Datenverarbeitung für Technische Physiker II

Lineare Algebra

- Matrix-, Vektormanipulationen
- Lineare Gleichungssysteme
- Eigenwertanalyse, Eigenvektoren

Matrix-, Vektormanipulationen

z.B.: $\mathbf{A}\vec{x} = \vec{y}$ $\quad y_i = \sum_j a_{ij} x_j$

Indizes: C/C++: Anordnung zeilenweise
Fortran: Anordnung spaltenweise

```
DO I=1,N_row
   Y(I) = 0.D0
   DO J=1,N_col
      Y(I) = Y(I) + X(J)*A(I,J)
   END DO
END DO
```

Matrix-, Vektormanipulationen

z.B.: $\mathbf{A}\vec{x} = \vec{y}$ $\quad y_i = \sum_j a_{ij} x_j$

besser:
```
Y = 0.D0
DO J=1,N_col
   IF (X(J) .NE. 0.D0) THEN
      DO I=1,N_row
         Y(I) = Y(I) + X(J)*A(I,J)
      END DO
   END IF
END DO
```

Meaning of prefixes

S - REAL	C - COMPLEX
D - DOUBLE PRECISION	Z - COMPLEX*16 (this may not be supported by all machines)

For the Level 2 BLAS a set of extended-precision routines with the prefixes ES, ED, EC, EZ may also be available.

Level 1 BLAS
In addition to the listed routines there are two further extended-precision dot product routines DQDOTI and DQDOTA.

Level 2 and Level 3 BLAS
Matrix types:

GE - GEneral	GB - General Band	
SY - SYmmetric	SB - Sym. Band	SP - Sum. Packed
HE - HErmitian	HB - Herm. Band	HP - Herm. Packed
TR - TRiangular	TB - Triang. Band	TP - Triang. Packed

Level 2 and Level 3 BLAS Options
Dummy options arguments are declared as CHARACTER*1 and may be passed as character strings.

TRANx	= '**N**o transpose', '**T**ranspose', '**C**onjugate transpose' (X, X^T, X^H)
UPLO	= '**U**pper triangular', '**L**ower triangular'
DIAG	= '**N**on-unit triangular', '**U**nit triangular'
SIDE	= '**L**eft', '**R**ight' (A or op(A) on the left, or A or op(A) on the right)

For real matrices, TRANSx = 'T' and TRANSx = 'C' have the same meaning.
For Hermitian matrices, TRANSx = 'T' is not allowed.
For complex symmetric matrices, TRANSx = 'H' is not allowed.

References

C. Lawson, R. Hanson, D. Kincaid, and F. Krogh, "Basic Linear Algebra Subprograms for Fortran Usage," *ACM Trans. on Math. Soft.* 5 (1979) 308-325

J.J. Dongarra, J. DuCroz, S. Hammarling, and R. Hanson, "An Extended Set of Fortran Basic Linear Algebra Subprograms," *ACM Trans. on Math. Soft.* 14,1 (1988) 1-32

J.J. Dongarra, I. Duff, J. DuCroz, and S. Hammarling, "A Set of Level 3 Basic Linear Algebra Subprograms," *ACM Trans. on Math. Soft.* (1989)

Obtaining the Software via `netlib@ornl.gov`

To receive a copy of the single-precision software, type in a mail message:
```
send sblas from blas
send sblas2 from blas
send sblas3 from blas
```

To receive a copy of the double-precision software, type in a mail message:
```
send dblas from blas
send dblas2 from blas
send dblas3 from blas
```

To receive a copy of the complex single-precision software, type in a mail message:
```
send cblas from blas
send cblas2 from blas
send cblas3 from blas
```

To receive a copy of the complex double-precision software, type in a mail message:
```
send zblas from blas
send zblas2 from blas
send zblas3 from blas
```

Send comments and questions to `lapack@cs.utk.edu` .

Basic Linear Algebra Subprograms

A Quick Reference Guide

University of Tennessee
Oak Ridge National Laboratory
Numerical Algorithms Group Ltd.

May 11, 1997

Level 1 BLAS

		prefixes
SUBROUTINE xROTG (A, B, C, S)	Generate plane rotation	S, D
SUBROUTINE xROTMG(D1, D2, A, B, PARAM)	Generate modified plane rotation	S, D
SUBROUTINE xROT (N, X, INCX, Y, INCY, C, S)	Apply plane rotation	S, D
SUBROUTINE xROTM (N, X, INCX, Y, INCY, PARAM)	Apply modified plane rotation	S, D
SUBROUTINE xSWAP (N, X, INCX, Y, INCY)	$x \leftrightarrow y$	S, D, C, Z
SUBROUTINE xSCAL (N, ALPHA, X, INCX)	$x \leftarrow \alpha x$	S, D, C, Z, CS, ZD
SUBROUTINE xCOPY (N, X, INCX, Y, INCY)	$y \leftarrow x$	S, D, C, Z
SUBROUTINE xAXPY (N, ALPHA, X, INCX, Y, INCY)	$y \leftarrow \alpha x + y$	S, D, C, Z
FUNCTION xDOT (N, X, INCX, Y, INCY)	$dot \leftarrow x^T y$	S, D, DS
FUNCTION xDOTU (N, X, INCX, Y, INCY)	$dot \leftarrow x^T y$	C, Z
FUNCTION xDOTC (N, X, INCX, Y, INCY)	$dot \leftarrow x^H y$	C, Z
FUNCTION xxDOT (N, X, INCX, Y, INCY)	$dot \leftarrow \alpha + x^T y$	SDS
FUNCTION xNRM2 (N, X, INCX)	$nrm2 \leftarrow \|\|x\|\|_2$	S, D, SC, DZ
FUNCTION xASUM (N, X, INCX)	$asum \leftarrow \|\|re(x)\|\|_1 + \|\|im(x)\|\|_1$	S, D, SC, DZ
FUNCTION IxAMAX(N, X, INCX)	$amax \leftarrow 1^{st} k \ni \|re(x_k)\| + \|im(x_k)\|$ $= max(\|re(x_i)\| + \|im(x_i)\|)$	S, D, C, Z

