

Using the Newmat09 Library

It supports matrix of types

Matrix	(rectangular matrix)
UpperTriangularMatrix	
LowerTriangularMatrix	
DiagonalMatrix	
SymmetricMatrix	
RowVector	(derived from Matrix)
ColumnVector	(derived from Matrix)

and others

Only one element type (float or double) is supported.

To construct an $m \times n$ matrix, 'A', (m and n are integers) use

```
Matrix A(m,n);
```

The UpperTriangularMatrix, LowerTriangularMatrix, SymmetricMatrix and DiagonalMatrix types are square. To construct an $n \times n$ matrix use, for example

```
UpperTriangularMatrix UT(n);  
LowerTriangularMatrix LT(n);  
SymmetricMatrix S(n);  
DiagonalMatrix D(n);
```

You can also use a constructor to set a matrix equal to another matrix or matrix expression.

```
Matrix A = UT;  
Matrix A = UT * LT;
```

Accessing Elements

$A(i, j)$ for i, j ranging from 1 to n
and $A.element(i, j)$ for i, j ranging from 0 to $n-1$

Entering values

You can load the elements of a matrix from an array:

```
Matrix A(3,2);  
Real a[] = { 11,12,21,22,31,33 };  
A << a;
```

This construction does not check that the numbers of elements match correctly. This version of '`<<`' can be used with submatrices on the left hand side. It is not defined for band matrices.

Alternatively you can enter short lists using a sequence of numbers separated by '`<<`' .

```
Matrix A(3,2);  
A << 11 << 12  
    << 21 << 22  
    << 31 << 32;
```

Accessing Rows Columns and Submatrices

```
A.SubMatrix(fr,lr,fc,lc)
```

This selects a submatrix from 'A'. The arguments fr,lr,fc,lc are the first row, last row, first column, last column of the submatrix with the numbering beginning at 1. This may be used in any matrix expression or on the left hand side of '=', 'ij' or Inject. Inject does not check no information loss. You can also use the construction

```
Real c; .... A.SubMatrix(fr,lr,fc,lc) = c;
```

to set a submatrix equal to a constant.

The following are variants of SubMatrix:

```
A.SymSubMatrix(f,l)           // This assumes fr=fc and lr=lc.\\
A.Rows(f,l)                   // select rows\\
A.Row(f)                       // select single row\\
A.Columns(f,l)                 // select columns\\
A.Column(f)                    // select single column\\
```

Unary operators

```
X = -A;           // change sign of elements
X = A.t();        // transpose
X = A.i();        // inverse (of square matrix A)
X = A.Reverse(); // reverse order of elements of vector
                  // or matrix (not band matrix)
```

Binary operators

```
X = A + B;           // matrix addition
X = A - B;           // matrix subtraction
X = A * B;           // matrix multiplication
X = A.i() * B;       // equation solve (square matrix A)
X = A | B;           // concatenate horizontally (concatenate the rows)
X = A & B;           // concatenate vertically (concatenate the columns)
X = SP(A, B);        // elementwise product of A and B (Schur product)
bool b = A == B;     // test whether A and B are equal
bool b = A != B;     // ! (A == B)
A += B;              // A = A + B;
A -= B;              // A = A - B;
A *= B;              // A = A * B;
A |= B;              // A = A | B;
```

Scalar functions of a matrix

```
int m = A.Nrows();           // number of rows
int n = A.Ncols();          // number of columns
Real r = A.AsScalar();      // value of 1x1 matrix
Real ssq = A.SumSquare();   // sum of squares of elements
Real sav = A.SumAbsoluteValue(); // sum of absolute values
Real s = A.Sum();           // sum of values
Real mav = A.MaximumAbsoluteValue(); // maximum of absolute values
Real norm = A.Norm1();     // maximum of sum of absolute
                           // values of elements of a column
Real norm = A.NormInfinity(); // maximum of sum of absolute
                           // values of elements of a row
Real t = A.Trace();        // trace
LogAndSign ld = A.LogDeterminant(); // log of determinant
bool z = A.IsZero();       // test all elements zero
MatrixType mt = A.Type();  // type of matrix
```

Easy printing and Transparent Coding

Printing: For printing just write `cout << A` and it will print the whole Matrix in the conventional form .

```
Matrix A;
```

```
.....
```

```
cout << setw(10) << setprecision(5) << A;
```

To print several vectors or matrices in columns use a concatenation operator

```
ColumnVector A, B;
```

```
.....
```

```
cout << setw(10) << setprecision(5) << (A | B);
```

Transparent Coding:

```
XprimeXinv=(X.t()*X).i();
```

```
beta=XprimeXinv*X.t()*y;
```

or even

```
beta=((X.t()*X).i())*X.t()*y;
```

Using Splus from C

In the C-Program write

```
system("Splus BATCH graph.s graph.out");
```

this is the graph.s file

```
x_scan("OUT")
```

```
postscript("graph.ps")
```

```
hist(x,nclass=20,main=paste("mean of x is=", (mean(x,3))),xlab="")
```

```
y_scan("DIST")
```

```
abline(v=y)
```

```
dev.off()
```

Tricks Using C++

Calculating Length of an array and using default values

```
int sum(int *vector, int length)
/* This function calculates the sum of length numbers */
int i=0;
int sum=0;
for(i=0; i<length; i++) sum=sum+vector[i];
return sum;
```

```
int sum(int *vector)
/* this function is written to demonstrate a special property of
C++ if the second argument is provided ..by default it calculates
the sum of 52 numbers */
return(sum(vector,52));
```

```
int sum(int *vector)
/* This function calculates the length from the vector and then
calculates the sum */
int i=0,length;
int sum=0;
length=sizeof(vector)/sizeof(*vector);
for(i=0; i<length; i++) sum=sum+vector[i];
return sum;
```

To Keep you Guessing.....

Which of the following are correct ways to assign 32 to position 6 in the array ?

(a) `array[5] = 32;`

(b) `*array+5 = 32;`

(c) `5[array] = 32;`

ANS: All three

Reasoning Pointer arithmetic is interesting, suppose you have a pointer to an integer: `int *pInt`; The size of an integer is 2 bytes. So adding 1 to `pInt`: `pInt + 1` means that the pointer is now 2 bytes farther in to the computer's memory. In the same way

```
array[5] = 32;
```

```
*(array+5) = 32;
```

```
*(5+array) = 32;
```

```
5[array] = 32;
```

CAUTION Unconventional so don't use this