.2 (Lax-Wendroff method). *The Lax-Wendroff method is a finite difference method, second-order accurate both in time and in space, and was introduced by Peter Lax and Burton Wendroff. It solves the first-order hyperbolic equation*

$$\frac{\partial u}{\partial t}(x,t) + c\frac{\partial u}{\partial x}(x,t) = 0$$

by actually computing an approximate solution to a problem with a bit of diffusion

$$\frac{\partial w}{\partial t}(x,t) + c\frac{\partial w}{\partial x}(x,t) = \Delta t \frac{c^2}{2}\frac{\partial^2 w}{\partial x^2}(x,t).$$

One can motivate this by writing the Taylor approximation of $u(x,t+\Delta t)$ about $u(x,t)$ and using the original equation $u_t + cu_x = 0$ (hence $u_{tt} = c^2 u_{xx}$), namely

$$\begin{aligned} u(x,t+\Delta t) &= u(x,t) + \Delta t \frac{\partial u}{\partial t}(x,t) + \frac{1}{2}\Delta t^2 \frac{\partial^2 u}{\partial t^2}(x,t) + \mathscr{O}(\Delta t^3) \\ &= u(x,t) - \Delta t c \frac{\partial u}{\partial x}(x,t) + \frac{1}{2}\Delta t^2 c^2 \frac{\partial^2 u}{\partial x^2}(x,t) + \mathscr{O}(\Delta t^3), \end{aligned}$$

which is then approximated using the (first-order central) *and* (second-order central) *approximations of the first- and second- derivatives in space:*

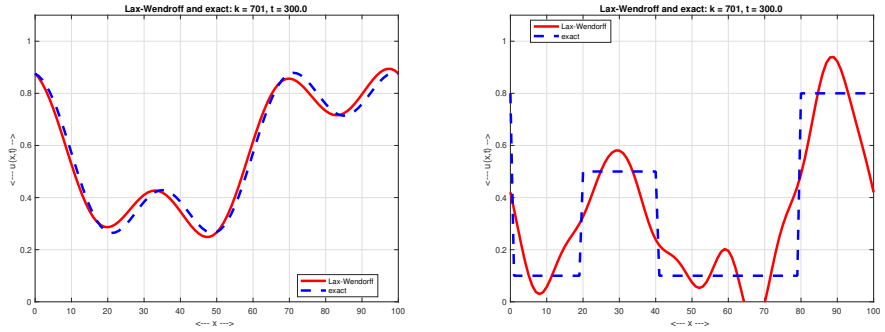
$$u(x,t+\Delta t) = u(x,t) - c\frac{\Delta t}{2h}\big(u(x+h,t) - u(x-h,t)\big) + c^2\frac{\Delta t^2}{2h^2}\big(u(x+h,t) - 2u(x,t) + u(x-h,t)\big) + \mathscr{O}(\Delta t^3 + h^2).$$

giving

$$v(x,t+\Delta t) = v(x,t) - c\frac{\Delta t}{2h}\big(v(x+h,t) - v(x-h,t)\big) + c^2\frac{\Delta t^2}{2h^2}\big(v(x+h,t) - 2v(x,t) + v(x-h,t)\big).$$

(Lax-Wendroff method)

See Lax-Wendroff.m for solving the (1-D transport equation) *using the* (Lax-Wendroff method) *method, for a smooth and a non-smooth initial condition.*



In 2 space dimensions, the transport equation writes

$$\frac{\partial u}{\partial t}(x,y,t) + \mathbf{v}\cdot\nabla u(x,y,t) = 0 \qquad \forall (x,y)\in\Omega\subset\mathbb{R}^2, \quad t\geq 0, \quad \text{(2-D transport equation)}$$

where Ω is a bounded domain in $\mathbb{R}^2$, $\mathbf{v} = (v_1, v_2)^T$ is a vector, and (using the (gradient) definition) $\mathbf{v}\cdot\nabla u \equiv v_1\cdot\partial_x u + v_2\cdot\partial_y u$.

REMARK 2.3 (Conservation law). *Let $u(x,t)$ in (1-D transport equation) represent the concentration of a substance at time t and point x in space (in the (2-D transport equation) would represent the concentration at the point (x,y) in space, transported by the vector $\mathbf{v}$). Integrating (1-D transport equation) on the whole space domain $x\in(0,1)$ we obtain*

$$\begin{aligned} 0 &= \int_0^1 \frac{\partial u}{\partial t}(x,t)dx + \int_0^1 \frac{\partial u}{\partial x}(x,t)dx = \frac{\partial}{\partial t}\Big(\int_0^1 u(x,t)dx\Big) + u(x,t)\Big|_{x=0}^{x=1} \\ &= \frac{\partial}{\partial t}\Big(\int_0^1 u(x,t)dx\Big) + \underbrace{(u(1,t)-u(0,t))}_{=0 \text{ by (B.C. wave eq)}} = \frac{\partial}{\partial t}\Big(\underbrace{\int_0^1 u(x,t)dx}_{\text{total mass}}\Big), \end{aligned}$$

which means that the total mass

$$m_{\text{total}}(t) := \int_0^1 u(x,t)dx \qquad \text{(total mass)}$$

is constant in time, as its derivative with respect to time is 0.

