Computational Astrophysics I: Introduction and basic concepts

Helge Todt

Astrophysics
Institute of Physics and Astronomy
University of Potsdam

SoSe 2024, 16.7.2024

Matrices and Linear Algebra

Methods to solve matrix problems (e.g., inversion) useful for ODEs and PDEs, e.g., eigenvalue problem or radiative transfer with Feautrier scheme

Example: Vibrational spectrum of a molecule 1

n degrees of vibrational freedom $\rightarrow$ potential energy

$$U(q_1, q_2, \ldots, q_n) \simeq \frac{1}{2} \sum_{j,k}^{n} A_{jk} q_j q_k \tag{1}$$

in generalized coordinates around equilibrium state up to 2nd order term, coupling/potential parameter A_{jk} (e.g., spring constant).
Kinetic energy with generalized mass M_{jk}

$$T(\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n) \simeq \frac{1}{2} \sum_{j,k}^{n} M_{jk} \dot{q}_j \dot{q}_k \tag{2}$$

Example: Vibrational spectrum of a molecule 2

Apply Lagrange equation of 2nd kind

$$\frac{\partial \mathcal{L}}{\partial q_j} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_j} = 0 \qquad \text{with } \mathcal{L} = T - U \tag{3}$$

$$\text{Hence, equations of motion, for } k = 1, \ldots, n\text{: } \sum_{j=1}^{n}(A_{jk} q_j + M_{jk}\ddot{q}_j) = 0 \tag{4}$$

Assume an oscillatory motion $q_j = x_j\, e^{\imath\omega t} \rightarrow \frac{d^2}{dt^2}(x_j\, e^{\imath\omega t}) = -x_j\omega^2\, e^{\imath\omega t}$

$$\rightarrow \sum_{j=1}^{n}(A_{jk} - M_{jk}\omega^2)x_j = 0 \quad \text{or with } k = 1, \ldots, n \;:\quad \boldsymbol{A}\boldsymbol{x} = \omega^2 \boldsymbol{M}\boldsymbol{x} \tag{5}$$

set of linear homogenous equations. Nontrivial solution $\rightarrow$ determinant of coefficient matrix $\stackrel{!}{=} 0$
$\rightarrow \omega_k = \sqrt{\lambda_k}$ $(k = 1, \ldots, n)$ from equation

$$\det(\boldsymbol{A} - \lambda \boldsymbol{M}) = 0 \tag{6}$$

Matrix $\boldsymbol{A}$ with elements A_{ij} and $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n \rightarrow m \times n$ matrix.

$$\begin{array}{c} \\ m \\ \text{rows} \\ \downarrow \end{array} \begin{array}{c} n \text{ columns} \rightarrow \\ \begin{pmatrix} A_{11} & A_{12} & \ldots & A_{1n} \\ A_{21} & \ldots & & \\ \ldots & & & \\ A_{m1} & & & A_{mn} \end{pmatrix} \end{array}$$

If $m = n \rightarrow$ square matrix

Remember: Computer stores array in memory sequentially (1d), for C/C++ stored by rows (last index runs first)

$$A_{11}, A_{12}, \ldots, A_{1n}, A_{21}, \ldots, A_{mn} \tag{7}$$

whereas for Fortran stored by column (first index runs first):

$$A_{11}, A_{21}, \ldots, A_{m1}, A_{12}, \ldots, A_{mn} \tag{8}$$

Variable array $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$: $n \times 1$ matrix. Hence set of linear equations for $i = 1, 2, \ldots, n$, where x_i is unknown:

$$A_{i1}\, x_1 + A_{i2}\, x_2 + \ldots + A_{in}\, x_n = b_i \tag{9}$$

with coefficients A_{ij} and constants b_i, so express Eq. (9) in matrix form

$$\boldsymbol{A}\, \boldsymbol{x} = \boldsymbol{b} \tag{10}$$

with $\boldsymbol{A}\, \boldsymbol{x}$ from standard matrix multiplication for $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}$, i.e.

$$C_{ij} = \sum_k A_{ik} B_{kj} \tag{11}$$

(number of columns of $\boldsymbol{A} \stackrel{!}{=}$ number of rows of $\boldsymbol{B}$)

Example: Population numbers from statistical equilibrium (non-LTE)

