# Computational Astrophysics I: Introduction and basic concepts 

Helge Todt

Astrophysics
Institute of Physics and Astronomy
University of Potsdam

SoSe 2023, 24.7.2023


## 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

$$
\begin{equation*}
U\left(q_{1}, q_{2}, \ldots, q_{n}\right) \simeq \frac{1}{2} \sum_{j, k}^{n} A_{j k} q_{j} q_{k} \tag{1}
\end{equation*}
$$

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

$$
\begin{equation*}
T\left(\dot{q}_{1}, \dot{q}_{2}, \ldots, \dot{q}_{n}\right) \simeq \frac{1}{2} \sum_{j, k}^{n} M_{j k} \dot{q}_{j} \dot{q}_{k} \tag{2}
\end{equation*}
$$

## Matrices in physics II

## Example: Vibrational spectrum of a molecule 2

Apply Lagrange equation of 2nd kind

$$
\begin{equation*}
\frac{\partial \mathcal{L}}{\partial q_{j}}-\frac{d}{d t} \frac{\partial \mathcal{L}}{\partial \dot{q}_{j}}=0 \quad \text { with } \mathcal{L}=T-U \tag{3}
\end{equation*}
$$

$$
\begin{equation*}
\text { Hence, equations of motion, for } k=1, \ldots, n: \sum_{j=1}^{n}\left(A_{j k} q_{j}+M_{j k} \ddot{q}_{j}\right)=0 \tag{4}
\end{equation*}
$$

Assume an oscillatory motion $q_{j}=x_{j} e^{\imath \omega t} \rightarrow \frac{d^{2}}{d t^{2}}\left(x_{j} e^{\imath \omega t}\right)=-x_{j} \omega^{2} e^{\imath \omega t}$

$$
\begin{equation*}
\rightarrow \sum_{j=1}^{n}\left(A_{j k}-M_{j k} \omega^{2}\right) x_{j}=0 \quad \text { or with } k=1, \ldots, n: \quad \boldsymbol{A} \boldsymbol{x}=\omega^{2} \boldsymbol{M} \boldsymbol{x} \tag{5}
\end{equation*}
$$

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

$$
\begin{equation*}
\operatorname{det}(\boldsymbol{A}-\lambda \boldsymbol{M})=0 \tag{6}
\end{equation*}
$$

## Matrix operations I

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

$$
\underset{\boldsymbol{c}}{\quad n \text { columns } \rightarrow} \text { rows }\left(\begin{array}{cccc}
A_{11} & A_{12} & \ldots & A_{1 n} \\
A_{21} & \ldots & & \\
\ldots & & & \\
A_{m 1} & & & A_{m n}
\end{array}\right)
$$

If $m=n \rightarrow$ square matrix
Remember: Computer stores array in memory sequentially (1d), for $\mathrm{C} / \mathrm{C}++$ stored by rows (last index runs first)

$$
\begin{equation*}
A_{11}, A_{12}, \ldots, A_{1 n}, A_{21}, \ldots, A_{m n} \tag{7}
\end{equation*}
$$

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

$$
\begin{equation*}
A_{11}, A_{21}, \ldots, A_{m 1}, A_{12}, \ldots, A_{m n} \tag{8}
\end{equation*}
$$

## Matrix operations II

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

$$
\begin{equation*}
A_{i 1} x_{1}+A_{i 2} x_{2}+\ldots+A_{i n} x_{n}=b_{i} \tag{9}
\end{equation*}
$$

with coefficients $A_{i j}$ and constants $b_{i}$, so express Eq. (9) in matrix form

$$
\begin{equation*}
\boldsymbol{A x}=\boldsymbol{b} \tag{10}
\end{equation*}
$$

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

$$
\begin{equation*}
C_{i j}=\sum_{k} A_{i k} B_{k j} \tag{11}
\end{equation*}
$$