Level 2 BLAS

options	dim	b-width	scalar	matrix	vector	scalar	vector		
xGEMV (TRANS,	M, N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$	S, D, C, Z
xGBMV (TRANS,	M, N,	KL, KU,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$	S, D, C, Z
xHEMV (UPLO,	N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xHBMV (UPLO,	N,	K,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xHPMV (UPLO,	N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xSYMV (UPLO,	N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xSBMV (UPLO,	N,	K,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xSPMV (UPLO,	N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xTRMV (UPLO, TRANS, DIAG,	N,			A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTBMV (UPLO, TRANS, DIAG,	N,	K,		A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTPMV (UPLO, TRANS, DIAG,	N,			AP,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTRSV (UPLO, TRANS, DIAG,	N,			A, LDA,	X, INCX)			$x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$	S, D, C, Z
xTBSV (UPLO, TRANS, DIAG,	N,	K,		A, LDA,	X, INCX)			$x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$	S, D, C, Z
xTPSV (UPLO, TRANS, DIAG,	N,			AP,	X, INCX)			$x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$	S, D, C, Z

options	dim	scalar	vector	matrix		
xGER (	M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^T + A, A - m \times n$	S, D
xGERU (	M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^T + A, A - m \times n$	C, Z
xGERC (	M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^H + A, A - m \times n$	C, Z
xHER (UPLO,	N,	ALPHA,	X, INCX,		A, LDA)	$A \leftarrow \alpha xx^H + A$	C, Z
xHPR (UPLO,	N,	ALPHA,	X, INCX,		AP)	$A \leftarrow \alpha xx^H + A$	C, Z
xHER2 (UPLO,	N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^H + y(\alpha x)^H + A$	C, Z
xHPR2 (UPLO,	N,	ALPHA,	X, INCX,	Y, INCY,	AP)	$A \leftarrow \alpha xy^H + y(\alpha x)^H + A$	C, Z
xSYR (UPLO,	N,	ALPHA,	X, INCX,		A, LDA)	$A \leftarrow \alpha xx^T + A$	S, D
xSPR (UPLO,	N,	ALPHA,	X, INCX,		AP)	$A \leftarrow \alpha xx^T + A$	S, D
xSYR2 (UPLO,	N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^T + \alpha yx^T + A$	S, D
xSPR2 (UPLO,	N,	ALPHA,	X, INCX,	Y, INCY,	AP)	$A \leftarrow \alpha xy^T + \alpha yx^T + A$	S, D

Level 3 BLAS

options	dim	scalar	matrix	matrix	scalar	matrix		
xGEMM (TRANSA, TRANSB,	M, N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha op(A)op(B) + \beta C, op(X) = X, X^T, X^H, C - m \times n$	S, D, C, Z
xSYMM (SIDE, UPLO,	M, N,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^T$	S, D, C, Z
xHEMM (SIDE, UPLO,	M, N,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^H$	C, Z
xSYRK (UPLO, TRANS,	N, K,	ALPHA,	A, LDA,		BETA,	C, LDC)	$C \leftarrow \alpha AA^T + \beta C, C \leftarrow \alpha A^T A + \beta C, C - n \times n$	S, D, C, Z
xHERK (UPLO, TRANS,	N, K,	ALPHA,	A, LDA,		BETA,	C, LDC)	$C \leftarrow \alpha AA^H + \beta C, C \leftarrow \alpha A^H A + \beta C, C - n \times n$	C, Z
xSYR2K(UPLO, TRANS,	N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB^T + \bar{\alpha} BA^T + \beta C, C \leftarrow \alpha A^T B + \bar{\alpha} B^T A + \beta C, C - n \times n$	S, D, C, Z
xHER2K(UPLO, TRANS,	N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB^H + \bar{\alpha} BA^H + \beta C, C \leftarrow \alpha A^H B + \bar{\alpha} B^H A + \beta C, C - n \times n$	C, Z
xTRMM (SIDE, UPLO, TRANSA, DIAG,	M, N,	ALPHA,	A, LDA,	B, LDB)			$B \leftarrow \alpha op(A)B, B \leftarrow \alpha Bop(A), op(A) = A, A^T, A^H, B - m \times n$	S, D, C, Z
xTRSM (SIDE, UPLO, TRANSA, DIAG,	M, N,	ALPHA,	A, LDA,	B, LDB)			$B \leftarrow \alpha op(A^{-1})B, B \leftarrow \alpha Bop(A^{-1}), op(A) = A, A^T, A^H, B - m \times n$	S, D, C, Z

LAPACK Quick Reference Guide to the Driver Routines

Release 3.0

Simple Drivers

Simple Driver Routines for Linear Equations

Matrix Type	Routine						
General	SGESV(	N,	NRHS, A, LDA,	IPIV, B, LDB,			INFO)
	CGESV(	N,	NRHS, A, LDA,	IPIV, B, LDB,			INFO)
General Band	SGBSV(	N, KL, KU,	NRHS, AB, LDAB,	IPIV, B, LDB,			INFO)
	CGBSV(	N, KL, KU,	NRHS, AB, LDAB,	IPIV, B, LDB,			INFO)
General Tridiagonal	SGTSV(	N,	NRHS, DL, D, DU,	B, LDB,			INFO)
	CGTSV(	N,	NRHS, DL, D, DU,	B, LDB,			INFO)
Symmetric/Hermitian Positive Definite	SPOSV(	UPLO, N,	NRHS, A, LDA,	B, LDB,			INFO)
	CPOSV(	UPLO, N,	NRHS, A, LDA,	B, LDB,			INFO)
Symmetric/Hermitian Positive Definite (Packed Storage)	SPPSV(	UPLO, N,	NRHS, AP,	B, LDB,			INFO)
	CPPSV(	UPLO, N,	NRHS, AP,	B, LDB,			INFO)
Symmetric/Hermitian Positive Definite Band	SPBSV(	UPLO, N, KD,	NRHS, AB, LDAB,	B, LDB,			INFO)
	CPBSV(	UPLO, N, KD,	NRHS, AB, LDAB,	B, LDB,			INFO)
Symmetric/Hermitian Positive Definite Tridiagonal	SPTSV(	N,	NRHS, D, E,	B, LDB,			INFO)
	CPTSV(	N,	NRHS, D, E,	B, LDB,			INFO)
Symmetric/Hermitian Indefinite	SSYSV(	UPLO, N,	NRHS, A, LDA,	IPIV, B, LDB,	WORK,	LWORK,	INFO)
	CSYSV(	UPLO, N,	NRHS, A, LDA,	IPIV, B, LDB,	WORK,	LWORK,	INFO)
	CHESV(	UPLO, N,	NRHS, A, LDA,	IPIV, B, LDB,	WORK,	LWORK,	INFO)
Symmetric/Hermitian Indefinite (Packed Storage)	SSPSV(	UPLO, N,	NRHS, AP,	IPIV, B, LDB,			INFO)
	CSPSV(	UPLO, N,	NRHS, AP,	IPIV, B, LDB,			INFO)
	CHPSV(	UPLO, N,	NRHS, AP,	IPIV, B, LDB,			INFO)

Simple Driver Routines for Standard and Generalized Linear Least Squares Problems