3. Elliptic Equations

We recall some classical elliptic equations:

$$\Delta u(\mathbf{x}) = 0, \qquad \forall \mathbf{x} = (x_1,x_2,x_3)^T \in \Omega\subset\mathbb{R}^3 \qquad \text{(Laplace equation)}$$

(here $u(\mathbf{x})$ describes the steady distribution of heat, or the steady distribution of electrical charge in a bounded domain Ω.)

$$-\Delta u(\mathbf{x}) = g(x) \qquad \text{(Poisson equation)}$$

$$\Delta u(\mathbf{x}) + f(\mathbf{x})u(\mathbf{x}) = g(\mathbf{x}) \qquad \text{(Helmholtz equation)}$$

(looking for oscillatory solutions of the wave equation)

Usually these equations are endowed with the following types of boundary conditions:

- *Dirichlet boundary conditions, where the values of the unknown function are prescribed:*

$$u(\mathbf{x}) = \Phi(\mathbf{x}) \qquad \forall x\in\partial\Omega \qquad \text{(Dirichlet B.C.)}$$

  where $\Phi(\cdot)$ is given on the $\partial\Omega$, the boundary of the domain Ω.
- *Neumann boundary conditions, where the values of the flux are prescribed:*

$$\frac{\partial u}{\partial \mathbf{n}} := \nabla u(\mathbf{x})\cdot\mathbf{n} = \Psi(\mathbf{x}) \qquad \forall x\in\partial\Omega \qquad \text{(Neumann B.C.)}$$

  where $\Psi(\cdot)$ is given on the $\partial\Omega$, the boundary of the domain Ω, and $\mathbf{n}$ is the normal vector to the boundary $\partial\Omega$.
- *Robin (mixed Dirichlet and Neumann) boundary conditions, where the flux depends linearly on the unknown function:*

$$\nabla u(\mathbf{x})\cdot\mathbf{n} + \alpha u(\mathbf{x}) = \Theta(\mathbf{x}) \qquad \text{(Robin B.C.)}$$

where $\Theta(\cdot)$ is given, $\alpha \in \mathbb{R}$.

REMARK 3.1 (Finite Difference approximations of the Laplacian in 1-D).

In 1-D space dimension, using the central difference formula (second-order central)*, the Laplacian has the following approximation*

$$u''(x) = \frac{u(x-\Delta x) - 2u(x) + u(x+\Delta x)}{\Delta x^2} + \frac{u^{(4)}(\xi)}{12}\Delta x^2,$$

where $\xi \in (x-\Delta x, x+\Delta x)$ is an intermediary arbitrary point.

REMARK 3.2 (Finite Difference approximations of the Laplacian in 2-D).

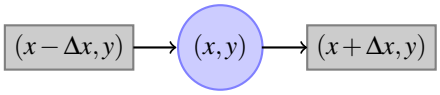


FIGURE 2. 1-D stencil

In 2-D space dimension, similarly - we use the central difference formula (second-order central) *to approximate the second-order partial derivatives with respect to x and y*

$$u_{xx}(x,y) = \frac{u(x-\Delta x, y) - 2u(x,y) + u(x+\Delta x, y)}{\Delta x^2} + \frac{u^{(4)}_{xxxx}(\xi, y)}{12}\Delta x^2,$$
$$u_{yy}(x,y) = \frac{u(x, y-\Delta y) - 2u(x,y) + u(x, y+\Delta y)}{\Delta y^2} + \frac{u^{(4)}_{yyyy}(x, \theta)}{12}\Delta y^2,$$

where $\xi \in (x-\Delta x, x+\Delta x), \theta \in (y-\Delta y, y+\Delta y)$ are intermediary arbitrary points. Let now assume that the 2-D mesh is equally spaced, i.e., $\Delta x = \Delta y = h$. Then, using a five-point stencil, the Laplacian $\Delta u(x,y)$ has the following approximation

$$\Delta_5 u(x,y) := \frac{1}{h^2}\Big(u(x-h,y) + u(x+h,y) + u(x,y-h) + u(x,y+h) - 4u(x,y)\Big), \tag{five-point formula}$$

with the approximation error:

$$-\frac{h^2}{12}\Big(\frac{\partial^4 u}{\partial x^4}u(\xi, y) + \frac{\partial^4 u}{\partial y^4}u(x, \theta)\Big).$$

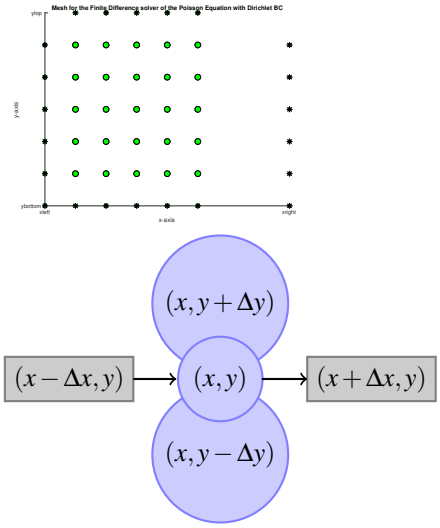


FIGURE 3. 2-D stencil

On a square space domain $\Omega = [a,b] \times [c,d]$*, we consider the following mesh points*

$$x_i = i\Delta x, \quad i = 0 : n+1,$$
$$y_j = j\Delta y, \quad j = 0 : m+1,$$

where the mesh sizes are

$$\Delta x = \frac{b-a}{n}, \qquad \Delta y = \frac{d-c}{m}.$$

This convention yields the following mesh nodes:

$$(x_i, y_j), \qquad i = 0 : n+1, \; j = 0 : m+1. \quad \text{(mesh nodes)}$$

We note that nodes with coordinates x_0 *and* x_{n+1} *are on the vertical left and right boundaries, while the nodes with coordinates* y_0 *and* y_{m+1} *are on the horizontal bottom and top boundaries. As before, we denote the approximation solutions*

$$u(x_i, y_j) \approx u_{ij}$$

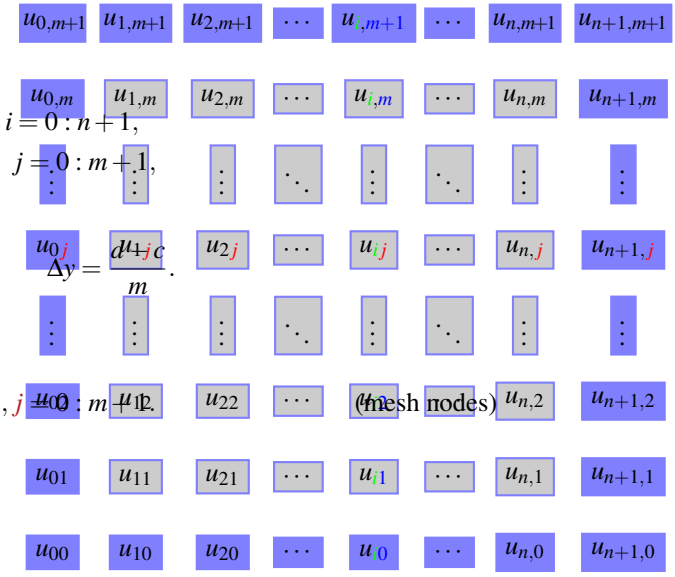


FIGURE 4. Natural ordering: left $\rightarrow$ right, bottom $\rightarrow$ top

while the exact values of the given functions f, g *on the grid nodes are*

$$f(x_i, y_j) := f_{ij}, \qquad g(x_i, y_j) := g_{ij}.$$

Let consider now the approximation of the (Helmholtz equation)