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

## Matrix operations III

## 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)

$$
\begin{align*}
\sum_{\substack{i=1 \\
i \neq j}}^{N} n_{i} P_{i j} & =\sum_{\substack{i=1 \\
i \neq j}}^{N} n_{j} P_{j i} \quad \forall j=1, \ldots, N  \tag{12}\\
\boldsymbol{n} \boldsymbol{P} & =0 \quad \text { with } P_{i j}:=-\sum_{j \neq i} P_{i j} \tag{13}
\end{align*}
$$

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

$$
\begin{equation*}
\boldsymbol{A}^{-1} \boldsymbol{A}=\boldsymbol{A A}^{-1}=\boldsymbol{I} \tag{14}
\end{equation*}
$$

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

$$
\begin{equation*}
A_{i j}^{T}=A_{j i} \tag{15}
\end{equation*}
$$

## Matrix operations IV

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

$$
\begin{equation*}
\operatorname{Tr} \boldsymbol{A}=\sum_{i=1}^{n} A_{i i} \tag{16}
\end{equation*}
$$

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

$$
\begin{equation*}
\operatorname{det}(\boldsymbol{A})=\sum_{i=1}^{n}(-1)^{i+j} A_{i j} \operatorname{det}\left(\boldsymbol{R}_{i j}\right) \tag{17}
\end{equation*}
$$

where $\boldsymbol{R}_{i j}$ is residual matrix of $\boldsymbol{A}$ with $i$ th row and $j$ th column removed ( $\rightarrow$ recursive computation)

$$
\text { e.g., } \operatorname{det}\left(\begin{array}{ll}
A_{11} & A 12  \tag{18}\\
A_{21} & A_{22}
\end{array}\right)=A_{11} A_{22}-A_{12} A_{21}
$$

## Matrix operations V

Important properties of the determinant:

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

$$
\begin{equation*}
A_{i j}^{-1}=(-1)^{i+j} \frac{\operatorname{det}\left(\boldsymbol{R}_{i j}\right)}{\operatorname{det}(\boldsymbol{A})} \tag{19}
\end{equation*}
$$

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

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

- the matrix

$$
\boldsymbol{A}=\left(\begin{array}{ll}
1 & 2  \tag{20}\\
2 & 3
\end{array}\right)
$$

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

$$
\boldsymbol{A}^{-1}=\left(\begin{array}{cc}
-3 & 2  \tag{21}\\
2 & -1
\end{array}\right)
$$

- the matrix

$$
\boldsymbol{B}=\left(\begin{array}{ll}
1 & 2  \tag{22}\\
0 & 0
\end{array}\right)
$$

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

$$
\boldsymbol{B} \cdot \boldsymbol{M}=\left(\begin{array}{ll}
1 & 2  \tag{23}\\
0 & 0
\end{array}\right) \cdot\left(\begin{array}{ll}
a & b \\
c & d
\end{array}\right)=\left(\begin{array}{cc}
1 a+2 c & 1 b+2 d \\
0 & 0
\end{array}\right) \neq \boldsymbol{I}
$$

- the matrix

$$
\boldsymbol{C}=\left(\begin{array}{ll}
1 & 2  \tag{24}\\
2 & 4
\end{array}\right)
$$

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

## Matrix operations VIII

Moreover, it can be useful to perform the following transformations, represented by a matrix multiplications: $\boldsymbol{A}^{\prime}=\boldsymbol{M A}$
(1) interchanging two rows $i$ and $j$, elements: $M_{i j}=1 ; M_{j i}=1 ; M_{k k}=1$ for $k \neq i, j$ other elements $=0 \rightarrow \operatorname{det}(\boldsymbol{M} \boldsymbol{A})=-\operatorname{det}(\boldsymbol{A})$
(2) multiply one row by $\lambda: M_{k k}=1$ for $k \neq i ; M_{i i}=\lambda \neq 0$, all other elements $=0$ $\rightarrow \operatorname{det}(\boldsymbol{M} \boldsymbol{A})=\operatorname{det}(\boldsymbol{M}) \operatorname{det}(\boldsymbol{A})=\lambda \operatorname{det}(\boldsymbol{A})$
(3) add a row (or column) to another row (or column) multiplied by a factor $\lambda$ : $M_{i i}=1, M_{i j}=\lambda, M_{k l}=0$. This can be also be written as

$$
\begin{equation*}
A_{i j}^{\prime}=A_{i j}+\lambda A_{k j} \quad \text { for } j=1,2, \ldots, n \tag{25}
\end{equation*}
$$

and $i$ and $k$ are row indices, which can be the same. The determinant is preserved $\operatorname{det}\left(\boldsymbol{A}^{\prime}\right)=\operatorname{det}(\boldsymbol{A})$.
$\rightarrow$ see below for Gaussian elimination and matrix decomposition