Problem Type	Routine						
Solve Using Orthogonal Factor, Assuming Full Rank	SGELS(	TRANS, M, N, NRHS, A, LDA, B, LDB,			WORK,	LWORK,	INFO)
	CGELS(	TRANS, M, N, NRHS, A, LDA, B, LDB,			WORK,	LWORK,	INFO)
Solve LSE Problem Using GRQ	SGGLSE(	M, N, P,	A, LDA, B, LDB, C, D, X,		WORK,	LWORK,	INFO)
	CGGLSE(	M, N, P,	A, LDA, B, LDB, C, D, X,		WORK,	LWORK,	INFO)
Solve GLM Problem Using GQR	SGGGLM(	N, M, P,	A, LDA, B, LDB, D, X, Y,		WORK,	LWORK,	INFO)
	CGGGLM(	N, M, P,	A, LDA, B, LDB, D, X, Y,		WORK,	LWORK,	INFO)

Simple and Divide and Conquer Driver Routines for Standard Eigenvalue and Singular Value Problems

Matrix/Problem Type	Routine														
Symmetric/Hermitian	SSYEV(	JOBZ,	UPLO,	N,	A,	LDA,	W,				WORK,	LWORK,			INFO)
Eigenvalues/vectors	CHEEV(	JOBZ,	UPLO,	N,	A,	LDA,	W,				WORK,	LWORK,	RWORK,		INFO)
Divide and Conquer	SSYEVD(	JOBZ,	UPLO,	N,	A,	LDA,	W,				WORK,	LWORK,		IWORK, LIWORK,	INFO)
	CHEEVD(	JOBZ,	UPLO,	N,	A,	LDA,	W,				WORK,	LWORK,	RWORK, LRWORK,	IWORK, LIWORK,	INFO)
Symmetric/Hermitian	SSPEV(	JOBZ,	UPLO,	N,	AP,		W,	Z,	LDZ,		WORK,				INFO)
(Packed Storage)	CHPEV(	JOBZ,	UPLO,	N,	AP,		W,	Z,	LDZ,		WORK,	RWORK,			INFO)
Eigenvalues/vectors															
Divide and Conquer	SSPEVD(	JOBZ,	UPLO,	N,	AP,		W,	Z,	LDZ,		WORK,	LWORK,		IWORK, LIWORK,	INFO)
	CHPEVD(	JOBZ,	UPLO,	N,	AP,		W,	Z,	LDZ,		WORK,	LWORK,	RWORK, LRWORK,	IWORK, LIWORK,	INFO)
Symmetric/Hermitian Band	SSBEV(	JOBZ,	UPLO,	N,	KD,	AB,	LDAB,	W,	Z,	LDZ,	WORK,				INFO)
Eigenvalues/vectors	CHBEV(	JOBZ,	UPLO,	N,	KD,	AB,	LDAB,	W,	Z,	LDZ,	WORK,	RWORK,			INFO)
Divide and Conquer	SSBEVD(	JOBZ,	UPLO,	N,	KD,	AB,	LDAB,	W,	Z,	LDZ,	WORK,	LWORK,		IWORK, LIWORK,	INFO)
	CHBEVD(	JOBZ,	UPLO,	N,	KD,	AB,	LDAB,	W,	Z,	LDZ,	WORK,	LWORK,	RWORK, LRWORK,	IWORK, LIWORK,	INFO)
Symmetric Tridiagonal	SSTEV(	JOBZ,		N,	D,	E,		Z,	LDZ,		WORK,				INFO)
Eigenvalues/vectors															
Divide and Conquer	SSTEVD(	JOBZ,		N,	D,	E,		Z,	LDZ,		WORK,	LWORK,		IWORK, LIWORK,	INFO)
General	SGEES(	JOBVS,	SORT,	SELECT,	N,	A,	LDA,	SDIM,	WR,	WI,	VS,	LDVS,	WORK,	LWORK,	BWORK, INFO)
Schur Factorization	CGEES(	JOBVS,	SORT,	SELECT,	N,	A,	LDA,	SDIM,	W,		VS,	LDVS,	WORK,	LWORK, RWORK,	BWORK, INFO)
General	SGEEV(	JOBVL,	JOBVR,		N,	A,	LDA,		WR,	WI,	VL,	LDVL,	VR,	LDVR, WORK, LWORK,	INFO)
Eigenvalues/vectors	CGEEV(	JOBVL,	JOBVR,		N,	A,	LDA,		W,		VL,	LDVL,	VR,	LDVR, WORK, LWORK, RWORK,	INFO)
General	SGESVD(	JOBU,	JOBVT,		N,	N,	A,	LDA,	S,	U,	LDU,	VT,	LDVT,	WORK, LWORK,	INFO)
Singular Values/Vectors	CGESVD(	JOBU,	JOBVT,		N,	N,	A,	LDA,	S,	U,	LDU,	VT,	LDVT,	WORK, LWORK, RWORK,	INFO)
Divide and Conquer	SGESDD(	JOBZ,			N,	N,	A,	LDA,	S,	U,	LDU,	VT,	LDVT,	WORK, LWORK,	IWORK, INFO)
	CGESDD(	JOBZ,			N,	N,	A,	LDA,	S,	U,	LDU,	VT,	LDVT,	WORK, LWORK, RWORK,	IWORK, INFO)

Simple and Divide and Conquer Driver Routines for Generalized Eigenvalue and Singular Value Problems

