Introduction to

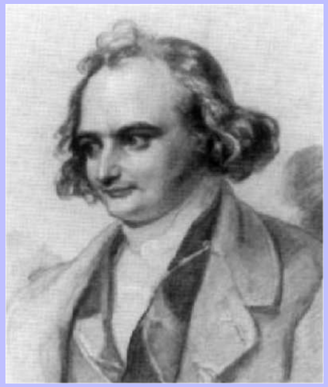
Computational Physics

University of Maribor,
Summer Term 2003

Franz J. Vesely
University of Vienna

www.ap.univie.ac.at/users/Franz.Vesely/

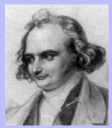
2. Linear Algebra



Carl Gustav Jacob Jacobi taught us to relax

- Subject too large, excellent textbooks
- Many library subroutines exist
- But: "physical" matrices often simple in structure
- Specific algorithms that may (may!) be self-programmed
- We will concentrate on **Relaxation Methods**

But before that, some general remarks $\Longrightarrow$



Given $f(x)$, introduce *finite differences*

$\Longrightarrow$ *Vector* $\mathbf{f} \equiv (f_k\,;\; k = 1, \ldots, M)$

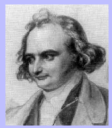
Similarly, given $f(x, y)$ or $f(x, t)$

$\Longrightarrow$ *Matrix* $\mathbf{F} \equiv [f_{i,j}] \equiv [f(x_i, y_j)\,;\; i = 1, \ldots M;\, j = 1, \ldots N]$

Approximate the various *differentials* by *differences*:

$\Longrightarrow$ Convert *Partial Differential Equations* (PDEs) into *Systems of Linear Equations* $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$.

As a rule $\mathbf{A}$ has a simple structure: *sparse*, *diagonally dominated*, *positive definite*, etc.



Fundamental manipulations:

- Invert a matrix:

$$\mathbf{A} \iff \mathbf{A}^{-1}$$

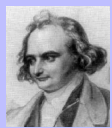
- Find the solution to the system of equations:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

- Find the eigenvalues λ_i and the eigenvectors $\mathbf{a}_i$ of a quadratic matrix:

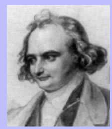
$$\left.\begin{array}{rcl} |\mathbf{A} - \lambda_i \mathbf{I}| & = & 0 \\ (\mathbf{A} - \lambda_i \mathbf{I}) \cdot \mathbf{a}_i & = & 0 \end{array}\right\} \qquad i = 1, \ldots N$$

  (will be skipped in this course)



Solve $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ exactly:

- Gauss Elimination and Back Substitution
- Householder Transformation
- LU Decomposition
- Recursion Method



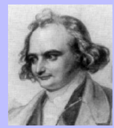
Gauss Elimination and Back Substitution:

$$\begin{pmatrix} a_{11} & a_{12} & . & . & a_{1N} \\ a_{21} & a_{22} & & & \\ . & & . & & \\ . & & & . & \\ . & & & & a_{NN} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ . \\ . \\ . \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ . \\ . \\ . \\ b_N \end{pmatrix}$$

Convert this to *triangular* form:

$$\begin{pmatrix} a'_{11} & a'_{12} & . & . & . \\ 0 & a'_{22} & & & \\ . & . & . & & \\ . & & . & . & \\ 0 & . & . & 0 & a'_{NN} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ . \\ . \\ . \\ x_N \end{pmatrix} = \begin{pmatrix} b'_1 \\ . \\ . \\ . \\ b'_N \end{pmatrix}$$

Then solve the system by *Back Substitution*.



LU Decomposition:

Split $\mathbf{A}$ into a *Lower* and an *Upper* triangular matrix:

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U}$$

Then solve by substitution.

Householder Transformation:

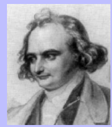
Systematic procedure to strip off elements in rows or columns of $\mathbf{A}$:

Given $\mathbf{A} \longrightarrow \mathbf{A}'$ *triangular*, *tridiagonal*, or otherwise *simple*.

Recursion:

Find solution $\mathbf{x}$ if $\mathbf{A}$ is *tri-diagonal* (maybe after Householder).