$$\Delta u(\mathbf{x}) + f(\mathbf{x})u(\mathbf{x}) = g(\mathbf{x}) \qquad \text{(Helmholtz equation)}$$

evaluated at a generic 'interior' node (x_i, y_j)*:*

$$\Delta u(x_i, y_j) + f(x_i, y_j)u(x_i, y_j) = g(x_i, y_j), \qquad \forall i = 1 : n, j = 1 : m$$

by finite differences, with the (five-point formula)

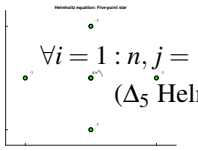
$$(\Delta)_5\, u_{i,j} + f_{ij}u_{ij} = g_{ij},$$

namely

$$u_{i+1,j} + u_{i-1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij} + h^2 f_{ij}u_{ij} = h^2 g_{ij},$$

which by multiplication with -1 *and rearrangements writes*

$$-u_{i+1,j} - u_{i-1,j} - u_{i,j-1} - u_{i,j+1} + \Big(4 - h^2 f_{ij}\Big)u_{ij} = -h^2 g_{ij},$$



$$\forall i = 1:n, j = 1:m.$$

(Δ_5 Helmholtz)

The equation above approximates the continuous problem at the interior nodes, which leaves the boundary conditions to be approximated. For simplicity, we consider here the homogeneous (Dirichlet B.C.)

$$u(\mathbf{x}) = \Phi(\mathbf{x}) \qquad \forall \mathbf{x} \in \partial\Omega$$

which evaluated on the boundary nodes is

$$u_{ij} = \Phi_{ij} \qquad \forall (x_i, y_j) \in \partial\Omega \qquad \text{(Dirichlet B.C.)}$$

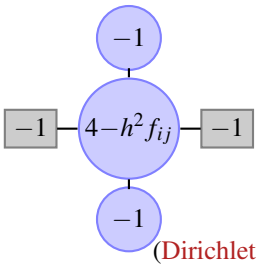


FIGURE 5. Coefficients of the grid

As seen in Figure 4, the 2D natural ordering of the unknowns and equations is from left-to-right and bottom-to-top. With this convention, we arrange the unknowns $\{u_{ij}\}_{i=1:n,j=1:m}$ *in vector form as follows*

$$\mathbf{u} = \begin{bmatrix} u_{11} \\ u_{21} \\ \cdots \\ u_{n1} \\ \hline u_{12} \\ u_{22} \\ \vdots \\ u_{1n} \\ \hline u_{1j} \\ u_{2j} \\ \vdots \\ u_{nj} \\ \hline u_{1m} \\ u_{2m} \\ \vdots \\ u_{nm} \end{bmatrix}$$

To fix ideas, let us write down, from the bottom row upward what is the information we have.

$$u_{i,0} = \Phi_{i0}, \quad \forall i = 0:m+1 \qquad (j = 0, \text{ from (Dirichlet B.C.)})$$

$$u_{01} = \Phi_{01}, \quad u_{n+1,1} = \Phi_{n+1,1}, \qquad (j = 1,\ i = 0 \text{ and } i = n+1 \text{ from (Dirichlet B.C.)})$$

$$-u_{i+1,1}-u_{i-1,1}-\underbrace{u_{i,0}}_{=\Phi_{i0}}-u_{i,2}+\Big(4-h^2f_{i1}\Big)u_{i1}=-h^2g_{i1},,\quad \forall i=1:n$$

$$(j=1, i=1:n, \text{ from } (\Delta_5 \text{ Helmholtz}))$$

$$u_{02}=\Phi_{02},\quad u_{n+1,2}=\Phi_{n+1,2},\qquad (j=2,\ i=0 \text{ and } i=n+1 \text{ from (Dirichlet B.C.)})$$

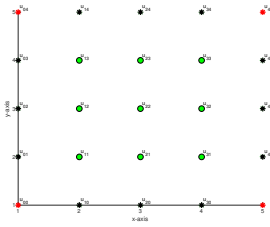
$$-u_{i+1,2}-u_{i-1,2}-u_{i,1}-u_{i,3}+\Big(4-h^2f_{i2}\Big)u_{i2}=-h^2g_{i2},,\quad \forall i=1:n$$

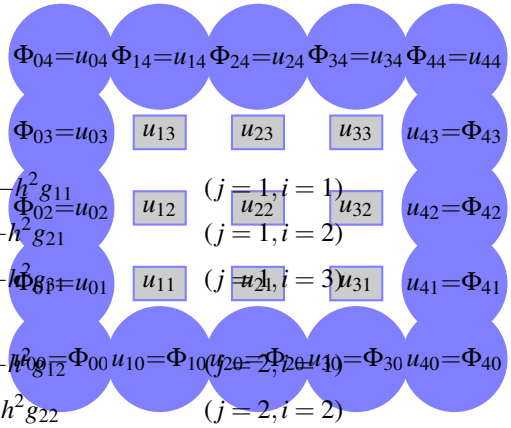
$$(j=2, i=1:n, \text{ from } (\Delta_5 \text{ Helmholtz}))$$

$\vdots$

For example, let $n=3$ and $m=3$. Then the unknowns on the grid are

$$\mathbf{u}=\left[\begin{array}{c} u_{11}\\ u_{21}\\ u_{31}\\ \hline u_{12}\\ u_{22}\\ u_{32}\\ \hline u_{13}\\ u_{23}\\ u_{33}\end{array}\right]$$