Matrix/Problem Type	Routine																
Symmetric-definite	SSYGV(	ITYPE,	JOBZ,	UPLO,	N,	A,	LDA,	B,	LDB,	W,				WORK,	LWORK,		INFO)
Eigenvalues/vectors	CHEGV(	ITYPE,	JOBZ,	UPLO,	N,	A,	LDA,	B,	LDB,	W,				WORK,	LWORK,	RWORK,	INFO)
Divide and Conquer	SSYGVD(	ITYPE,	JOBZ,	UPLO,	N,	A,	LDA,	B,	LDB,	W,				WORK,	LWORK,	IWORK, LIWORK,	INFO)
	CHEGVD(	ITYPE,	JOBZ,	UPLO,	N,	A,	LDA,	B,	LDB,	W,				WORK,	LWORK,	RWORK, LRWORK, IWORK, LIWORK,	INFO)
Symmetric-definite	SSPGV(	ITYPE,	JOBZ,	UPLO,	N,	AP,	BP,			W,		Z,	LDZ,	WORK,			INFO)
(Packed Storage)	CHPGV(	ITYPE,	JOBZ,	UPLO,	N,	AP,	BP,			W,		Z,	LDZ,	WORK,	RWORK,		INFO)
Eigenvalues/vectors																	
Divide and Conquer	SSPGVD(	ITYPE,	JOBZ,	UPLO,	N,	AP,	BP,			W,		Z,	LDZ,	WORK,	LWORK,	IWORK, LIWORK,	INFO)
	CHPGVD(	ITYPE,	JOBZ,	UPLO,	N,	AP,	BP,			W,		Z,	LDZ,	WORK,	LWORK, RWORK, LRWORK, IWORK, LIWORK,		INFO)
Symmetric-definite	SSBGV(		JOBZ,	UPLO,	N,	KA,	KB,	AB,	LDAB,	BB,	LDBB,	W,	Z,	LDZ,	WORK,		INFO)
(Band Storage)	CHBGV(		JOBZ,	UPLO,	N,	KA,	KB,	AB,	LDAB,	BB,	LDBB,	W,	Z,	LDZ,	WORK,	RWORK,	INFO)
Eigenvalues/vectors																	
Divide and Conquer	SSBGVD(		JOBZ,	UPLO,	N,	KA,	KB,	AB,	LDAB,	BB,	LDBB,	W,	Z,	LDZ,	WORK,	LWORK,	IWORK, LIWORK, INFO)
	CHBGVD(		JOBZ,	UPLO,	N,	KA,	KB,	AB,	LDAB,	BB,	LDBB,	W,	Z,	LDZ,	WORK,	LWORK, RWORK, LRWORK, IWORK, LIWORK,		INFO)
General	SGGES(	JOBVSL,	JOBVSR,	SORT,	SELCTG,	N,	A,	LDA,	B,	LDB,	SDIM,	ALPHAR,	ALPHAI,	BETA,	VSL,	LDVSL, VSR, LDVSR, WORK, LWORK,	BWORK,	INFO)
Schur Factorization	CGGES(	JOBVSL,	JOBVSR,	SORT,	SELCTG,	N,	A,	LDA,	B,	LDB,	SDIM,	ALPHA,		BETA,	VSL,	LDVSL, VSR, LDVSR, WORK, LWORK, RWORK,	BWORK,	INFO)
General	SGGEV(	JOBVL,	JOBVR,			N,	A,	LDA,	B,	LDB,		ALPHAR,	ALPHAI,	BETA,	VL,	LDVL, VR, LDVR, WORK, LWORK,		INFO)
Eigenvalues/vectors	CGGEV(	JOBVL,	JOBVR,			N,	A,	LDA,	B,	LDB,		ALPHA,		BETA,	VL,	LDVL, VR, LDVR, WORK, LWORK, RWORK,		INFO)
General	SGGSVD(	JOBU,	JOBV,	JOBQ,	M,	N,	P,	K,	L,	A,	LDA,	B,	LDB,	ALPHA,	BETA,	U, LDU, V, LDV, Q, LDQ, WORK,	IWORK,	INFO)
Singular Values/Vectors	CGGSVD(	JOBU,	JOBV,	JOBQ,	M,	N,	P,	K,	L,	A,	LDA,	B,	LDB,	ALPHA,	BETA,	U, LDU, V, LDV, Q, LDQ, WORK, RWORK,	IWORK,	INFO)

Expert Drivers

Expert Driver Routines for Linear Equations

Matrix Type	Routine																		
General	SGESVX(	FACT,	TRANS,	N,	NRHS,	A,	LDA,	AF,	LDAF,	IPIV,	EQUED,	R,	C,	B,	LDB,	X,	LDX,	RCOND,	FERR, BERR, WORK, IWORK, INFO)
	CGESVX(	FACT,	TRANS,	N,	NRHS,	A,	LDA,	AF,	LDAF,	IPIV,	EQUED,	R,	C,	B,	LDB,	X,	LDX,	RCOND,	FERR, BERR, WORK, RWORK, INFO)
General Band	SGBSVX(	FACT,	TRANS,	N,	KL,	KU,	NRHS,	AB,	LDAB,	AFB,	LDAFB,	IPIV,	EQUED,	R,	C,	B,	LDB,	X,	LDX, RCOND, FERR, BERR, WORK, IWORK, INFO)
	CGBSVX(	FACT,	TRANS,	N,	KL,	KU,	NRHS,	AB,	LDAB,	AFB,	LDAFB,	IPIV,	EQUED,	R,	C,	B,	LDB,	X,	LDX, RCOND, FERR, BERR, WORK, RWORK, INFO)
General Tridiagonal	SGTSVX(	FACT,	TRANS,	N,	NRHS,	DL,	D,	DU,	DLF,	DF,	DUF,	DU2,	IPIV,	B,	LDB,	X,	LDX,	RCOND,	FERR, BERR, WORK, IWORK, INFO)
	CGTSVX(	FACT,	TRANS,	N,	NRHS,	DL,	D,	DU,	DLF,	DF,	DUF,	DU2,	IPIV,	B,	LDB,	X,	LDX,	RCOND,	FERR, BERR, WORK, RWORK, INFO)
Symmetric/Hermitian	SPOSVX(	FACT,	UPLO,	N,	NRHS,	A,	LDA,	AF,	LDAF,	EQUED,	S,	B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR,	WORK, IWORK, INFO)
Positive Definite	CPOSVX(	FACT,	UPLO,	N,	NRHS,	A,	LDA,	AF,	LDAF,	EQUED,	S,	B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR,	WORK, RWORK, INFO)
Symmetric/Hermitian	SPPSVX(	FACT,	UPLO,	N,	NRHS,	AP,		AFP,		EQUED,	S,	B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR,	WORK, IWORK, INFO)
Positive Definite (Packed Storage)	CPPSVX(	FACT,	UPLO,	N,	NRHS,	AP,		AFP,		EQUED,	S,	B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR,	WORK, RWORK, INFO)
Symmetric/Hermitian	SPBSVX(	FACT,	UPLO,	N,	KD,	NRHS,	AB,	LDAB,	AFB,	LDAFB,	EQUED,	S,	B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR, WORK, IWORK, INFO)
Positive Definite Band	CPBSVX(	FACT,	UPLO,	N,	KD,	NRHS,	AB,	LDAB,	AFB,	LDAFB,	EQUED,	S,	B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR, WORK, RWORK, INFO)
Symmetric/Hermitian	SPTSVX(	FACT,		N,	NRHS,	D,	E,	DF,	EF,			B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR,	WORK, INFO)
Positive Definite Tridiagonal	CPTSVX(	FACT,		N,	NRHS,	D,	E,	DF,	EF,			B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR,	WORK, RWORK, INFO)
Symmetric/Hermitian	SSYSVX(	FACT,	UPLO,	N,	NRHS,	A,	LDA,	AF,	LDAF,	IPIV,		B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR,	WORK, LWORK, IWORK, INFO)
Indefinite	CSYSVX(	FACT,	UPLO,	N,	NRHS,	A,	LDA,	AF,	LDAF,	IPIV,		B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR,	WORK, LWORK, RWORK, INFO)
	CHESVX(	FACT,	UPLO,	N,	NRHS,	A,	LDA,	AF,	LDAF,	IPIV,		B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR,	WORK, LWORK, RWORK, INFO)
Symmetric/Hermitian	SSPSVX(	FACT,	UPLO,	N,	NRHS,	AP,		AFP,	IPIV,			B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR,	WORK, IWORK, INFO)
Indefinite (Packed Storage)	CSPSVX(	FACT,	UPLO,	N,	NRHS,	AP,		AFP,	IPIV,			B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR,	WORK, RWORK, INFO)
	CHPSVX(	FACT,	UPLO,	N,	NRHS,	AP,		AFP,	IPIV,			B,	LDB,	X,	LDX,	RCOND,	FERR,	BERR,	WORK, RWORK, INFO)