## Eigenvalue problems I

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

$$
\begin{equation*}
\boldsymbol{A} \boldsymbol{x}=\lambda \boldsymbol{x} \tag{26}
\end{equation*}
$$

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

$$
\begin{align*}
\boldsymbol{A} \boldsymbol{x} & =\omega^{2} \boldsymbol{M} \boldsymbol{x} \quad \mid \quad \boldsymbol{B}:=\boldsymbol{M}^{-1} \boldsymbol{A}  \tag{27}\\
\rightarrow \boldsymbol{B} \boldsymbol{x} & =\omega^{2} \boldsymbol{x} \tag{28}
\end{align*}
$$

## Eigenvalue problems II

$\rightarrow$ Matrix eigenvalue problem $=$ linear equation set problem
$\rightarrow$ e.g., iterative solution

$$
\begin{equation*}
\boldsymbol{A} \boldsymbol{x}_{n+1}=\lambda_{n} \boldsymbol{x}_{n} \tag{29}
\end{equation*}
$$

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

$$
\begin{align*}
\boldsymbol{B} & =\boldsymbol{S}^{-1} \boldsymbol{A} \boldsymbol{S}  \tag{30}\\
\rightarrow \boldsymbol{B} \boldsymbol{y} & =\lambda \boldsymbol{y} \quad \Leftrightarrow \quad \boldsymbol{A} \boldsymbol{x}=\lambda \boldsymbol{x} \quad \text { for } \boldsymbol{x}=\boldsymbol{S} \boldsymbol{y}  \tag{31}\\
\rightarrow \operatorname{det}(\boldsymbol{B}) & =\operatorname{det}(\boldsymbol{A})=\prod_{i=1}^{n} \lambda_{i} \tag{32}
\end{align*}
$$

$\rightarrow$ computation of eigenvalues \& eigenvectors usually complicated ...

The general problem:

$$
\begin{equation*}
\boldsymbol{A x}=\boldsymbol{b} \tag{33}
\end{equation*}
$$

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

- Cramer's rule:

$$
\begin{equation*}
x_{i}=\frac{\operatorname{det}\left(\boldsymbol{A}_{i}\right)}{\operatorname{det}(\boldsymbol{A})} \tag{34}
\end{equation*}
$$

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
$\rightarrow t \approx 3$ a only for multiplications (also note large accumulation of roundoff error)

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

$$
\begin{equation*}
\boldsymbol{x}=\boldsymbol{A}^{-1} \boldsymbol{b} \tag{35}
\end{equation*}
$$

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

$$
\begin{align*}
7 x & =21  \tag{36}\\
x & =\frac{21}{7}=3 \quad(\text { direct division })  \tag{37}\\
x & =\left(7^{-1}\right)(21) \quad \text { (compute inverse) }  \tag{38}\\
& =(.142857)(21)=2.999997 \quad \text { (less accurate) } \tag{39}
\end{align*}
$$

computation of the inverse, e.g., via Cramer's rule (see above) or
with Gauß-Jordan elimination (see below) for system $\boldsymbol{A A}^{-1}=\boldsymbol{I}$ :

$$
\left(\begin{array}{ccc}
a_{11} & \ldots & a_{1 n}  \tag{40}\\
\vdots & & \vdots \\
a_{n 1} & \ldots & a_{n n}
\end{array}\right) \cdot\left(\begin{array}{ccc}
\hat{a}_{11} & \ldots & \hat{a}_{1 n} \\
\vdots & & \vdots \\
\hat{a}_{n 1} & \ldots & \hat{a}_{n n}
\end{array}\right)=\left(\begin{array}{ccc}
1 & & 0 \\
& \ddots & \\
0 & & 1
\end{array}\right)
$$