The discrete equations above, at the grid points, for determining the unknowns are:

$$\begin{aligned}
-u_{21}-\Phi_{01}-\Phi_{10}-u_{12}+(4-h^2f_{11})u_{11}&=-h^2g_{11} && (j=1,i=1)\\
-u_{31}-u_{11}-\Phi_{20}-u_{22}+(4-h^2f_{21})u_{21}&=-h^2g_{21} && (j=1,i=2)\\
-u_{41}-u_{21}-\Phi_{30}-u_{32}+(4-h^2f_{31})u_{31}&=-h^2g_{31} && (j=1,i=3)\\
\\
-u_{22}-\Phi_{02}-u_{11}-u_{13}+(4-h^2f_{12})u_{12}&=-h^2g_{12} && (j=2,i=1)\\
-u_{32}-u_{12}-u_{21}-u_{23}+(4-h^2f_{22})u_{22}&=-h^2g_{22} && (j=2,i=2)\\
-\Phi_{42}-u_{22}-u_{31}-u_{33}+(4-h^2f_{32})u_{32}&=-h^2g_{32} && (j=2,i=3)\\
\\
-u_{23}-\Phi_{03}-u_{12}-\Phi_{14}+(4-h^2f_{13})u_{13}&=-h^2g_{13} && (j=3,i=1)\\
-u_{33}-u_{13}-u_{22}-\Phi_{24}+(4-h^2f_{23})u_{23}&=-h^2g_{23} && (j=3,i=2)\\
-\Phi_{43}-u_{23}-u_{32}-\Phi_{34}+(4-h^2f_{33})u_{33}&=-h^2g_{33} && (j=3,i=3)
\end{aligned}$$

equivalently

$$\begin{aligned}
-u_{21}-u_{12}+(4-h^2f_{11})u_{11}&=-h^2g_{11}+\Phi_{01}+\Phi_{10}\\
-u_{31}-u_{11}-u_{22}+(4-h^2f_{21})u_{21}&=-h^2g_{21}+\Phi_{20}\\
-u_{41}-u_{21}-u_{32}+(4-h^2f_{31})u_{31}&=-h^2g_{31}+\Phi_{30}\\
-u_{22}-u_{11}-u_{13}+(4-h^2f_{12})u_{12}&=-h^2g_{12}+\Phi_{02}
\end{aligned}$$

$$
\begin{aligned}
-u_{32}-u_{12}-u_{21}-u_{23}+(4-h^2f_{22})u_{22} &= -h^2g_{22}\\
-u_{22}-u_{31}-u_{33}+(4-h^2f_{32})u_{32} &= -h^2g_{32}+\Phi_{42}\\
-u_{23}-u_{12}+(4-h^2f_{13})u_{13} &= -h^2g_{13}+\Phi_{03}+\Phi_{14}\\
-u_{33}-u_{13}-u_{22}+(4-h^2f_{23})u_{23} &= -h^2g_{23}+\Phi_{24}\\
-u_{23}-u_{32}+(4-h^2f_{33})u_{33} &= -h^2g_{33}+\Phi_{43}+\Phi_{34}.
\end{aligned}
$$

In matrix form this writes

$$
\left[\begin{array}{ccc|ccc|ccc}
4-h^2f_{11} & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0\\
-1 & 4-h^2f_{21} & -1 & 0 & -1 & 0 & 0 & 0 & 0\\
0 & -1 & 4-h^2f_{31} & 0 & 0 & -1 & 0 & 0 & 0\\
\hline
-1 & 0 & 0 & 4-h^2f_{12} & -1 & 0 & -1 & 0 & 0\\
0 & -1 & 0 & -1 & 4-h^2f_{22} & -1 & 0 & -1 & 0\\
0 & 0 & -1 & 0 & -1 & 4-h^2f_{32} & 0 & 0 & -1\\
\hline
0 & 0 & 0 & -1 & 0 & 0 & 4-h^2f_{13} & -1 & 0\\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 4-h^2f_{23} & -1\\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4-h^2f_{33}
\end{array}\right]
\left[\begin{array}{c}
u_{11}\\ u_{21}\\ u_{31}\\ \hline u_{12}\\ u_{22}\\ u_{32}\\ \hline u_{13}\\ u_{23}\\ u_{33}
\end{array}\right]
$$

$$
=\left[\begin{array}{c}
-h^2g_{11}+\Phi_{01}+\Phi_{10}\\
-h^2g_{21}+\Phi_{20}\\
-h^2g_{31}+\Phi_{30}\\ \hline
-h^2g_{12}+\Phi_{02}\\
-h^2g_{22}\\
-h^2g_{32}+\Phi_{42}\\ \hline
-h^2g_{13}+\Phi_{03}+\Phi_{14}\\
-h^2g_{23}+\Phi_{24}\\
-h^2g_{33}+\Phi_{34}+\Phi_{34}
\end{array}\right],
$$

i.e.,

$$
A\mathbf{u}=\mathbf{b}, \tag{3.1}
$$

where

$$
A=\left[\begin{array}{ccc|ccc|ccc}
4-h^2f_{11} & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0\\
-1 & 4-h^2f_{21} & -1 & 0 & -1 & 0 & 0 & 0 & 0\\
0 & -1 & 4-h^2f_{31} & 0 & 0 & -1 & 0 & 0 & 0\\
\hline
-1 & 0 & 0 & 4-h^2f_{12} & -1 & 0 & -1 & 0 & 0\\
0 & -1 & 0 & -1 & 4-h^2f_{22} & -1 & 0 & -1 & 0\\
0 & 0 & -1 & 0 & -1 & 4-h^2f_{32} & 0 & 0 & -1\\
\hline
0 & 0 & 0 & -1 & 0 & 0 & 4-h^2f_{13} & -1 & 0\\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 4-h^2f_{23} & -1\\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4-h^2f_{33}
\end{array}\right],
$$

$$\mathbf{b} = \begin{bmatrix} -h^2 g_{11} + \Phi_{01} + \Phi_{10} \\ -h^2 g_{21} + \Phi_{20} \\ -h^2 g_{31} + \Phi_{30} \\ \hline -h^2 g_{12} + \Phi_{02} \\ -h^2 g_{22} \\ -h^2 g_{32} + \Phi_{42} \\ \hline -h^2 g_{13} + \Phi_{03} + \Phi_{14} \\ -h^2 g_{23} + \Phi_{24} \\ -h^2 g_{33} + \Phi_{34} + \Phi_{34} \end{bmatrix}.$$

REMARK 3.3. *Note that matrix A in (3.1) is a sparse matrix, with*

$$a_{ij} = 0 \quad \textit{for all} \quad |i-j| > 3.$$

If the mesh-size h is halved (i.e., $n = m = 6$), the the size of the matrix is approximately **quadrupled**.