Divide and Conquer and Expert Driver Routines for Linear Least Squares Problems

Problem Type	Routine													
Solve Using Orthogonal Factor	SGELSY(	M,	N,	NRHS,	A,	LDA,	B,	LDB,	JPVT,	RCOND,	RANK,	WORK,	LWORK,	INFO)
	CGELSY(	M,	N,	NRHS,	A,	LDA,	B,	LDB,	JPVT,	RCOND,	RANK,	WORK,	LWORK, RWORK,	INFO)
Solve Using SVD,	SGELSS(	M,	N,	NRHS,	A,	LDA,	B,	LDB,	S,	RCOND,	RANK,	WORK,	LWORK,	INFO)
Allowing for Rank-Deficiency	CGELSS(	M,	N,	NRHS,	A,	LDA,	B,	LDB,	S,	RCOND,	RANK,	WORK,	LWORK, RWORK,	INFO)
Solve Using D&C SVD,	SGELSD(	M,	N,	NRHS,	A,	LDA,	B,	LDB,	S,	RCOND,	RANK,	WORK,	LWORK,	IWORK, INFO)
Allowing for Rank-Deficiency	CGELSD(	M,	N,	NRHS,	A,	LDA,	B,	LDB,	S,	RCOND,	RANK,	WORK,	LWORK, RWORK,	IWORK, INFO)

Expert and RRR Driver Routines for Standard and Generalized Symmetric Eigenvalue Problems

Matrix/Problem Type	Routine	
Symmetric/Hermitian Eigenvalues/vectors	SSYEVX(JOBZ, RANGE, UPLO, N, A, LDA,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, WORK, LWORK, IWORK, IFAIL, INFO)
	CHEEVX(JOBZ, RANGE, UPLO, N, A, LDA,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, WORK, LWORK, RWORK, IWORK, IFAIL, INFO)
	SSYEYR(JOBZ, RANGE, UPLO, N, A, LDA,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, ISUPPZ, WORK, LWORK, IWORK, LIWORK, INFO)
	CHEEYR(JOBZ, RANGE, UPLO, N, A, LDA,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, ISUPPZ, WORK, LWORK, RWORK, LRWORK, IWORK, LIWORK, INFO)
	SSYGVX(ITYPE, JOBZ, RANGE, UPLO, N, A, LDA, B, LDB,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, WORK, LWORK, IWORK, IFAIL, INFO)
	CHEGVX(ITYPE, JOBZ, RANGE, UPLO, N, A, LDA, B, LDB,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, WORK, LWORK, RWORK, IWORK, IFAIL, INFO)
Symmetric/Hermitian (Packed Storage) Eigenvalues/vectors	SSPEVX(JOBZ, RANGE, UPLO, N, AP,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, WORK, IWORK, IFAIL, INFO)
	CHPEVX(JOBZ, RANGE, UPLO, N, AP,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, WORK, RWORK, IWORK, IFAIL, INFO)
	SSPGVX(ITYPE, JOBZ, RANGE, UPLO, N, AP, BP,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, WORK, IWORK, IFAIL, INFO)
	CHPGVX(ITYPE, JOBZ, RANGE, UPLO, N, AP, BP,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, WORK, RWORK, IWORK, IFAIL, INFO)
Symmetric/Hermitian Band Eigenvalues/vectors	SSBEVX(JOBZ, RANGE, UPLO, N, KD, AB, LDAB, Q, LDQ,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, WORK, IWORK, IFAIL, INFO)
	CHBEVX(JOBZ, RANGE, UPLO, N, KD, AB, LDAB, Q, LDQ,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, WORK, RWORK, IWORK, IFAIL, INFO)
	SSBGVX(JOBZ, RANGE, UPLO, N, KA, KB, AB, LDAB, BB, LDBB, Q, LDQ,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, WORK, IWORK, IFAIL, INFO)
	CHBGVX(JOBZ, RANGE, UPLO, N, KA, KB, AB, LDAB, BB, LDBB, Q, LDQ,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, WORK, RWORK, IWORK, IFAIL, INFO)
Symmetric Tridiagonal Eigenvalues/vectors	SSTEVX(JOBZ, RANGE, N, D, E,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, WORK, IWORK, IFAIL, INFO)
	SSTEYR(JOBZ, RANGE, N, D, E,	VL, VU, IL, IU, ABSTOL, M, W, Z, LDZ, ISUPPZ, WORK, LWORK, IWORK, LIWORK, INFO)

Expert Driver Routines for Standard and Generalized Nonsymmetric Eigenvalue Problems

Problem Type	Routine	
Schur Factorization	SGEESX(JOBVS, SORT, SELECT, SENSE, N, A, LDA, SDIM, WR, WI, VS, LDVS,	RCONDE, RCONDV, WORK, LWORK, IWORK, LIWORK, BWORK, INFO)
	CGEESX(JOBVS, SORT, SELECT, SENSE, N, A, LDA, SDIM, W, VS, LDVS,	RCONDE, RCONDV, WORK, LWORK, RWORK, BWORK, INFO)
	SGGESX(JOBVSL, JOBVSR, SORT, SELCTG, SENSE, N, A, LDA, B, LDB, SDIM, ALPHAR, ALPHAI, BETA, VSL, LDVSL, VSR, LDVSR, RCONDE, RCONDV,	WORK, LWORK, IWORK, LIWORK, BWORK, INFO)
	CGGESX(JOBVSL, JOBVSR, SORT, SELCTG, SENSE, N, A, LDA, B, LDB, SDIM, ALPHAR, ALPHAI, BETA, VSL, LDVSL, VSR, LDVSR, RCONDE, RCONDV,	WORK, LWORK, RWORK, IWORK, LIWORK, BWORK, INFO)
Eigenvalues/ vectors	SGEEVX(BALANC, JOBVL, JOBVR, SENSE, N, A, LDA, WR, WI, VL, LDVL, VR, LDVR, ILO, IHI, SCALE, ABNRM, RCONDE, RCONDV,	WORK, LWORK, IWORK, INFO)
	CGEEVX(BALANC, JOBVL, JOBVR, SENSE, N, A, LDA, W, VL, LDVL, VR, LDVR, ILO, IHI, SCALE, ABNRM, RCONDE, RCONDV,	WORK, LWORK, RWORK, INFO)
	SGGEVX(BALANC, JOBVL, JOBVR, SENSE, N, A, LDA, B, LDB, ALPHAR, ALPHAI, BETA, VL, LDVL, VR, LDVR, ILO, IHI, LSCALE, RSCALE, ABNRM, BBNRM, RCONDE, RCONDV, WORK, LWORK, IWORK, BWORK, INFO)	
	CGGEVX(BALANC, JOBVL, JOBVR, SENSE, N, A, LDA, B, LDB, ALPHAR, ALPHAI, BETA, VL, LDVL, VR, LDVR, ILO, IHI, LSCALE, RSCALE, ABNRM, BBNRM, RCONDE, RCONDV, WORK, LWORK, RWORK, IWORK, BWORK, INFO)	

