Fifth SimLab Short Course on

Parallel Numerical Simulation

Belgrade, October 1-7, 2006

Iterative Solution of Linear Systems

October 3, 2006

Hans-Joachim Bungartz
Department of Computer Science – Chair V
Technische Universität München, Germany

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

3.1. Standard Iterative Solvers of SLE

- iterative solution of large linear systems is one of the most important numerical tasks in scientific computing (they occur in the discretization of both ODE and PDE)
- direct solvers are often not competitive:
  - too large number of unknowns (cf. PDE in 3D)
  - sparse matrices (classical elimination destroys sparsity)
  - anyway only approximations, thus no need for exact solution (this holds esp. in the *nonlinear case*, where a system of linear equations occurs in each step of some outer iteration for the nonlinearity)
- objective: "for 3 digits, you need 10 steps" – no matter how big the number n of unknowns is
- however typically: speed of convergence deteriorates with increasing n!

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Principal Remarks on Iterations

- consider an iterative scheme, starting from $x^{(0)}$, and, hopefully converging to the solution x of $Ax = b$:

$$x^{(0)} \to x^{(0)} \to \cdots \to x^{(i+1)} \to \cdots \to \lim_{i \to \infty} x^{(i)} = x$$

- speed of convergence: $\|x - x^{(i+1)}\| < \gamma \cdot \|x - x^{(i)}\|$ for some $0 < \gamma < 1$
- typical behaviour of iterative schemes:

$$\gamma = O(1 - n^{-k}), \; k \in \{0, 1, 2, \dots\}$$

- strategy: look for iterative methods with
  - only $O(n)$ arithmetic operations per step of iteration (cost)
  - a convergence behaviour like $\gamma < 1 - \text{const} \ll 1$
- two big families: **relaxation** and **Krylov subspace** methods

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Relaxation Techniques 1

- sometimes also called smoothing methods:
  - **Richardson** iteration
  - **Jacobi** iteration
  - **Gauß-Seidel** iteration
  - **successive over relaxation (SOR)** or **damped** methods
- All start from the residual r after i steps of iteration:

$$r^{(i)} = b - Ax^{(i)} = Ax - Ax^{(i)} = A(x - x^{(i)}) = -Ae^{(i)}$$

(the error e is not known, so r is used as an indicator)

- How to obtain an improvement?
  - Richardson: use the residual as it is as a correction
  - Jacobi/Gauß-Seidel: make one component of r vanish
  - SOR/damped: same, but do a bit less/more than indicated

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Relaxation Techniques 2

- Richardson iteration:

  repeat(i) : for $k = 1, \ldots, n$ do $x_k^{(i+1)} = x_k^{(i)} + r_k^{(i)}$

- (damped) Jacobi iteration:

  repeat(i) : for $k = 1, \ldots, n$ do $y_k = r_k^{(i)} / a_{k,k}$

  for $k = 1, \ldots, n$ do $x_k^{(i+1)} = x_k^{(i)} + \alpha \cdot y_k$

  (compute and store updates, apply them at the end)

- Gauß-Seidel or SOR, resp.:

  repeat(i) : for $k = 1, \ldots, n$ do

$$r_k^{(i)} := b_k - \sum_{j=1}^{k-1} a_{kj} x_j^{(i+1)} - \sum_{j=k}^{n} a_{kj} x_j^{(i)}$$

$$y_k := \frac{1}{a_{kk}} \cdot r_k^{(i)}, \quad x_k^{(i+1)} := x_k^{(i)} + \alpha \cdot y_k$$

  (compute same updates, but apply them at once)

- For an analysis, decompose A in its strictly lower, diagonal, and strictly upper part: $A = L_A + D_A + U_A$

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Relaxation Techniques 3

- all can be written in the form $Mx^{(i+1)} + (A - M)x^{(i)} = b$ or
$$x^{(i+1)} = M^{-1}b - (M^{-1}A - I)x^{(i)} = x^{(i)} + M^{-1}r^{(i)}$$
- how looks M?
  - Richardson: $M := I$,
  - Jacobi: $M := D_A$,
  - Gauß-Seidel: $M := D_A + L_A$,
  - SOR: $M := \frac{1}{\alpha} D_A + L_A$.
- some convergence results:
  - If the iteration converges at all, then towards x.
  - The crucial quantity is the *spectral radius* of $-M^{-1} \cdot (A - M)$ which is smaller than 1 if and only if the iteration converges.
  - necessary for SOR: $0 < \alpha < 2$
  - sufficient for Gauß-Seidel/SOR: A positive definite
  - sufficient for Jacobi: both A **and** $2D_A - A$ are positive definite