REMARK 3.4 (The Gauss-Seidel iterative method for solving the (Helmholtz equation)). *Let consider now solving the linear system generated by the finite differences (five-point formula) approximation of the (Helmholtz equation), namely system (Δ_5 Helmholtz) above:*

$$-u_{i+1,j} - u_{i-1,j} - u_{i,j-1} - u_{i,j+1} + \Big(4 - h^2 f_{ij}\Big) u_{ij} = -h^2 g_{ij}, \qquad \forall i = 1:n, j = 1:m.$$

This can be written as (3.1) and then solved by a direct method (e.g., Gaussian elimination). As noticed in Remark 3.3, it can happen that the mesh h must be chosen so small that Gaussian elimination (GE) is no longer practicable.

If the mesh is fairly fine, hence the size of the vector of unknowns and the size of the matrix is very large, the only choice might be to use an iterative method (recall the Jacobi method, Gauss-Seidel and GMRES).

Therefore we write the system above in the equivalent form

$$u_{ij} = \frac{1}{4 - h^2 f_{ij}} \Big(u_{i+1,j} + u_{i-1,j} + u_{i,j-1} + u_{i,j+1} - h^2 g_{ij} \Big), \qquad \forall i = 1:n, j = 1:m \textit{ (on the interior nodes).}$$

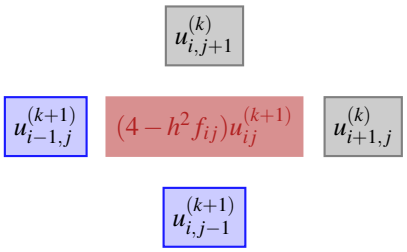
If we have approximate values of the unknowns at each grip point, this equation can be used to generate new values, by using an iterative method. For example, following the natural ordering in Figure 4 (left to right, bottom to top), the equation above can be solved iteratively by:

$$u_{ij}^{(k+1)} = \frac{1}{4 - h^2 f_{ij}} \Big(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k+1)} - h^2 g_{ij} \Big), \qquad \forall i = 1:n, j = 1:m \textit{ (on the interior nodes).}$$

The successive over-relaxation (SOR) method would then write

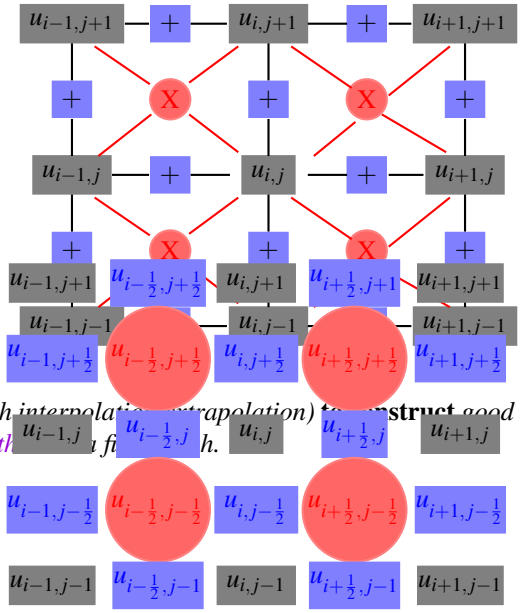
$$\begin{aligned} {}_{SOR}u_{ij}^{(k+1)} &= \omega u_{ij}^{(k)} + (1-\omega) u_{ij}^{(k+1)} \\ &= \omega u_{ij}^{(k)} + (1-\omega) \frac{1}{4 - h^2 f_{ij}} \Big(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k+1)} - h^2 g_{ij} \Big), \\ &= u_{ij}^{(k)} + (1-\omega) \frac{1}{4 - h^2 f_{ij}} \Big(\underbrace{u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k+1)} - u_{i,j}^{(k)} (4 - h^2 f_{ij}) - h^2 g_{ij}}_{\text{residual}} \Big). \end{aligned}$$

We remind that the SOR method converges for any initial guesses and $\forall\ 0 < \omega < 2$ *if and only if the matrix A is symmetric and positive definite (check !) provided A has positive diagonal elements, i.e., if and only if* $f_{ij} \leq 0$.



Therefore a **good strategy** *to solve Boundary Value Problems with linear Differential Equations is to*

- *start with a coarse mesh, and*
- *use Gaussian elimination (GE).*



If the accuracy is insufficient (h-mesh-size is too small), then one can use the results obtained on the coarse mesh (possibly with interpolation/extrapolation) to **construct** *good initial approximations for an iterative method for finer h.*

In constructing this initial approximations, one also needs the values of u at the midpoints of the squares.

