Numerisk programvara

Olika typer av programvara:

- Paket som Matlab, Maple, Mathematica.
- Generella rutinbibliotek, NAG, IMSL. Får tillhandahålla huvudprogram etc. och länka till en biblioteksfil.
- Bibliotek för speciella problemområden.
  Lapack (RISC-optimerat) rutiner för $Ax = b$,
  $\min_x \ ||Ax - b||_2$ och $Ax = \lambda x$.
  (Linpack äldre bibliotek för CISC.)

  BLAS, t.ex. $\sigma = x^T y$, $y = Ax$ och $C = AB$.
  Finns i NAG, IMSL men även i:
- Datorspecifika bibliotek, t.ex.
  - AMD: ACML (AMD Core Math Library)
  - Intel: MKL (Intel Math Kernel library).
  - Sun: Sunperf (Sun Performance Library).
  - SGI: complib.sgimath
    (Scientific and Mathematical Library).
  - IBM: ESSL
    (Engineering and Scientific Subroutine Library).
- Rutiner för speciella tillämpningar. Finns samlingar på www. Netlib (där man även finner Lapack),

Rutinerna är normalt skrivna i Fortran (finns Fortran66, 77, 90, 95). En mindre del i C/C++. Än mindre andel i Java etc.

Enkelt att blanda Fortran med C/C++ (om enkel lagring).

NAG - The Numerical Algorithms Group Ltd

`http://www.nag.com/`

NAG offers libraries in Fortran 77, Fortran 90, C, as well as an SMP Library and a Parallel Library for shared memory and distributed memory parallel computing respectively. We have most recently added Data Mining Components to our product range.

Innehåller så där 1100 rutiner.

List of Contents, Release 4

Chapter 1: Utilities
Chapter 3: Special Functions
Chapter 4: Matrix and Vector Operations

Chapter 5: Linear Equations

Module 5.1: Solves a general real or complex system of linear equations with one or many right-hand sides

Module 5.2: Solves a real or complex, symmetric or Hermitian system of linear equations with one or many right-hand sides

Module 5.3: Solves a real or complex triangular system of linear equations

Module 5.4: Solves a general real or complex banded system of linear equations, with one or many right-hand sides

Module 5.5: Solves a real symmetric or complex Hermitian positive definite banded system of linear equations, with one or many right-hand sides

Module 5.6: Sparse matrix preconditioner Set-up and Solve

Module 5.7: Iterative Solvers
General sparse linear system solver

Chapter 6: Eigenvalue and Least-squares Problems
Chapter 7: Transforms
Chapter 8: Curve and Surface Fitting
Chapter 9: Optimization
Chapter 10: Nonlinear Equations
Chapter 11: Quadrature
Chapter 12: Ordinary Differential Equations (ODE's)
Chapter 13: Partial Differential Equations (PDE's)
Chapter 19: Operations Research
Chapter 20: Statistical Distribution Functions
Chapter 21: Random Number Generation
Chapter 22: Basic Descriptive Statistics
Chapter 25: Correlation and Regression Analysis
Chapter 28: Multivariate Analysis

Ett annat kommersiellt paket är IMSL (The International Mathematical and Statistical Library),
`http://www.vni.com/products/imsl/index.html`

Finns i Fortran, C men även i Java:
`http://www.vni.com/products/imsl/jmsl.html`

Netlib

`http://www.netlib.org`

Main Index of Software Libraries

Netlib Libraries:

a	fishpack	matlab	research
access	fitpack	mds	scalapack
aicm	floppy	microscope	sched
alliant	fmm	minpack	scilib
amos	fn	misc	seispack
ampl	fortran	mpfun	sequent
anl-reports	fortran-m	mpi	sfmm
apollo	fp	na-digest-html	shpcc94
benchmark	gcv	napack	slap
bib	gmat	netsolve	slatec
bibnet	gnu	news	sminpack
bihar	go	numeralgo	sodepack
blacs	graphics	ode	sparse
blas	harwell	odepack	sparse-blas
blast	hence	odrpack	sparspak
bmp	hompack	opt	specfun
c	hpf	p4	spin
c++	hypercube	paragraph	srwn
cephes	ieeecss	paranoia	stoeplitz
chammp	ijsa	parkbench	stringsearch
cheney-kincaid	image	parmacs	svdpack
clapack	intercom	pascal	templates
commercial	itpack	pbwg	tennessee
confdb	jakef	pdes	textbook
conformal	kincaid-cheney	performance	toeplitz
contin	la-net	photo	toms
control	lanczos	picl	tomspdf
crc	lanz	pltmg	transform
cumulvs	lapack	poly2	typesetting
ddsv	lapack90	polyhedra	uncon
dierckx	laso	popi	vanhuffel
diffpack	lawson-hanson	port	vfftpack
domino	linalg	posix	vfnlib
eispack	linpack	pppack	voronoi
elefunt	list	presto	xmagic
env	lp	problem-set	xnetlib
f2c	machines	pvm3	y12m
fdlibm	magic	quadpack	
fftpack	maspar	random	