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

$$
\begin{equation*}
A \cdot \hat{a}_{j}=e_{j} \tag{41}
\end{equation*}
$$

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

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

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

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

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

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

or compute inverse with characteristic polynomial:

$$
\begin{equation*}
A^{-1}=\frac{-1}{\operatorname{det}(A)}\left(\alpha_{1} I_{n}+\alpha_{2} A+\ldots+\alpha_{n} A^{n-1}\right) \tag{45}
\end{equation*}
$$

where the coefficients of the chracteristical polynomial of $\boldsymbol{A}$ can be obtained from $\chi(t)=\operatorname{det}(t \boldsymbol{I}-\boldsymbol{A})=\alpha_{0}+\alpha_{1} \cdot t^{1}+\ldots+\alpha_{n} \cdot t^{n}$

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

$$
\left(\begin{array}{cccc}
R_{11} & R_{12} & \ldots & R_{1 n}  \tag{46}\\
0 & R_{22} & \ldots & R_{2 n} \\
& & \ddots & \\
0 & 0 & \ldots & R_{n n}
\end{array}\right) \cdot\left(\begin{array}{c}
x_{1} \\
x_{2} \\
\ldots \\
x_{n}
\end{array}\right)=\left(\begin{array}{c}
c_{1} \\
c_{2} \\
\ldots \\
c_{n}
\end{array}\right)
$$

via backward (forward) substitution, i.e. starting with $x_{n}=c_{n} / R_{n n}$ and

$$
\begin{equation*}
x_{i}=\frac{c_{i}-\sum_{j=i+1}^{n} R_{i j} x_{j}}{R_{i i}} \quad \text { for } i=n-1, \ldots, 1 \tag{47}
\end{equation*}
$$

$\rightarrow$ 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

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

where $A_{i j}^{(n-1)}=0$ for $i>j$ :
(1) multiply 1st equation (1st row $\boldsymbol{A}$ and $b_{1}^{(0)}$ ) by $-A_{i 1}^{(0)} / A_{11}^{(0)}$ and add to $i$ th equation (row) for $i>1 \rightarrow 1$ st element of every row except 1 st row eliminated $\rightarrow \boldsymbol{A}^{(1)}$
(2) multiply 2 nd equation by $-A_{i 2}^{(1)} / A_{22}^{(1)}$ and add to $i$ th equation for $i>2 \rightarrow 2$ nd element of every row except 1st \& 2nd row eliminated $\rightarrow \boldsymbol{A}^{(2)}$
(3)...
(9) upper triangular matrix $\boldsymbol{A}^{(n-1)}$
2. backward substitution according to Eq. (47)
ad 1.: all diagonal elements $A_{j j}$ are used in denominators $-A_{i j}^{(j-1)} / A_{j j}{ }^{(j-1)}$
$\rightarrow$ problems if diagonal elements $=0$ or $\approx 0$
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)

