UPPSALA UNIVERSITET

Numerical Methods in Scientific Computing

NGSSC

Uppsala University

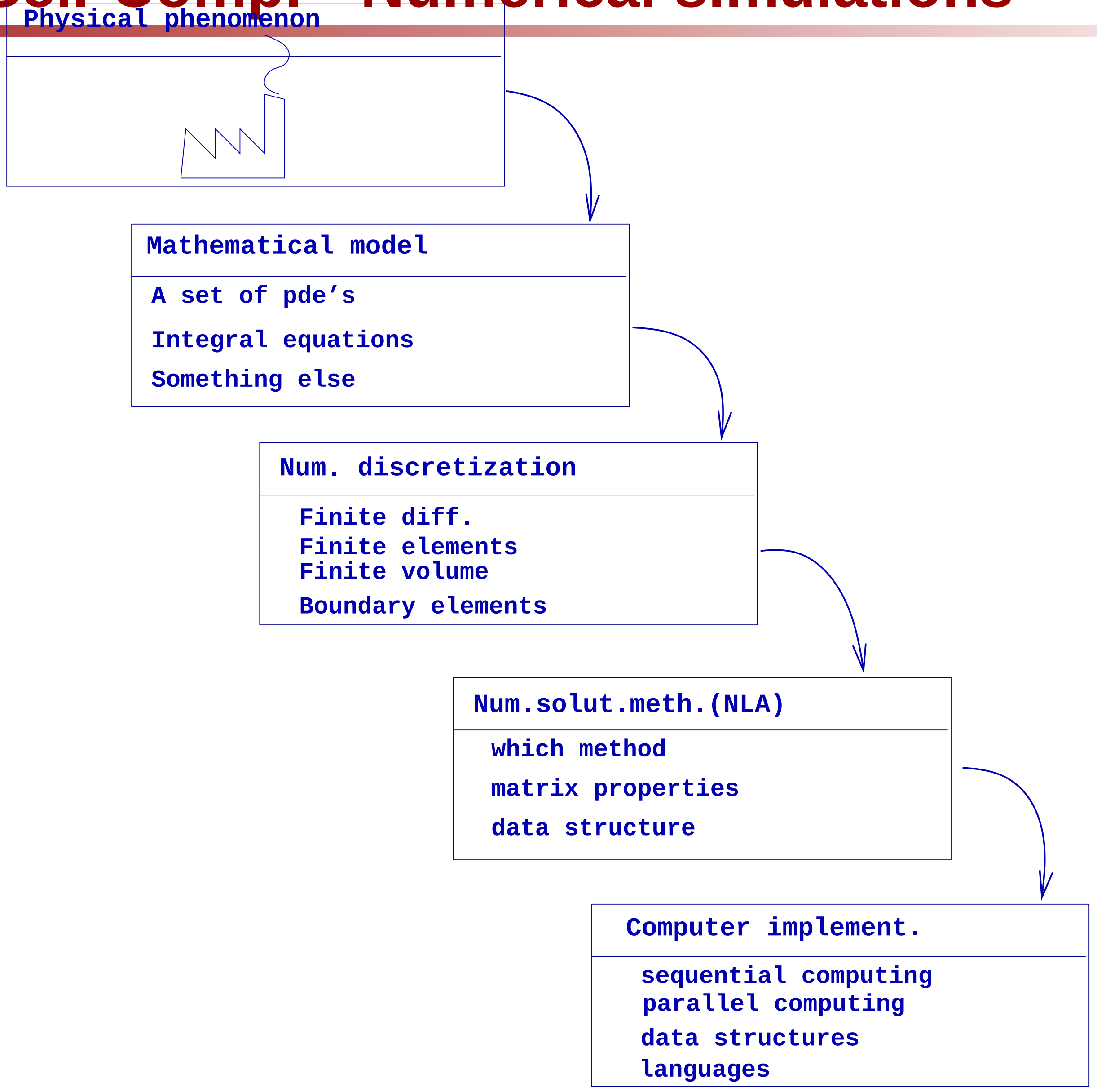
January 2012

NGSSC
FORSKARSKOLA I TEKNISK-VETENSKAPLIGA BERÄKNINGAR
NATIONAL GRADUATE SCHOOL IN SCIENTIFIC COMPUTING

Why this course ?