Meaning of prefixes

Routines beginning with "S" are available in:

S - REAL
D - DOUBLE PRECISION

Routines beginning with "C" are available in:

C - COMPLEX
Z - COMPLEX*16

Note: COMPLEX*16 may not be supported by all machines

Lineare Gleichungssysteme

typische Anwendungen: cubic spline:

$$(a_{i+1}-a_i)s_{i-1}+2(a_{i+1}-a_{i-1})s_i+(a_i-a_{i-1})s_{i+1}=$$
$$=3f[a_{i-1},a_i](a_{i+1}-a_i)+3f[a_i,a_{i+1}](a_i-a_{i-1})=y_i$$

$$\begin{pmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & a_{ii-1} & a_{ii} & a_{ii+1} & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & a_{n-1n-2} & a_{n-1n-1} & a_{n-1n} \\ 0 & 0 & 0 & 0 & 0 & a_{nn-1} & a_{nn} \end{pmatrix} \begin{pmatrix} s_1 \\ \vdots \\ s_{i-1} \\ s_i \\ s_{i+1} \\ \vdots \\ s_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ \vdots \\ y_i \\ \vdots \\ \vdots \\ y_n \end{pmatrix}$$

Lineare Gleichungssysteme

typische Anwendungen: Schrödingergleichung: $(e = m_e = \hbar = 1)$

$$\left(-\frac{\Delta}{2}+V(x)\right)\psi(x) = i\partial_t\psi(x) \rightarrow -\frac{1}{2}\left(\frac{\psi_{i-1}-2\psi_i+\psi_{i+1}}{\Delta x^2}\right)+V(x_i)\psi_i = \frac{i}{\Delta t}\left(\psi_i(t+\Delta t)-\psi_i(t)\right)$$

$$\left\{-\frac{1}{2\Delta x^2}\begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix} + V(x_i)\mathbf{1}\right\}\begin{pmatrix} \psi_1 \\ \vdots \\ \psi_{i-1} \\ \psi_i \\ \psi_{i+1} \\ \vdots \\ \psi_n \end{pmatrix} = \frac{i\mathbf{1}}{\Delta t}\left(\vec{\psi}(t+\Delta t)-\vec{\psi}(t)\right)$$

Wir erinnern uns …

$$\mathbf{A}\vec{x} = \vec{b}\,; \quad \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

1. $m < n$ (oder $m = n$, $\det(\mathbf{A}) = 0$) → System unterbestimmt
2. $m = n$, $\det(\mathbf{A}) \neq 0$ → eindeutige Lösung existiert
3. $m > n$ → System überbestimmt

Lösungsverfahren

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

$\det(\mathbf{A}) \neq 0$

direkte Verfahren
„exaktes“ Ergebnis in
endlich vielen Schritten → Gausselimination, LU-Zerlegung

indirekte Verfahren
„exaktes“ Ergebnis in
unendlich vielen Schritten → Jacobi-Methode, OR-Methoden

indirekte Verfahren

Jacobi-Methode:

$$A\mathbf{x} = \mathbf{b} \qquad A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

$$A = D + R \quad \text{mit} \quad D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}, \quad R = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}, \quad \mathbf{x}^{(k+1)} = D^{-1}(\mathbf{b} - R\mathbf{x}^{(k)})$$

aus: wikipedia

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.$$

“lower” “upper”

Verbesserung: Gauss-Seidel-Verfahren: ***R*** = ***L*** + ***U***

indirekte Verfahren

successive-over-relaxation (SOR) Methode:

$$A = D + L + U, \quad D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}, \quad L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$(D + \omega L)\mathbf{x} = \omega \mathbf{b} - [\omega U + (\omega - 1)D]\mathbf{x}$$

$$\mathbf{x}^{(k+1)} = (D + \omega L)^{-1}(\omega \mathbf{b} - [\omega U + (\omega - 1)D]\mathbf{x}^{(k)}).$$

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j>i} a_{ij}x_j^{(k)} - \sum_{j<i} a_{ij}x_j^{(k+1)}\right), \quad i = 1, 2, \ldots, n.$$

A* = *D* + *R → Jacobi over-relaxation (JOR) Methode

direkte Verfahren

Gauss-Elimination:

Transformation der Matrix **A** auf Dreiecksform

$$\mathbf{A}\vec{x} = \vec{b} \quad \rightarrow \quad \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$$

$$x_n = \frac{c_n}{u_{nn}}; \quad x_i = \frac{1}{u_{ii}}\left(c_i - \sum_{k=i+1}^{n} u_{ik}x_k\right); \quad \det(\mathbf{A}) = \prod_{i=1}^{n} u_{ii}$$

rückwärts einsetzen (backward substitution)

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right)$$

Zeile 2 - $\frac{a_{21}}{a_{11}}$ Zeile 1

…

Zeile n - $\frac{a_{n1}}{a_{11}}$ Zeile 1

$$\frac{a_{ki}}{a_{ii}} = \lambda_{ki}$$

$$\rightarrow \left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & \bar{a}_{22} & \cdots & \bar{a}_{2n} & \bar{b}_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \bar{a}_{n2} & \cdots & \bar{a}_{nn} & \bar{b}_n \end{array}\right)$$

Zeile 3 - $\frac{\bar{a}_{32}}{\bar{a}_{22}}$ Zeile 2

…

Zeile n - $\frac{\bar{a}_{n2}}{\bar{a}_{22}}$ Zeile 2

$$\rightarrow \left(\begin{array}{ccccc|c} a_{11} & a_{12} & \cdots & \cdots & a_{1n} & b_1 \\ 0 & \bar{a}_{22} & \bar{a}_{23} & \cdots & \bar{a}_{2n} & \bar{b}_2 \\ 0 & 0 & \bar{\bar{a}}_{33} & \cdots & \vdots & \bar{\bar{b}}_3 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \bar{\bar{a}}_{n3} & \cdots & \bar{\bar{a}}_{nn} & \bar{\bar{b}}_n \end{array}\right) \rightarrow \ldots \rightarrow \left(\begin{array}{ccccc|c} u_{11} & u_{12} & \cdots & \cdots & u_{1n} & c_1 \\ 0 & u_{22} & & \cdots & u_{2n} & c_2 \\ 0 & 0 & \ddots & & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & u_{nn} & c_n \end{array}\right)$$

Problem:

Pivot

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & \bar{a}_{22} & \cdots & \bar{a}_{2n} & \bar{b}_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \bar{a}_{n2} & \cdots & \bar{a}_{nn} & \bar{b}_n \end{array}\right)$$

Zeile 3 - $\frac{\bar{a}_{32}}{\bar{a}_{22}}$ Zeile 2

…

Zeile n - $\frac{\bar{a}_{n2}}{\bar{a}_{22}}$ Zeile 2

Pivot darf nicht 0 sein!

