DD_QD (MuPAT)

- Display — Display of DD and QD numbers
- Overloads — Available overloads for DD and QD real numbers
- concatenations — Horizontal and vertical concatenations of D, DD, QD arrays together
- size — sizes of a DD or QD array
- conversions
  - d2dd — type conversion from double to DD
  - d2qd — type conversion from double to QD
  - dd2qd — type conversion from DD to QD
  - dd2str — prints a DD number in a string
  - getHi — Get the highest part of dd,qd number
  - qd2dd — type conversion from QD to DD
  - qd2str — prints a QD number in a string
  - str2dd — parses a long input numeric string and converts it into a DD number
- DD: 16 bytes decimal arithmetics
  - dd — builds an array of DD numbers
  - exp — exponential
  - ddeye — Identity matrix
  - ddgqr — QR decomposition with modified Gram-Schmidt orthonormalization
  - ddip — inner product of DD vectors
  - lu — LU factorization of a square DD matrix
  - max — maximal value of an array of DD numbers
  - min — minimal value of an array of DD numbers
  - norm — Norm 1|2|p|inf|fro of a matrix of DD numbers
  - ddnrt — n-th root of a DD real number
  - ddones — generates a matrix made of DD ones
  - ddpow — n-th power of DD variable
  - qr — QR factorization of a DD matrix
  - ddrand — (Quasi) Pseudorandom DD number generator
  - ddzeros — generates a matrix made of DD zeros
- QD: 32 bytes decimal arithmetics
  - qd — builds an array of QD numbers
  - exp — exponential with 64 digits
  - qdeye — Identity matrix
  - qdgqr — QR decomposition with modified Gram-Schmidt orthonormalization
  - qdip — inner product of QD vectors
  - lu — LU factorization of a square QD matrix
  - max — maximal value of an array of QD numbers
  - min — minimal value of an array of QD numbers
  - norm — Norm 1|2|p|inf|fro of a matrix of QD numbers
  - qdones — QD matrix made of ones
  - qdpow — n-th power of a QD real number
  - qr — QR factorization of a QD matrix
  - qdrand — (Quasi) Pseudorandom QD number generator
  - qdzeros — QD matrix made of zeros
- (DLL mandatory)
  - ddGauss — Gaussian Elimination with pivoting for DD
  - ddinv — Inverse matrix of DD matrix
  - qdGauss — Gaussian Elimination with pivoting for QD
  - qdinv — Inverse matrix of QD matrix

Display

Display of DD and QD numbers

Illustration

Scalar DD and QD numbers are displayed with all their digits, without respect to format().

Otherwise, arrays are displayed in a compact way.

The default display in the console is also the one output with `disp(…)`.

Example for DD numbers:

```
--> exp(dd(1))
 ans  =
   2.718281828459045235360287471352E0


--> d = ddrand(2,3)
 d  =
[d1]
   0.5465335   0.7395657   0.5900573
   0.9885408   0.0037173   0.3096467

[d2] 10^-17 *
   2.817203   -1.2288763   5.1060442
   2.5962902   0.0088383   1.0787568
```

Example for QD numbers:

```
--> q = exp(qd(1))
 q  =
   2.71828182845904523536028747135266249775724709369995957496696763E0


--> dd2qd(d)
 ans  =
[d1]
   0.7444457   0.6836931   0.5053017
   0.2269504   0.9365073   0.2524815

[d2] 10^-17 *
   5.4052066   2.0205118  -1.540086
   0.9062805   5.2122164   1.931117

[d3] zeros(2,3)
[d4] zeros(2,3)


--> qdrand(1,3)*1e6
 ans  =
[d1] 10^5 *
   0.3401932   2.3805456   9.4920116
```

```
[d2] 10^-12 *
   0.000135   4.2883451   16.447086

[d3] 10^-29 *
   0.0006044   16.970184  -23.363434

[d4] 10^-46 *
   0.0025863  -59.536291  -118.0642
```

Overloads

Available overloads for DD and QD real numbers

For DD numbers (16 bytes)

All these features are illustrated in the example of the dd() page.

DD = array. **dd** = scalar. **SQ** = SQuare matrix. **col** = column vector.

Supported functions:

eye(DD), zeros(DD), ones(DD), rand(DD)

abs(DD), ceil(DD), floor(DD)

min(DD), min(DD,'r'|1|'c'|2'), max(DD), max(DD,'r'|1|'c'|2')

cos(dd), sin(dd), tan(dd)

`exp(dd), lu(SQ), norm(col), qr(DD), sqrt(dd)`

For QD numbers (32 bytes)

All these features are illustrated in the example of the qd() page.

QD = array. **qd** = scalar. **SQ** = SQuare matrix. **col** = column vector.

Supported functions:

eye(QD), zeros(QD), ones(QD), rand(QD)

abs(QD), ceil(QD), floor(QD)

min(QD), min(QD,'r'|1|'c'|2'), max(QD), max(QD,'r'|1|'c'|2')

cos(qd), sin(qd), tan(qd)

`exp(qd), lu(SQ), norm(col), qr(QD), sqrt(qd)`

Supported operators:

All these features are illustrated in the example of the dd() and qd() pages.

d = scalar decimal number (8 bytes). **D** = array of decimal numbers (8 bytes).

- **Display of numbers** :

  Scalar numbers are displayed with all their digits, without respect to format().

  Otherwise, arrays are displayed as raw tlist.

- **Transposition : DD', QD'**.

  Only real extended numbers are supported.

- **Addition A + B** :

A \ B	d	D	dd	DD	qd	QD
d	.	.	+	+	+	+
D	.	.	+	+	+	+
dd	+	+	+	+	+	+
DD	+	+	+	+	+	+
qd	+	+	+	+	+	+
QD	+	+	+	+	+	+

- **Opposition : -DD, -QD**
- **Substraction A - B** :

A \ B	d	D	dd	DD	qd	QD
d	.	.	-	-	-	-
D	.	.	-	-	-	-
dd	-	-	-	-	-	-
DD	-	-	-	-	-	-
qd	-	-	-	-	-	-
QD	-	-	-	-	-	-

- **Multiplication A * B** : A and B must have compatible sizes

A \ B	d	D	dd	DD	qd	QD
d	.	.	*	*	*	*
D	.	.	*	*	*	*
dd	*	*	*	*	*	*
DD	*	*	*	*	*	*
qd	*	*	*	*	*	*
QD	*	*	*	*	*	*

- **Elementwise multiplication A .* B** : not supported.
- **Division A / b** : b must be scalar.

A / b	d	D	dd	DD	qd	QD
d	.	.	/		/	
D	.	.	/		/	
dd	/		/		/	
DD	/		/		/	
qd	/		/		/	
QD	/		/		/	

- **Comparisons** : Both operands a and b must be scalar.

  All comparisons **a<b**, **a<=b**, **a==b**, **a~=b**, **a>=b**, **a>b** are supported.

a / b	d	D	dd	DD	qd	QD
d	.	.	c		c	
D	.	.				
dd	c		c		c	
DD						
qd	c		c		c	
QD						

- **Extraction**
- **Insertion**
- Concatenations, size()

concatenations

Horizontal and vertical concatenations of D, DD, QD arrays together

Syntax

```
[A , B]
[A ; B]
```

[A , B]	D	DD	QD		[A ; B]	D	DD	QD
D	.	x	x	....	D	.	x	x
DD	x	x	x		DD	x	x	x
QD	x	x	x		QD	x	x	x

Parameters

D
: Array of Decimal numbers (Doubles)(8 bytes/number)

DD
: Array of Double-Double numbers (16 bytes/number)

QD
: Array of Quadruple-Double numbers (32 bytes/number)

Description

The concatenations of D, DD, and QD together have the same meaning as for other regular arrays. When more than 2 operands are embraced, the concatenator is automatically applied in an iterative way.

When both neighboring operands have not the same typeof, the one having the lower resolution is automatically promoted to the other operand's typeof: D -> DD -> QD

Examples

Homogeneous concatenations:

```
a = ddrand(2,1), b = ddrand(2,2)
[b a]            // Horizontal
[a ; b(:,2)]     // Vertical
```

```
--> a = ddrand(2,1), b = ddrand(2,2)
 a  =
[d1]
   0.8833888
   0.6525135

[d2] 10^-17 *
  -4.191495
```

```
   1.3547731

 b  =
[d1]
   0.2146008   0.3616361
   0.312642    0.2922267

[d2] 10^-17 *
  -0.809155   -1.762794
  -0.5817424  -2.1572964

--> [b a]           // Horizontal
 ans  =
[d1]
   0.2146008   0.3616361   0.8833888
   0.312642    0.2922267   0.6525135

[d2] 10^-17 *
  -0.809155   -1.762794   -4.191495
  -0.5817424  -2.1572964   1.3547731


--> [a ; b(:,2)]    // Vertical
 ans  =
[d1]
   0.8833888
   0.6525135
   0.3616361
   0.2922267

[d2] 10^-17 *
  -4.191495
   1.3547731
  -1.762794
  -2.1572964
```

Heterogeneous concatenations, with promotions

```
[%pi  ddpi() qdpi()]
[%e ; exp(dd(1))]
```

```
--> [%pi  ddpi() qdpi()]
 ans  =
[d1]
   3.1415927   3.1415927   3.1415927

[d2] 10^-17 *
   0.   12.246468   12.246468

[d3] 10^-34 *
   0.   0.  -29.947698

[d4] 10^-51 *
   0.   0.   111.24542


--> [%e ; exp(dd(1))]
 ans  =
[d1]
   2.7182818
   2.7182818

[d2] 10^-17 *
   0.
   14.456469
```

See Also

- dd — builds an array of DD numbers
- qd — builds an array of QD numbers
- size
- DD & QD overloads — Available overloads for DD and QD real numbers

Authors

size

sizes of a DD or QD array

Syntax

```
s = size(A)
nr = size(A, "r"|1)
nc = size(A, "c"|2)
n = size(A, "*")
[nr, nc] = size(A)
```

Arguments

A
: Array of DD or QD real numbers

s
: vector `[nr, nc]`

nr
: single integer: number of rows

nc
: single integer: number of columns

n
: single integer: number of elements (= nr*nc).

Description

`size()` works with DD and QD arrays as for arrays of regular data types.

Examples

```
a = ddzeros(3,5);
s = size(a)
[r,c] = size(a)
r = size(a, 1)
c = size(a, 2)
n = size(a, "*")
```

```
--> a = ddzeros(3,5);
--> s = size(a)
 s  =
   3.   5.

--> [r,c] = size(a)
 c  =
   5.
 r  =
   3.

--> r = size(a, 1)
 r  =
```

```
   3.

--> c = size(a, 2)
 c  =
   5.

--> n = size(a, "*")
 n  =
   15.
```

See Also

- dd — builds an array of DD numbers
- qd — builds an array of QD numbers
- concatenations
- DD & QD overloads — Available overloads for DD and QD real numbers

Authors

conversions

d2dd

type conversion from double to DD

Syntax

```
b = d2dd(a)
```

Parameters

a

array of real decimal numbers

b

array of real DD numbers, equal to **a** ones.