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

3.2. Towards Krylov: Steepest Descent

- alternative point of view for positive definite A:

  $$x \text{ solves } Ax = b \Leftrightarrow x \text{ minimizes } f(x) = 0.5 \cdot x^T Ax - b^T x + c$$

  (uniqueness of minimum due to positive definiteness)

- hence new strategy: look for minimum of f

- possible way: method of *steepest descent* (looks for the best improvement in the direction of the negative gradient)

  $$\text{repeat}(i) : \alpha_i = \frac{r^{(i)T} r^{(i)}}{r^{(i)T} Ar^{(i)}}; x^{(i+1)} = x^{(i)} + \alpha_i r^{(i)}; r^{(i+1)} = r^{(i)} - \alpha_i Ar^{(i)};$$

  (1D minimization along search direction $-f'(x^{(i)}) = r^{(i)}$)

- even simpler: search along coordinate axes (is Gauß-Seidel!)

- slow convergence (progress may get lost again!)

- crucial quantity: spectral condition number of A

  $$\kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Improvement: Conjugate Directions

- further enhancement:
  - orthogonal search directions, error after i steps shall be orthogonal to all previous search directions
  - nothing destroyed, hence: in principle a direct method
  - however, since n iterations are too much in practice: used as an iterative method (therefore called *semi-iterative* method)
  - new search directions: $x^{(i+1)} = x^{(i)} + \alpha_i d^{(i)}$
  - optimum orthogonality: $0 = {d^{(i)}}^T e^{(i+1)}$, but error is missing
  - therefore *conjugation*: $0 = {d^{(i)}}^T A e^{(i+1)}$
    (u and v are called *A-orthogonal* or *conjugate*, if $u^T A v = 0$)
  - algorithm: start with $d^{(0)} = r^{(0)}$ and iterate:

$$\text{repeat}(i) : \alpha_i = \frac{{d^{(i)}}^T r^{(i)}}{{d^{(i)}}^T A d^{(i)}}; x^{(i+1)} = x(i) + \alpha_i d^{(i)};$$

$$r^{(i+1)} = r^{(i)} - \alpha_i A d^{(i)};$$

  - still to be done: construction of the conjugate directions $d^{(i)}$

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Finally: Conjugate Gradients

- above method + efficient construction of the conjugate directions
- principle of construction: Gram-Schmidt conjugation of r's
- no detailed derivation here, just the algorithm:

$$\begin{aligned}
\text{repeat}(i):\ \alpha_i &= \frac{d^{(i)^T} r^{(i)}}{d^{(i)^T} A d^{(i)}};\\
x^{(i+1)} &= x^{(i)} + \alpha_i d^{(i)};\\
r^{(i+1)} &= r^{(i)} - \alpha_i A d^{(i)};\\
\beta_{i+1} &= \frac{r^{(i+1)^T} r^{(i+1)}}{r^{(i)^T} r^{(i)}};\\
d^{(i+1)} &= r^{(i+1)} + \beta_{(i+1)} d^{(i)};
\end{aligned}$$

- faster than steepest descent, but still depending on n!
- search spaces form a so-called *Krylov sequence*:

$$\begin{aligned}
\text{span}\{d^{(0)}, \ldots, d^{(i-1)}\} &= \text{span}\{d^{(0)}, Ad^{(0)}, \ldots, A^{i-1}d^{(0)}\}\\
&= \text{span}\{r^{(0)}, Ar^{(0)}, \ldots, A^{i-1}r^{(0)}\}
\end{aligned}$$

- other famous Krylov methods: **GMRES, Bi-CGSTAB**

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

3.3. Fast Iterative Solvers of Systems of Linear Equations

- crucial drawback of solvers discussed so far: they become slower if we discretize more accurate!
- now: look for possible remedies
  - **relaxation**: explicit application of the *multigrid* principle
  - **Krylov/cg**: *preconditioning* (typically also following multigrid)
- let us start with preconditioning:
  - crucial quantity for cg's convergence: condition number
  - PDE: condition of system matrix increases dramatically with n
  - therefore: look for a modified matrix with better condition

$$Ax = b \Leftrightarrow M^{-1}Ax = M^{-1}b \Leftrightarrow W^{-1}AW^{-T}y = W^{-1}b,$$

where

M s.p.d., $WW^T = M,\ y = W^T x,\ M^{-1}A$ and $W^{-1}AW^{-T}$ similar

(no need to construct M or W explicitly, must be applied only)

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Strategies for Preconditioners