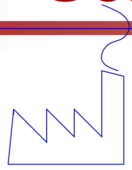
What to expect ?

Sci. Comp. - Numerical simulations



Sci. Comp. - Numerical simulations

Physical phenomenon



Mathematical model

- A set of pde's
- Integral equations
- Something else

Model error

Num. discretization

- Finite diff.
- Finite elements
- Finite volume
- Boundary elements

Discretization error

Stability wrt discretization parameters

Num.solut.meth.(NLA)

- which method
- matrix properties
- data structure

Solution error (iterations, stop.crit.)

Robustness

Method parameters

Computer implement.

- sequential computing
- parallel computing
- data structures
- languages

Scalability

Execution bottlenecks, memory

parallelization

Day 1 and 2:

TARGET 1: solve a linear system of equations

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

GOAL: get a global overview of what the needs are, how far we can go, and with which methods.

- ➜ how large systems we want to solve nowadays;
- ➜ what kind of systems;
- ➜ what are the methods of choice;
- ➜ how far we can go with a particular technique.

Day 3 and 4:

TARGET 2: Discretization of ordinary and partial differential equations
GOAL: Understand how the type of the problem influences the choice of discretization parameters

- ➜ Systems of ordinary differential equations (ODEs)
- ➜ Elliptic PDEs;
- ➜ Parabolic PDEs;
- ➜ Hyperbolic PDEs.

Day 5:

TARGET 3: Monte Carlo methods
GOAL: Get basic understanding of the MC methods and their potential (and limitations) in particular for high dimensional problems.

UPPSALA UNIVERSITET

Day 1 and 2:

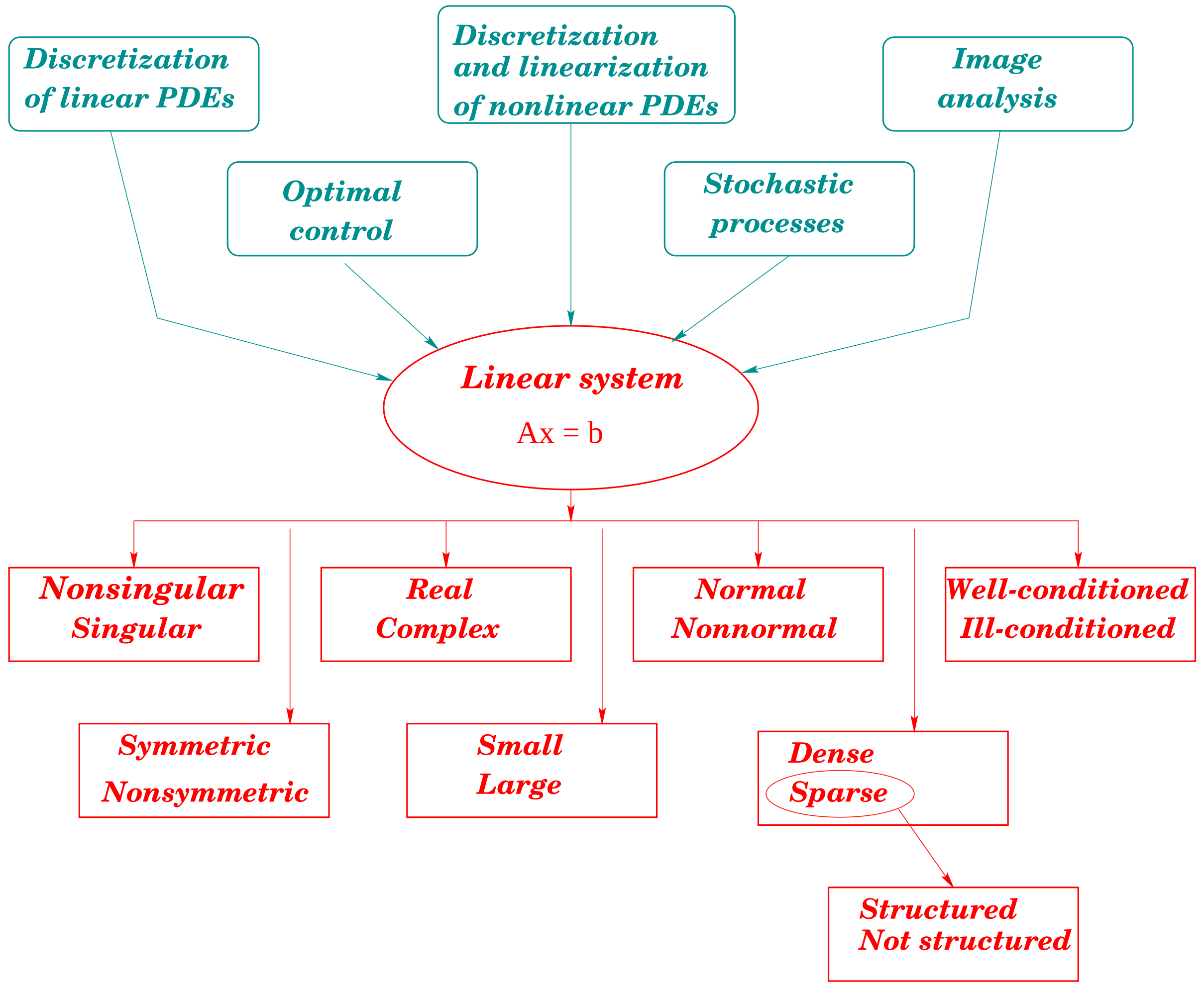
- ➜ **Introduction**
- ➜ **Direct solution methods** for linear systems – dense and sparse matrices
- ➜ **Iterative solution methods**
  - ➤ Basic techniques
  - ➤ Projection methods
    - ➢ The Conjugate Gradient method (CG)
    - ➢ The Generalized Conjugate Gradient method (GCG)
  - ➤ Preconditioning
    - ➢ (Block-)ILU
    - ➢ Approximate inverses
    - ➢ Block preconditioners (block-triangular, block-factorized, Schur complement-based)
    - ➢ Multilevel methods

Partial list of application fields

Stiff ODEs
Differential-Algebraic Equations
Linear programming (simplex, interior point methods)
Optimization
Nonlinear equations
Elliptic PDEs
Eigensolutions
Two-point boundary value problems
Least Squares calculations

More application fields

acoustic scattering	demography	network flow
air traffic control	economics	oceanography
astrophysics	electrical eng.	petroleum eng.
biochemical	electric nets	reactor modellin
chemical eng.	climate/pollution studies	statistics
chemical kinetics	fluid flow	structural eng
circuit physics	laser optics	survey data
computer simulations	linear programming	signal processin


Discretization of linear PDEs
Discretization and linearization of nonlinear PDEs
Image analysis
Optimal control
Stochastic processes
Linear system
Ax = b
Nonsingular Singular
Real Complex
Normal Nonnormal
Well-conditioned Ill-conditioned
Symmetric Nonsymmetric
Small Large
Dense Sparse
Structured Not structured

When talking about the solution of a linear system of equations:

- computer demands (computing time and memory consumption)
- numerical efficiency (number of iterations)
- computational complexity
- robustness wrt to (problem, discretization and method) parameters