"inflow" to level n_j (from all other levels) balanced by "outflow" from level n_j (to all other levels)

$$\sum_{\substack{i=1 \\ i\neq j}}^{N} n_i P_{ij} = \sum_{\substack{i=1 \\ i\neq j}}^{N} n_j P_{ji} \quad \forall\, j = 1, \ldots, N \tag{12}$$

$$\boldsymbol{n}\,\boldsymbol{P} = 0 \quad \text{with } P_{ii} := -\sum_{j\neq i} P_{ij} \tag{13}$$

Remember definitions: Inverse of a matrix $\boldsymbol{A}$ is $\boldsymbol{A}^{-1}$:

$$\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I} \tag{14}$$

with $I_{ij} = \delta_{ij}$.
The transpose of a matrix $\boldsymbol{A}^T$ is with column and row indices of $\boldsymbol{A}$ interchanged

$$A^T_{ij} = A_{ji} \tag{15}$$

Matrix operations IV

Trace of $\boldsymbol{A}$ ($\mathrm{Tr}\,\boldsymbol{A}$) is summation of diagonal elements of $\boldsymbol{A}$

$$\mathrm{Tr}\,\boldsymbol{A} = \sum_{i=1}^{n} A_{ii} \tag{16}$$

The determinant of square matrix $\boldsymbol{A}$

$$\det(\boldsymbol{A}) = \sum_{i=1}^{n} (-1)^{i+j} A_{ij} \det(\boldsymbol{R}_{ij}) \tag{17}$$

where $\boldsymbol{R}_{ij}$ is residual matrix of $\boldsymbol{A}$ with ith row and jth column removed ($\rightarrow$ recursive computation)

$$\text{e.g., } \det \begin{pmatrix} A_{11} & A12 \\ A_{21} & A_{22} \end{pmatrix} = A_{11}A_{22} - A_{12}A_{21} \tag{18}$$

Matrix operations V

Important properties of the determinant:

- Determinant of a 1×1 matrix = element itself.
- Determinant of a triangular matrix (lower or upper) is the product of diagonal elements: $\det(\boldsymbol{A}) = \prod_{i=1}^{n} A_{ii}$
- $\det(\boldsymbol{BA}) = \det(\boldsymbol{B}) \cdot \det(\boldsymbol{A})$ (if both $n \times n$)
- $\det(\boldsymbol{A}^{-1}) = \frac{1}{\det(\boldsymbol{A})}$ $\rightarrow$ integer entries for $\boldsymbol{A}$ and $\boldsymbol{A}^{-1} \Leftrightarrow \det(\boldsymbol{A}) = \pm 1$
- $\det(\boldsymbol{A}^{T}) = \det(\boldsymbol{A})$
- The determinant is an *n-linear function* of the n columns (rows). It is moreover an *alternating form*. Together with $\det(\boldsymbol{A}^{T}) = \det(\boldsymbol{A})$, this means:
  Interchanging any pair of columns or rows of a matrix multiplies its determinant by -1.

Inverse of $\boldsymbol{A}$ via (Cramer's rule)

$$A_{ij}^{-1} = (-1)^{i+j} \frac{\det(\boldsymbol{R}_{ij})}{\det(\boldsymbol{A})} \tag{19}$$

$\rightarrow$ if $\boldsymbol{A}^{-1}$ exists or $\det(\boldsymbol{A}) \neq 0$ $\rightarrow$ nonsingular matrix, singular otherwise ().

Examples for singular / non-singular (=regular) matrices:

- the matrix

$$\boldsymbol{A} = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \tag{20}$$

is non-singular, its determinant is $\det(\boldsymbol{A}) = -1$ and its inverse is

$$\boldsymbol{A}^{-1} = \begin{pmatrix} -3 & 2 \\ 2 & -1 \end{pmatrix} \tag{21}$$

- the matrix

$$\boldsymbol{B} = \begin{pmatrix} 1 & 2 \\ 0 & 0 \end{pmatrix} \tag{22}$$

is singular, its determinant is $\det(\boldsymbol{A}) = 0$ and there exists no inverse

$$\boldsymbol{B} \cdot \boldsymbol{M} = \begin{pmatrix} 1 & 2 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1a + 2c & 1b + 2d \\ 0 & 0 \end{pmatrix} \neq \boldsymbol{I} \tag{23}$$

- the matrix

$$\boldsymbol{C} = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} \tag{24}$$

is singular, its determinant is $\det(\boldsymbol{A}) = 0$, as two of its lines are linearly dependent

Moreover, it can be useful to perform the following transformations, represented by a matrix multiplications : $\boldsymbol{A}' = \boldsymbol{M}\boldsymbol{A}$