## Gaussian elimination IV

## 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
```


## Gaussian elimination $V$

## Example: Gaussian elimination by hand I

$$
\left(\begin{array}{rrr}
10 & -7 & 0  \tag{49}\\
-3 & 2 & 6 \\
5 & -1 & 5
\end{array}\right)\left(\begin{array}{l}
x_{1} \\
x_{2} \\
x_{3}
\end{array}\right)=\left(\begin{array}{l}
7 \\
4 \\
6
\end{array}\right)
$$

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

$$
\left(\begin{array}{rrr}
10 & -7 & 0  \tag{50}\\
0 & -0.1 & 6 \\
0 & 2.5 & 5
\end{array}\right)\left(\begin{array}{l}
x_{1} \\
x_{2} \\
x_{3}
\end{array}\right)=\left(\begin{array}{l}
7 \\
6.1 \\
2.5
\end{array}\right)
$$

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)

$$
\left(\begin{array}{rrr}
10 & -7 & 0  \tag{51}\\
0 & 2.5 & 5 \\
0 & -0.1 & 6
\end{array}\right)\left(\begin{array}{l}
x_{1} \\
x_{2} \\
x_{3}
\end{array}\right)=\left(\begin{array}{l}
7 \\
2.5 \\
6.1
\end{array}\right)
$$

## Gaussian elimination VI

## Example: Gaussian elimination by hand II

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

$$
\left(\begin{array}{rrl}
10 & -7 & 0  \tag{52}\\
0 & 2.5 & 5 \\
0 & 0 & 6.2
\end{array}\right)\left(\begin{array}{l}
x_{1} \\
x_{2} \\
x_{3}
\end{array}\right)=\left(\begin{array}{l}
7 \\
2.5 \\
6.2
\end{array}\right)
$$

Finally: backward substitution, starting with last row:

$$
\begin{align*}
6.2 x_{3} & =6.2 \rightarrow x_{3}=1  \tag{53}\\
2.5 x_{2}+5 \cdot 1 & =2.5 \rightarrow x_{2}=-1  \tag{54}\\
10 x_{1}+(-7) \cdot(-1)+0 & =7 \rightarrow x_{1}=0 \tag{55}
\end{align*}
$$

This can be also expressed in matrix notation: Let

$$
\boldsymbol{M}_{1}=\left(\begin{array}{ccc}
1 & 0 & 0  \tag{56}\\
0.3 & 1 & 0 \\
-0.5 & 0 & 1
\end{array}\right) \rightarrow \boldsymbol{M}_{1} \boldsymbol{A}=\left(\begin{array}{rrr}
10 & -7 & 0 \\
0 & -0.1 & 6 \\
0 & 2.5 & 5
\end{array}\right), \quad \boldsymbol{M}_{1} \boldsymbol{b}=\left(\begin{array}{l}
7 \\
6.1 \\
2.5
\end{array}\right)
$$

## Gaussian elimination VII

## Example: Gaussian elimination by hand III

Let then

$$
\begin{align*}
\boldsymbol{P}_{2} & =\left(\begin{array}{lll}
1 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0
\end{array}\right), \quad \boldsymbol{M}_{2}=\left(\begin{array}{lll}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0.04 & 1
\end{array}\right)  \tag{57}\\
\rightarrow \boldsymbol{M}_{2} \boldsymbol{P}_{2} \boldsymbol{M}_{1} \boldsymbol{A} & =\left(\begin{array}{rrr}
10 & -7 & 0 \\
0 & 2.5 & 5 \\
0 & 0 & 6.2
\end{array}\right)=\boldsymbol{U}, \quad \boldsymbol{M}_{2} \boldsymbol{P}_{2} \boldsymbol{M}_{1} \boldsymbol{b}=\left(\begin{array}{l}
7 \\
2.5 \\
6.2
\end{array}\right)=\boldsymbol{c} \tag{58}
\end{align*}
$$

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 $k$ th step, and $\boldsymbol{M}_{k}$ is multiplication matrix, inferred from identy matrix by inserting mulitpliers used in $k$ th step below diagonal in $k$ th column $\rightarrow \boldsymbol{M}_{k}$ are lower triangular matrices

$$
\begin{align*}