More on Recursion $\Longrightarrow$



Recursion Method:

With

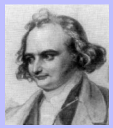
$$
\mathbf{A} \equiv \begin{pmatrix} \beta_1 & \gamma_1 & 0 & . & . & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & 0 & . & 0 \\ 0 & \alpha_3 & \beta_3 & \gamma_3 & 0 & . \\ . & . & . & . & . & . \\ . & . & . & \alpha_{N-1} & \beta_{N-1} & \gamma_{N-1} \\ . & . & . & 0 & \alpha_N & \beta_N \end{pmatrix}
$$

the system of equations reads

$$
\begin{aligned}
\beta_1 x_1 + \gamma_1 x_2 &= b_1 \\
\alpha_i x_{i-1} + \beta_i x_i + \gamma_i x_{i+1} &= b_i\,; \quad i = 2, \ldots, N-1 \\
\alpha_N x_{N-1} + \beta_N x_N &= b_N
\end{aligned}
$$

Introducing auxiliary variables g_i and h_i by the recursive ansatz

$$
x_{i+1} = g_i x_i + h_i\,; \; i = 1, \ldots, N-1
$$



we find the "downward recursion formulae"

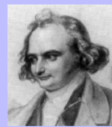
$$g_{N-1} = \frac{-\alpha_N}{\beta_N} \quad , \quad h_{N-1} = \frac{b_N}{\beta_N}$$

$$g_{i-1} = \frac{-\alpha_i}{\beta_i + \gamma_i \, g_i} \quad , \quad h_{i-1} = \frac{b_i - \gamma_i \, h_i}{\beta_i + \gamma_i \, g_i} \, ; \quad i = N-1, \ldots, 2$$

Having arrived at g_1 and h_1 we insert the known values of g_i, h_i in the "upward recursion formulae"

$$\begin{aligned} x_1 &= \frac{b_1 - \gamma_1 \, h_1}{\beta_1 + \gamma_1 \, g_1} \\ x_{i+1} &= g_i \, x_i + h_i \, ; \quad i = 1, \ldots, N-1 \end{aligned}$$

(The equation for the starting value x_1 follows from $\beta_1 x_1 + \gamma_1 x_2 = b_1$ and $x_2 = g_1 x_1 + h_1$.)



Example: In $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, let

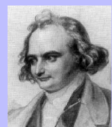
$$\mathbf{A} \equiv \begin{pmatrix} \beta_1 & \gamma_1 & 0 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & 0 \\ 0 & \alpha_3 & \beta_3 & \gamma_3 \\ 0 & 0 & \alpha_4 & \beta_4 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 2 & 3 & 1 & 0 \\ 0 & 1 & 4 & 2 \\ 0 & 0 & 1 & 3 \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

Downward recursion: $g_3 = -\alpha_4/\beta_4 = -1/3$, $h_3 = b_4/\beta_4 = 4/3$, and

$$\begin{array}{lll} i = 3: \quad g_2 = -3/10 & , & h_2 = 1/10 \\ i = 2: \quad g_1 = -20/27 & , & h_1 = 19/27 \end{array}$$

Upward recursion: $x_1 = 8/34$, and

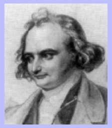
$$\begin{array}{lcl} i = 1: \quad x_2 & = & 9/17 \\ i = 2: \quad x_3 & = & -1/17 \\ i = 3: \quad x_4 & = & 23/17 \end{array}$$



Solve $\mathrm{A} \cdot \mathrm{x} = \mathrm{b}$ by iteration:

- Jacobi Relaxation
- Gauss-Seidel Relaxation (GSR)
- Successive Over-Relaxation (SOR)

But first: "Iterative Improvement". $\Longrightarrow$



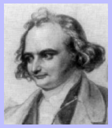
Iterative Improvement

Let $\mathbf{x}$ be the exact solution of $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$,
and let $\mathbf{x}'$ be an inaccurate (or estimated) solution vector, such that $\mathbf{x} \equiv \mathbf{x}' + \delta\,\mathbf{x}$.

Inserting this into the given equation we find

$$\mathbf{A} \cdot \delta\,\mathbf{x} \;=\; \mathbf{b} - \mathbf{A} \cdot \mathbf{x}' \equiv \mathbf{c}$$

which may be solved for $\delta\mathbf{x}$. (Use *double precision*!)