1. interchanging two rows i and j, elements: $M_{ij} = 1; M_{ji} = 1; M_{kk} = 1$ for $k \neq i, j$ other elements $= 0 \rightarrow \det(\boldsymbol{M}\,\boldsymbol{A}) = -\det(\boldsymbol{A})$
2. multiply one row by λ: $M_{kk} = 1$ for $k \neq i$; $M_{ii} = \lambda \neq 0$, all other elements $= 0$ $\rightarrow \det(\boldsymbol{M}\,\boldsymbol{A}) = \det(\boldsymbol{M})\det(\boldsymbol{A}) = \lambda \det(\boldsymbol{A})$
3. add a row (or column) to another row (or column) multiplied by a factor λ: $M_{ii} = 1,\ M_{ij} = \lambda,\ M_{kl} = 0$. This can be also be written as

$$A'_{ij} = A_{ij} + \lambda A_{kj} \qquad \text{for } j = 1, 2, \ldots, n \tag{25}$$

and i and k are row indices, which can be the same. The determinant is preserved $\det(\boldsymbol{A}') = \det(\boldsymbol{A})$.

$\rightarrow$ see below for Gaussian elimination and matrix decomposition

The matrix eigenvalue problem is for a given matrix $\boldsymbol{A}$

$$\boldsymbol{A}\,\boldsymbol{x} = \lambda \boldsymbol{x} \tag{26}$$

with eigenvector $\boldsymbol{x}$ and corresponding eigenvalue λ of the matrix.
Also for the example of the vibrating molecules:

$$\boldsymbol{A}\boldsymbol{x} = \omega^2 \boldsymbol{M}\boldsymbol{x} \quad | \quad \boldsymbol{B} := \boldsymbol{M}^{-1}\boldsymbol{A} \tag{27}$$

$$\rightarrow \boldsymbol{B}\boldsymbol{x} = \omega^2 \boldsymbol{x} \tag{28}$$

→ Matrix eigenvalue problem = linear equation set problem
→ e.g., iterative solution

$$\boldsymbol{A}\,\boldsymbol{x}_{n+1} = \lambda_n \boldsymbol{x}_n \tag{29}$$

Moreover, the eigenvalues are preserved under a similarity transformation with a non-singular matrix $\boldsymbol{S}$

$$\boldsymbol{B} = \boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{S} \tag{30}$$

$$\rightarrow\ \boldsymbol{B}\,\boldsymbol{y} = \lambda \boldsymbol{y} \quad \Leftrightarrow \quad \boldsymbol{A}\,\boldsymbol{x} = \lambda \boldsymbol{x} \qquad \text{for } \boldsymbol{x} = \boldsymbol{S}\boldsymbol{y} \tag{31}$$

$$\rightarrow\ \det(\boldsymbol{B}) = \det(\boldsymbol{A}) = \prod_{i=1}^{n} \lambda_i \tag{32}$$

→ computation of eigenvalues & eigenvectors usually complicated ...

The general problem:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{33}$$

where matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ given and vector $\boldsymbol{x}$ unknown.

Straightforward solutions:

- Cramer's rule:

$$x_i = \frac{\det(\boldsymbol{A}_i)}{\det(\boldsymbol{A})} \tag{34}$$

where in $\boldsymbol{A}_i$ the i-th column is replaced by $\boldsymbol{b}$
$\rightarrow$ for a system of n equations: need to compute $n+1$ determinants, each of order n (see above), i.e., compute $n!$ terms each with $(n-1)$ multiplications
$\rightarrow (n+1) \times n! \times (n-1)$ multiplications,
e.g., for $n = 20 \rightarrow 10^{21}$ multiplications and for a computer with, e.g., 10 TFLOPS
$\rightarrow t \approx 3\,\text{a}$ only for multiplications (also note large accumulation of roundoff error)

- find the inverse $\boldsymbol{A}^{-1}$

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b} \tag{35}$$

$\rightarrow$ also time-consuming and instable, e.g., ($n = 1$, float)

$$7x = 21 \tag{36}$$

$$x = \frac{21}{7} = 3 \quad \text{(direct division)} \tag{37}$$

$$x = (7^{-1})(21) \quad \text{(compute inverse)} \tag{38}$$

$$= (.142857)(21) = 2.999997 \quad \text{(less accurate)} \tag{39}$$

computation of the inverse, e.g., via Cramer's rule (see above) or

with *Gauß-Jordan elimination* (see below) for system $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$:

$$\begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} \hat{a}_{11} & \dots & \hat{a}_{1n} \\ \vdots & & \vdots \\ \hat{a}_{n1} & \dots & \hat{a}_{nn} \end{pmatrix} = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix} \tag{40}$$