Computational complexity issues: Cramer's rule

$$A\mathbf{x} = \mathbf{b}, \quad A(n \times n), \quad det(A) \neq 0$$

$$\begin{bmatrix} a_{11} & \cdots & a_{1,i-1} & a_{1,i} & a_{1,i+1} & \cdots & a_{1,n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i1} & \cdots & a_{i,i-1} & a_{i,i} & a_{i,i+1} & \cdots & a_{i,n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & \cdots & a_{n,i-1} & a_{n,i} & a_{n,i+1} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ \dots \\ x_i \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \dots \\ b_i \\ \dots \\ b_n \end{bmatrix}$$

Computational complexity issues: Cramer's rule

$$x_i = \frac{1}{det(A)} \left(\begin{bmatrix} a_{11} & \cdots & a_{1,i-1} & b_1 & a_{1,i+1} & \cdots & a_{1,n} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{i1} & \cdots & a_{i,i-1} & b_i & a_{i,i+1} & \cdots & a_{i,n} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{n1} & \cdots & a_{n,i-1} & b_n & a_{n,i+1} & \cdots & a_{n,n} \end{bmatrix} \right)$$

UPPSALA UNIVERSITET

Computational complexity issues

Consider products of n elements of A,

$$a_{1,\alpha_1}, a_{2,\alpha_2}, \cdots, a_{n,\alpha_n},$$

where $\alpha_1, \alpha_2, \cdots, \alpha_n$ is a permutation of $1, 2, \cdots, n$.
The number of all these products is $\boxed{n!}$.

$$det(A) = \sum_{i=1}^{n!} \prod_{j=1}^{n} (-1)^\gamma a_{j,\alpha_j},$$

thus, the computational complexity to solve the system is $n!$.
To be more precise: $(n+1)(n!) = (n+1)!$ multiplications and
$(n+1)(n!) = (n+1)!$ additions.

Computational complexity issues: computing $det(A)$

Method	Multiplications	Additions
Gaussian Elimination	$\frac{1}{3}n^3 + n^2 - \frac{1}{3}n$	$\frac{1}{3}n^3 + \frac{1}{2}n^2 - \frac{5}{6}n$
Gauss-Jordan Elimination	$\frac{1}{3}n^3 + n^2 - \frac{5}{2}n + 2$	$\frac{1}{3}n^3 - \frac{3}{2}n^2 + 1$
Solving using the factors obtained by Gaussian elimination	$\frac{5}{6}n^3 + 2n^2 - \frac{5}{6}n$	$\frac{4}{3}n^3 - \frac{1}{2}n^2 - \frac{5}{6}n$
Solving using the factors obtained by Gauss-Jordan elimination	$\frac{3}{2}n^3 + \frac{1}{2}n - 1$	$\frac{3}{2}n^3 - 2n^2 - \frac{1}{2}n$
Cramer’s Rule	$n!$	$n!$

Computational complexity issues:

Matrix inversion	Gauss-Jordan elimination	$O(n^3)$
	Strassen algorithm	$O(n^{2.807})$
	Coppersmith-Winograd algorithm	$O(n^{2.376})$
Determinant	Laplace expansion	$O(n!)$
	LU decomposition	$O(n^3)$
	Bareiss algorithm	$O(n^3)$
	Fast matrix multiplication	$O(n^{2.376})$

Computational complexity issues: Cramer against Gauss

A comparison of the amount of time to solve $A\mathbf{x} = \mathbf{b}$ on a Cray J90. The Cray J90 performs one trillion operations per second (one teraflop).

n	Gaussian Elimination	Cramer's Rule
2	6 x 10^{-12} secs	6 x 10^{-12} secs
3	1.7 x 10^{-11} secs	2.4 x 10^{-11} secs
4	3.6 x 10^{-11} secs	1.2 x 10^{-10} secs
5	6.5 x 10^{-11} secs	7.2 x 10^{-10} secs
6	1.06 x 10^{-10} secs	5.04 x 10^{-09} secs
10	4.3 x 10^{-10} secs	3.99168 x 10^{-05} secs
20	3.06 x 10^{-9} secs	1.622 years
100	3.433 x 10^{-7} secs	2.9889 x 10^{138} centuries
1000	3.3433 x 10^{-4} secs	

Computational complexity issues: Cramer against Gauss

In November 2011, it was announced that Japan had achieved 10.51 petaflops with its K computer.
Will tera-, peta-computers change much?