\boldsymbol{M} & :=\boldsymbol{M}_{n-1} \boldsymbol{P}_{n-1} \ldots \boldsymbol{M}_{1} \boldsymbol{P}_{1}  \tag{59}\\
\boldsymbol{U} & =\boldsymbol{M} \boldsymbol{A} \quad \text { ("triangular decomposition" of } \boldsymbol{A}) \tag{60}
\end{align*}
$$

## LU decomposition I

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

$$
\begin{equation*}
A=L U \tag{61}
\end{equation*}
$$

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

$$
\begin{align*}
& \qquad \boldsymbol{A} \boldsymbol{x}=\boldsymbol{L} \boldsymbol{U} \boldsymbol{x}=\boldsymbol{b}  \tag{62}\\
& \rightarrow \text { first, solve 1. } \boldsymbol{L} \boldsymbol{y}=\boldsymbol{b} \rightarrow \boldsymbol{y}  \tag{63}\\
& \text { then 2. } \boldsymbol{U x}=\boldsymbol{y} \rightarrow \boldsymbol{x} \tag{64}
\end{align*}
$$

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):

$$
\begin{align*}
\boldsymbol{P A} & =\boldsymbol{L} \boldsymbol{U}, \text { then }  \tag{65}\\
\boldsymbol{L} \boldsymbol{U} \boldsymbol{x} & =\boldsymbol{P} \boldsymbol{b}  \tag{66}\\
\text { 1. } \boldsymbol{L} \boldsymbol{y} & =\boldsymbol{P} \boldsymbol{b} \rightarrow \boldsymbol{y}  \tag{67}\\
\text { 2. } \boldsymbol{U} \boldsymbol{x} & =\boldsymbol{y} \rightarrow \boldsymbol{x} \tag{68}
\end{align*}
$$

e.g. $\rightarrow$ Crout's method
start with $L_{i 1}=A_{i 1}$ and $U_{1 j}=A_{1 j} / A_{11}$, then recursively:

$$
\begin{align*}
& L_{i j}=A_{i j}-\sum_{k=1}^{j-1} L_{i k} U_{k j}  \tag{69}\\
& U_{i j}=\frac{1}{L_{i i}}\left(A_{i j}-\sum_{k=1}^{i-1} L_{i k} U_{k j}\right) \tag{70}
\end{align*}
$$

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

| $E_{i}[\mathrm{MeV}]$ | 0 | 25 | 50 | 75 | 100 | 125 | 150 | 175 | 200 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| $\sigma\left(E_{i}\right)[\mathrm{Mb}]$ | 10.6 | 16.0 | 45.0 | 83.5 | 52.8 | 19.9 | 10.8 | 8.25 | 4.7 |
| $\sigma_{\sigma\left(E_{i}\right)}[\mathrm{Mb}]$ | 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

$$
\begin{equation*}
f(E)=\frac{f_{r}}{\left(E-E_{r}\right)^{2}+\Gamma^{2} / 4} \tag{71}
\end{equation*}
$$

## Application: Interpolating data II

## 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 $f_{r}, E_{r}, \Gamma$, (taking errors into account), i.e., minimizing $\chi^{2}$
- Fourier analysis (next semester lecture)

Already seen for linear regression:
We have $N_{D}$ data points

$$
\begin{equation*}
\left(x_{i}, y_{i} \pm \sigma_{i}\right) \quad i=1, \ldots, N_{D} \tag{72}
\end{equation*}
$$

and a function $y=g(x)$ (=model) with parameters $\left\{a_{m}\right\}$; fit function to data, such that $\chi^{2}=m i n:$

$$
\begin{equation*}
\chi^{2}:=\sum_{i=1}^{N_{D}}\left(\frac{y_{i}-g\left(x_{i} ;\left\{a_{m}\right\}\right)}{\sigma_{i}}\right)^{2} \tag{73}
\end{equation*}
$$

i.e. for $M_{P}$ parameters $\left\{a_{m}, m=1 \ldots M_{P}\right\}$

$$
\begin{equation*}
\frac{\partial \chi^{2}}{\partial a_{m}} \stackrel{!}{=} 0 \Rightarrow \sum_{i=1}^{N_{D}} \frac{\left[y_{i}-g\left(x_{i}\right)\right]}{\sigma_{i}^{2}} \frac{\partial g\left(x_{i}\right)}{\partial a_{m}}=0 \quad\left(m=1, \ldots, M_{P}\right) \tag{74}