hence, the j-th column of the inverse $\hat{a}_j = (\hat{a}_{1j}, \hat{a}_{2j}, \dots, \hat{a}_{nj})^T$ is solution of the system of linear equations

$$A \cdot \hat{a}_j = e_j \tag{41}$$

These equations are solved simultaneously by extending matrix $\boldsymbol{A}$ with $\boldsymbol{I}$:

$$(\, A \,|\, I \,) = \left(\begin{array}{ccc|ccc} a_{11} & \dots & a_{1n} & 1 & & 0 \\ \vdots & & \vdots & & \ddots & \\ a_{n1} & \dots & a_{nn} & 0 & & 1 \end{array} \right) \tag{42}$$

$\rightarrow$ elementary row operations $\rightarrow$ matrix $\boldsymbol{A}$ into upper triangular form (forward elimination)

$$(\,D\,|\,B\,) = \left(\begin{array}{ccc|ccc} * & \dots & * & * & \dots & * \\ & \ddots & \vdots & \vdots & & \vdots \\ 0 & & * & * & \dots & * \end{array}\right) \tag{43}$$

$\rightarrow$ if no zeros on diagonal $\rightarrow$ invertible, bring into diagonal form:

$$(\,I\,|\,A^{-1}\,) = \left(\begin{array}{ccc|ccc} 1 & & 0 & \hat{a}_{11} & \dots & \hat{a}_{1n} \\ & \ddots & & \vdots & & \vdots \\ 0 & & 1 & \hat{a}_{n1} & \dots & \hat{a}_{nn} \end{array}\right) \tag{44}$$

or compute inverse with *characteristic polynomial*:

$$A^{-1} = \frac{-1}{\det(A)}\left(\alpha_1 I_n + \alpha_2 A + \ldots + \alpha_n A^{n-1}\right) \tag{45}$$

where the coefficients of the chracteristical polynomial of $\boldsymbol{A}$ can be obtained from $\chi(t) = \det(t\boldsymbol{I} - \boldsymbol{A}) = \alpha_0 + \alpha_1 \cdot t^1 + \ldots + \alpha_n \cdot t^n$

Matrix problems can be easily solved for an upper (lower) triangular matrix, for which elements below (above) the diagonal = 0,

$$\begin{pmatrix} R_{11} & R_{12} & \dots & R_{1n} \\ 0 & R_{22} & \dots & R_{2n} \\ & & \ddots & \\ 0 & 0 & \dots & R_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \dots \\ c_n \end{pmatrix} \tag{46}$$

via backward (forward) substitution, i.e. starting with $x_n = c_n/R_{nn}$ and

$$x_i = \frac{c_i - \sum_{j=i+1}^{n} R_{ij} x_j}{R_{ii}} \qquad \text{for } i = n-1, \dots, 1 \tag{47}$$

$\rightarrow$ need algorithms for transformation into triangular form

Gaussian elimination

1. *Forward elimination:* Transform linear equation set $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ by a sequence of matrix operations j from original matrix $\boldsymbol{A} = \boldsymbol{A}^{(0)}$ to $\boldsymbol{A}^{(j)}$, hence after $n-1$ steps for a $n \times n$ matrix

$$\boldsymbol{A}^{(n-1)}\boldsymbol{x} = \boldsymbol{b}^{(n-1)} \tag{48}$$

where $A_{ij}^{(n-1)} = 0$ for $i > j$:

1. multiply 1st equation (1st row $\boldsymbol{A}$ and $b_1^{(0)}$) by $-A_{i1}^{(0)}/A_{11}^{(0)}$ and add to ith equation (row) for $i > 1 \rightarrow$ 1st element of every row except 1st row eliminated $\rightarrow \boldsymbol{A}^{(1)}$
2. multiply 2nd equation by $-A_{i2}^{(1)}/A_{22}^{(1)}$ and add to ith equation for $i > 2 \rightarrow$ 2nd element of every row except 1st & 2nd row eliminated $\rightarrow \boldsymbol{A}^{(2)}$
3. ...
4. upper triangular matrix $\boldsymbol{A}^{(n-1)}$

2. *backward substitution* according to Eq. (47)