LEMMA 3.1. *How to compute* $u_{i,j+\frac{1}{2}}$*?*

The (Helmholtz equation) evaluated at the node $(x_i, y_{j+\frac{1}{2}})$*:*

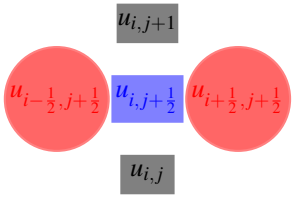
$$\Delta u(x_i, y_{j+\frac{1}{2}}) + f(x_i, y_{j+\frac{1}{2}})u(x_i, y_{j+\frac{1}{2}}) = g(x_i, y_{j+\frac{1}{2}})$$

is then approximated by (note that here we use (five-point formula) with half mesh-size $h/2$*)*

$$-\underbrace{u_{i,j+1}}_{\text{known by GE}} - \underbrace{u_{i,j}}_{\text{known by GE}} - \underbrace{u_{i-\frac{1}{2},j+\frac{1}{2}}}_{\text{unknown}} - \underbrace{u_{i+\frac{1}{2},j+\frac{1}{2}}}_{\text{unknown}} + \Big(4-(h/2)^2\underbrace{f_{i,j+\frac{1}{2}}}_{\text{data}}\Big)u_{i,j+\frac{1}{2}} = -(h/2)^2\underbrace{g_{i,j+\frac{1}{2}}}_{\text{data}},$$

hence $(x_i, y_{j+\frac{1}{2}})$ *is computable provided* $u_{i-\frac{1}{2},j+\frac{1}{2}}$ *and* $u_{i+\frac{1}{2},j+\frac{1}{2}}$ *are known!*

LEMMA 3.2. *How to compute* $u_{i+\frac{1}{2},j+\frac{1}{2}}$*?*



Use the (Δ_{cross}) *approximation of the Laplacian in the* (Helmholtz equation) *evaluated at the node* $(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}})$*:*

$$\Delta u(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}}) + f(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}}) u(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}}) = g(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}})$$

to obtain

$$-u_{i,j+1} - u_{i+1,j+1} - u_{ij} - u_{i+1,j} + \Big(4 - \frac{h^2}{2} f_{i+\frac{1}{2},j+\frac{1}{2}}\Big) u_{i+\frac{1}{2},j+\frac{1}{2}} = -\frac{h^2}{2} g_{i+\frac{1}{2},j+\frac{1}{2}}.$$

PROPOSITION 3.1 (Δ_{cross} approximation of the Laplacian).

$$\Delta_{\text{cross}} u_{i+\frac{1}{2},j+\frac{1}{2}} := \frac{u_{i,j+1} + u_{i+1,j+1} + u_{ij} + u_{i+1,j} - 4u_{i+\frac{1}{2},j+\frac{1}{2}}}{h^2/2} \qquad (\Delta_{\text{cross}})$$

PROOF. We use Taylor approximations in 2D:

$$\begin{aligned}\phi(x,y) &= \phi(x_0,y_0) + (x-x_0)\frac{\partial\phi}{\partial x}(x_0,y_0) + (y-y_0)\frac{\partial\phi}{\partial y}(x_0,y_0) \\ &\quad + \frac{1}{2}\Big[(x-x_0)^2\frac{\partial^2\phi}{\partial x^2}(x_0,y_0) + 2(x-x_0)(y-y_0)\frac{\partial^2\phi}{\partial x\partial y}(x_0,y_0) + (y-y_0)^2\frac{\partial^2\phi}{\partial y^2}(x_0,y_0)\Big] + \cdots\end{aligned}$$

to approximate

$$\begin{aligned}u(x_i,y_{j+1}) &= u(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}}) + \frac{h}{2}\Big(-\cancel{\frac{\partial u}{\partial x}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}})} + \cancel{\frac{\partial u}{\partial y}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}})}\Big) \\ &\quad + \frac{1}{2}\frac{h^2}{4}\Big[\frac{\partial^2 u}{\partial x^2}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}}) + \frac{\partial^2 u}{\partial y^2}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}}) - \cancel{2\frac{\partial^2 u}{\partial x\partial y}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}})}\Big] + \cdots, \\ u(x_{i+1},y_{j+1}) &= u(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}}) + \frac{h}{2}\Big(\cancel{\frac{\partial u}{\partial x}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}})} + \cancel{\frac{\partial u}{\partial y}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}})}\Big) \\ &\quad + \frac{1}{2}\frac{h^2}{4}\Big[\frac{\partial^2 u}{\partial x^2}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}}) + \frac{\partial^2 u}{\partial y^2}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}}) + \cancel{2\frac{\partial^2 u}{\partial x\partial y}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}})}\Big] + \cdots \\ u(x_i,y_j) &= u(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}}) + \frac{h}{2}\Big(-\cancel{\frac{\partial u}{\partial x}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}})} - \cancel{\frac{\partial u}{\partial y}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}})}\Big) \\ &\quad + \frac{1}{2}\frac{h^2}{4}\Big[\frac{\partial^2 u}{\partial x^2}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}}) + \frac{\partial^2 u}{\partial y^2}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}}) + \cancel{2\frac{\partial^2 u}{\partial x\partial y}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}})}\Big] + \cdots \\ u(x_{i+1},y_j) &= u(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}}) + \frac{h}{2}\Big(\cancel{\frac{\partial u}{\partial x}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}})} - \cancel{\frac{\partial u}{\partial y}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}})}\Big) \\ &\quad + \frac{1}{2}\frac{h^2}{4}\Big[\frac{\partial^2 u}{\partial x^2}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}}) + \frac{\partial^2 u}{\partial y^2}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}}) + \cancel{2\frac{\partial^2 u}{\partial x\partial y}(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}})}\Big] + \cdots\end{aligned}$$

Adding up and moving the $u(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}})$ terms to the left hand-side yields

$$u(x_i,y_{j+1}) + u(x_{i+1},y_{j+1}) + u(x_i,y_j) + u(x_{i+1},y_j) - 4u(x_{i+\frac{1}{2}},y_{j+\frac{1}{2}})$$

$$= \frac{h^2}{2}\Delta u(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}}) + \cdots,$$

and, finally, division by $\frac{h^2}{2}$ concludes the proof. □

DEFINITION 3.1 (nine-point discrete Laplacian [14]). *The nine-point discrete Laplacian is a linear combination of the* (five-point formula) *and* (Δ_{cross})*:*

$$\Delta_9 u(x,y) := \frac{2}{3}\Delta_5 u(x,y) + \frac{1}{3}\Delta_{\text{cross}} u_{x,y}. \qquad (\Delta_{\text{nine-point}})$$