Examples

```
a = d2dd(2)
b = 3
c = d2dd(b)
```

See Also

- dd — builds an array of DD numbers
- d2qd — type conversion from double to QD
- dd2qd — type conversion from DD to QD

Authors

d2qd

type conversion from double to QD

Syntax

```
b = d2qd(a)
```

Parameters

a

array of real decimal numbers

b

array of real QD numbers, equal to **a** ones.

Examples

```
a = d2qd(2)
b = 3
c = d2qd(b)
```

See Also

- qd — builds an array of QD numbers
- d2dd — type conversion from double to DD
- dd2qd — type conversion from DD to QD

Authors

Copyright (C) 2011 - Tsubasa Saito

dd2qd

type conversion from DD to QD

Syntax

```
b = dd2qd(a)
```

Parameters

a

array of DD numbers

b

array of QD numbers equal to **a** ones.

Examples

```
a = dd(2,2^-80)
b = dd2qd(a)
c = dd2qd(dd(2,2^-80))
```

See Also

- dd — builds an array of DD numbers
- qd — builds an array of QD numbers
- d2dd — type conversion from double to DD
- d2qd — type conversion from double to QD

Authors

dd2str

prints a DD number in a string

Syntax

```
b = dd2str(a)
```

Parameters

a

DD variable

z

strings

Description

dd2str(a)

displays DD variable.

Examples

```
a = sqrt(dd(2))
b = dd2str(a)
```

See Also

- dd — builds an array of DD numbers

Authors

getHi

Get the highest part of dd,qd number

Syntax

```
b = getHi(a)
```

Parameters

a

DD or QD number

b

double number

Description

getHi(a)

return the highest part of dd,qd number.

Examples

```
a = ddrand(1,1)
b = getHi(a)
A = qdrand(2,2)
B = getHi(A)
```

See Also

- dd — builds an array of DD numbers
- qd — builds an array of QD numbers
- ddrand — (Quasi) Pseudorandom DD number generator
- qdrand — (Quasi) Pseudorandom QD number generator

Authors

qd2dd

type conversion from QD to DD

Syntax

```
b = qd2dd(a)
```

Parameters

a

array of QD numbers

b

array of DD numbers equal to truncated **a** ones

Examples

```
q = qdrand(1,2)
d = qd2dd(q)
typeof(d)
```

```
--> q = qdrand(1,2)
 q  =
[d1]
   0.4368588   0.2693125

[d2] 10^-17 *
   2.1951823   1.6617343

[d3] 10^-34 *
   1.8738568   5.926026

[d4] 10^-51 *
   4.8713013   14.739443


--> d = qd2dd(q)
 d  =
[d1]
   0.4368588   0.2693125

[d2] 10^-17 *
   2.1951823   1.6617343

--> typeof(d)
 ans  =
 dd
```

See Also

- d2dd — type conversion from double to DD
- d2qd — type conversion from double to QD
- dd2qd — type conversion from DD to QD

Authors

qd2str

prints a QD number in a string

Syntax

```
b = qd2str(a)
```

Parameters

a

QD variable

z

strings

Description

qd2str(a)

displays QD variable.

Examples

```
a = sqrt(qd(2))
b = qd2str(a)
```

See Also

- qd — builds an array of QD numbers

Authors

str2dd

parses a long input numeric string and converts it into a DD number

Syntax

```
b = str2dd(a)
```

Parameters

a

string

z

DD variable

Description

str2dd(a)

input a DD variable.

Examples

```
str2dd('-3.00112233445566778899001122e-7')
str2dd('-1234.000111222333444555666777D22')
```

```
--> str2dd('-3.00112233445566778899001122e-7')
 ans  =
  -3.0011223344556677889900112200000E-7

--> str2dd('-1234.000111222333444555666777D22')
 ans  =
  -1.2340001112223334445556667776999E25
```

See Also

- dd — builds an array of DD numbers
- dd2str — prints a DD number in a string

Authors

DD: 16 bytes decimal arithmetics

- dd — builds an array of DD numbers
- exp — exponential
- ddeye — Identity matrix
- ddgqr — QR decomposition with modified Gram-Schmidt orthonormalization
- ddip — inner product of DD vectors
- lu — LU factorization of a square DD matrix
- max — maximal value of an array of DD numbers
- min — minimal value of an array of DD numbers
- norm — Norm 1|2|p|inf|fro of a matrix of DD numbers
- ddnrt — n-th root of a DD real number
- ddones — generates a matrix made of DD ones
- ddpow — n-th power of DD variable
- qr — QR factorization of a DD matrix
- ddrand — (Quasi) Pseudorandom DD number generator
- ddzeros — generates a matrix made of DD zeros

dd

builds an array of DD numbers

Syntax

```
a = dd(ahi)
a = dd(ahi,alo)
```

Parameters

ahi,alo
: arrays of decimal numbers, of same sizes

a
: array of DD numbers, of size(ahi)

Description

Generate DD number using double precision numbers. The author applied overloading to basic arithmetic operations and several Scilab functions.

Examples

```
// define dd variables
a = dd(1)
b = dd(2,2^-70)
c = 3
d = dd(c)
A = [1,2;3,4]
B = dd(A)
// ---------------------------------------------
// four basic arithmetic for dd
a + b
b + 1
2 + b
C = ddrand(2,2) // random matrix generator
A + C
-b
a - b
b - 4
c - b
C - A
b * d // scalar * scalar
2 * b
3 * A // scalar * matrix
A * C // matrix * matrix
a / b
a / 3
5 / b
A / 3
// ---------------------------------------------
// relational operators for dd
a == b
a ~= b
```

```
a <> b
a > b
a < b
a >= b
a <= b
a == 1
b ~= 2
a <> 1
b < 2
b <= 3
b > 3
b >= -1
5 < b
2.2 <= b
2.1 > b
2 >= b

// --------------------------------------------
// available functions for dd

// square root
a = dd(2)
b = sqrt(a)

//n-th root
b = ddnrt(a,3)

//absolute value
c = -b
abs(c)

// cealing,floor
d = b*10
ceil(d)
floor(d)

//sin,cos,tan
sin(d)
cos(d)
tan(d)

// matrix functions
A = ddrand(3,3)
A(2,1) //extraction
v = A(:,2)
A(2,1) = dd(5) // insertion
A(3,:) = ddrand(1,3)

norm(v,2)

B = ddrand(3,3)
[L,U] = lu(B)
[Q,R] = qr(B)
```

See Also

- qd — builds an array of QD numbers

Authors

exp

exponential

Syntax

```
a = exp(dd)
```

Parameters

dd, a

: scalar DD numbers

Examples

```
dde = exp(dd(1))
```

```
--> dde = exp(dd(1))
 dde  =
   2.7182818284590452353602874713 52E0
```

Authors

Copyright (C) 2011 - Tsubasa Saito

ddeye

Identity matrix

Syntax

```
a = ddeye(m,n)
a = eye(DDmat)
```

Parameters

m, n

positive integers

DDmat

matrix of DD numbers

a

DD matrix of size [m,n] or size(DDmat), with DD ones on the diagonal.

Description

ddeye(m,n)

returns a `(m,n)` identity matrix of DD.

Examples

```
ddeye(1,1) //scalar
ddeye(3,3) //matrix
```

See Also

- dd — builds an array of DD numbers
- ddones — builds an array of DD numbers
- ddzeros — generates a matrix made of DD zeros

Authors

Copyright (C) 2011 - Tsubasa Saito

ddgqr

QR decomposition with modified Gram-Schmidt orthonormalization

Syntax

```
[Q,R] = ddgqr(A)
```

Parameters

A

DD matrix

Q, R

DD matrices: Q is orthogonal, R is upper-Right, and `Q*R==A`.

Description

ddgqr(A)

returns a (m,n) orthogonal matrix Q, and a (n,n) upper Right triangular matrix R, such that A = Q*R.

Examples

Decomposing a square matrix:

```
A = ddrand(5,5);
[Q, R] = ddgqr(A);
R       // upper-Right
Q'*Q    // Q orthogonal
Q*R-A   // {Q, R} such that A = Q*R
```

```
--> A = ddrand(5,5);
--> [Q, R] = ddgqr(A);
--> R       // upper-Right
 R  =
[d1]
   1.0276952   1.0883915   0.4438709   0.6298701   0.8929215
   0.          0.4053251   0.405679    0.5015382   0.4345841
   0.          0.          0.6123556   0.0610015   0.0478677
   0.          0.          0.          0.3527555   0.35924
   0.          0.          0.          0.          0.407706

[d2] 10^-17 *
  -5.5416962   8.6289724  -2.1701687   1.0175454   1.0208458
   0.         -2.5160916   1.045223    4.3587077  -0.4703223
   0.          0.          0.0479763   0.3100273  -0.3052711
   0.          0.          0.          1.0067234   0.8982596
```

```
   0.          0.          0.          0.         -1.0127866

--> Q'*Q   // Q orthogonal
 ans  =
[d1]
   1.   0.   0.   0.   0.
   0.   1.   0.   0.   0.
   0.   0.   1.   0.   0.
   0.   0.   0.   1.   0.
   0.   0.   0.   0.   1.

[d2] 10^-17 *
  -2.465D-15   0.          0.   0.          0.
   0.          1.079D-15   0.   0.          0.
   0.          0.          0.   0.          0.
   0.          0.          0.  -9.244D-16   0.
   0.          0.          0.   0.         -2.003D-15

--> Q*R-A  // {Q, R} such that A = Q*R
 ans  =
[d1] 10^-33 *
   0.   0.         -1.540744   0.          0.
   0.   3.8518599   0.         3.0814879   0.
   0.   0.          0.        -3.0814879  -6.1629758
   0.   0.          0.         0.          9.2444637
   0.   0.          0.         0.         -3.0814879

[d2] zeros(5,5)
```