Gaussian elimination III

ad 1.: all diagonal elements A_{jj} are used in denominators $-A_{ij}^{(j-1)}/A_{jj}{}^{(j-1)}$
$\rightarrow$ problems if diagonal elements $= 0$ or ≈ 0
Solution: pivoting (from french pivot=center of rotation) $\rightarrow$ interchange rows/columns to put always largest (absolut value) element on diagonal
full pivoting: interchange columns and rows, need to keep track of order ...
partial pivoting: only search for pivot in remaining elements of the current column (swap rows only)
$\rightarrow$ partial pivoting usually good compromise between speed and accuracy
$\rightarrow$ use index to record order of pivot elements instead of physically interchanging
$\rightarrow$ *rescaling*: rescale all elements from a row by its largest element before comparing to find pivot (reduces rounding errors)

Example: Gaussian elimination in Fortran - code sniplet

```
! partial pivot. Gaussian elimin.
DIMENSION A(N,N),INDX(N),C(N)
DO I = 1, N
 INDX(I) = I ! init. index
 C1 = 0.0
 DO J = 1, N ! rescale coeff.
  C1 = AMAX1(C1,ABS(A(I,J)))
 ENDDO
 C(I) = C1
ENDDO

DO J = 1, N-1 ! search pivots
 PI1 = 0.0
 DO I = J, N
  PI = ABS(A(INDX(I),J)) / C(INDX(I))
  IF (PI.GT.PI1) THEN
   PI1 = PI
   K   = I
  ENDIF
 ENDDO
 ITMP = INDX(J)
 INDX(J) = INDX(K)
 INDX(K) = ITMP
 DO I = J + 1, N ! elimin. subdiagonal
  PJ = A(INDX(I),J) / A(INDX(J),J)
  A(INDX(I),J) = PJ
  DO L = J + 1, N
   A(INDX(I),L) = A(INDX(I),L) - &
   PJ * A(INDX(J),L)
  ENDDO
 ENDDO
ENDDO
```

Example: Gaussian elimination by hand I

$$\begin{pmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \\ 6 \end{pmatrix} \tag{49}$$

1.) eliminate x_1 from row 2 & 3 $\rightarrow$ add $3/10 = 0.3\times$ 1st row to 2nd row & add $-5/10 = -0.5\times$ 1st row to 3rd row:

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 2.5 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 6.1 \\ 2.5 \end{pmatrix} \tag{50}$$

2.) eliminate x_2 from row 3 $\rightarrow$ a) pivoting: interchange row 2 & 3 so that coefficient of x_2 in row 2 is largest (because of roundoff errors $\rightarrow$ only for computers necessary)

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & -0.1 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 2.5 \\ 6.1 \end{pmatrix} \tag{51}$$

Example: Gaussian elimination by hand II

2.b) now add $0.1/2.5 = 0.04\times$ 2nd row to 3rd row:

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 2.5 \\ 6.2 \end{pmatrix} \tag{52}$$

Finally: backward substitution, starting with last row:

$$6.2\, x_3 = 6.2 \rightarrow x_3 = 1 \tag{53}$$

$$2.5\, x_2 + 5 \cdot 1 = 2.5 \rightarrow x_2 = -1 \tag{54}$$

$$10\, x_1 + (-7) \cdot (-1) + 0 = 7 \quad \rightarrow x_1 = 0 \tag{55}$$

This can be also expressed in matrix notation: Let

$$\boldsymbol{M}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0.3 & 1 & 0 \\ -0.5 & 0 & 1 \end{pmatrix} \rightarrow \boldsymbol{M}_1\boldsymbol{A} = \begin{pmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 2.5 & 5 \end{pmatrix}, \quad \boldsymbol{M}_1\boldsymbol{b} = \begin{pmatrix} 7 \\ 6.1 \\ 2.5 \end{pmatrix} \tag{56}$$

Example: Gaussian elimination by hand III

Let then

$$\boldsymbol{P}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \boldsymbol{M}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0.04 & 1 \end{pmatrix} \tag{57}$$

$$\rightarrow \boldsymbol{M}_2\boldsymbol{P}_2\boldsymbol{M}_1\boldsymbol{A} = \begin{pmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.2 \end{pmatrix} = \boldsymbol{U}, \quad \boldsymbol{M}_2\boldsymbol{P}_2\boldsymbol{M}_1\boldsymbol{b} = \begin{pmatrix} 7 \\ 2.5 \\ 6.2 \end{pmatrix} = \boldsymbol{c} \tag{58}$$