Example:

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \ \mathbf{b} = \begin{pmatrix} 3 \\ 2 \end{pmatrix} \text{ and } \mathbf{x}' = \begin{pmatrix} -3 \\ 4 \end{pmatrix}$$

From

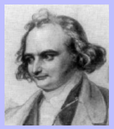
$$\mathbf{A} \cdot \delta\, \mathbf{x} = \begin{pmatrix} 3 \\ 2 \end{pmatrix} - \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} -3 \\ 4 \end{pmatrix} = \begin{pmatrix} -2 \\ -5 \end{pmatrix}$$

we find, using the decomposition

$$\mathbf{L} = \begin{pmatrix} 1 & 0 \\ 3 & 1 \end{pmatrix} \text{ and } \mathbf{U} = \begin{pmatrix} 1 & 2 \\ 0 & -2 \end{pmatrix}$$

the correction vector

$$\delta\, \mathbf{x} = \begin{pmatrix} -1 \\ -\frac{1}{2} \end{pmatrix} \quad \text{so} \quad \text{that} \quad \mathbf{x} = \begin{pmatrix} -4 \\ \frac{7}{2} \end{pmatrix}$$



Relaxation methods:

Now interpret the improvement equation as an iterative formula:

$$\mathbf{A} \cdot (\mathbf{x}_{k+1} - \mathbf{x}_k) \quad = \quad \mathbf{b} - \mathbf{A} \cdot \mathbf{x}_k$$

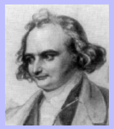
Replace $\mathbf{A}$ *on the left hand side* by an easily invertible matrix $\mathbf{B}$ close to $\mathbf{A}$:

$$\mathbf{B} \cdot (\mathbf{x}_{k+1} - \mathbf{x}_k) \quad = \quad \mathbf{b} - \mathbf{A} \cdot \mathbf{x}_k$$

or

$$\mathbf{x}_{k+1} \quad = \quad \mathbf{B}^{-1} \cdot \mathbf{b} + \mathbf{B}^{-1} \cdot [\mathbf{B} - \mathbf{A}] \cdot \mathbf{x}_k$$

This procedure converges to the solution of $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ if $|\mathbf{x}_{k+1} - \mathbf{x}_k| < |\mathbf{x}_k - \mathbf{x}_{k-1}|$. This is the case if all eigenvalues of the matrix $\mathbf{B}^{-1} \cdot [\mathbf{B} - \mathbf{A}]$ are situated within the unit circle.



Jacobi Relaxation:

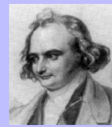
Divide the given matrix according to $\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{R}$ where $\mathbf{D}$ contains only the diagonal elements of $\mathbf{A}$, while $\mathbf{L}$ and $\mathbf{R}$ are the left and right parts of $\mathbf{A}$, respectively.

Choose $\mathbf{B} = \mathbf{D}$ and write the iteration formula as

$$\boxed{\mathbf{D} \cdot \mathbf{x}_{k+1} = \mathbf{b} + [\mathbf{D} - \mathbf{A}] \cdot \mathbf{x}_k}$$

or

$$a_{ii}\, x_i^{(k+1)} \quad = \quad b_i - \sum_{j \neq i} a_{ij}\, x_j^{(k)}\,; \quad i = 1, \ldots, N$$



Example: In $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ let

$$\mathbf{A} = \begin{pmatrix} 3 & 1 \\ 2 & 4 \end{pmatrix}; \quad \mathbf{b} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

Starting from the estimated solution

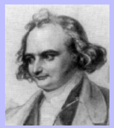
$$\mathbf{x}_0 = \begin{pmatrix} 1.2 \\ 0.2 \end{pmatrix}$$

and using the diagonal part of $\mathbf{A}$,

$$\mathbf{D} = \begin{pmatrix} 3 & 0 \\ 0 & 4 \end{pmatrix}$$

in the iteration we find the increasingly more accurate solutions

$$\mathbf{x}_1 = \begin{pmatrix} 0.933 \\ -0.100 \end{pmatrix}; \quad \mathbf{x}_2 = \begin{pmatrix} 1.033 \\ 0.033 \end{pmatrix} \; etc. \; \rightarrow \mathbf{x}_\infty = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



Convergence rate:

Writing the Jacobi scheme in the form

$$\mathbf{x}_{k+1} = \mathbf{D}^{-1} \cdot \mathbf{b} + \mathbf{D}^{-1} \cdot [\mathbf{D} - \mathbf{A}] \cdot \mathbf{x}_k \equiv \mathbf{D}^{-1} \cdot \mathbf{b} + \mathbf{J} \cdot \mathbf{x}_k$$

with the *Jacobi block matrix*

$$\mathbf{J} \equiv \mathbf{D}^{-1} \cdot [\mathbf{D} - \mathbf{A}] = -\mathbf{D}^{-1} \cdot [\mathbf{L} + \mathbf{R}]$$

convergence requires that all eigenvalues of $\mathbf{J}$ be smaller than one (by absolute value). Denoting the largest eigenvalue (the *spectral radius*) of $\mathbf{J}$ by λ_J, we have for the asymptotic rate of convergence

$$r_J \equiv \frac{|\mathbf{x}_{k+1} - \mathbf{x}_k|}{|\mathbf{x}_k - \mathbf{x}|} \approx |\lambda_J - 1|$$

In the above example $\lambda_J = 0.408$ and $r \approx 0.59$.