ode innehåller 51 paket (jag har inte listat alla). Jag har också avlägsnat en del rader i de paket jag tagit med.

```
file    daspk.tgz
for     differential-algebraic system solver
by      Brown, Hindmarsh, Petzold
prec    double and single
alg     BDF methods with direct and preconditioned
        Krylov linear solvers
ref     SIAM J. Sci. Comp.: 15, 6, 1467 (1994) and 19,

lib     rksuite
alg     Runge-Kutta
for     initial value problem for first order ordinary
        differential equations. A suite of codes for
        solving IVPs in ODEs. A choice of RK methods
        is available. Includes an error assessment
        facility and a sophisticated stiffness checker.
        Template programs and example results provided.
by      R.W. Brankin (NAG), I. Gladwell and L.F. Shampi
lang    Fortran
prec    double

file    colnew.f
by      Ascher and Bader
for     ordinary differential equation boundary-value
        problem solver
alg     truncated powers collocation

file    ddverk.f
by      Enright and Hayashi <hiroshi@cs.toronto.edu>
for     delay differential equations
```



```
file    parsodes.tar.gz
by      Claus.Bendtsen@uni-c.dk
for     large systems of stiff ordinary differential
        equations
alg     multiimplicit Runge-Kutta with "across the
        method" parallelization
lang    Fortran90, MPI

file    odeToJava.tgz
for     initial-value problems for stiff and non-stiff
        ordinary differential equations
alg     explicit Runge-Kutta, linearly implicit
        implicit-explicit (IMEX)
by      Murray Patterson and Raymond J. Spiteri <spiter
lang    Java
```


odepack

ODEPACK is a collection of Fortran solvers for the initial value problem for ordinary differential equation systems. It consists of nine solvers, namely a basic solver called LSODE and eight variants of it – LSODES, LSODA, LSODAR, LSODPK, LSODKR, LSODI, LSOIBT, and LSODIS. The collection is suitable for both stiff and nonstiff systems. It includes solvers for systems given in explicit form, dy/dt = f(t,y), and also solvers for systems given in linearly implicit form, A(t,y) dy/dt = g(t,y). Two of the solvers use general sparse matrix solvers for the linear systems that arise. Two others use iterative (preconditioned Krylov) methods instead of direct methods for these linear systems. The most recent addition is LSODIS, which solves implicit problems with general sparse treatment of all matrices involved.

1. LSODE (Livermore Solver for Ordinary Differential Equations) is the basic solver of the collection. It solves stiff and nonstiff systems of the form dy/dt = f. In the stiff case, it treats the Jacobian matrix df/dy as either a dense (full) or a banded matrix, and as either user-supplied or internally approximated by difference quotients. It uses Adams methods (predictor-corrector) in the nonstiff case, and Backward Differentiation Formula (BDF) methods in the stiff case. The linear systems that arise are solved by direct methods (LU factor/solve).

2. LSODES, written jointly with A. H. Sherman, solves systems dy/dt = f and in the stiff case treats the Jacobian matrix in general sparse form. It determines the sparsity structure on its own, or optionally accepts this information from the user. It then uses parts of the Yale Sparse Matrix Package (YSMP) to solve the linear systems that arise, by a sparse (direct) LU factorization/backsolve method.

3. LSODA, written jointly with L. R. Petzold, solves systems dy/dt = f with a dense or banded Jacobian when the problem is stiff, but it automatically selects between nonstiff (Adams) and stiff (BDF) methods. It uses the nonstiff method initially, and dynamically monitors data in order to decide which method to use.

4. LSODAR, also written jointly with L. R. Petzold, is a variant of LSODA with a rootfinding capability added. Thus it solves problems dy/dt = f with dense or banded Jacobian and automatic method selection, and at the same time, it finds the roots of any of a set of given functions of the form g(t,y). This is often useful for finding stop conditions, or for finding points at which a switch is to be made in the function f.

5. LSODPK, written jointly with Peter N. Brown, is a variant of LSODE in which the direct solvers for the linear systems have been replaced by a selection of four preconditioned Krylov (iterative) solvers. The user must supply a pair of routine to evaluate, preprocess, and solve the (left and/or right) preconditioner matrices. LSODPK also includes an option for a user-supplied linear system solver to be used without Krylov iteration.