PROPOSITION 3.2 (see [Fundamental Solutions of 9-point Discrete Laplacians] by Robert E. Lynch, 1992 [14]). *(See also* [Nine-point difference solutions for Poisson's equation] by J. Barkley Rosser, 1976 *in [16])*
Provided $u \in C^8(\Omega)$, the error in the ($\Delta_{\text{nine-point}}$) *approximation of the* (Laplacian) *operator is*

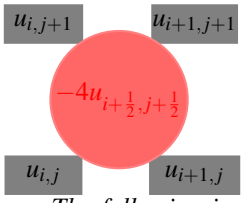
$$\Delta_9 u_{i+\frac{1}{2},j+\frac{1}{2}} - \Delta u(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}}) = \frac{1}{12}h^2\nabla^4 u(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}}) + \frac{1}{360}h^4\big(\nabla^6 u + 2\frac{\partial^4 u}{\partial x^2 \partial y^2}\nabla^2 u\big)(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}}) + \mathscr{O}(h^6), \tag{3.2}$$

which gives "optimal" 6-th order accuracy for harmonic functions.

3.1. Fundamental Solutions of 9-point Discrete Laplacians,by Robert E. Lynch, 1992**.** *Let L_α the 9-point difference operator defined by*

$$L_\alpha U_{j,k} = (2\alpha - 4)U_{j,k} \cdots$$

EXAMPLE 3.1.



The following is a simple reaction-diffusion model with solution - a spiral rotating around the center of the spatial domain (see e.g., [11, page 301] and the reference therein).

$$\begin{cases} \dfrac{\partial u}{\partial t} = \Delta u + \dfrac{1}{\varepsilon}u(1-u)\big(u - \dfrac{v+\beta}{\alpha}\big), \\ \dfrac{\partial v}{\partial t} = \delta\Delta v + u - v, \end{cases} \qquad (x,y) \in \Omega = (0,80)^2,\ t > 0,$$

$$\begin{cases} u_0(x,y,0) = \begin{cases} 0, & x < 40, \\ 1, & x \geq 0 \end{cases} \\ v_0(x,y,0) = \begin{cases} 0, & y < 40, \\ \frac{1}{2}\alpha, & y \geq 0 \end{cases} \end{cases} \qquad \text{(I.C.)}$$

$$\frac{\partial u}{\partial n} = \frac{\partial v}{\partial n} = 0, \qquad \text{(homogeneous Neumann B.C.)}$$

where the parameters are:

$$\delta = 0, \quad \varepsilon = 0.002, \quad \alpha = 0.25, \quad \beta = 0.001.$$

On a fixed spatial grid of 400×400 *($h = 0.2$), Figure 6 shows the plots of the solutions at $t = 10$ obtained with a finite difference 5-point Laplace discretization, and respectively with the 9-point Laplacian. Observe that the 5-point Laplacian exhibits a spiral of a 'square'*

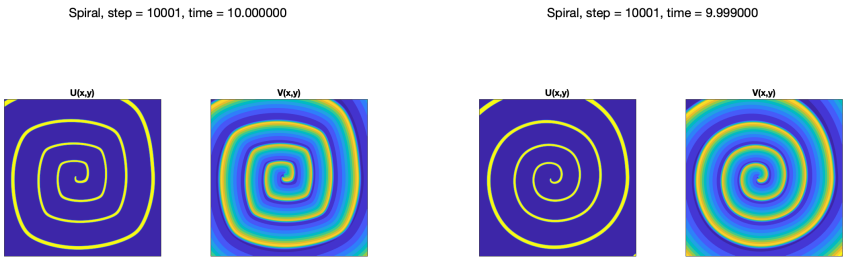


FIGURE 6. The spiral solutions of the reaction-diffusion model in Example 3.1 using the 5-point and 9-point Laplacians

form, aligned to the grid, while the 9-point Laplacian solution gives a 'round' spiral. With these parameters, the PDE is not well resolved. When finer grids are used, the differene between the results becomes smaller, both solutions converging to the exact solution. (See Spiral Reaction Diffusion.m.)

EXAMPLE 3.2. `Reaction diffusion equations`*: the* `Gray-Scott model`.

See Reaction-diffusion system for the following text:

"Reaction-diffusion systems are mathematical models which correspond to several physical phenomena. The most common is the change in space and time of the concentration of one or more chemical substances: local chemical reactions in which the substances are transformed into each other, and diffusion which causes the substances to spread out over a surface in space.

Reaction-diffusion systems are naturally applied in chemistry. However, the system can also describe dynamical processes of non-chemical nature. Examples are found in biology, geology and physics (neutron diffusion theory) and ecology. Mathematically, reaction-diffusion systems take the form of semi-linear parabolic partial differential equations (see Section 1). They can be represented in the general form

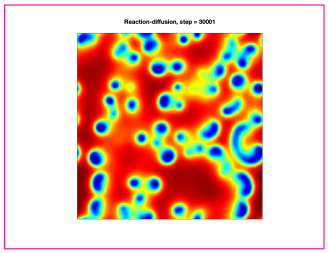
$$\partial_t \boldsymbol{q} = \boldsymbol{D}\nabla^2 \boldsymbol{q} + \boldsymbol{R}(\boldsymbol{q}),$$

where $\boldsymbol{q}(\boldsymbol{x},t)$ represents the unknown vector function, $\boldsymbol{D}$ is a diagonal matrix of diffusion coefficients, and R accounts for all local reactions. The solutions of reaction-diffusion equations display a wide range of behaviours, including the formation of travelling waves and wave-like phenomena as well as other self-organized patterns like stripes, hexagons or more intricate structure like dissipative solitons. Such patterns have been dubbed "Turing patterns". Each function, for which a reaction diffusion differential equation holds, represents in fact a concentration variable.