With a rectangular matrix:

```
A = ddrand(3,5);
[Q, R] = ddgqr(A);
R      // upper-Right
Q'*Q   // Q orthogonal
Q*R-A  // {Q, R} such that A = Q*R
```

```
--> A = ddrand(3,5);
--> [Q, R] = ddgqr(A);
--> R      // upper-Right
 R  =
[d1]
   0.8640718   0.0470916   0.7822416   0.7741461   0.9556921
   0.          0.8553543   0.4750534   0.2601096   0.4186977
   0.          0.          0.0190578  -0.7748508  -0.0248745
   0.          0.          0.          0.          0.6761419
   0.          0.          0.          0.          0.6761419

[d2] 10^-17 *
   0.3460269   0.0749481   0.9294711   5.1624504   4.3993537
   0.         -4.753316    0.2413107  -2.4295448   1.9963482
   0.          0.         -0.0536768  -3.2012722   0.1156425
   0.          0.          0.          0.          5.4033633
   0.          0.          0.          0.          5.4033633

--> Q'*Q   // Q orthogonal
 ans  =
[d1]
   1.          0.          0.          0.386515   -0.386515
   0.          1.          0.          0.6967344  -0.6967344
   0.          0.          1.         -0.6042908   0.6042908
   0.386515    0.6967344  -0.6042908   1.         -1.
  -0.386515   -0.6967344   0.6042908  -1.          1.

[d2] 10^-17 *
  -1.233D-15   0.          0.          1.4786666  -1.4786666
```

```
   0.          -2.465D-15   0.          1.0797994  -1.0797994
   0.           0.          1.233D-15   5.3605861  -5.3605861
   1.4786666    1.0797994   5.3605861   0.          1.849D-15
  -1.4786666   -1.0797994  -5.3605861   1.849D-15  -3.081D-15


--> Q*R-A  // {Q, R} such that A = Q*R
 ans  =
[d1] 10^-32 *
   0.           0.          0.   -1.2325952   0.6162976
   0.          -2.4651903   0.   -0.1540744   0.6162976
  -0.3081488    0.          0.    0.          0.

[d2] zeros(3,5)
```

See Also

- qdgqr — QR decomposition with modified Gram-Schmidt orthonormalization
- lu — LU factorization of a square DD matrix
- dd — builds an array of DD numbers

Authors

ddip

inner product of DD vectors

Syntax

```
z = ddip(x,y)
```

Parameters

x,y

2 column vectors of real DD numbers, of same lengths.

z

DD number = x'*y

Examples

```
x = ddrand(10,1);
y = ddrand(10,1);
z = ddip(x,y)
z == x' * y
sqrt(ddip(x,x)) == norm(x,2)
```

```
--> z = ddip(x,y)
 z  =
   2.4071674257475003630014625521 51E0

--> z == x' * y
 ans  =
  T

--> sqrt(ddip(x,x)) == norm(x,2)
 ans  =
  T
```

See Also

- dd — builds an array of DD numbers
- overloads — Available overloads for DD and QD real numbers

Authors

lu

LU factorization of a square DD matrix

Syntax

```
[L,U] = lu(A)
```

Parameters

A

square matrix of DD real numbers, of size [n,n].

U

Upper triangular square matrix of DD real numbers, of size [n,n].

L

square matrix of DD real numbers, of size [m,n], Lower triangular after some rows permutations.

Description

`[L,U]= lu(A)` computes the matrices **L**L and **U** such that `A = L*U`, with **U** Upper triangular, and **L** Lower triangular after some rows permutations.

The implementation for matrices of DD numbers is restricted to square matrices.

Examples

:

```
A = ddrand(4,4)
[L,U] = lu(A)
L*U - A
```

```
--> A = ddrand(4,4)
 A  =
[d1]
   0.2113249   0.6653811   0.8782165   0.7263507
   0.7560439   0.6283918   0.068374    0.1985144
   0.0002211   0.8497452   0.5608486   0.5442573
   0.3303271   0.685731    0.6623569   0.2320748

[d2] 10^-17 *
   0.6832332  -4.191495   -1.7567766  -1.762794
  -2.5063366   1.3547731   0.0425332   0.6182611
  -0.0005011   2.2655615   1.9664026  -2.098293
```

```
  -1.039939   -1.3751321  -0.5817424  -1.2407185


--> [L,U] = lu(A)
 U  =
[d1]
    0.2113249   0.6653811   0.8782165   0.7263507
    0.         -1.7521009  -3.0735666  -2.4001053
    0.          0.         -0.9294872  -0.6195676
    0.          0.          0.         -0.3587091

[d2] 10^-17 *
    0.6832332  -4.191495   -1.7567766  -1.762794
    0.          9.5531966  -15.971936  -10.783318
    0.          0.         -0.1814096  -0.9867343
    0.          0.          0.          0.3761884

 L  =
[d1]
    1.          0.          0.          0.
    3.5776379   1.          0.          0.
    0.0010464  -0.4845891   1.          0.
    1.5631246   0.2022387   0.0955482   1.
[d2] 10^-17 *
    0.          0.          0.          0.
    11.073338   0.          0.          0.
  -0.008786     1.16019     0.          0.
    0.3666115  -0.8467752  -0.3476148   0.


--> L*U - A
 ans  =
[d1] 10^-32 *
    0.          0.          0.   0.
  -1.2325952    2.4651903   0.   0.
    0.          0.          0.   0.
    0.          0.          0.   0.3081488
[d2] zeros(4,4)
```

See Also

- ddqr — QR factorization of a DD matrix
- ddgqr — QR decomposition with modified Gram-Schmidt orthonormalization
- dd — builds an array of DD numbers
- DD overloads — Available overloads for DD and QD real numbers
- norm — Norm 1|2|p|inf|fro of a matrix of DD numbers

Authors

max

maximal value of an array of DD numbers

Syntax

```
dd = max(DD)
DDrow = max(DD, "r"|1)
DDcol = max(DD, "c"|2)
[v, k] = max(DD..)
```

Parameters

DD
: array of DD-encoded real numbers

dd
: DD-encoded number.

DDrow
: row of DD-encoded numbers.

DDcol
: column of DD-encoded numbers.

v
: One of the results dd, DDrow, DDcol, according to the used syntax.

k
: vector: linearized index, indices of rows, or indices of columns in **DD** where the (first) maximum is found. Has the shape of **v**.

Description

`v = max(DD)`, `[v, k] = max(DD)`, and its input directional option `v = max(DD,'r'|1|'c'|2)` and `[v, k] = max(DD,'r'|1|'c'|2)` work for DD arrays as for usual decimal numbers. Please refer to the examples below and to the max() page to see details. "r" and 1 option values are equivalent ; "c" and 2 ones as well.

Please note that the syntax `max(DD1,DD2,..)` is not supported.

Examples

```
hi = grand(3,4,"uin",-2,2); lo = grand(3,4,"uin",0,9).*sign(hi);
d = dd(hi, lo*1e-18)

[v, k] = max(d)
[v, k] = max(d, "r")
[v, k] = max(d, "c")
```

```
--> hi = grand(3,4,"uin",-2,2); lo = grand(3,4,"uin",0,9).*sign(hi);
--> d = dd(hi, lo*1e-18)
 d  =
[d1]
  -2.   1.   0.   2.
  -1.   0.  -1.   2.
```

```
   1.   2.  -2.   1.

[d2] 10^-17 *
  -0.6   0.2   0.    0.1
  -0.9   0.   -0.7   0.7
   0.1   0.1  -0.5   0.7


--> [v, k] = max(d)
 k  =
   11.
 v  =
   2.000000000000000070000000000000E0

--> [v, k] = max(d, "r")
 k  =
   3.   3.   1.   2.

 v  =
[d1]
   1.   2.   0.   2.

[d2] 10^-17 *
   0.1   0.1   0.   0.7


--> [v, k] = max(d, "c")
 k  =
   4.
   4.
   2.

 v  =
[d1]
   2.
   2.
   2.
[d2] 10^-17 *
   0.1
   0.7
   0.1
```

See Also

- max(QD) — maximal value of an array of QD numbers
- max
- min
- min(DD) — minimal value of an array of DD numbers
- min(QD) — minimal value of an array of QD numbers
- dd — builds an array of DD numbers
- overloads
- floor

Authors

min

minimal value of an array of DD numbers

Syntax

```
dd = min(DD)
DDrow = min(DD, "r"|1)
DDcol = min(DD, "c"|2)
[v, k] = min(DD..)
```

Parameters

DD
: array of DD-encoded real numbers

dd
: DD-encoded number.

DDrow
: row of DD-encoded numbers.

DDcol
: column of DD-encoded numbers.

v
: One of the results dd, DDrow, DDcol, according to the used syntax.

k
: vector: linearized index, indices of rows, or indices of columns in **DD** where the (first) minimum is found. Has the shape of **v**.

Description

`v = min(DD)`, `[v, k] = min(DD)`, and its input directional option `v = min(DD,'r'|1|'c'|2)` and `[v, k] = min(DD,'r'|1|'c'|2)` work for DD arrays as for usual decimal numbers. Please refer to the examples below and to the min() page to see details. "r" and 1 option values are equivalent ; "c" and 2 ones as well.

Please note that the syntax `min(DD1,DD2,..)` is not supported.

Examples

```
hi = grand(3,4,"uin",-2,2); lo = grand(3,4,"uin",0,9).*sign(hi);
d = dd(hi, lo*1e-18)

[v, k] = min(d)
[v, k] = min(d, "r")
[v, k] = min(d, "c")
```

```
--> hi = grand(3,4,"uin",-2,2); lo = grand(3,4,"uin",0,9).*sign(hi);
--> d = dd(hi, lo*1e-18)
 d  =
[d1]
   0.   1.   0.   0.
   0.  -2.  -1.   2.
```

```
   1.  -1.  -1.  -1.

[d2] 10^-17 *
   0.    0.8   0.    0.
   0.   -0.2  -0.2   0.5
   0.6   0.   -0.1  -0.2

--> [v, k] = min(d)
 k  =
   5.
 v  =
  -2.00000000000000000200000000000000E0


--> [v, k] = min(d, "r")
 k  =
   1.   2.   2.   3.

 v  =
[d1]
   0.  -2.  -1.  -1.

[d2] 10^-17 *
   0.  -0.2  -0.2  -0.2

--> [v, k] = min(d, "c")
 k  =
   1.
   2.
   4.

 v  =
[d1]
   0.
  -2.
  -1.

[d2] 10^-17 *
   0.
  -0.2
  -0.2
```

See Also

- min(QD) — minimal value of an array of QD numbers
- min
- max
- max(DD) — maximal value of an array of DD numbers
- max(QD) — maximal value of an array of QD numbers
- dd — builds an array of DD numbers
- overloads
- floor

Authors

norm

Norm 1|2|p|inf|fro of a matrix of DD numbers

Syntax

```
dd = norm(A)
dd = norm(A, 1)
dd = norm(A, 2)
dd = norm(A, p)
dd = norm(A, %inf|'inf')
dd = norm(A, 'fro'|'f')
```

Parameters

A

matrix of DD real numbers. Row vectors are transposed, in order to make their norm matching the correspond matrix-case definition.
⚠ The 2-norm is presently restricted to input vectors.

dd

single DD number: the norm of **A**.

Description

The native `norm()` Scilab function is overloaded in order to process DD and QD matrices.