Factorials...

In 2001, the value of 1000! was currently too large to be stored as a single number in the memory of a computer. (*Computational Science: Tools for a Changing World* by R.A. Tapia, C. Lanius, 2001, Rice.)

The scientific calculator in Windows XP is able to calculate factorials up to at least 100000!.
(look-up tables)

mine

UPPSALA
UNIVERSITET

Grand challenge problems

Environment

- High-resolution weather forecasting: more accurate, faster and longer range predictions
- Pollution studies, including cross-pollutant interactions
- Global atmosphere-ocean-biosphere and long range climate modelling

  $\Rightarrow$ Risø National Laboratory & Technical University of Denmark, Roskilde

UPPSALA UNIVERSITET

DEM - air pollution model

Danish Eulerian model: to perform numerical simulations and to study the air pollution over Europe.
Task: establish reliable control strategies for the air pollution.

DEM - air pollution model

Danish Eulerian model: to perform numerical simulations and to study the air pollution over Europe.

Task: establish reliable control strategies for the air pollution.

Basic scheme: pollutants are emitted in the air from various sources

UPPSALA UNIVERSITET

DEM - air pollution model

Danish Eulerian model: to perform numerical simulations and to study the air pollution over Europe.
Task: establish reliable control strategies for the air pollution.
Basic scheme: pollutants (many of them) are emitted in the air from various sources (many of them) and

- transported by the wind - (diffusion, advection, horizontal vertical)
- get deposited on the Earth surface
- transform due to chemical reactions
  - factors: winds, temperature, humidity, day/night, ...

UPPSALA UNIVERSITET

DEM - air pollution model

$$\frac{\partial c_s}{\partial t} + - \sum_{i=1}^{3} \frac{\partial u_i \, c_s}{\partial x_i} + \sum_{i=1}^{3} \frac{\partial}{\partial x_i} \left(K_{x_i} \frac{\partial c_s}{\partial x_i} \right) + E_s + k_s c_s + Q(c_1, \cdots, c_q),$$

where $c_s, \quad s = 1, \cdots, q$ are the unknown concentrations of q species of the air pollutants to be followed.

DEM - air pollution model

Demands:

UPPSALA UNIVERSITET

DEM - air pollution model

Demands:

Spatial domain: 4800 km^2, 96×96 or 480×480 regular mesh (horizontal resolution)

UPPSALA UNIVERSITET

DEM - air pollution model

Demands:

Spatial domain: 4800 km^2, 96×96 or 480×480 regular mesh (horizontal resolution)
10 vertical layers, 1 km each

35 pollutants

UPPSALA UNIVERSITET

DEM - air pollution model

Demands:

Spatial domain: 4800 km^2, 96×96 or 480×480 regular mesh (horizontal resolution)
10 vertical layers, 1 km each

35 pollutants

3D: $35 * 10 * 96^2 = 3225600$ unknowns

3D: $35 * 10 * 480^2 = 80640000$ unknowns

about 36000 time-steps, 'only one month simulated'

UPPSALA UNIVERSITET

DEM - air pollution model

Demands:

Spatial domain: 4800 km^2, 96×96 or 480×480 regular mesh (horizontal resolution)
10 vertical layers, 1 km each

35 pollutants

3D: $35 * 10 * 96^2 = 3225600$ unknowns

3D: $35 * 10 * 480^2 = 80640000$ unknowns

about 36000 time-steps, 'only one month simulated'

Computers at DCAMM: 16 PEs on Newton:

60 000 sec = 1000 min $\approx$ 17 h   Is this much or not?