Hence $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{c}$, with upper triangular matrix $\boldsymbol{U}$.
The matrices $\boldsymbol{P}_k, k = 1, \ldots, n-1$ are the permutations matrices, inferred from the identity matrix $\boldsymbol{I}$ by interchanging rows in same way as for $\boldsymbol{A}$ in the kth step, and $\boldsymbol{M}_k$ is multiplication matrix, inferred from identy matrix by inserting mulitpliers used in kth step below diagonal in kth column $\rightarrow \boldsymbol{M}_k$ are lower triangular matrices

$$\boldsymbol{M} := \boldsymbol{M}_{n-1}\boldsymbol{P}_{n-1} \ldots \boldsymbol{M}_1\boldsymbol{P}_1 \tag{59}$$

$$\boldsymbol{U} = \boldsymbol{M}\boldsymbol{A} \quad (\text{“triangular decomposition” of } \boldsymbol{A}) \tag{60}$$

LU decomposition I

More general approach: decompose *nonsingular* matrix $\boldsymbol{A}$ into two triangular matrices

$$\boldsymbol{A} = \boldsymbol{L}\,\boldsymbol{U} \tag{61}$$

with lower (left) triangular matrix $\boldsymbol{L}$ and upper (right) triangular matrix $\boldsymbol{U}$ (or $\boldsymbol{R}$), hence

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{L}\boldsymbol{U}\boldsymbol{x} = \boldsymbol{b} \tag{62}$$

$$\rightarrow \text{ first, solve 1. } \boldsymbol{L}\boldsymbol{y} = \boldsymbol{b} \rightarrow \boldsymbol{y} \tag{63}$$

$$\text{then 2. } \boldsymbol{U}\boldsymbol{x} = \boldsymbol{y} \rightarrow \boldsymbol{x} \tag{64}$$

i.e. once $\boldsymbol{A} = \boldsymbol{L}\,\boldsymbol{U}$ obtained $\rightarrow$ easy to solve for *any* b.
More general case: re-order matrix $\boldsymbol{A}$ by, e.g., row-permutations (partial pivoting):

$$\boldsymbol{P}\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}, \text{ then} \tag{65}$$

$$\boldsymbol{L}\boldsymbol{U}\boldsymbol{x} = \boldsymbol{P}\boldsymbol{b} \tag{66}$$

$$1.\ \boldsymbol{L}\boldsymbol{y} = \boldsymbol{P}\boldsymbol{b} \rightarrow \boldsymbol{y} \tag{67}$$

$$2.\ \boldsymbol{U}\boldsymbol{x} = \boldsymbol{y} \rightarrow \boldsymbol{x} \tag{68}$$

e.g. $\rightarrow$ Crout's method

start with $L_{i1} = A_{i1}$ and $U_{1j} = A_{1j}/A_{11}$, then recursively:

$$L_{ij} = A_{ij} - \sum_{k=1}^{j-1} L_{ik} U_{kj} \tag{69}$$

$$U_{ij} = \frac{1}{L_{ii}} \left(A_{ij} - \sum_{k=1}^{i-1} L_{ik} U_{kj} \right) \tag{70}$$

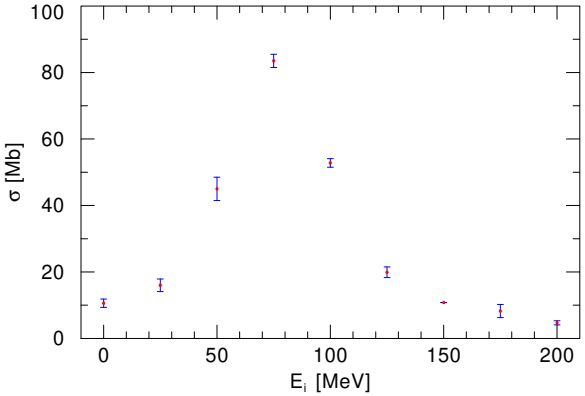
Usually no need to implement by yourself, instead use libraries, e.g., `LINPACK`:

- `DGEFA` performs LU decomposition by Gaussian elimination
- `DGESL` uses that decomposition to solve the given system of linear equations
- `DGEDI` uses decomposition to compute inverse of a matrix

Application: Interpolating data I

Remember following measurement of a cross section

E_i [MeV]	0	25	50	75	100	125	150	175	200
$\sigma(E_i)$ [Mb]	10.6	16.0	45.0	83.5	52.8	19.9	10.8	8.25	4.7
$\sigma_{\sigma(E_i)}$ [Mb]	1.26	1.9	3.5	2.0	1.3	1.6	0.04	1.96	0.61



The cross section can be described by Breit-Wigner formula

$$f(E) = \frac{f_r}{(E - E_r)^2 + \Gamma^2/4} \tag{71}$$

Interpolation problem

We want to determine $\sigma(E)$ for values of E which lie between measured values of E