- the simplest choice: $M = I$ (cheap, but useless)
- the best choice: $M = A$ (perfect, but expensive)
- some possibilities in-between:
  - **diagonal** or **Jacobi** preconditioner: $M = D_A$
  - GS or SOR are not used due to lack of symmetry
  - **SSOR** preconditioner:

$$M^{(1/2)} = \alpha^{-1} D_A + L_A;\ M^{(1)} = \alpha^{-1} D_A + U_A;$$

$$M = \frac{\alpha}{\alpha - 2} \left(M^{(1/2)} \right)^{-1} D_A^{-1} M^{(1)}$$

  - **incomplete factorization**, e.g. **ILU**: compute approximate factors L and U instead of exact ones in direct methods
  - **sparse approximate inverse**: look for some cheap B with

$$\min_B \|I - AB\|^2,\ M^{-1} = B$$

  - **multilevel** preconditioners: following the multigrid principle

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

The Multigrid Principle

- starting point: *Fourier mode analysis* of the errors
  - decompose the error $e^{(i)} = x^{(i)} - x$ into its Fourier components (Fourier transform)
  - observe how they change/decrease under a standard relaxation like Jacobi or Gauß-Seidel (in a two-band sense):
    - The *high* frequency part (with respect to the underlying grid) is reduced quite quickly.
    - The *low* frequency part (w.r.t. the grid) decreases only very slowly; actually the slower, the finer the grid is.
  - This behaviour is annoying
    - the low frequencies are not expected to make troubles, but we can hardly get rid of them on a fine grid;

    but also encouraging
    - the low frequencies can be represented and, hopefully tackled, on a coarser grid – there is no need for the fine resolution.

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

A Simple Example

- 1D Laplace equation, $u(0) = u(1) = 0$ (exact solution 0)
- equidistant grid, 65 points, 3-point stencil, damped Jacobi method with damping parameter 0.5
- start with random values in $[0, 1]$ for u in the grid points
- After 100 (!) steps, there is still a maximum error bigger than 0.1 due to low-frequency components!
- therefore the name *smoothers* for relaxation schemes:
  - They reduce the strongly oscillating parts of the error quite efficiently.
  - They, thus, produce a **smooth** error which is very resistent.
- the idea: work on grids of different resolution

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Coarse Grid Correction 1

- sequence of equidistant grids on our domain:

$$\Omega_l,\ l = 1, 2, \ldots, L, \text{ with mesh width } h_l = 2^{-l}$$

- let $A_l, b_l, \ldots$ denote corresponding matrix, right-hand side,...
- combine work on two grids with a *correction scheme*:

smooth the current solution x_l ;
form the residual $r_l = b_l - A_l x_l$;
restrict r_l to the coarse grid Ω_{l-1} ;
provide a solution to $A_{l-1}\, e_{l-1} = r_{l-1}$;
prolongate e_{l-1} to the fine grid Ω_l;
add the resulting correction to x_l;
if necessary, smooth again ;

Standard Iterative ...
Towards Krylov: ...
Fast Iterative Solvers ...
Tools: Libraries and ...
Tools for Algorithm ...
Literature

Coarse Grid Correction 2

- the different steps of this *2-grid algorithm*:
  - the **pre-smoothing**: reduce high-frequency error components, smooth error, and prepare residual for transfer to coarse grid
  - the **restriction**: transfer from fine grid to coarse grid
    - *injection*: inherit the coarse grid values and forget the others
    - *(full) weighting*: apply some averaging process
  - the **coarse grid correction**: provide an (approximate) solution on the coarse grid (direct, if coarse enough; some smoothing steps otherwise)
  - the **prolongation**: transfer from coarse grid to fine grid
    - usually some *interpolation* method
  - the **post-smoothing**: sometimes reasonable to avoid new high-frequency error components
- recursive application leads to **multigrid methods**

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

The V-Cycle

- now, the coarse grid equation is solved by coarse grid correction, too; the resulting algorithmic scheme is called *V-cycle*:

smooth the current solution x_l ;
form the residual $r_l = b_l - A_l x_l$;
restrict r_l to the coarse grid Ω_{l-1} ;
solve $A_{l-1}\, e_{l-1} = r_{l-1}$ by coarse grid correction ;
prolongate e_{l-1} to the fine grid Ω_l;
add the resulting correction to x_l;
if necessary, smooth again ;

- on the finest grid: direct solution
- number of smoothing steps: typically small (1 or 2)