→ Suche nach geeignetem Pivot-Element

→ evtl. Zeilenvertauschung („Spalten-Pivotsuche“)
üblicherweise größtes Element in der ersten Spalte der Submatrix

Einschub: Gleitkommazahlen

Gleitkomma-Darstellung von Zahlen: $x = (-1)^s * m * b^e$

Exponent

Mantisse

Basis

Darstellung: $s, e_0 e_1 e_2 \ldots, m_0.m_1 m_2 m_3 m_4 m_5 m_6 \ldots$

Normierung: $m_0 \neq 0$

Addition: 1.) Angleichen der Exponenten
2.) Addition der Mantissen, danach Normierung

→ Rundungs- oder Abschneidefehler (z.B. 3-stellige Mantisse):

- unterschiedliche Größenordnung:
  1.00e2 + 1.00e-2=1.00e2 + 0.0001e2=1.00e2
- fast gleich große Zahlen:
  π – 3.14 = 3.14159…e0 – 3.14e0 = 0.00e0

Computer: Basis $b = 2$ → $m_0 = 1$ muss nicht dargestellt werden
+ Einsparung des Vorzeichens des Exponenten (Bias)

Norm: IEEE 754-2008

$$e = \lfloor \log_2(|x|) \rfloor$$
$$m = \left(\frac{|x|}{2^e} - 1 \right) \cdot 2^{t-1}$$

↔

t	e_{min}	e_{max}	bias	ε	
24	−126	127	127	6×10^{-8}	SP
53	−1022	1023	1023	10^{-16}	DP

"hidden bit"

z.B.:

$x = 4711.0815$

$\rightarrow s = 0; \quad e = (12 + 127)_{10}; \quad m = 1259686.912_{10}$

$\rightarrow s = 0; \quad e = 10001011; \quad m = 10011001110001010011 0.11101\ldots_2$

Pivotsuche auch für verbesserte Genauigkeit:

$$\left(\begin{array}{cc|c} 0.005 & 1 & 0.5 \\ 1 & 1 & 1 \end{array}\right) \rightarrow x = \begin{pmatrix} 0.502512... \\ 0.497487... \end{pmatrix}$$

3-stellige Mantisse, b = 10:

$$\left(\begin{array}{cc|c} 5.00\text{e-}3 & 1.00\text{e}0 & 5.00\text{e-}1 \\ 1.00\text{e}0 & 1.00\text{e}0 & 1.00\text{e}0 \end{array}\right) \rightarrow \left(\begin{array}{cc|c} 5.00\text{e-}3 & 1.00\text{e}0 & 5.00\text{e-}1 \\ 0.00\text{e}0 & -1.99\text{e}2 & -9.90\text{e}1 \end{array}\right) \rightarrow x = \begin{pmatrix} 6.00\text{e-}1 \\ 4.97\text{e-}1 \end{pmatrix}$$

$$\left(\begin{array}{cc|c} 1.00\text{e}0 & 1.00\text{e}0 & 1.00\text{e}0 \\ 5.00\text{e-}3 & 1.00\text{e}0 & 5.00\text{e-}1 \end{array}\right) \rightarrow \left(\begin{array}{cc|c} 1.00\text{e}0 & 1.00\text{e}0 & 1.00\text{e}0 \\ 0.00\text{e}0 & 9.90\text{e-}1 & 4.95\text{e-}1 \end{array}\right) \rightarrow x = \begin{pmatrix} 5.00\text{e-}1 \\ 5.00\text{e-}1 \end{pmatrix}$$

neben **Spaltenpivotsuche** auch **Totalpivotsuche** (Spalten- und Zeilenvertauschung): nicht viel besser (stabiler)
verbesserte Kriterien zur Pivotsuche → Eigenschaften der Restmatrix

mehrere Ergebnisvektoren

LU-Faktorisierung:
Zerlegung der Matrix **A** in Dreiecksmatrizen

$$\mathbf{A}\vec{x} = \vec{b} \quad \rightarrow \quad \mathbf{LU}\vec{x} = \vec{b}$$

aus Gauss-Elimination:

$$\rightarrow \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \lambda_{21} & 1 & 0 & \cdots & 0 \\ \lambda_{31} & \lambda_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \lambda_{n1} & \lambda_{n2} & \cdots & \lambda_{nn-1} & 1 \end{pmatrix}; \quad \mathbf{U} = \begin{pmatrix} u_{11} & u_{12} & \cdots & \cdots & u_{1n} \\ 0 & u_{22} & & \cdots & u_{2n} \\ 0 & 0 & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & u_{nn} \end{pmatrix}$$

mehrere Ergebnisvektoren

Lösung des Gleichungssystems:

$$\mathbf{A}\vec{x} = \vec{b} \quad \rightarrow \quad \mathbf{L}\left(\mathbf{U}\vec{x}\right) = \vec{b}$$

$$\mathbf{U}\vec{x} = \vec{y} \quad \rightarrow \quad \mathbf{L}\vec{y} = \vec{b}$$

vorwärts einsetzen (forward substitution)

$$y_1 = b_1; \quad y_i = \left(b_i - \sum_{k=1}^{i-1} \lambda_{ik} y_k \right)$$

rückwärts einsetzen (backward substitution)

$$x_n = \frac{y_n}{u_{nn}}; \quad x_i = \frac{1}{u_{ii}} \left(y_i - \sum_{k=i+1}^{n} u_{ik} x_k \right)$$

wichtige Anwendung: Berechnung der inversen Matrix

Lösung der *n* Probleme

$$\mathbf{A}_{n\times n} x_n = b_n \quad \text{mit} \quad b_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}; b_2 = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}; \cdots; b_n = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix};$$

$$\rightarrow \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} (A^{-1})_{1i} \\ (A^{-1})_{2i} \\ \vdots \\ (A^{-1})_{ni} \end{pmatrix} = \begin{pmatrix} \delta_{1i} \\ \delta_{2i} \\ \vdots \\ \delta_{ni} \end{pmatrix}$$