6. LSODKR is a variant of LSODPK with the addition of the same rootfinding capability as in LSODAR, and also of automatic switching between functional and Newton iteration. The nonlinear iteration method-switching differs from the method-switching in LSODA and LSODAR, but provides similar savings by using the cheaper method in the non-stiff regions of the problem. LSODKR also improves on the Krylov methods in LSODPK by offering the option to save and reuse the approximate Jacobian data underlying the preconditioner.

BLAS - Basic Linear Algebra Subprograms

- Blas 1: vektor-vektor-rutiner.

  $y := y + a\ x$ **daxpy**.

  Innerprodukt, skalning.

- Blas 2: matris-vektor-rutiner.

  $y := \alpha Ax + \beta y$ eller $y := \alpha A^T x + \beta y$ **dgemv**

  (eller $y := \alpha A^H x + \beta y$ om komplext. $A^H = \bar{A}^T$).

- Blas 3: matris-matris-rutiner.

  $C := \alpha\ op(A)\ op(B) + \beta C$ **dgemm**, där

  $op(M) = M$, $op(M) = M^T$ eller $op(M) = M^H$.

Fyra datatyper, **namn = gemm** t.ex.:

- **snamn** enkel precision
- **dnamn** dubbel precision
- **cnamn** enkel precision komplex
- **znamn** dubbel precision komplex

De datorspecifika biblioteken innehåller optimerade BLAS-rutiner. BLAS från Netlib är inte optimerat.

LAPACK - Linear Algebra PACKage

Ickeglesa problem, men finns stöd för bandmatriser.

- $Ax = b$
- $\min_x\ ||Ax - b||_2$
- $Ax = \lambda x$.

(plus en del närliggande problem vi ej har pratat om).

Två slag av rutiner:

- "driver routines", ett anrop för att lösa ett fullständigt problem. Det finns "simple drivers" med standardvärden på vissa parametrar respektive "expert drivers" med fler inställningsmöjligheter. Kan få tillbaks en uppskattning av konditionstalet till exempel.
- "computational routines", som utför delsteg i en beräkning, t.ex. LU-faktorisering eller lösning givet LU-faktoriseringen.
- "auxiliary routines", diverse hjälprutiner på låg nivå

Drivers för $Ax = b$:

- general
- general band
- general tridiagonal
- symmetric/Hermitian positive definite
- symmetric/Hermitian positive definite (packed storage)
- symmetric/Hermitian positive definite band
- symmetric/Hermitian positive definite tridiagonal
- symmetric/Hermitian indefinite
- complex symmetric
- symmetric/Hermitian indefinite (packed storage)
- complex symmetric (packed storage)

Här är en lista över "computational routines" för "symmetric/Hermitian positive definite":

- factorize, **POTRF**
- solve using factorization, **POTRS**
- estimate condition number, **POCON**
- error bounds for solution, **PORFS**
- invert using factorization, **POTRI**
- equilibrate, **POEQU**

Samma namngivningskonvention som för BLAS.

AMD Core Math Library (ACML)

Från hemsidan: **http://developer.amd.com/acml.jsp**

- A full implementation of Level 1, 2 and 3 Basic Linear Algebra Subroutines (BLAS), with key routines optimized for high performance on AMD Opteron processors.
- A full suite of Linear Algebra (LAPACK) routines. As well as taking advantage of the highly-tuned BLAS kernels, a key set of LAPACK routines has been further optimized to achieve considerably higher performance than standard LAPACK implementations.
- A comprehensive suite of Fast Fourier Transforms (FFTs) in both single-, double-, single-complex and double-complex data types.
- Fast scalar, vector, and array math transcendental library routines optimized for high performance on AMD Opteron processors.
- Random Number Generators in both single- and double-precision.

Tillgängligt för flera system och kompilatorer:
Linux & Windows: gnu, Intel, PGI.
Linux: gfortran, NAG, PathScale
Sun: Solaris

Intel Math Kernel Library (Intel MKL), Linux & Windows

Från:
`http://www.intel.com/cd/software/products/asmo-na/eng/307757.htm`

Optimerat för Intels processorer.
Känner av vilken CPU som används.
Trådat och färdigt för CPUer med flera kärnor.
Fortran- och C-interface.