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Multigrid Algorithms

- the V-cycle is not the only multigrid scheme:
  - the **W-cycle**: after each prolongation, visit the coarse grid once more, before moving on to the next finer grid
  - the **nested iteration**: start on coarsest grid Ω_1, smooth, prolongate to Ω_2, smooth, prolongate to Ω_3, and so on, until finest grid is reached; now start V-cycle
  - **full multigrid**: replace 'smooth steps above by 'apply a V-cycle; combination of improved start solution and multigrid solver
- multigrid idea is not limited to rectangular or structured grids: we just need a hierarchy of nested grids (works for triangles or tetrahedra, too)
- also without underlying geometry: **algebraic** multigrid

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Basic Convergence Results

- Cost (storage and computing time):
  - 1D: $c \cdot n + c \cdot n/2 + c \cdot n/4 + c \cdot n/8 + \cdots \leq 2c \cdot n = O(n)$
  - 2D: $c \cdot n + c \cdot n/4 + c \cdot n/16 + c \cdot n/64 + \cdots \leq 4/3c \cdot n = O(n)$
  - 3D: $c \cdot n + c \cdot n/8 + c \cdot n/64 + c \cdot n/512 + \cdots \leq 8/7c \cdot n = O(n)$
  - i.e.: work on coarse grids is negligible compared to finest grid
- Benefit (speed of convergence):
  - always significant acceleration compared with pure use of smoother (relaxation method)
  - in most cases even ideal behaviour $\gamma = O(1 - \text{const})$
  - effect:
    - ⋆ constant number of multigrid steps to obtain a given number of digits
    - ⋆ overall computational work increases only linearly with n

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

3.4. Tools: Libraries and Software

- In addition to standard tools like editors, compilers, or debuggers, there is a lot of (commercial or public domain) support available:
  - **Modelling:** Computer algebra programs like Mathematica, Maple, Axiom, or Reduce support derivations and proofs of theorems via symbolic means.
  - **Numerics:** Mathematica, Maple, or MATLAB support the development, testing, and analysis of (numerical) algorithms and allow an efficient prototyping.
  - **Implementation:** A zoo of (numerical) libraries provide up-to-date modules for standard tasks (numerical linear algebra etc.), tailored to specific target architectures.
  - **Visualization:** Packages like IDL, IRIS Explorer, or AVS/Express offer (nearly) all you want.

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Libraries and Collections

- **GAMS:** Guide to Available Mathematical Software
  - service offered by the National Institute of Standards & Technology
  - see http://gams.nist.gov/
  - catalogue and database of more than 100 packages and libraries with together several tens of thousands of routines
  - Topics range from number theory to statistics!
  - majority: FORTRAN programs for numerical tasks (systems of linear equations, eigenvalues, roots, differential equations, . . .)
  - includes both public domain material (at NIST or at NETLIB, see below) and commercial (licenced) products.
  - good user guidance

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Libraries and Collections

- Matrix market
  - see http://math.nist.gov/MatrixMarket/
  - repository of test data for use in comparative studies of algorithms for numerical linear algebra
  - features nearly 500 (sparse) matrices from various fields of applications (chemical engineering, fluid flow, power system networks, quantum physics, or structural engineering, e.g.)
  - provides also matrix generation tools
  - classification according to matrix properties:
    - ⋆ number field: real or complex
    - ⋆ nonzero structure: dense, banded sparse, tridiagonal, …
    - ⋆ symmetry: none, symmetric, skewsymmetric, SPD, SSPD,…
    - ⋆ shape: square, more rows than columns,…

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Libraries and Collections

- **NETLIB:** repository of free software for numerical purposes
  - see http://www.netlib.org/
  - offered by University of Tennessee and Oak Ridge Nat'l Lab
  - several mirrored copies all over the world
  - about 135 million requests since 1985, > 40 million in 2000
  - > 90% http, rest ftp and email
  - about 160 different libraries, among which
    - ⋆ BLAS (Basic Linear Algebra Subprograms)
    - ⋆ LAPACK (Linear Algebra PACKage)
    - ⋆ ODEPACK (ordinary differential equations)
    - ⋆ MPI (message passing interface, for parallelization)
    - ⋆ PLTMG (elliptic boundary value problems)

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

BLAS

- collection of robust, efficient, and portable modules for elementary vector and matrix operations
- basis for LAPACK routines, for example
- allows plug-and-play for numerical subroutines
- FORTRAN, to be used from FORTRAN/C/C++
- Java BLAS available, too
- levels:
  - **Level 1**: vector and vector-vector operations (norm, scalar product, vector addition, SAXPY, . . .)
  - **Level 2** matrix-vector operations (rank-1-modifications, matrix-vector product, tridiagonal systems); vector processors
  - **Level 3** matrix-matrix operations; parallel computers!