\end{equation*}
$$

$\rightarrow$ solve $M_{P}$ equations, usually nonlinear in $a_{m}$
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 $\sigma_{i}$ 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)

$$
\begin{equation*}
f(E)=\frac{f_{r}}{\left(E-E_{r}\right)^{2}+\Gamma^{2} / 4} \tag{75}
\end{equation*}
$$

$\rightarrow$ determine $f_{r}, E_{r}, \Gamma$
$\rightarrow$ nonlinear equations in the parameters

$$
\begin{align*}
& a_{1}=f_{r} \quad a_{2}=E_{r} \quad a_{3}=\Gamma^{2} / 4  \tag{76}\\
& \Rightarrow g(x)=\frac{a_{1}}{\left(x-a_{2}\right)^{2}+a_{3}}  \tag{77}\\
& \frac{\partial g}{\partial a_{1}}=\frac{1}{\left(x-a_{2}\right)^{2}+a_{3}}, \quad \frac{\partial g}{\partial a_{2}}=\frac{-2 a_{1}\left(x-a_{2}\right)}{\left[\left(x-a_{2}\right)^{2}+a_{3}\right]^{2}}, \quad \frac{\partial g}{\partial a_{3}}=\frac{-a_{1}}{\left[\left(x-a_{2}\right)^{2}+a_{3}\right]^{2}} \tag{78}
\end{align*}
$$

Insert into Eq. (74):

$$
\begin{align*}
& \sum_{i=1}^{9} \frac{y_{i}-g\left(x_{i}, a\right)}{\left(x_{i}-a_{2}\right)^{2}+a_{3}}=0 \quad \sum_{i=1}^{9} \frac{\left[y_{i}-g\left(x_{i}, a\right)\right]\left(x_{i}-a_{2}\right)}{\left[\left(x_{i}-a_{2}\right)^{2}+a_{3}\right]^{2}}=0 \\
& \sum_{i=1}^{9} \frac{y_{i}-g\left(x_{i}, a\right)}{\left[\left(x_{i}-a_{2}\right)^{2}+a_{3}\right]^{2}}=0 \tag{79}
\end{align*}
$$

$\rightarrow$ 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

$$
\begin{equation*}
f_{i}\left(a_{1}, \ldots, a_{M}\right)=0 \quad i=1, \ldots, M \tag{80}
\end{equation*}
$$

## Least square fitting

So

$$
\begin{align*}
& f_{1}\left(a_{1}, a_{2}, a_{3}\right)=\sum_{i=1}^{9} \frac{y_{i}-g\left(x_{i}, a\right)}{\left(x_{i}-a_{2}\right)^{2}+a_{3}}=0  \tag{81}\\
& f_{2}\left(a_{1}, a_{2}, a_{3}\right)=\sum_{i=1}^{9} \frac{\left[y_{i}-g\left(x_{i}, a\right)\right]\left(x_{i}-a_{2}\right)}{\left[\left(x_{i}-a_{2}\right)^{2}+a_{3}\right]^{2}}=0  \tag{82}\\
& f_{3}\left(a_{1}, a_{2}, a_{3}\right)=\sum_{i=1}^{9} \frac{y_{i}-g\left(x_{i}, a\right)}{\left[\left(x_{i}-a_{2}\right)^{2}+a_{3}\right]^{2}}=0 \tag{83}
\end{align*}
$$

with intial guesses for $a_{1}, a_{2}, a_{3}$.

## Least square fitting VI

Newton-Raphson method for a system of nonlinear equations
Remember for 1dim Newton-Raphson method, correction for $\Delta x$ :

$$
\begin{align*}
& f\left(x_{0}\right)+f^{\prime}\left(x_{0}\right) \cdot \Delta x \stackrel{!}{=} 0  \tag{84}\\
& \Delta x=-\frac{f\left(x_{0}\right)}{f^{\prime}\left(x_{0}\right)} \tag{85}
\end{align*}
$$

For our system of equations $f_{i}\left(a_{1}, \ldots, a_{M}\right)=0$, we assume that for our approximation (intial guess) $\left\{a_{i}\right\}$ corrections $\left\{\Delta x_{i}\right\}$ exist so that

$$
\begin{equation*}
f_{i}\left(a_{1}+\Delta a_{1}, a_{2}+\Delta a_{2}, a_{3}+\Delta a_{3}\right)=0 \quad i=1,2,3 \tag{86}
\end{equation*}
$$

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

$$
\begin{equation*}