Innehåller (bland annat):

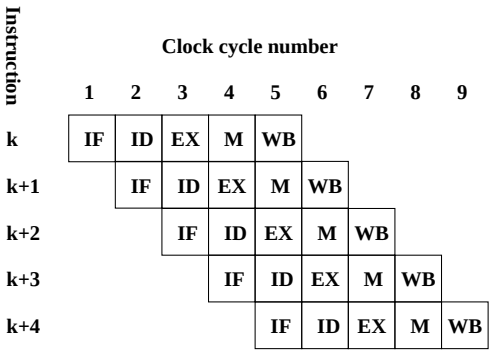
- BLAS and LAPACK. Optimerade för Intels CPUer.
- ScaLAPACK, parallell Lapack. Available in the Cluster Edition only.
- Sparse solvers. Solve large, sparse, symmetric, and asymmetric linear systems of equations on shared-memory multiprocessors with the PARDISO Direct Sparse Solver...
  Intel MKL also includes a Conjugate Gradient iterative solver...
- Fast Fourier Transforms (FFT).
- Vector Math Library. Increase application speeds with vectorized implementations of computationally intensive core mathematical functions (power, trigonometric, exponential, hyperbolic, logarithmic, and so on).
- Vector Random Number Generators.

Väldigt kort om hur man skriver snabba program

Dagens CPUer är väldigt snabba, de utnyttjar parallellitet på flera nivåer. Här några viktiga begrepp:

RISC (Reduced Instruction Set Computer) David Patterson, Berkeley (1980), John Hennessy, Stanford (1981). Förenkling av assemblerspråket vilket bla möjliggjorde höjd klockfrekvens.

CPUerna är superskalära, dvs. de kan exkvera flera instruktioner parallellt, t.ex. två heltalsinstruktioner och två flyttalsinstruktioner.
Pipelining utnyttjas för att arbeta på olika delar av instruktioner parallellt.

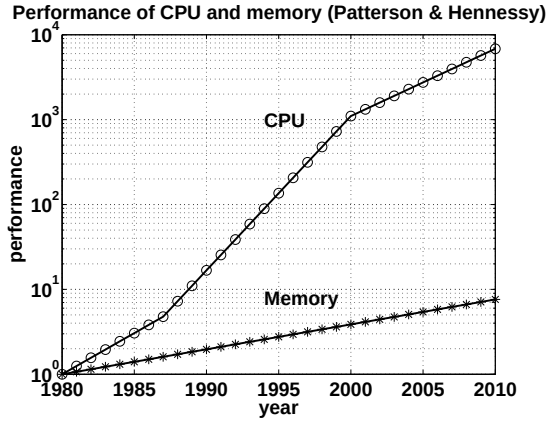


Pipelining används även för addition och multiplikation (men inte division) av flyttal.

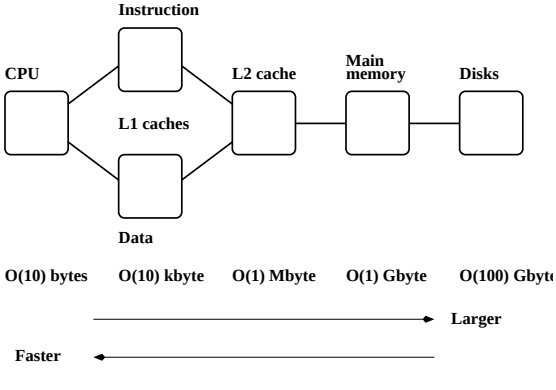
Det är också vanligt med stöd för vektoroperationer, t.ex SSE-SSE4 (Streaming SIMD Extensions), 3DNow!.

Flera kärnor (multi-core).

Primärminnet är mycket långsammare än CPUn:



För att minska gapet används cacheminnen:



För att programmen skall bli snabba måste man utnyttja cacheminnena på ett effektivt sätt. Här är en del av ett enkelt exempel i Fortran90. A är en 2000 × 2000-matris.

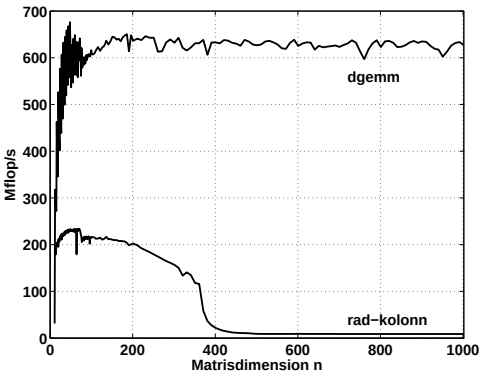
```
...
  do repeat = 1, 20  ! for more accurate times
    s = 0.0
    do row = 1, n
      do col = 1, n
        s = s + A(row, col)
      end do
    end do
  end do

  do repeat = 1, 20
    s = 0.0
    do col = 1, n    ! loop-interchange
      do row = 1, n
        s = s + A(row, col)
      end do
    end do
  end do

  do repeat = 1, 20
    s = sum(A)       ! builtin
  end do
...
% a.out
time = 1.32
time = 0.32
time = 0.32
```

Detta visar på vikten av minneslokalitet.