By

- numerical interpolation (assumption of data representation by polynomial in E):
  $\rightarrow$ see previous lectures
  $\rightarrow$ ignores errors in measurement (noise)
- fitting parameters of an underlying model, e.g., Breit-Wigner with f_r, E_r, Γ, (taking errors into account), i.e., minimizing χ^2
- Fourier analysis (next semester lecture)

Already seen for linear regression:
We have N_D data points

$$(x_i, y_i \pm \sigma_i) \quad i = 1, \ldots, N_D \tag{72}$$

and a function $y = g(x)$ (=model) with parameters $\{a_m\}$; fit function to data, such that $\chi^2 = min$:

$$\chi^2 := \sum_{i=1}^{N_D} \left(\frac{y_i - g(x_i; \{a_m\})}{\sigma_i} \right)^2 \tag{73}$$

i.e. for M_P parameters $\{a_m, m = 1 \ldots M_P\}$

$$\frac{\partial \chi^2}{\partial a_m} \stackrel{!}{=} 0 \Rightarrow \sum_{i=1}^{N_D} \frac{[y_i - g(x_i)]}{\sigma_i^2} \frac{\partial g(x_i)}{\partial a_m} = 0 \quad (m = 1, \ldots, M_P) \tag{74}$$

$\rightarrow$ solve M_P equations, usually nonlinear in a_m

Least square fitting II

goodness of fit, assumptions

- deviations to model only due to random errors
- Gaussion distribution of errors

$\rightarrow$ then, fit is good when $\chi^2 \approx N_D - M_P$ (degrees of freedom)

- if $\chi^2 \ll N_D - M_P \rightarrow$ probably too many parameters or errors σ_i to large (fitting random scatter)
- if $\chi^2 \gg N_D - M_P \rightarrow$ model not good or underestimated errors or non-random errors

$\rightarrow$ for linear fit see above

Non-linear fit

remember Breit-Wigner resonance formula Eq. (71)

$$f(E) = \frac{f_r}{(E - E_r)^2 + \Gamma^2/4} \tag{75}$$

$\rightarrow$ determine f_r, E_r, Γ
$\rightarrow$ nonlinear equations in the parameters

$$a_1 = f_r \qquad a_2 = E_r \qquad a_3 = \Gamma^2/4 \tag{76}$$

$$\Rightarrow g(x) = \frac{a_1}{(x - a_2)^2 + a_3} \tag{77}$$

$$\frac{\partial g}{\partial a_1} = \frac{1}{(x - a_2)^2 + a_3}, \quad \frac{\partial g}{\partial a_2} = \frac{-2a_1(x - a_2)}{[(x - a_2)^2 + a_3]^2}, \quad \frac{\partial g}{\partial a_3} = \frac{-a_1}{[(x - a_2)^2 + a_3]^2} \tag{78}$$

Insert into Eq. (74):

$$\sum_{i=1}^{9} \frac{y_i - g(x_i, a)}{(x_i - a_2)^2 + a_3} = 0 \qquad \sum_{i=1}^{9} \frac{[y_i - g(x_i, a)](x_i - a_2)}{[(x_i - a_2)^2 + a_3]^2} = 0$$
$$\sum_{i=1}^{9} \frac{y_i - g(x_i, a)}{[(x_i - a_2)^2 + a_3]^2} = 0 \tag{79}$$

$\rightarrow$ *three* nonlinear equations for unknown a_1, a_2, a_3, i.e. cannot be solved by linear algebra but can be solved with help of Newton-Raphson method, i.e. find the roots for the equations above

$$f_i(a_1, \ldots, a_M) = 0 \qquad i = 1, \ldots, M \tag{80}$$

So

$$f_1(a_1, a_2, a_3) = \sum_{i=1}^{9} \frac{y_i - g(x_i, a)}{(x_i - a_2)^2 + a_3} = 0 \tag{81}$$

$$f_2(a_1, a_2, a_3) = \sum_{i=1}^{9} \frac{[y_i - g(x_i, a)](x_i - a_2)}{[(x_i - a_2)^2 + a_3]^2} = 0 \tag{82}$$

$$f_3(a_1, a_2, a_3) = \sum_{i=1}^{9} \frac{y_i - g(x_i, a)}{[(x_i - a_2)^2 + a_3]^2} = 0 \tag{83}$$

with intial guesses for a_1, a_2, a_3.