As a reminder, here are the norm definitions:

1	L-1 norm: `max(sum(abs(A),'r'))`
2	Euclidian norm.
p	L-p norm: `sum(abs(A).^p)^(1/p)`, with p>2: Not implemented
inf	Infinity norm: `max(sum(abs(A),'c'))`. For a vector (forced to a column A), this is `max(abs(A))`
fro	Frobenius norm: `sqrt(sum(diag(A'*A)))`. For a vector (forced to a column A), this matches the euclidian norm.

Examples

Norm of a matrix:

```
q = d2dd([-2:2; 1:5])
format(20)

norm(q,1)
norm(q.d,1)

norm(q, %inf)
```

```
norm(q.d, %inf)

norm(q, 'fro')
norm(q.d, 'fro')

format(10)
```

```
--> q = d2dd([-2:2; 1:5])
 q  =
[d1]
  -2.  -1.   0.   1.   2.
   1.   2.   3.   4.   5.
[d2] zeros(2,5)
[d3] zeros(2,5)
[d4] zeros(2,5)

--> format(20)

--> norm(q,1)
 ans  =
   7.0000000000000000000000000000000000000000000000000000000000000000E0

--> norm(q.d,1)
 ans  =
   7.


--> norm(q, %inf)
 ans  =
   1.5000000000000000000000000000000000000000000000000000000000000000E1

--> norm(q.d, %inf)
 ans  =
   15.


--> norm(q, 'fro')
 ans  =
   8.0622577482985496523666132303037711311343963056085733879365923 9E0

--> norm(q.d, 'fro')
 ans  =
   8.0622577482985509 0
```

L2-Norm of a vector:

```
format(20)

q = d2dd(1:3)'
norm(q)
norm(q.d)

format(10)
```

```
--> q = d2dd(1:3)'
 q  =
[d1]
   1.
   2.
   3.
[d2] zeros(3,1)
[d3] zeros(3,1)
[d4] zeros(3,1)

--> norm(q)
 ans  =
```

```
   3.741657386773941385583748732316549301756019807778726946303745 47E0

--> norm(q.d)
 ans  =
   3.741657386773941 38677394090
```

See Also

- dd — builds an array of DD numbers
- DD overloads — Available overloads for DD and QD real numbers

Authors

Copyright (C) 2011 - Tsubasa Saito
Copyright (C) 2018 - Samuel GOUGEON

ddnrt

n-th root of a DD real number

Syntax

```
r = ddnrt(a,n)
```

Parameters

a
: DD number

n
: integer > 0

r
: DD number

Description

ddnrt(a,n)

return a n-th root of **a**.

Examples

```
a = ddrand(1,1)*100
r = ddnrt(a,3)
(r*r*r-a).hi
```

```
--> a = ddrand(1,1)*100
 a  =
   6.561381467862602087669074544900E1

--> r = ddnrt(a,3)
 r  =
   4.033342437368291873928469084975E0

--> (r*r*r-a).hi
 ans  =
  -7.889D-31
```

See Also

- ddpow — n-th power of DD variable
- dd — builds an array of DD numbers

Authors

ddones

generates a matrix made of DD ones

Syntax

```
a = ddones(m,n)
a = ones(DDmat)
```

Parameters

m,n

positive integers: numbers of rows and columns

DDmat

a matrix of DD numbers

a

matrix full of DD ones, of size [m,n] or size(DDmat).

Examples

```
ddones(1,1)   // scalar
ddones(1,4)   // vector
ddones(2,3)   // matrix

d = ddrand(2,4);
ones(d)
```

```
--> ddones(1,1)   // scalar
 ans  =
   1.000000000000000000000000000000E0

--> ddones(1,4)   // vector
 ans  =
[d1]
   1.   1.   1.   1.

[d2] zeros(1,4)


--> ddones(2,3)   // matrix
 ans  =
[d1]
   1.   1.   1.
   1.   1.   1.

[d2] zeros(2,3)
```

```
--> d = ddrand(2,4);
--> ones(d)
 ans  =
[d1]
   1.   1.   1.   1.
   1.   1.   1.   1.

[d2] zeros(2,4)
```

See Also

- ddzeros — generates a matrix made of DD zeros
- ddeye — Identity matrix
- ddrand — (Quasi) Pseudorandom DD number generator
- dd — builds an array of DD numbers
- qdones — QD matrix made of ones

Authors

ddpow

n-th power of DD variable

Syntax

```
p = ddpow(a,n)
```

Parameters

a

single DD real number

n

integer > 0

p

DD real number

Description

ddpow(a,n)

return the n-th power of **a**.

Examples

```
a = - ddrand(1,1)
p = ddpow(a,3)
p == a*a*a

ddpow(a, 384)
```

```
--> a = - ddrand(1,1)
 a  =
  -6.2839178842607950787059963100064E-1

--> p = ddpow(a,3)
 p  =
  -2.4813698651058543441627099623368E-1

--> p == a*a*a
 ans  =
  T

--> ddpow(a, 384)
 ans  =
   3.3152208685004742680009482061I3E-78
```

See Also

- ddnrt — n-th root of a DD real number
- multiplication — Available overloads for DD and QD real numbers
- dd — builds an array of DD numbers

Authors

qr

QR factorization of a DD matrix

Syntax

```
[Q, R] = qr(A)
```

Parameters

A

matrix of DD real numbers, of size [m,n].

Q

Orthogonal matrix of DD real numbers, of size [m,m].

R

upper Right triangular matrix of DD real numbers, of size [m,n] = size(A).

Description

`[Q,R]= qr(A)` computes the matrices **`Q`** and **`R`** such that `A = Q*R`, with **`Q`** orthogonal, and **`R`** upper-Right triangular.

Examples

Square input matrix:

```
A = ddrand(3,3)
[Q,R] = qr(A)
Q'*Q     // Q orthogonal
Q*R - A
```

```
--> A = ddrand(3,3)
 A  =
[d1]
   0.6912788   0.76934     0.9561172
   0.7656859   0.5477634   0.2207409
   0.357265    0.0962289   0.0143259

[d2] 10^-17 *
  -2.1563392  -3.3012259   0.0754278
   3.3926528  -4.8015079   0.5082632
  -1.6398934  -0.3715576  -0.0100407


--> [Q,R] = qr(A)
```

```
 R  =
[d1]
  -1.0916865  -0.9028437  -0.7649447
   0.          0.2933632   0.5700987
   0.          0.         -0.2300846

[d2] 10^-17 *
  -9.0947905   2.7862969   1.1048161
   0.          1.7805365   0.8939544
   0.          0.          0.6188504


 Q  =
[d1]
  -0.633221    0.6737055  -0.3809883
  -0.7013789  -0.2913525   0.6505239
  -0.3272597  -0.6791425  -0.6570133

[d2] 10^-17 *
   0.6394485  -0.9725349  -2.2612669
   5.1128525   1.4099617  -2.095428
   1.0405078  -4.7400587  -2.6590128


--> Q'*Q     // Q orthogonal
 ans  =
[d1]
   1.   0.   0.
   0.   1.   0.
   0.   0.   1.

[d2] 10^-17 *
  -1.849D-15   0.          0.
   0.          3.698D-15   0.
   0.          0.         -4.314D-15


--> Q*R - A
 ans  =
[d1] 10^-32 *
  -1.2325952   1.2325952  -0.3081488
   1.2325952   0.          0.9244464
   0.         -0.2311116  -1.3866696

[d2] zeros(3,3)
```

Rectangular input matrix:

```
A = ddrand(3,4)
[Q,R] = qr(A)
Q'*Q     // Q orthogonal
(Q*R)(2), A(2)
(Q*R)(1:$-1,:) - A
```

```
--> A = ddrand(3,4)
 A  =
[d1]
   0.7883861   0.9709819   0.8525161   0.028486
   0.3453042   0.8875248   0.6744698   0.2367841
   0.2659857   0.2066753   0.9152874   0.7015344

[d2] 10^-17 *
  -0.3388527   1.1452892  -4.2847864   0.0475098
   2.485645    2.0398681   3.9345457  -0.3019807
   1.5545584   0.5803466  -2.9278544   3.8157225
```

```
--> [Q,R] = qr(A)
 R  =
[d1]
  -0.900853  -1.2509781  -1.2748612  -0.3228259
   0.        -0.4563709  -0.0454238   0.0461185
   0.         0.         -0.6261867  -0.6653459

[d2] 10^-17 *
   5.1095656  -8.5066542  -4.2787347   0.6116295
   0.         -2.7577039  -0.1083261   0.2324751
   0.          0.          0.8326262   5.4850138


 Q  =
[d1]
  -0.8751551   0.2713101   0.4006174
  -0.3833081  -0.8940419  -0.2318707
  -0.2952599   0.3564827  -0.8864207
   0.          0.          0.

[d2] 10^-17 *
  -2.6423348   1.9069959  -0.586978
   1.729251    1.0532369   0.2681887
   0.8431917   2.658457    4.3158254
   0.          0.          0.


--> Q'*Q     // Q orthogonal
 ans  =
[d1]
   1.   0.   0.
   0.   1.   0.
   0.   0.   1.

[d2] 10^-17 *
   3.081D-16   0.          0.
   0.         -8.936D-15   0.
   0.          0.          2.465D-15


--> (Q*R)(2), A(2)
 ans  =
   0.7883861   0.9709819   0.8525161   0.028486
   0.3453042   0.8875248   0.6744698   0.2367841
   0.2659857   0.2066753   0.9152874   0.7015344
   0.          0.          0.          0.

 ans  =
   0.7883861   0.9709819   0.8525161   0.028486
   0.3453042   0.8875248   0.6744698   0.2367841
   0.2659857   0.2066753   0.9152874   0.7015344


--> (Q*R)(1:$-1,:) - A
 ans  =
[d1] 10^-32 *
   0.6933348  -5.0844551  -2.4651903   0.6162976
  -0.9244464  -7.395571   -3.6977855  -2.0800043
   0.          0.3081488   1.2325952  -1.2325952

[d2] zeros(3,4)
```

See Also

- ddgqr — QR decomposition with modified Gram-Schmidt orthonormalization

- lu — LU factorization of a square DD matrix
- dd — builds an array of DD numbers
- DD overloads — Available overloads for DD and QD real numbers
- norm — Norm 1|2|p|inf|fro of a matrix of DD numbers

Authors

Copyright (C) 2011 - Tsubasa Saito Copyright (C) 2018 - Samuel GOUGEON

ddrand

(Quasi) Pseudorandom DD number generator

Syntax

```
a = ddrand(m,n)
a = rand(DDmat)
```

Parameters

m,n

positive integers: numbers of rows and columns

DDmat

a matrix of DD numbers

a

matrix of size [m,n] or size(DDmat) of random DD numbers, with uniform distribution on [0,1[.

Description

ddrand() uses the rand() generator. To set the generator, please refer to the rand() page.