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

LAPACK

- popular collection of FORTRAN subroutines for standard problems from numerical linear algebra like linear systems, regression, eigenvalues, SVD, ...
- dense and band matrices (not general sparse ones)
- successor of EISPACK and LINPACK, tuning for modern microprocessors and supercomputer architectures (reduction of memory accesses, block operations, ...)
- LAPACK routines use BLAS modules
- variants:
  - LAPACK90, CLAPACK, LAPACK++
  - ScaLAPACK (MIMD systems, scalability!)

Standard Iterative ...
Towards Krylov: ...
Fast Iterative Solvers ...
Tools: Libraries and ...
Tools for Algorithm ...
Literature

Libraries and Collections

- Visual Numerics:
  - see http://www.vni.com
  - mathematical libraries
  - predecessor: IMSL (The Int'l Math. & Statist. Library)
- Diffpack:
  - offered by Numerical Objects, see http://www.nobjects.com
  - environment for the development of code for numerical simulation problems plus libraries of efficient routines
  - object-oriented concept, available for most UNIX platforms
- NAG (Numerical Algorithms Group):
  - see http://www.nag.co.uk/
  - non-profit software house, spin-off of Oxford University
  - FORTRAN/FORTRAN90/C/Parallel libraries;
  - AXIOM; IRIS Explorer (visualization); Fastflo (CFD and more)

Standard Iterative ...
Towards Krylov: ...
Fast Iterative Solvers ...
Tools: Libraries and ...
Tools for Algorithm ...
Literature

Libraries and Collections

- Numerical Recipes:
  - see http://www.nr.com
  - book series *The Art of Scientific Computing* (CU Press)
  - sophisticated algorithms and their implementations
  - available for FORTRAN 77, FORTRAN 90, C, Pascal, . . .
  - corresponding software is licenced and commercial
  - about 350 routines for topics like
    - solution of linear systems
    - interpolation and extrapolation
    - numerical quadrature
    - differentiation and approximation
    - roots and extrema
    - eigenvalues, differential equations, and more

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

3.5. Tools for Algorithm Development

- Libraries offer tested and efficient (w.r.t. both storage and run-time) standard modules for competitive simulation codes (*do something classical cheap!*)
- Another problem is algorithm development (*develop something new and cheaper!*):
  - design of algorithms
  - testing and rapid prototyping
  - analysis (convergence behaviour etc.)
  - not yet: production runs, memory or runtime optimization
- widespread solutions:
  - computer algebra programs like Maple or Mathematica
  - MATLAB

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

Maple

- by Waterloo Maple Inc., a spin-off of the University of Waterloo in Ontario (see http://www.maplesoft.com)
- originally a mere computer algebra program, today "interactive environment for mathematical problem solving and programming"
- focus:
  - symbolic computations, formula manipulation
  - numerical computations with arbitrary accuracy
  - 2D and 3D graphical output
  - straightforward programming for algorithm development
- structure:

  kernel + main library + mixed library + packages

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

MATLAB

- by The MathWorks (see http://www.mathworks.com/)
- originally: primarily for use in (maths) education
- today: “high-performance numerical computation and visualization software”, standard tool for scientific computing research groups:
  - development, prototyping, programming
  - computations
  - visualization
- singular success story: >500 employees, >100 countries, >2000 universities and research institutes
- structure: basic program plus a collection of specialized tool boxes

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature

References

[1] Walter Gander and Jiři Hřebíček. Solving Problems in Scientific Computing Using Maple and MATLAB. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., second edition, 1995.

[2] Werner Krabs. Mathematische Modellierung. Eine Einführung in die Problematik. Teubner-Verlag, 1997.

[3] Gene H. Golub and James M. Ortega. Scientific Computing and Differential Equations. Academic Press, Boston, MA, USA, 1992.

[4] Jack J. Dongarra, Iain S. Duff, Danny C. Sorensen, and Henk A. van der Vorst. Numerical linear algebra for high-performance computers. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, 1998.

[5] Kai Hwang. Advanced Computer Architecture: Parallelism, Scalability, Programmability. McGraw-Hill, New York, 1993.

[6] Michael Griebel, Thomas Dornseifer, and Tilman Neunhoeffer. Numerical Simulation in Fluid Dynamics: A Practical Introduction. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, 1997.

Standard Iterative . . .
Towards Krylov: . . .
Fast Iterative Solvers . . .
Tools: Libraries and . . .
Tools for Algorithm . . .
Literature