Newton-Raphson method for a system of nonlinear equations
Remember for 1dim Newton-Raphson method, correction for Δx:

$$f(x_0) + f'(x_0) \cdot \Delta x \stackrel{!}{=} 0 \tag{84}$$

$$\Delta x = -\frac{f(x_0)}{f'(x_0)} \tag{85}$$

For our system of equations $f_i(a_1, \ldots, a_M) = 0$, we assume that for our approximation (intial guess) $\{a_i\}$ corrections $\{\Delta x_i\}$ exist so that

$$f_i(a_1 + \Delta a_1, a_2 + \Delta a_2, a_3 + \Delta a_3) = 0 \qquad i = 1, 2, 3 \tag{86}$$

$\rightarrow$ linear approximation (two terms of Taylor series):

$$f_i(a_1 + \Delta a_1, \ldots) \simeq f_i(a_1, a_2, a_3) + \sum_{j=1}^{3} \frac{\partial f_i}{\partial a_j} \Delta a_j = 0 \qquad i = 1, 2, 3 \tag{87}$$

$\rightarrow$ set of 3 linear equations in 3 unknowns

as explicit equations:

$$f_1 + \partial f_1/\partial a_1 \Delta a_1 + \partial f_1/\partial a_2 \Delta a_2 + \partial f_1/\partial a_3 \Delta a_3 = 0 \quad (88)$$
$$f_2 + \partial f_2/\partial a_1 \Delta a_1 + \partial f_2/\partial a_2 \Delta a_2 + \partial f_2/\partial a_3 \Delta a_3 = 0 \quad (89)$$
$$f_3 + \partial f_3/\partial a_1 \Delta a_1 + \partial f_3/\partial a_2 \Delta a_2 + \partial f_3/\partial a_3 \Delta a_3 = 0 \quad (90)$$

Or as single matrix equation:

$$\begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} + \begin{pmatrix} \partial f_1/\partial a_1 & \partial f_1/\partial a_2 & \partial f_1/\partial a_3 \\ \partial f_2/\partial a_1 & \partial f_2/\partial a_2 & \partial f_2/\partial a_3 \\ \partial f_3/\partial a_1 & \partial f_3/\partial a_2 & \partial f_3/\partial a_3 \end{pmatrix} \begin{pmatrix} \Delta a_1 \\ \Delta a_2 \\ \Delta a_3 \end{pmatrix} = 0 \quad (91)$$

Or in matrix notation

$$\mathrm{f} + \mathrm{F}' \, \boldsymbol{\Delta a} = 0 \Rightarrow \mathrm{F}' \, \boldsymbol{\Delta a} = -\boldsymbol{f} \quad (92)$$

Where we want to solve for $\boldsymbol{\Delta a}$ (the corrections)
Matrix $\boldsymbol{F}'$ sometimes written as $\boldsymbol{J}$ is called the *Jacobian* matrix (with entries $f'_{ij} = \partial f_i/\partial a_j$).

Equation $\boldsymbol{F}'\Delta a = -\boldsymbol{f}$ corresponds to standard form $\boldsymbol{Ax} = \boldsymbol{b}$ for systems of linear equations. Formally solution obtained by multiplying with inverse of $\boldsymbol{F}'$

$$\boldsymbol{\Delta a} = -\boldsymbol{F}'^{-1}\boldsymbol{f} \tag{93}$$

$\rightarrow$ inverse must exist for unique solution
$\rightarrow$ same form as for 1d Newton-Raphson: $\Delta x = -(1/f')f$
$\rightarrow$ iterate as for 1d Newton-Raphson till $\boldsymbol{f} \approx 0$

compute derivatives for the system numerically

$$f'_{ij} = \frac{\partial f_i}{\partial a_j} \simeq \frac{f_i(a_j + \Delta a_j) - f_i(a_j)}{\Delta a_j} \tag{94}$$

with Δa_j sufficiently small, e.g., 1% of a

Nonlinear fit with Newton-Raphson

In our nonlinear fit problem the Newton step

$$\mathrm{F}' \, \boldsymbol{\Delta a} = -\boldsymbol{f} \tag{95}$$

can be solved for $\boldsymbol{\Delta a}$ with help of `DGEFA` and `DGESL` (see p. 27):

```
CALL DGEFA(FPRIME, NDIM, NDIM, IPVT, INFO)
IF (INFO .NE. 0) STOP 'JACOBIAN MATRIX WITH 0 ON DIAGONAL'
CALL DGESL(FPRIME, NDIM, NDIM, IPVT, F)
```

where the solution $\boldsymbol{\Delta a}$ is written to vector `F`