Examples

```
ddrand(1,1)    // scalar
ddrand(1,4)    // vector
ddrand(2,3)    // matrix

d = d2dd([1 2 3 ; 4 5 6]);  // template
rand(d)
```

```
--> ddrand(1,1)    // scalar
 ans  =
   3.948993250361848903819918622785E-1


--> ddrand(1,4)    // vector
 ans  =
[d1]
   0.706149   0.6787831   0.4132936   0.1402291

[d2] 10^-17 *
  -4.6128991  -0.491101   0.3302046   0.9724391
```

```
--> ddrand(2,3)    // matrix
 ans  =
[d1]
   0.2512136   0.3921976   0.3361603
   0.3389102   0.4681552   0.5336877

[d2] 10^-17 *
   0.5967803  -0.5662747   2.5888306
   0.4666782  -1.0359787   4.0207999


--> d = dd([1 2 3 ; 4 5 6]);  // template
--> rand(d)
 ans  =
[d1]
   0.3184586   0.4254902   0.251896
   0.5761894   0.9761982   0.4391129

[d2] 10^-17 *
  -0.0354794  -2.3726398   0.9995478
   5.2432006  -3.7086932  -0.0837781
```

See Also

- rand
- qdrand — (Quasi) Pseudorandom QD number generator
- ddzeros — generates a matrix made of DD zeros
- ddones — generates a matrix made of DD ones
- ddeye — Identity matrix
- dd — builds an array of DD numbers

Authors

ddzeros

generates a matrix made of DD zeros

Syntax

```
a = ddzeros(m,n)
a = zeros(DDmat)
```

Parameters

m,n

positive integers: numbers of rows and columns

DDmat

a matrix of DD numbers

a

matrix of DD zeros, of size [m,n] or size(DDmat).

Examples

```
ddzeros(1,1) // scalar
ddzeros(1,4) // vector
ddzeros(2,3) // matrix

d = ddrand(2,4);
zeros(d)
```

```
-->  ddzeros(1,1) // scalar
 ans  =
   0.0000000000000000000000000000000E0

-->  ddzeros(1,4) // vector
 ans  =
[d1] zeros(1,4)
[d2] zeros(1,4)

-->  ddzeros(2,3) // matrix
 ans  =
[d1] zeros(2,3)
[d2] zeros(2,3)


--> d = ddrand(2,4);
--> zeros(d)
 ans  =
[d1] zeros(2,4)
[d2] zeros(2,4)
```

See Also

- qdzeros — QD matrix made of zeros
- ddones — generates a matrix made of DD ones
- ddeye — Identity matrix
- ddrand — (Quasi) Pseudorandom DD number generator
- dd — builds an array of DD numbers

Authors

QD: 32 bytes decimal arithmetics

- qd — builds an array of QD numbers
- exp — exponential with 64 digits
- qdeye — Identity matrix
- qdgqr — QR decomposition with modified Gram-Schmidt orthonormalization
- qdip — inner product of QD vectors
- lu — LU factorization of a square QD matrix
- max — maximal value of an array of QD numbers
- min — minimal value of an array of QD numbers
- norm — Norm 1|2|p|inf|fro of a matrix of QD numbers
- qdones — QD matrix made of ones
- qdpow — n-th power of a QD real number
- qr — QR factorization of a QD matrix
- qdrand — (Quasi) Pseudorandom QD number generator
- qdzeros — QD matrix made of zeros

qd

builds an array of QD numbers

Syntax

```
a = qd(a0)
a = qd(a0,a1)
a = qd(a0,a1,a2)
a = qd(a0,a1,a2,a3)
```

Parameters

a0,a1,a2,a3
: arrays of decimal numbers, of same sizes

a
: array of QD numbers, of size(a0)

Description

- Generate QD number using double precision numbers. The author applied overloading to basic arithmetic operations and several Scilab functions.

Examples

```
// define qd variables
a = qd(1)
b = qdrand(1,1) //// random value generator
c = 3
d = qd(c)
A = [1,2;3,4]
B = qd(A)
// -----------------------------------------------
// four basic arithmetic for qd
a + b
b + 1
2 + b
C = qdrand(2,2)
A + C
-b
a - b
b - 4
c - b
C - A
b * d // scalar * scalar
2 * b
3 * A // scalar * matrix
A * C // matrix * matrix
a / b
a / 3
5 / b
A / 3
// -----------------------------------------------
// relational operators for qd
```

```
a == b
a ~= b
a <> b
a > b
a < b
a >= b
a <= b
a == 1
b ~= 2
a <> 1
b < 2
b <= 3
b > 3
b >= -1
5 < b
2.2 <= b
2.1 > b
2 >= b

// ---------------------------------------------
// available functions for qd

// square root
a = qd(2)
b = sqrt(a)

//n-th root: not yet implemented
//b = qdnrt(a,3)

//absolute value
c = -b
abs(c)

// cealing,floor
d = b*10
ceil(d)
floor(d)

//sin,cos,tan
sin(d)
cos(d)
tan(d)

// matrix functions
A = qdrand(3,3)
A(2,1) //extraction
v = A(:,2)
A(2,1) = qd(5) // insertion
A(3,:) = qdrand(1,3)

norm(v,2)

B = qdrand(3,3)
[L,U] = lu(B)
[Q,R] = qr(B)
```

See Also

- dd — builds an array of DD numbers

Authors

exp

exponential with 64 digits

Syntax

```
a = exp(qd)
```

Parameters

qd, a

: scalar QD numbers

Examples

```
qde = exp(qd(1))
```

```
--> qde = exp(qd(1))
 qde  =
   2.718281828459045235360287471352662497757247093699959574966967 63E0
```

Authors

qdeye

Identity matrix

Syntax

```
a = qdeye(m,n)
a = eye(QDmat)
```

Parameters

m,n

positive integers

QDmat

matrix of QD numbers

a

QD matrix of size [m,n] or size(QDmat), with QD ones on the diagonal.

Description

qdeye(m,n)

returns a `(m,n)` QD-encoded identity matrix.

Examples

```
qdeye(1,1)  // scalar
qdeye(3,4)  // matrix

q = qdrand(4,5);
eye(q)
```

```
--> qdeye(1,1)  // scalar
 ans  =
   1.000000000000000000000000000000000000000000000000000000000000000E0

--> qdeye(3,4)  // matrix
 ans  =
[d1]
   1.   0.   0.   0.
   0.   1.   0.   0.
   0.   0.   1.   0.
[d2] zeros(3,4)
[d3] zeros(3,4)
[d4] zeros(3,4)
```

```
--> q = qdrand(4,5);
--> eye(q)
 ans  =
[d1]
   1.   0.   0.   0.   0.
   0.   1.   0.   0.   0.
   0.   0.   1.   0.   0.
   0.   0.   0.   1.   0.
[d2] zeros(4,5)
[d3] zeros(4,5)
[d4] zeros(4,5)
```

See Also

- qdzeros — QD matrix made of zeros
- qdones — QD matrix made of ones
- qdrand — (Quasi) Pseudorandom QD number generator
- qd — builds an array of QD numbers

Authors

qdgqr

QR decomposition with modified Gram-Schmidt orthonormalization

Syntax

```
[Q,R] = qdgqr(A)
```

Parameters

A

QD matrix

Q, R

QD matrices: Q is orthogonal, R is upper-Right, and `Q*R==A`.

Description

qdgqr(A)

returns a (m,n) orthogonal matrix Q and a (n,n) upper Right triangular matrix R, such that A = Q*R.

Examples

Decomposing a square matrix:

```
A = qdrand(4,4);
[Q, R] = qdgqr(A);
R       // upper-Right
Q'*Q    // Q orthogonal
Q*R-A   // {Q, R} such that A = Q*R
```

```
--> A = qdrand(4,4);
--> [Q, R] = qdgqr(A);
--> R       // upper-Right
 R  =
[d1]
   1.4029373   1.1278077   0.5983773   1.0920244
   0.          0.2197976  -0.0226192   0.4762974
   0.          0.          0.3030321   0.8666805
   0.          0.          0.          0.026052

[d2] 10^-17 *
   4.4497528   0.4789732   3.4040136  -0.5176044
   0.         -1.3391656  -0.059136   -1.9788856
   0.          0.          0.8363172   4.9765785
   0.          0.          0.          0.0736546
```

```
[d3] 10^-34 *
   18.975848  -0.589498   -5.0376968   1.3392224
   0.         -1.7725521   0.2809045   3.4130812
   0.          0.         -5.2942099  -30.61483
   0.          0.          0.          0.4252575

[d4] 10^-51 *
   112.06023  -3.6135207   5.1691145  -0.3923409
   0.         -8.3671828   2.3851052   10.45023
   0.          0.          14.713691  -139.16724
   0.          0.          0.         -2.4250226

--> Q'*Q   // Q orthogonal
 ans  =
[d1]
   1.   0.   0.   0.
   0.   1.   0.   0.
   0.   0.   1.   0.
   0.   0.   0.   1.

[d2] 10^-17 *
   0.   0.   0.   0.
   0.   0.   0.   0.
   0.   0.   0.   0.
   0.   0.   0.   0.

[d3] zeros(4,4)
[d4] zeros(4,4)

--> Q*R-A  // {Q, R} such that A = Q*R
 ans  =
[d1] 10^-65 *
   5.2225623   9.2581786   3.7982271  -3.7982271
  -4.5103947   0.9495568   0.          0.
  -1.8991135  -5.5489724  -0.2373892  -4.3917001
  -0.3857574   0.5341257  -1.7210717   3.7982271

[d2] zeros(4,4)
[d3] zeros(4,4)
[d4] zeros(4,4)
```