Gauss-Seidel Relaxation (GSR):

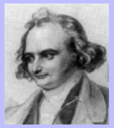
Somewhat faster convergent than Jacobi.
Choose $\mathbf{B} = \mathbf{D} + \mathbf{L}$ (i. e. lower triangle):

$$\boxed{[\mathbf{D} + \mathbf{L}] \cdot \mathbf{x}_{k+1} = \mathbf{b} - \mathbf{R} \cdot \mathbf{x}_k}$$

Solving the set of *implicit* equations

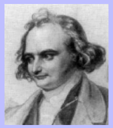
$$a_{ii}\, x_i^{(k+1)} + \sum_{j<i} a_{ij}\, x_i^{(k+1)} \quad = \quad b_i - \sum_{j>i} a_{ij}\, x_j^{(k)}\,;\; i = 1, \ldots, N$$

is not quite as simple as solving the *explicit* Jacobi equations. However, since the matrix $\mathbf{D} + \mathbf{L}$ is triangular the additional effort is affordable.



Example: With the same data as in the previous example we find the first two improved solutions

$$\mathbf{x}_1 = \begin{pmatrix} 0.933 \\ 0.033 \end{pmatrix}; \ \mathbf{x}_2 = \begin{pmatrix} 0.989 \\ 0.006 \end{pmatrix}.$$



The *convergence rate* of the GSR scheme is governed by the matrix

$$\mathbf{G} \equiv -[\mathbf{D}+\mathbf{L}]^{-1} \cdot \mathbf{R}$$

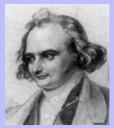
It can be shown that the spectral radius of $\mathbf{G}$ is given by

$$\lambda_G = \lambda_J^2$$

so that the rate of convergence is now

$$r_G \approx \left|\lambda_J^2 - 1\right|$$

In our example $\lambda_G = 0.17$ and $r \approx 0.83$.



Successive Over-Relaxation (SOR):

At each iteration step, compute the new vector $\mathbf{x}_{k+1}$ using GSR; then "mix it" with the previous vector $\mathbf{x}_k$:

$$\mathbf{x}_{k+1}^{SOR} \quad = \quad \omega\, \mathbf{x}_{k+1}^{GSR} + (1-\omega)\mathbf{x}_k$$

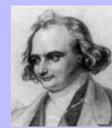
The "relaxation parameter" ω may be varied within the range $0 \leq \omega \leq 2$ to optimize the method.

The complete iteration formula is

$$\boxed{[\mathbf{D}+\mathbf{L}] \cdot \mathbf{x}_{k+1} = \omega\, \mathbf{b} - [\mathbf{R} - (1-\omega)\, \mathbf{A}] \cdot \mathbf{x}_k}$$

A single row in this system of equations reads

$$\begin{aligned} a_{ii}\, x_i^{(k+1)} + \sum_{j<i} a_{ij}\, x_j^{(k+1)} \quad &= \quad \omega\, b_i - \omega \sum_{j>i} a_{ij}\, x_j^{(k)} + \\ &+(1-\omega) \sum_{j\leq i} a_{ij} x_j(k) \qquad i = 1, \ldots, N \end{aligned}$$



The *rate of convergence* of this procedure is governed by the matrix

$$\mathbf{S} \quad \equiv \quad -\left[\mathbf{D}+\mathbf{L}\right]^{-1} \cdot \left[\mathbf{R} - (1-\omega)\,\mathbf{A}\right]$$

The optimal value of ω is given by

$$\omega_{opt} \quad = \quad \frac{2}{1+\sqrt{1-\lambda_J^2}}$$

yielding

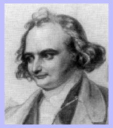
$$\lambda_S \quad = \quad \left[\frac{\lambda_J}{1+\sqrt{1-\lambda_J^2}}\right]^2$$