f_{i}\left(a_{1}+\Delta a_{1}, \ldots\right) \simeq f_{i}\left(a_{1}, a_{2}, a_{3}\right)+\sum_{j=1}^{3} \frac{\partial f_{i}}{\partial a_{j}} \Delta a_{j}=0 \quad i=1,2,3 \tag{87}
\end{equation*}
$$

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

## Least square fitting VII

as explicit equations:

$$
\begin{align*}
& f_{1}+\partial f_{1} / \partial a_{1} \Delta a_{1}+\partial f_{1} / \partial a_{2} \Delta a_{2}+\partial f_{1} / \partial a_{3} \Delta a_{3}=0  \tag{88}\\
& f_{2}+\partial f_{2} / \partial a_{1} \Delta a_{1}+\partial f_{2} / \partial a_{2} \Delta a_{2}+\partial f_{2} / \partial a_{3} \Delta a_{3}=0  \tag{89}\\
& f_{3}+\partial f_{3} / \partial a_{1} \Delta a_{1}+\partial f_{3} / \partial a_{2} \Delta a_{2}+\partial f_{3} / \partial a_{3} \Delta a_{3}=0 \tag{90}
\end{align*}
$$

Or as single matrix equation:

$$
\left(\begin{array}{l}
f_{1}  \tag{91}\\
f_{2} \\
f_{3}
\end{array}\right)+\left(\begin{array}{lll}
\partial f_{1} / \partial a_{1} & \partial f_{1} / \partial a_{2} & \partial f_{1} / \partial a_{3} \\
\partial f_{2} / \partial a_{1} & \partial f_{2} / \partial a_{2} & \partial f_{2} / \partial a_{3} \\
\partial f_{3} / \partial a_{1} & \partial f_{3} / \partial a_{2} & \partial f_{3} / \partial a_{3}
\end{array}\right)\left(\begin{array}{c}
\Delta a_{1} \\
\Delta a_{2} \\
\Delta a_{3}
\end{array}\right)=0
$$

Or in matrix notation

$$
\begin{equation*}
\mathrm{f}+\mathrm{F}^{\prime} \boldsymbol{\Delta} \boldsymbol{a}=0 \Rightarrow \mathrm{~F}^{\prime} \boldsymbol{\Delta} \boldsymbol{a}=-\boldsymbol{f} \tag{92}
\end{equation*}
$$

Where we want to solve for $\boldsymbol{\Delta} \boldsymbol{a}$ (the corrections) Matrix $\boldsymbol{F}^{\prime}$ sometimes written as $\boldsymbol{J}$ is called the Jacobian matrix (with entries $f_{i j}^{\prime}=\partial f_{i} / \partial a_{j}$ ).

## Least square fitting VIII

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

$$
\begin{equation*}
\Delta \boldsymbol{a}=-\boldsymbol{F}^{\prime-1} \boldsymbol{f} \tag{93}
\end{equation*}
$$

$\rightarrow$ inverse must exist for unique solution
$\rightarrow$ same form as for 1d Newton-Raphson: $\Delta x=-\left(1 / f^{\prime}\right) f$
$\rightarrow$ iterate as for 1d Newton-Raphson till $\boldsymbol{f} \approx 0$
compute derivatives for the system numerically

$$
\begin{equation*}
f_{i j}^{\prime}=\frac{\partial f_{i}}{\partial a_{j}} \simeq \frac{f_{i}\left(a_{j}+\Delta a_{j}\right)-f_{i}\left(a_{j}\right)}{\Delta a_{j}} \tag{94}
\end{equation*}
$$

with $\Delta a_{j}$ sufficiently small, e.g., $1 \%$ of $a$

## Nonlinear fit with Newton-Raphson

In our nonlinear fit problem the Newton step

$$
\begin{equation*}
F^{\prime} \boldsymbol{\Delta} \boldsymbol{a}=-\boldsymbol{f} \tag{95}
\end{equation*}
$$

can be solved for $\boldsymbol{\Delta} \boldsymbol{a}$ with help of DGEFA and DGESL (see p. 27):
CALL DGEFA(FPRIME, NDIM, NDIM, IPVT, INFO)
IF (INFO .NE. O) STOP 'JACOBIAN MATRIX WITH O ON DIAGONAL'
CALL DGESL(FPRIME, NDIM, NDIM, IPVT, F)
where the solution $\boldsymbol{\Delta} \boldsymbol{a}$ is written to vector F