Ibland kan detta tas ett steg vidare. När man multiplicerar två matriser kan man återanvända element som redan hämtats till L1-cachen.

$$AB = \begin{bmatrix} \boxed{a_{1,1}} & a_{1,2} & a_{1,3} & a_{1,4} \\ \boxed{a_{2,1}} & a_{2,2} & a_{2,3} & a_{2,4} \\ \boxed{a_{3,1}} & a_{3,2} & a_{3,3} & a_{3,4} \\ \boxed{a_{4,1}} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} \begin{bmatrix} \boxed{b_{1,1}} & \boxed{b_{1,2}} & \boxed{b_{1,3}} & \boxed{b_{1,4}} \\ b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} \\ b_{3,1} & b_{3,2} & b_{3,3} & b_{3,4} \\ b_{4,1} & b_{4,2} & b_{4,3} & b_{4,4} \end{bmatrix}$$

Så, $a_{1,1}$ används för att multiplicera hela första raden i B och $b_{1,1}$ multiplicerar första kolonnen i A etc. Man måste ta hänsyn till detta för att få snabba rutiner annars kan man råka ut för följande (kappkörning mellan en trimmad rutin, **dgemm**, och den vanliga "rad gånger kolonn"-varianten).



BLAS 1: $\mathcal{O}(n)$ data, $\mathcal{O}(n)$ operationer
BLAS 2: $\mathcal{O}(n^2)$ data, $\mathcal{O}(n^2)$ operationer
BLAS 3: $\mathcal{O}(n^2)$ data, $\mathcal{O}(n^3)$ operationer kan återanvända data

En av de tekniker som används är sk "blocking", att dela in en matris i delar och arbeta med dessa delar.

x = A \ b i Matlab

När man skapar algoritmer försöker man arbeta med matris-matris-operationer snarare än med vektor-operationer just för att det finns större chans att återanvända element.

Låt oss titta lite kort på hur detta används i Matlab när vi skriver **x = A \ b**.

Antag att A är en kvadratisk ickesingulär matris. Om man läser Matlabs dokumentation framgår att systemet löses med LAPACK-rutinen **dgesv** (dessutom används två rutiner för att uppskatta $\kappa_1(A)$ givet LU-faktoriseringen från **dgesv**).

```
% man dgesv
NAME
 DGESV - compute the solution to a real system of
         linear equations A * X = B,

SYNOPSIS
 SUBROUTINE DGESV(N, NRHS, A, LDA, IPIV, B, LDB, INFO)
   INTEGER          INFO, LDA, LDB, N, NRHS
   INTEGER          IPIV( * )
   DOUBLE PRECISION A( LDA, * ), B( LDB, * )

PURPOSE
 DGESV computes the solution to a real system of linear
 equations A * X = B, where A is an N-by-N matrix and
 X and B are N-by-NRHS matrices.

 The LU decomposition with partial pivoting and row
 interchanges is used to factor A as
    A = P * L * U,
 where P is a permutation matrix, L is unit lower
 triangular, and U is upper triangular. The factored
 form of A is then used to solve the system of
 equations A * X = B.
etc.
```


LAPACK är byggt ovanpå BLAS och man försöker, om möjligt, använda BLAS3-rutiner. Det är viktigt att dessa är optimerade (Netlib har långsamma referens-versioner skrivna i Fortran som man inte skall använda). **dgesv** anropar bland annat **dgemm**. Här är en profilering med långsamma BLAS:

```
 %     self              self     total
time   seconds    calls   s/call   s/call  name
87.01    6.63       15     0.44     0.44  dgemm
 6.69    0.51       17     0.03     0.03  dtrsm
 3.81    0.29      984     0.00     0.00  dger
 1.18    0.09       32     0.00     0.00  dlaswp
etc.
```

dgemm utför olika matrismultiplikationer. **dtrsm** används för att lösa triangulära system. **dger** utför sk rang-ett uppdateringar (används för att beräkna LU-faktoriseringen). **dlaswp** utför radpermutationer. Totalt används 16 Fortranrutiner.

```
% grep -v ^\* *.f | wc -l
1636
```

Exempel på riktigt snabba BLAS är Goto BLAS av Kazushige Goto **http://www.tacc.utexas.edu/tacc-projects/gotoblas2/downloads**.
Ett annat bra alternativ är ATLAS (Automatically Tuned Linear Algebra Software). **http://math-atlas.sourceforge.net/**
BLAS i MKL och ACML är också snabba.