The topics of the Sci.Comp. course - a cross-disciplinary point

Conducting a numerical simulation:

- Modelling
- Discretization
- Choice of a solution method
- Computer implementation
- Postprocessing

The topics of the Sci.Comp. course - a cross-disciplinary point

- Modelling:modelling error

The topics of the Sci.Comp. course - a cross-disciplinary point

- Modelling: modelling error
- Discretization: discretization error (space, time,stability...)

The topics of the Sci.Comp. course - a cross-disciplinary point

- Modelling: modelling error
- Discretization: discretization error (space, time, stability...)
- Choice of a solution method: iteration error (robustness wrt discretization parameters,)

Large dense matrices

An idea what matrix dimensions might have been considered very large for a dense, direct matrix computation through the years:

n	Year	Source
20	1950	Wilkinson
200	1965	Forsythe&Moler
2000	1980	LINPACK
20000	1995	LAPACK
$> 200\,000$	today	(Umeå)

J. Wilkinson, The algebraic eigenvalue problem, 1965
G. Forsythe& C. Moler, Computer solutions of linear algebraic systems, 1967.

Matrix factorizations (dense matrices)

- Schur's form: $A = U^T R U$
- LU
- LL^T or $U^T U$
- LDU

UPPSALA UNIVERSITET

LU factorization $A(m,n)$

```
$for\ k = 1, 2 \cdots m-1$
    $d = 1/a_{kk}^{(k)}$
    $for\ i = k+1, \cdots m$
        $\ell_{ik}^{(k)} = -a_{ik}^{(k)}\ d$
        $for\ j = k+1, \cdots n$
            $a_{ij}^{(k)} = a_{ij}^{(k)} + \ell_{ik} a_{kj}^{(k)}$
        $end$
    $end$
$end$
```

The operational count for the LU factorization can be obtained by integrating the loops:

$$Flops_{LU} = \int_1^{m-1} \int_k^{m} \int_k^{n} d_j d_i d_k \approx n^3/3\ (m=n)$$

The LDU factorization

Theorem:

If all leading principal minors of A are nonsingular, then there exist unique lower-triangular matrix L, diagonal matrix D and upper-triangular matrix U, such that $A = LDU$.

How do we compute $A = LDU$?

Note: If e_j is the jth unit vector, then $Me_j = M(:,j)$.

Assume that the first $j-1$ columns of L, $j-1$ elements of D and $j-1$ rows of U are computed.

$$\begin{bmatrix} 1 & & & & & \\ * & \ddots & & & & \\ * & * & 1 & & & \\ ? & ? & ? & 1 & & \\ ? & ? & ? & ? & \ddots & \\ ? & ? & ? & ? & ? & 1 \end{bmatrix} \begin{bmatrix} d_1 & & & & & \\ & \ddots & & & & \\ & & d_{j-1} & & & \\ & & & ? & & \\ & & & & \ddots & \\ & & & & & ? \end{bmatrix} \begin{bmatrix} 1 & * & * & ? & ? & ? \\ & \ddots & * & ? & ? & ? \\ & & 1 & ? & ? & ? \\ & & & 1 & ? & ? \\ & & & & \ddots & ? \\ & & & & & 1 \end{bmatrix}$$

How do we compute $A = LDU$?

We equate the jth column of $A = LDU$:
Denote $v = DUe_j$. Then $Ae_j = Lv$

- $v(1:j) = L(1:j,1:j)^{-1}A(1:j,j)$ - known data
- $d(j) = v(j)$
- $U(i,j) = v(i)/d(i), \; i = 1:j-1$
- $L(j+1:n,j)v(1:j) = A(j+1,j)$

Why bother about dense factorizations and the tricks behind?

What about this...