The asymptotic rate of convergence is

$$r_S \quad \approx \quad |\lambda_S - 1|$$

Example: With the same data as before we find an optimal relaxation parameter $\omega_{opt} = 1.046$, and from that $r_s = 0.95$. The first two iterations yield

$$\mathbf{x}_1 = \begin{pmatrix} 0.921 \\ 0.026 \end{pmatrix}; \ \mathbf{x}_2 = \begin{pmatrix} 0.994 \\ 0.003 \end{pmatrix}.$$



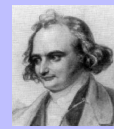
Chebyscheff Acceleration:

During the first few iterative steps the SOR procedure may give rise to overshooting corrections – particularly if ω is distinctly larger than 1. $\Longrightarrow$ Adjust ω on the fly: Start out with $\omega = 1$, then approach ω_{opt}.

- Split the solution vector $\mathbf{x}$ in even and odd elements: $\mathbf{x}_e$, $\mathbf{x}_o$; do the same with $\mathbf{b}$.

- The two subvectors $\mathbf{x}_e$ and $\mathbf{x}_o$ are iterated in alternating succession, with the relaxation parameter being adjusted according to

$$\begin{aligned} \omega^{(0)} &= 1 \\ \omega^{(1)} &= \frac{1}{1 - \lambda_J^2/2} \\ \omega^{(k+1)} &= \frac{1}{1 - \lambda_J^2 \omega^{(k)}/4}, \quad k = 1, \ldots \end{aligned}$$



Sample Application of Linear Algebra: Thermal Conduction

Again, discretize the equation of thermal conduction,

$$\frac{\partial T(x,t)}{\partial t} \;=\; \lambda \frac{\partial^2 T(x,t)}{\partial x^2}$$

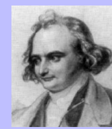
Earlier we applied DNGF to the l.h.s. and DDST *at time* t_n to the r.h.s.:

$$\frac{\partial T(x,t)}{\partial x^2} \;\approx\; \frac{\delta_i^2 T_i^n}{(\Delta x)^2}$$

In this manner we arrived at the “FTCS-” formula.

Now we may use the DDST formula *at time* t_{n+1},

$$\frac{\partial T(x,t)}{\partial x^2} \;\approx\; \frac{\delta_i^2 T_i^{n+1}}{(\Delta x)^2}$$



This leads us to the “implicit scheme of first order”

$$\frac{1}{\Delta t}[T_i^{n+1} - T_i^n] = \frac{\lambda}{(\Delta x)^2}[T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}]$$

which may be written, using $a \equiv \lambda\,\Delta t/(\Delta x)^2$,

$$\boxed{-aT_{i-1}^{n+1} + (1+2a)T_i^{n+1} - aT_{i+1}^{n+1} = T_i^n}$$

or

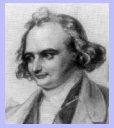
$$\mathbf{A} \cdot \mathbf{T}^{n+1} = \mathbf{T}^n$$

where (for fixed T_0 and T_N)

$$\mathbf{A} \equiv \begin{pmatrix} 1 & 0 & 0 & . & . & 0 \\ -a & 1+2a & -a & 0 & . & 0 \\ 0 & . & . & . & 0 & . \\ . & . & . & . & . & . \\ . & . & . & 0 & 0 & 1 \end{pmatrix}$$

Invert this tridiagonal system by the *Recursion Method*.

<u>Exercise:</u> Redo the earlier exercise on *One-dimensional thermal conduction* by applying the implicit scheme in place of the FTCS method. Use various values of Δt (and therefore a.) Compare the efficiencies and stabilities of the two methods.



Sample Application of Linear Algebra: Potential Equation

Discretize the *elliptic* PDE

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\rho$$

$\Longrightarrow$

$$\frac{1}{(\Delta x)^2}\left[u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right] = -\rho_{i,j}$$
$$i = 1, \ldots N;\ j = 1, \ldots M$$

Combining the N row vectors $\{u_{i,j}\,;\ j = 1, \ldots M\}$ sequentially to a vector $\mathbf{v}$ of length $N.M$ we may write these equations in the form

$$\mathbf{A} \cdot \mathbf{v} = \mathbf{b}$$

where $\mathbf{A}$ is a sparse matrix, and where the vector $\mathbf{b}$ contains the charge density ρ and the given boundary values of the potential function u.

Solve by applying any of the *Relaxation Methods*.