With a rectangular matrix:

```
A = qdrand(3,4);
[Q, R] = qdgqr(A);
R       // upper-Right
Q'*Q    // Q orthogonal
Q*R-A   // {Q, R} such that A = Q*R
```

```
--> A = qdrand(3,4);
--> [Q, R] = qdgqr(A);
--> R       // upper-Right
 R  =
[d1]
   0.8468846   0.0480157   0.8400779   0.6050737
   0.          0.8722477   0.6473926   0.2390266
   0.          0.          0.7514185   0.163592
   0.          0.          0.          0.

[d2] 10^-17 *
  -4.3691617  -0.0955502   0.7306212   3.959902
   0.         -2.0875525   5.385732    1.3817957
   0.          0.          4.967595   -0.2752122
   0.          0.          0.          0.

[d3] 10^-34 *
```

```
   22.209151  -0.3864361  -4.8143222  -4.4520127
   0.          10.240153   26.609205   3.0490374
   0.          0.          29.742515   0.3813211
   0.          0.          0.          0.

[d4] 10^-51 *
  -35.347197  -0.6653356  -14.770851  -19.897968
   0.         -0.9747981  -154.11853  -2.916253
   0.          0.         -44.504195   1.6654111
   0.          0.          0.          0.


--> Q'*Q   // Q orthogonal
 ans  =
[d1]
   1.          0.          0.          0.5834909
   0.          1.          0.          0.0253955
   0.          0.          1.         -0.8117225
   0.5834909   0.0253955  -0.8117225   1.

[d2] 10^-17 *
   0.          0.          0.       -4.1491468
   0.          0.          0.       -0.0820512
   0.          0.          0.       -1.68034
  -4.1491468  -0.0820512  -1.68034   0.

[d3] 10^-34 *
   0.          0.          0.          0.6303418
   0.          0.          0.         -0.0776001
   0.          0.          0.         -4.6005905
   0.6303418  -0.0776001  -4.6005905   0.

[d4] 10^-51 *
   0.          0.          0.         -0.3767355
   0.          0.          0.          0.1611753
   0.          0.          0.          1.284789
  -0.3767355   0.1611753   1.284789   0.


--> Q*R-A  // {Q, R} such that A = Q*R
 ans  =
[d1] 10^-65 *
   0.0556381   3.7982271   2.3738919   0.5341257
  -3.7982271  -0.0593473   0.          0.2967365
   0.341247   -0.4747784   2.6112811  -0.2670628

[d2] zeros(3,4)
[d3] zeros(3,4)
[d4] zeros(3,4)
```

See Also

- ddgqr — QR decomposition with modified Gram-Schmidt orthonormalization
- qd — builds an array of QD numbers

Authors

qdip

inner product of QD vectors

Syntax

```
z = qdip(x,y)
```

Parameters

x,y

2 column vectors of real QD numbers, of same lengths.

z

QD number = x'*y

Examples

```
x = qdrand(10,1);
y = qdrand(10,1);
z = qdip(x,y)
z == x' * y
sqrt(qdip(x,x)) == norm(x,2)
```

```
--> z = qdip(x,y)
 z  =
   3.56017539281641690795690874906301351451295011400674390166429558E0

--> z == x' * y
 ans  =
  T

--> sqrt(qdip(x,x)) == norm(x,2)
 ans  =
  T
```

See Also

- qd — builds an array of QD numbers
- overloads — Available overloads for DD and QD real numbers

Authors

lu

LU factorization of a square QD matrix

Syntax

```
[L,U] = lu(A)
```

Parameters

A

square matrix of QD real numbers, of size [n,n].

U

Upper triangular square matrix of QD real numbers, of size [n,n].

L

square matrix of QD real numbers, of size [m,n], Lower triangular after some rows permutations.

Description

`[L,U]= lu(A)` computes the matrices **L**L and **U** such that `A = L*U`, with **U** Upper triangular, and **L** Lower triangular after some rows permutations.

The implementation for matrices of QD numbers is restricted to square matrices.

Examples

:

```
A = qdrand(4,4)
[L,U] = lu(A)
L*U - A
```

```
--> A = qdrand(4,4)
 A  =
[d1]
   0.2113249   0.6653811   0.8782165   0.7263507
   0.7560439   0.6283918   0.068374    0.1985144
   0.0002211   0.8497452   0.5608486   0.5442573
   0.3303271   0.685731    0.6623569   0.2320748
[d2] 10^-17 *
   0.9179115  -2.5457833   2.8569193  -0.0895016
  -0.5581342   4.4783202   0.079312    1.0642382
   0.0007956  -0.7443819   1.1313369  -5.3577353
  -1.362032   -2.042491    3.8557987  -0.7230452
```

```
[d3] 10^-34 *
  -3.1771128   5.926026   -28.470856   0.4634389
   0.9909281   11.387791   0.278016    0.5853533
  -0.0069856   4.883837    2.3716206  -9.5272717
   1.8738568  -11.904973   23.61867    6.6879229
[d4] 10^-51 *
  -5.5450448  -2.4259894   143.85887  -1.9487679
   6.3859594   39.266399  -1.5289045   4.2949033
  -0.04034     29.550156  -11.818576  -34.428087
  -7.5784425  -15.544252   63.266317  -34.142348


--> [L,U] = lu(A)
 U  =
[d1]
   0.2113249   0.6653811   0.8782165   0.7263507
   0.         -1.7521009  -3.0735666  -2.4001053
   0.          0.         -0.9294872  -0.6195676
   0.          0.          0.         -0.3587091
[d2] 10^-17 *
   0.9179115  -2.5457833   2.8569193  -0.0895016
   0.          9.1176704   4.0771268   7.7802232
   0.          0.          1.9475187  -5.3156318
   0.          0.          0.         -0.8330074
[d3] 10^-34 *
  -3.1771128   5.926026   -28.470856   0.4634389
   0.          44.225787  -30.388578  -51.514873
   0.          0.         -15.231586   6.9457524
   0.          0.          0.         -4.9963616
[d4] 10^-51 *
  -5.5450448  -2.4259894   143.85887  -1.9487679
   0.         -173.8784    90.755608  -325.22583
   0.          0.          60.278397   32.035422
   0.          0.          0.          40.514566

 L  =
[d1]
   1.          0.          0.          0.
   3.5776379   1.          0.          0.
   0.0010464  -0.4845891   1.          0.
   1.5631246   0.2022387   0.0955482   1.
[d2] 10^-17 *
   0.          0.          0.          0.
   13.646401   0.          0.          0.
   0.0086084  -0.8552402   0.          0.
  -6.5616244   0.5924347   0.6097856   0.
[d3] 10^-34 *
   0.          0.          0.          0.
   101.23008   0.          0.          0.
  -0.0090916  -1.1613989   0.          0.
   42.858922   0.3631728   2.6062432   0.
[d4] 10^-51 *
   0.          0.          0.          0.
   252.9254    0.          0.          0.
   0.0413223  -8.8502451   0.          0.
  -290.23453   0.7679612  -16.352271   0.


--> L*U - A
 ans  =
[d1] 10^-65 *
   0.          0.          0.          0.
   6.5282028   1.4243352   0.0890209   0.6528203
   0.0018546  -2.8486703  -0.2373892   0.
   1.3056406   1.186946    0.          0.
[d2] zeros(4,4)
[d3] zeros(4,4)
[d4] zeros(4,4)
```

See Also

- qdqr — QR factorization of a QD matrix
- qdgqr — QR decomposition with modified Gram-Schmidt orthonormalization
- qd — builds an array of QD numbers
- QD overloads — Available overloads for DD and QD real numbers
- norm — Norm 1|2|p|inf|fro of a matrix of QD numbers

Authors

max

maximal value of an array of QD numbers

Syntax

```
qd = max(QD)
QDrow = max(QD, "r"|1)
QDcol = max(QD, "c"|2)
[v, k] = max(QD..)
```

Parameters

QD

array of QD-encoded real numbers

qd

QD-encoded number.

QDrow

row of QD-encoded numbers.

QDcol

column of QD-encoded numbers.

v

One of the results qd, QDrow, QDcol, according to the used syntax.

k

vector: linearized index, indices of rows, or indices of columns in **`QD`** where the (first) maximum is found. Has the shape of **`v`**.

Description

`v = max(QD)`, `[v, k] = max(QD)`, and its input directional option `v = max(QD,'r'|1|'c'|2)` and `[v, k] = max(QD,'r'|1|'c'|2)` work for QD arrays as for usual decimal numbers. Please refer to the examples below and to the max() page to see details. "r" and 1 option values are equivalent ; "c" and 2 ones as well.

Please note that the syntax `max(QD1,QD2,..)` is not supported.

Examples

```
h = grand(3,5,"uin",-1,1); d = grand(3,5,3,"uin",0,2);
s = sign(h); d = d.*cat(3, s, s, s);
q = qd(h, d(:,:,1)*1e-17, d(:,:,2)*1e-34, d(:,:,3)*1e-51)

[v, k] = max(q)
[v, k] = max(q, "r")
[v, k] = max(q, 2)
```

```
--> h = grand(3,5,"uin",-1,1); d = grand(3,5,3,"uin",0,2);
--> s = sign(h); d = d.*cat(3, s, s, s);
--> q = qd(h, d(:,:,1)*1e-17, d(:,:,2)*1e-34, d(:,:,3)*1e-51)
 q  =
[d1]
   0.   0.   1.   1.   1.
   0.   1.   0.   0.   0.
  -1.   0.   1.   1.  -1.

[d2] 10^-17 *
   0.   0.   0.   1.   1.
   0.   0.   0.   0.   0.
  -1.   0.   0.   0.   0.

[d3] 10^-34 *
   0.   0.   0.   2.   0.
   0.   0.   0.   0.   0.
  -2.   0.   1.   2.   0.

[d4] 10^-51 *
   0.   0.   2.   1.   1.
   0.   2.   0.   0.   0.
  -1.   0.   1.   1.   0.


--> [v, k] = max(q)
 k  =
   10.

 v  =
   1.000000000000000010000000000000000091542424054621923162020134021E0


--> [v, k] = max(q, "r")
 k  =
   1.   2.   3.   1.   1.

 v  =
[d1]
   0.   1.   1.   1.   1.

[d2] 10^-17 *
   0.   0.   0.   1.   1.

[d3] 10^-34 *
   0.   0.   1.   2.   0.

[d4] 10^-51 *
   0.   2.   1.   1.   1.


--> [v, k] = max(q, 2)
 k  =
   4.
   2.
   4.

 v  =
[d1]
   1.
```

```
   1.
   1.

[d2] 10^-17 *
   1.
   0.
   0.

[d3] 10^-34 *
   2.
   0.
   2.

[d4] 10^-51 *
   1.
   2.
   1.
```

See Also

- max(DD) — maximal value of an array of DD numbers
- max
- min
- min(DD) — minimal value of an array of DD numbers
- min(QD) — minimal value of an array of QD numbers
- qd — builds an array of QD numbers
- overloads
- floor

Authors

min

minimal value of an array of QD numbers

Syntax

```
qd = min(QD)
QDrow = min(QD, "r"|1)
QDcol = min(QD, "c"|2)
[v, k] = min(QD..)
```

Parameters

QD

array of QD-encoded real numbers

qd

QD-encoded number.

QDrow

row of QD-encoded numbers.

QDcol

column of QD-encoded numbers.

v

One of the results qd, QDrow, QDcol, according to the used syntax.

k

vector: linearized index, indices of rows, or indices of columns in `QD` where the (first) minimum is found. Has the shape of `v`.

Description

`v = min(QD)`, `[v, k] = min(QD)`, and its input directional option `v = min(QD,'r'|1|'c'|2)` and `[v, k] = min(QD,'r'|1|'c'|2)` work for QD arrays as for usual decimal numbers. Please refer to the examples below and to the min() page to see details. "r" and 1 option values are equivalent ; "c" and 2 ones as well.

Please note that the syntax `min(QD1,QD2,..)` is not supported.