"Dense Matrix Factorization of Linear Complexity for Impedance Extraction of Large-Scale 3-D Integrated Circuits"
Wenwen Chai, Dan Jiao School of Electrical and Computer Engineering, Purdue University
IEEE Xplore, July 2010

Abstract: A fast LU factorization of linear complexity is developed to directly solve a dense system of linear equations for the interconnect extraction of any arbitrary shaped 3-D structure embedded in inhomogeneous materials. The proposed solver successfully factorizes dense matrices that involve more than one million unknowns in fast CPU run time and modest memory consumption. Comparisons with state-of-the-art integral equation- based interconnect extraction tools have demonstrated its clear advantages.

Factorizing symmetric positive definite matrices

Factorizing symmetric positive definite matrices

Factorize $A = LL^T$, L – lower-triangular
Cholesky factorization

UPPSALA UNIVERSITET

Factorizing symmetric marices

The mathematician after whom the Cholesky factorisation is named

Major Andre-Louis Cholesky (1875-1918)

Cholesky was born in France, and worked in the Geodesic section of the Geographic service to the French army's artillery branch.

At this time the system of triangulation used in France, and based on the meridian line of Paris, was being revised; new methods were needed in order to facilitate what was not yet a quick or convenient process.

Cholesky invented computation procedures based on the method of least squares, for the solution of certain data-fitting problems in geodesy, to be put into practice in his triangulation of the French and British parts of Crete, and in his work in Algeria and Tunisia.

His mathematical work was posthumously published on his behalf in 1924 by a fellow officer, Benoit.

Cholesky factorization ...

```
% Maya’s version of Cholesky - to compare execution time
% ---------------------------------------------------------------
function [U]=my_chol(A)
A = triu(A);
n = size(A,1);
for k=1:n,
    A(1:k-1,k) = A(1:k-1,1:k-1)’\A(1:k-1,k);
    A(k,k) = sqrt(A(k,k) - A(1:k-1,k)’*A(1:k-1,k));
end
U = triu(A);
return
```

Cholesky factorization ...

size(A)	chol Matlab	chol mine	Ratio
10	$1.9\,10^{-5}$	0.00334	$1.76\,10^2$
50	$5\,10^{-5}$	0.00224	45
100	$1.5\,10^{-4}$	0.00517	34.5
500	0.0053	0.234918	44.32
1000	0.0259	2.117073	81.87
5000	2.7078	391.9548	144.8

GAXPY Cholesky

```
$for \quad k = 1:n$
    $if\ k > 1$
        $A(k:n,k) = A(k:n,k) - A(k:n,k-1) * A(k,1:k-1)^T$
    $endif$
    $A(k:n,k) = A(k:n,k)/\sqrt{A(k,k)}$
$end$
```

Outer Product Cholesky

```
$for \quad k = 1 : n$
    $A(k,k) = \sqrt{A(k,k)}$
    $A(k+1:n,k) = A(k+1:n,k) - A(n:k,k-1)/A(k,k)$
    $for\ j = k+1 : n$
        $A(j:n,j) = A(j:n,j) - A(j:n,j)A(j,k)$
    $end$
$end$
```

put together

```
$for$   $k = 1 : n$
        $if\ k > 1$
            $A(k:n,k) = A(k:n,k) - A(k:n,k-1) * A(k,1:k-1)^T$
        $endif$
        $A(k:n,k) = A(k:n,k)/\sqrt{A(k,k)}$
$end$
```

```
$for$   $k = 1 : n$
        $A(k,k) = \sqrt{A(k,k)}$
        $A(k+1:n,k) = A(k+1:n,k) - A(n:k,k-1)/A(k,k)$
        $for\ j = k+1 : n$
            $A(j:n,j) = A(j:n,j) - A(j:n,j)A(j,k)$
        $end$
$end$
```

Example of implementing Cholesky factorization

```
for k=1:n
    xeuitb(A(1:k-1,k),A(1:k-1,1:k-1),A(1:k-1,k))
    A(k,k) = sqrt(A(k,k) - A(1:k-1,k)^T*A(1:k-1,k)
end
```

Computes U (which overwrites A).

BLAS `xeuitb(X,U,B)` computes $X = U^{-1}B$