Spezialfall: tridiagonale Matrix

$$\mathbf{A}x = \begin{pmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & & & a_n & b_n \end{pmatrix} x \quad = \quad \mathbf{L}(\mathbf{U}x) = \mathbf{L}y = f$$

$$\rightarrow \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & & & 0 \\ \lambda_2 & 1 & 0 & & \\ & \lambda_3 & 1 & \ddots & \\ & & \ddots & \ddots & 0 \\ 0 & & & \lambda_n & 1 \end{pmatrix}; \quad \mathbf{U} = \begin{pmatrix} d_1 & c_1 & 0 & & 0 \\ 0 & d_2 & c_2 & & \\ & \ddots & \ddots & & \\ & & 0 & d_{n-1} & c_{n-1} \\ 0 & 0 & \cdots & 0 & d_n \end{pmatrix}$$

$d_1 = b_1$

Vorwärtsrekursion:

$$\lambda_j = a_j / d_{j-1}; \quad d_j = b_j - \lambda_j c_{j-1}$$

$$y_1 = f_1 \quad \rightarrow \quad y_i = f_i - \lambda_i y_{i-1}$$

Rückwärtsrekursion:

$$x_n = y_n / d_n \quad \rightarrow \quad x_i = (y_i - c_i x_{i+1}) / d_i$$

tridiagonale Matrix: Thomas-Algorithmus

$$\mathbf{A}x = \begin{pmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & & & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_i \\ \vdots \\ f_n \end{pmatrix}$$

Vorwärtsrekursion:

$$c_i' = \frac{c_i}{b_i - c_{i-1}' a_i}; \quad c_1' = \frac{c_1}{b_1}$$

$$f_i' = \frac{f_i - f_{i-1}' a_i}{b_i - c_{i-1}' a_i}; \quad f_1 = \frac{f_1}{b_1}$$

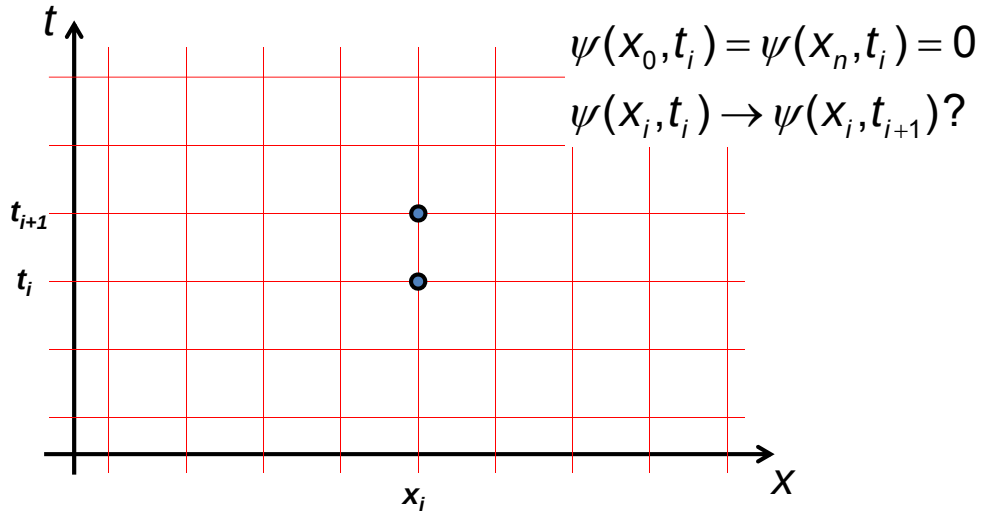
Rückwärtsrekursion:

$$x_n = f_n' \quad \rightarrow \quad x_i = f_i' - c_i' x_{i+1}$$

Anwendung: zeitabhängige Schrödingergleichung

Diskretisierung in x und t; Randbedingung $\psi_0 = \psi_n = 0$

$$i\partial_t\psi(x,t) = \left\{-\Delta/2 + V(x)\right\}\psi(x,t); \quad e = \hbar = m_e = 1$$



explizites Euler-Verfahren (konsistent, nur für $\Delta t \le \Delta x^2/2$ stabil):

$$\psi''(x_i,t_j) = \frac{\psi(x_{i-1},t_j) - 2\psi(x_i,t_j) + \psi(x_{i+1},t_j)}{\Delta x^2}; \quad \dot\psi(x_i,t_j) = \frac{\psi(x_i,t_{j+1}) - \psi(x_i,t_j)}{\Delta t}$$

$$\begin{pmatrix} 0 \\ \psi_{1,j+1} \\ \\ \psi_{i,j+1} \\ \\ \psi_{n-1,j+1} \\ 0 \end{pmatrix} = \left\{ \delta_{ij} - \frac{i\Delta t}{\Delta x^2} \begin{pmatrix} 1 & -1/2 & & & & & 0 \\ -1/2 & 1 & -1/2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & -1/2 & 1 & -1/2 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -1/2 & 1 & -1/2 \\ 0 & & & & & -1/2 & -2 \end{pmatrix} - \delta_{ij} i\Delta t V_j \right\} \begin{pmatrix} 0 \\ \psi_{1,j} \\ \vdots \\ \psi_{i-1,j} \\ \psi_{i,j} \\ \psi_{i+1,j} \\ \vdots \\ \psi_{n-1,j} \\ 0 \end{pmatrix}$$

$$\psi(x_i,t_{j+1}) = (\mathbf{1} - i\tau\mathbf{A} - i\mathbf{1}\Delta t V(x_i))\,\psi(x_i,t_j) \;\rightarrow\; \text{Matrix-Vektor-Multiplikation}$$

implizites Euler-Verfahren (konsistent, immer stabil):

$$\psi''(x_i,t_j)=\frac{\psi(x_{i-1},t_{j+1})-2\psi(x_i,t_{j+1})+\psi(x_{i+1},t_{j+1})}{\Delta x^2};\quad \dot{\psi}(x_i,t_j)=\frac{\psi(x_i,t_{j+1})-\psi(x_i,t_j)}{\Delta t}$$

Umformung:

$$(\mathbf{1}+i\tau\mathbf{A})\psi(x_i,t_{j+1})=-i\mathbf{1}\Delta tV(x_i)\,\psi(x_i,t_j)$$ → lin. Gleichungssystem

Crank-Nicolson-Verfahren (konsistent, immer stabil):

$$\psi''(x_i,t_j)=\frac{1}{2}\left(\frac{\psi(x_{i-1},t_j)-2\psi(x_i,t_j)+\psi(x_{i+1},t_j)}{\Delta x^2}+\frac{\psi(x_{i-1},t_{j+1})-2\psi(x_i,t_{j+1})+\psi(x_{i+1},t_{j+1})}{\Delta x^2}\right)$$

$$\rightarrow\quad (\mathbf{1}+\frac{i\tau}{2}\mathbf{A})\,\psi(x_i,t_{j+1})=(\mathbf{1}-\frac{i\tau}{2}\mathbf{A}-i\mathbf{1}\Delta tV(x_i))\,\psi(x_i,t_j)$$

→ Matrix-Vektor-Multiplikation + lin. Gleichungssystem