Examples

```
h = grand(3,5,"uin",-1,1); d = grand(3,5,3,"uin",0,2);
s = sign(h); d = d.*cat(3, s, s, s);
q = qd(h, d(:,:,1)*1e-17, d(:,:,2)*1e-34, d(:,:,3)*1e-51)

[v, k] = min(q)
[v, k] = min(q, "r")
[v, k] = min(q, 2)
```

```
--> h = grand(3,5,"uin",-1,1); d = grand(3,5,3,"uin",0,2);
--> s = sign(h); d = d.*cat(3, s, s, s);
--> q = qd(h, d(:,:,1)*1e-17, d(:,:,2)*1e-34, d(:,:,3)*1e-51)
 q  =
[d1]
  -1.   0.  -1.   0.   1.
  -1.  -1.   1.  -1.  -1.
  -1.   0.  -1.  -1.  -1.
[d2] 10^-17 *
  -1.   0.   0.   0.   0.
   0.  -2.   0.  -1.   0.
   0.   0.   0.  -2.   0.
[d3] 10^-34 *
  -1.   0.  -2.   0.   2.
  -2.  -2.   2.   0.  -1.
  -1.   0.   0.  -2.   0.
[d4] 10^-51 *
  -2.   0.  -1.   0.   2.
  -2.  -2.   0.  -2.  -2.
  -2.   0.   0.   0.   0.


--> [v, k] = min(q)
 k  =
   5.

 v  =
  -1.00000000000000002000000000000000163084848109243847770548190206E0


--> [v, k] = min(q, "r")
 k  =
   1.   2.   1.   3.   2.

 v  =
[d1]
  -1.  -1.  -1.  -1.  -1.
[d2] 10^-17 *
  -1.  -2.   0.  -2.   0.
[d3] 10^-34 *
  -1.  -2.  -2.  -2.  -1.
[d4] 10^-51 *
  -2.  -2.  -1.   0.  -2.


--> [v, k] = min(q, 2)
 k  =
   1.
   2.
   4.

 v  =
[d1]
  -1.
  -1.
  -1.
[d2] 10^-17 *
  -1.
  -2.
  -2.
```

```
[d3] 10^-34 *
  -1.
  -2.
  -2.
[d4] 10^-51 *
  -2.
  -2.
   0.
```

See Also

- min(DD) — minimal value of an array of DD numbers
- min
- max
- max(DD) — maximal value of an array of DD numbers
- max(QD) — maximal value of an array of QD numbers
- qd — builds an array of QD numbers
- overloads
- floor

Authors

norm

Norm 1|2|p|inf|fro of a matrix of QD numbers

Syntax

```
qd = norm(A)
qd = norm(A, 1)
qd = norm(A, 2)
qd = norm(A, p)
qd = norm(A, %inf|'inf')
qd = norm(A, 'fro'|'f')
```

Parameters

A

matrix of QD real numbers. Row vectors are transposed, in order to make their norm matching the correspond matrix-case definition.
⚠ The 2-norm is presently restricted to input vectors.

qd

single QD number: the norm of **A**.

Description

The native `norm()` Scilab function is overloaded in order to process DD and QD matrices.

As a reminder, here are the norm definitions:

1	L-1 norm: `max(sum(abs(A),'r'))`
2	Euclidian norm.
p	L-p norm: `sum(abs(A).^p)^(1/p)`, with p>2: Not implemented
inf	Infinity norm: `max(sum(abs(A),'c'))`. For a vector (forced to a column A), this is `max(abs(A))`
fro	Frobenius norm: `sqrt(sum(diag(A'*A)))`. For a vector (forced to a column A), this matches the euclidian norm.

Examples

Norm of a matrix:

```
q = d2qd([-2:2; 1:5])
format(20)

norm(q,1)
norm(q.d,1)

norm(q, %inf)
```

```
norm(q.d, %inf)

norm(q, 'fro')
norm(q.d, 'fro')

format(10)
```

```
--> q = d2qd([-2:2; 1:5])
 q  =
[d1]
  -2.  -1.   0.   1.   2.
   1.   2.   3.   4.   5.
[d2] zeros(2,5)
[d3] zeros(2,5)
[d4] zeros(2,5)

--> format(20)

--> norm(q,1)
 ans  =
   7.0000000000000000000000000000000000000000000000000000000000000000E0

--> norm(q.d,1)
 ans  =
   7.


--> norm(q, %inf)
 ans  =
   1.5000000000000000000000000000000000000000000000000000000000000000E1

--> norm(q.d, %inf)
 ans  =
   15.


--> norm(q, 'fro')
 ans  =
   8.06225774829854965236661323030377113113439630560857338793659239E0

--> norm(q.d, 'fro')
 ans  =
   8.06225774829855090
```

L2-Norm of a vector:

```
format(20)

q = d2qd(1:3)'
norm(q)
norm(q.d)

format(10)
```

```
--> q = d2qd(1:3)'
 q  =
[d1]
   1.
   2.
   3.
[d2] zeros(3,1)
[d3] zeros(3,1)
[d4] zeros(3,1)

--> norm(q)
 ans  =
```

```
   3.7416573867739413855837487323165493017560198077787269463037454 7E0

--> norm(q.d)
 ans  =
   3.741657386773 94090
```

See Also

- qd — builds an array of QD numbers
- QD overloads — Available overloads for DD and QD real numbers

Authors

Copyright (C) 2011 - Tsubasa Saito Copyright (C) 2018 - Samuel GOUGEON

qdones

QD matrix made of ones

Syntax

```
a = qdones(m,n)
a = ones(QDmat)
```

Parameters

m, n

: positive integers

QDmat

: a matrix of QD numbers

a

: matrix full of QD ones, of size [m,n] or size(QDmat).

Examples

```
qdones(1,1)   // scalar
qdones(1,4)   // vector
qdones(2,3)   // matrix

d = qdrand(2,4);
ones(d)
```

```
--> qdones(1,1)   // scalar
 ans  =
   1.0000000000000000000000000000000000000000000000000000000000000000E0

--> qdones(1,4)   // vector
 ans  =
[d1]
   1.   1.   1.   1.

[d2] zeros(1,4)
[d3] zeros(1,4)
[d4] zeros(1,4)

--> qdones(2,3)   // matrix
 ans  =
[d1]
   1.   1.   1.
   1.   1.   1.

[d2] zeros(2,3)
[d3] zeros(2,3)
```

```
[d4] zeros(2,3)


--> d = qdrand(2,4);
--> ones(d)
 ans  =
[d1]
   1.   1.   1.   1.
   1.   1.   1.   1.

[d2] zeros(2,4)
[d3] zeros(2,4)
[d4] zeros(2,4)
```

See Also

- ddones — generates a matrix made of DD ones
- qdeye — Identity matrix
- qdzeros — QD matrix made of zeros
- qdrand — (Quasi) Pseudorandom QD number generator
- qd — builds an array of QD numbers
- QD overloads — Available overloads for DD and QD real numbers

Authors

qdpow

n-th power of a QD real number

Syntax

```
p = qdpow(a,n)
```

Parameters

a
: QD real number

n
: integer > 0

p
: QD real number

Description

qdpow(a,n)

return a n-th power of QD number.

Examples

```
a = qdrand(1,1)
p = qdpow(a,3)
p == a*a*a

qdpow(-a, 371)
```

```
--> a = qdrand(1,1)
 a  =
   2.11324865464121859330597333610137131197531777868435698370881634E-1

--> p = qdpow(a,3)
 p  =
   9.43738784555168966366516175376947828621009167870893058967049250E-3

--> p == a*a*a
 ans  =
  T

--> qdpow(-a, 371)
 ans  =
  -3.60307275195960225336041475423518556241935332765288708739794485E-251
```

See Also

- ddnrt — n-th root of a DD real number

- multiplication — Available overloads for DD and QD real numbers
- qd — builds an array of QD numbers

Authors

qr

QR factorization of a QD matrix

Syntax

```
[Q, R] = qr(A)
```

Parameters

A
: matrix of QD real numbers, of size [m,n].

Q
: Orthogonal matrix of QD real numbers, of size [m,m].

R
: upper Right triangular matrix of QD real numbers, of size [m,n] = size(A).

Description

`[Q,R]= qr(A)` computes the matrices **Q** and **R** such that `A = Q*R`, with **Q** orthogonal, and **R** upper-Right triangular.

Examples

Square input matrix:

```
A = qdrand(3,3)
[Q,R] = qr(A)
Q'*Q      // Q orthogonal
Q*R - A
```

```
--> A = qdrand(3,3)
 A  =
[d1]
   0.6744698   0.2367841   0.8287412
   0.9152874   0.7015344   0.3161073
   0.028486    0.1202527   0.5305191
[d2] 10^-17 *
   1.640597   -0.2982776   1.8937044
   4.7801543  -5.3886078   0.2948024
  -0.0866402  -0.3056266   0.6539004
[d3] 10^-34 *
   9.2600373   0.8368638   8.5577923
  -8.6402125  -14.758006  -0.109292
  -0.0023412   1.0305      0.2138737
[d4] 10^-51 *
   25.767596   0.3439711  -35.969429
   56.108312  -69.010269   0.4279924
  -0.010919    3.5944152   1.3115704


--> [Q,R] = qr(A)
```

```
 R  =
[d1]
  -1.1373091  -0.7080176  -0.7591624
   0.         -0.247768    0.2248517
   0.          0.         -0.6643045
[d2] 10^-17 *
  -10.804238   1.2040685  -2.8589822
   0.          1.2657967  -0.8216278
   0.          0.         -3.6599586
[d3] 10^-34 *
  -19.301747   2.4665605  -28.301214
   0.         -4.7654354  -3.8427116
   0.          0.          4.9568623
[d4] 10^-51 *
   63.015136  -6.9150735   22.641026
   0.         -31.756367   1.6983272
   0.          0.          8.3776666

 Q  =
[d1]
  -0.59304     0.738992   -0.3196786
  -0.8047833  -0.5316812   0.2638918
  -0.0250468  -0.4137704  -0.9100367
[d2] 10^-17 *
   0.4723805  -4.8932887  -2.3383947
  -0.1250943   1.2879175   0.387895
  -0.1676178  -0.6446533   2.2015095
[d3] 10^-34 *
  -3.1201395   28.63755    14.857217
  -0.4783138   0.228468    0.0035934
  -0.3751341   3.6522336   7.8505864
[d4] 10^-51 *
  -18.082799  -74.785685  -18.170933
   0.4497962  -0.1707274  -0.017484
   1.6422523   7.858092   -80.851926


--> Q'*Q     // Q orthogonal
 ans  =
[d1]
   1.   0.   0.
   0.   1.   0.
   0.   0.   1.

[d2] 10^-17 *
   0.   0.   0.
   0.   0.   0.
   0.   0.   0.

[d3] zeros(3,3)
[d4] zeros(3,3)

--> Q*R - A
 ans  =
[d1] 10^-64 *
  -0.6172119   0.4376863  -5.0326509
  -0.0949557   0.284867    3.8108384
   0.0093194  -0.0593473  -8.2477908
[d2] zeros(3,3)
[d3] zeros(3,3)
[d4] zeros(3,3)
```