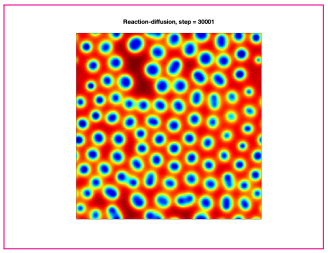
In particular, the `Gray-Scott model` *writes as:*

Equations :

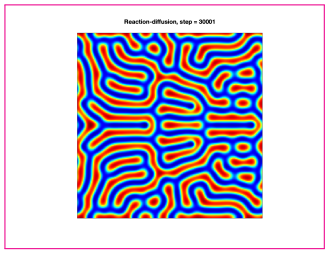
$$\frac{\partial u}{\partial t} = r_u \Delta u - uv^2 + f(1-u),$$
$$\frac{\partial v}{\partial t} = r_v \Delta v + uv^2 - (f+k)v,$$



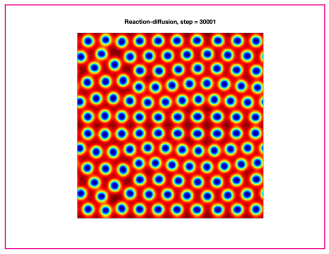
(A) "bubbles": $f = 0.012, k = 0.05$.



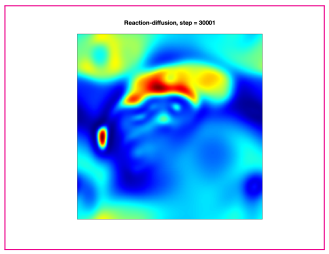
(B) "spots": $f = 0.022, k = 0.059$.



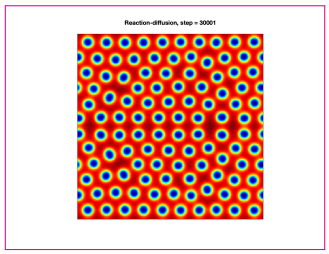
(C) "coral": $f = 0.0545, k = 0.0620$.



(D) "splits": $f = 0.0367, k = 0.0649$.



(E) "waves": $f = 0.025, k = 0.05$.



(F) "leopard skin": $f = 0.035, k = 0.065$.

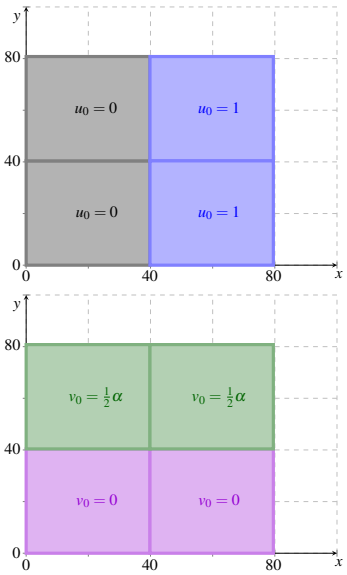
FIGURE 7. Gray-Scott

Chemical Reaction:

$$U + 2V \longrightarrow 3V,$$

$$V \longrightarrow P$$

where U, V, P, are chemical species, u, v represent their concentrations, $r_u r_v$ are their diffusion rates, k is the rate of conversion of V to P, f is the rate of the process that feeds U and drains U, V and P.


y
80
40
0
$u_0 = 0$
$u_0 = 1$
$u_0 = 0$
$u_0 = 1$
0
40
80
x
y
80
40
0
$v_0 = \frac{1}{2}\alpha$
$v_0 = \frac{1}{2}\alpha$
$v_0 = 0$
$v_0 = 0$
0
40
80
x

Bibliography

[1] O. AXELSSON, *Iterative solution methods*, Cambridge University Press, Cambridge, 1994.

[2] V. BARBU, *Partial differential equations and boundary value problems*, vol. 441 of Mathematics and its Applications, Kluwer Academic Publishers, Dordrecht, 1998. Translated and revised from the 1993 Romanian original by the author.

[3] J. BURKARDT AND C. TRENCHEA, *Refactorization of the midpoint rule*, Appl. Math. Lett., 107 (2020), p. 106438.

[4] G. G. DAHLQUIST AND Å. BJÖRCK, *Numerical methods*, Dover Publications, Inc., Mineola, NY, 2003. Translated from the Swedish by Ned Anderson, Reprint of the 1974 English translation.

[5] J. W. DEMMEL, *Applied numerical linear algebra*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[6] C. W. GEAR, *Numerical initial value problems in ordinary differential equations*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1971.

[7] G. H. GOLUB AND C. F. VAN LOAN, *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, third ed., 1996.

[8] D. F. GRIFFITHS AND D. J. HIGHAM, *Numerical methods for ordinary differential equations*, Springer Undergraduate Mathematics Series, Springer-Verlag London, Ltd., London, 2010. Initial value problems.

[9] W. HACKBUSCH, *Iterative solution of large sparse systems of equations*, vol. 95 of Applied Mathematical Sciences, Springer, [Cham], second ed., 2016.

[10] A. L. HODGKIN AND A. F. HUXLEY, *A quantitative description of membrane current and its application to conduction and excitation in nerve*, The Journal of Physiology, 117 (1952), pp. 500–544.

[11] W. HUNDSDORFER AND J. VERWER, *Numerical solution of time-dependent advection-diffusion-reaction equations*, vol. 33 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2003.

[12] W. LAYTON, W. PEI, AND C. TRENCHEA, *Time step adaptivity in the method of Dahlquist, Liniger and Nevanlinna*, Advances in Computational Science and Engineering, 1 (2023), pp. 320–350.

[13] W. LAYTON AND M. SUSSMAN, *Numerical Linear Algebra*, World Scientific, 2020.

[14] R. E. LYNCH, *Fundamental solutions of nine-point discrete Laplacians*, Appl. Numer. Math., 10 (1992), pp. 325–334. A Festschrift to honor Professor Garrett Birkhoff on his eightieth birthday.

[15] A. QUARTERONI, R. SACCO, AND F. SALERI, *Numerical mathematics*, vol. 37 of Texts in Applied Mathematics, Springer-Verlag, Berlin, second ed., 2007.

[16] J. B. ROSSER, *Nine-point difference solutions for Poisson's equation*, in Computers and mathematics with applications, 1976, pp. 351–360.

[17] L. N. TREFETHEN AND D. BAU, III, *Numerical linear algebra*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[18] R. S. VARGA, *Matrix iterative analysis*, vol. 27 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, expanded ed., 2000.