Rectangular input matrix:

```
A = qdrand(3,4)
[Q,R] = qr(A)
Q'*Q     // Q orthogonal
```

```
(Q*R)(2), A(2)
(Q*R)(1:$-1,:) - A
```

```
--> A = qdrand(3,4)
 A  =
[d1]
   0.025871    0.2413538   0.2893728   0.3454984
   0.5174468   0.5064435   0.0887932   0.7064868
   0.3916873   0.4236102   0.6212882   0.5211472
[d2] 10^-17 *
  -0.0924012   0.5787125   2.0737583   0.7253082
  -1.5854307  -5.4430809   0.3560689  -3.1813871
  -2.2888826   0.9501206  -3.1495401   0.4583247
[d3] 10^-34 *
  -0.630647   -2.9446788   4.4190639  -0.7377624
  -2.1015986   9.4105055  -3.629774   -5.6435068
  -1.6176924   5.7156986  -14.931267  -1.5105702
[d4] 10^-51 *
  -5.1663189  -19.009761  -39.773168   3.9272681
   4.0770955  -25.115063  -20.69295   -40.244413
  -10.359079  -29.836918   49.48939   -8.2666715


--> [Q,R] = qr(A)
 R  =
[d1]
  -0.6494917  -0.6685603  -0.4569463  -0.8909036
   0.          0.2172772   0.3336651   0.3052483
   0.          0.         -0.396847    0.0565813

[d2] 10^-17 *
  -0.9183674   4.0863942   0.5246183   1.3400133
   0.         -0.3118598   1.4280393   0.6187622
   0.          0.         -1.2854773  -0.3282915

[d3] 10^-34 *
   7.1565659   9.9115492   2.756013   -7.3313577
   0.          0.3002421  -11.93012    2.9081976
   0.          0.          4.7812511   1.1232049

[d4] 10^-51 *
   36.453962  -78.332762  -0.4709708   20.104321
   0.         -1.6776815  -26.866888  -4.9235185
   0.          0.          -29.109985   5.7078902


 Q  =
[d1]
  -0.0398327   0.9882458   0.1475928
  -0.796695   -0.1205612   0.5922348
  -0.6030675   0.0939961  -0.7921328
   0.          0.          0.
[d2] 10^-17 *
  -0.0947659  -3.8210273  -0.4146059
  -0.3772909   0.2168924   0.9494928
  -0.2265137  -0.3053764  -0.4491638
   0.          0.          0.
[d3] 10^-34 *
   0.1521926  -4.416197    0.5529143
   0.1815863  -1.1499104   3.0882364
   0.8058953   0.4462959  -0.1152056
   0.          0.          0.
[d4] 10^-51 *
  -0.4732434  -19.136316   0.9216502
   0.5295538  -7.6458796   15.242305
  -3.6577173  -2.288692    0.351394
   0.          0.          0.
```

```
--> Q'*Q      // Q orthogonal
 ans  =
[d1]
   1.   0.   0.
   0.   1.   0.
   0.   0.   1.

[d2] 10^-17 *
   0.   0.   0.
   0.   0.   0.
   0.   0.   0.

[d3] zeros(3,3)
[d4] zeros(3,3)


--> (Q*R)(2), A(2)
 ans  =
   0.025871    0.2413538   0.2893728   0.3454984
   0.5174468   0.5064435   0.0887932   0.7064868
   0.3916873   0.4236102   0.6212882   0.5211472
   0.          0.          0.          0.

 ans  =
   0.025871    0.2413538   0.2893728   0.3454984
   0.5174468   0.5064435   0.0887932   0.7064868
   0.3916873   0.4236102   0.6212882   0.5211472


--> (Q*R)(1:$-1,:) - A
 ans  =
[d1] 10^-64 *
  -0.0534126   0.593473    0.6646897   1.4836825
   0.4035616   0.0474778  -0.3323449  -0.1899114
   0.6172119  -0.1899114   1.2344238   0.2255197

[d2] zeros(3,4)
[d3] zeros(3,4)
[d4] zeros(3,4)
```

See Also

- qdgqr — QR decomposition with modified Gram-Schmidt orthonormalization
- lu — LU factorization of a square QD matrix
- qd — builds an array of QD numbers
- QD overloads — Available overloads for DD and QD real numbers
- norm — Norm 1|2|p|inf|fro of a matrix of QD numbers

Authors

qdrand

(Quasi) Pseudorandom QD number generator

Syntax

```
a = qdrand(m,n)
a = rand(QDmat)
```

Parameters

m,n

integers

QDmat

a matrix of QD numbers

a

matrix of size [m,n] or size(QDmat) of random QD numbers, with uniform distribution on [0,1[.

Description

qdrand() uses the rand() generator. To set the generator, please refer to the rand() page.

Examples

```
qdrand(1,1)    // scalar
qdrand(1,4)    // vector
qdrand(2,3)    // matrix

q = d2qd([1 2 3 ; 4 5 6]);  // template
rand(q)
```

```
--> qdrand(1,1)    // scalar
 ans  =
   2.2074085660278798582251789048396118785382092513036287904835328 8E-1

--> qdrand(1,4)    // vector
 ans  =
[d1]
   0.6561381   0.2445539   0.5283124   0.8468926

[d2] 10^-17 *
   1.0506091  -1.2569616   1.1229981   1.223734

[d3] 10^-34 *
  -5.6297094   3.1317322   0.5624801   2.1559544
```

```
[d4] 10^-51 *
  -31.699615  -7.739013   4.5088728   4.9954088


--> qdrand(2,3)    // matrix
 ans  =
[d1]
   0.3161073   0.5715175   0.824862
   0.5305191   0.0478015   0.5798843

[d2] 10^-17 *
   0.1625074   1.8937044   0.6539004
  -4.4689629   0.2948024   3.7193957

[d3] 10^-34 *
  -0.9364927   4.6887237  -2.8213599
   11.553239   0.6494335   8.5577923

[d4] 10^-51 *
  -1.3157813   25.767596  -17.926795
  -37.604663   2.6530177  -85.1845


--> q = d2qd([1 2 3 ; 4 5 6]);  // template
--> rand(q)
 ans  =
[d1]
   0.2039064   0.0181815   0.0105835
   0.158999    0.4098371   0.196531

[d2] 10^-17 *
  -0.4996219  -0.1327166   0.0438558
   1.0698596   2.3638705  -1.0858258

[d3] 10^-34 *
   1.9138771   0.6324264  -0.1820345
   0.1360263   0.7458167  -5.6961792

[d4] 10^-51 *
  -6.67334    -3.2283822   1.2898989
   0.1737281  -2.4198306   9.7320647
```

See Also

- rand
- ddrand — (Quasi) Pseudorandom DD number generator
- qdzeros — QD matrix made of zeros
- qdones — QD matrix made of ones
- qdeye — Identity matrix
- qd — builds an array of QD numbers

Authors

qdzeros

QD matrix made of zeros

Syntax

```
a = qdzeros(m,n)
a = zeros(QDmat)
```

Parameters

m, n

positive integers

QDmat

a matrix of QD numbers

a

matrix of QD zeros, of size [m,n] or size(QDmat).

Examples

```
qdzeros(1,1)     // scalar
qdzeros(1,4)     // vector
qdzeros(2,3)     // matrix

q = qdrand(2,4); // template
zeros(q)
```

```
--> qdzeros(1,1)     // scalar
 ans  =
   0.000000000000000000000000000000000000000000000000000000000000000E0

--> qdzeros(1,4)     // vector
 ans  =
[d1] zeros(1,4)
[d2] zeros(1,4)
[d3] zeros(1,4)
[d4] zeros(1,4)

--> qdzeros(2,3)     // matrix
 ans  =
[d1] zeros(2,3)
[d2] zeros(2,3)
[d3] zeros(2,3)
[d4] zeros(2,3)


--> q = qdrand(2,4); // template
--> zeros(q)
```

```
 ans  =
[d1] zeros(2,4)
[d2] zeros(2,4)
[d3] zeros(2,4)
[d4] zeros(2,4)
```

See Also

- ddzeros — generates a matrix made of DD zeros
- qdones — QD matrix made of ones
- qdeye — Identity matrix
- qdrand — (Quasi) Pseudorandom QD number generator
- qd — builds an array of QD numbers

Authors

(DLL mandatory)

- ddGauss — Gaussian Elimination with pivoting for DD
- ddinv — Inverse matrix of DD matrix
- qdGauss — Gaussian Elimination with pivoting for QD
- qdinv — Inverse matrix of QD matrix

ddGauss

Gaussian Elimination with pivoting for DD

Syntax

```
x = ddGauss(A,b)
```

Parameters

A

DD matrix

b

DD vector

x

DD vector

Description

ddGauss(A,b)

returns a apploximation vector x from Gaussian elimination.

Examples

```
A = ddrand(10,10)
b = ddrand(10,1)
x = ddGauss(A,b)
```

See Also

- dd — builds an array of DD numbers
- ddrand — (Quasi) Pseudorandom DD number generator

Authors

ddinv

Inverse matrix of DD matrix

Syntax

```
X = ddinv(A)
```

Parameters

A

DD matrix

X

DD matrix

Description

ddinv(A)

returns an inverse matrix of DD matrix A.

Examples

```
A = ddrand(10,10)
X = ddinv(A)
```

See Also

- ddGauss — Gaussian Elimination with pivoting for DD
- lu
- dd — builds an array of DD numbers
- ddrand — (Quasi) Pseudorandom DD number generator

Authors

Copyright (C) 2011 - Tsubasa Saito

qdGauss

Gaussian Elimination with pivoting for QD

Syntax

```
x = qdGauss(A,b)
```

Parameters

A

QD matrix

b

QD vector

x

QD vector

Description

qdGauss(A,b)

returns a apploximation vector x from Gaussian elimination.

Examples

```
A = qdrand(10,10)
b = qdrand(10,1)
x = qdGauss(A,b)
```

See Also

- qd — builds an array of QD numbers
- qdrand — (Quasi) Pseudorandom QD number generator

Authors

qdinv

Inverse matrix of QD matrix

Syntax

```
X = qdinv(A)
```

Parameters

A

QD matrix

X

QD matrix

Description

qdinv(A)

returns an inverse matrix of QD matrix A.

Examples

```
A = qdrand(10,10)
X = qdinv(A)
```

See Also

- qdGauss — Gaussian Elimination with pivoting for QD
- lu
- qd — builds an array of QD numbers
- qdrand — (Quasi) Pseudorandom DD number generator

Authors