



Computer Integrated Surgery Lab
NSF Engineering Research Center for Computer Integrated
Surgical Systems and Technology

The Johns Hopkins University; Baltimore, Maryland, USA
Massachusetts Institute of Technology; Cambridge, Mass.
Brigham & Women's Hospital; Boston, Mass.
Carnegie-Mellon University; Pittsburgh, Pennsylvania
Shadyside Hospital; Pittsburgh, Pennsylvania

A Crash Course in the Matrix-Vector Library

$$\begin{bmatrix} P_{N,0}(v_0) & \cdots & P_{N,N}(v_0) \\ \vdots & \ddots & \vdots \\ P_{N,0}(v_m) & \cdots & P_{N,N}(v_m) \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_m \end{bmatrix} \cong \begin{bmatrix} y_0 \\ \vdots \\ y_m \end{bmatrix}$$

A Crash Course in the Vecs Library

Ver 1.0

Document CIS0.1

Date 12/01/98

Dr. Russell H Taylor

Director, Center for Computer-Integrated Surgical Systems and Technology

Room 315, New Engineering Building

3400, North Charles Street

Baltimore MD 21218

USA

Web: <http://cisstweb.cs.jhu.edu>

Phone: +01 (410) 516 0740

Fax: +01 (410) 516 5553

Email: rht@cs.jhu.edu

Developmental Software—not certified for medical use. The CISST ERC makes no warranty that the software or other information described in this document is fit for any particular purpose. Users assume full responsibility for any consequences arising from its use.

© 1998,1999 CISST ERC

This document contains proprietary and confidential information of the Engineering Research Center for Computer-Integrated Surgical Systems and Technology (CISST ERC). The contents of this document may not be disclosed to third parties, translated, copied, or duplicated in any form without the express written permission of the CISST ERC Director or other duly authorized representative of the CISST ERC. Queries should be directed to the CISST ERC Director.

1 Introduction

This document is intended to be a short reference for the Matrix-Vector library. It is not by any means a comprehensive guide to every nuance of the library's functionality.

The Matrix-Vector library was designed to allow easy and efficient representations of arbitrary length vectors (arrays) and matrices of arbitrary objects. The library is implemented by a set of C++ templated classes. Instantiations of the templates are available for numerical elements (int, float, double) and Vec3 objects.

The "Vector" data types are

```
BasicVector<class scalar>(int N);           // a vector of N objects of type "scalar"  
intVector(int N);                          // a vector of N integers  
floatVector(int N);                        // a vector of N floats  
doubleVector(int N);                       // a vector of N doubles
```

The "Matrix" data types are

```
BasicMatrix<class scalar>(int M, int N);   // a matrix of N columns of M objects of type "scalar"  
intMatrix(int M, int N);                  // a matrix of M columns of N integers  
floatMatrix(int M, int N);                // a matrix of M columns of N floats  
doubleMatrix(int M, int N);               // a matrix of M columns of N doubles
```

Vectors and matrices use 0-origin indexing. Matrices are stored in column order, for compatibility with Fortran packages and common practice in numerical software. The package overloads indexing and common arithmetic operators such as "+" and "-". Inner products are overloaded on "*". Matrix and vector indexing is bounds checked, although hooks are available to bypass these checks.

The Matrix-Vector package contains hooks for overlaying matrix-vector indexing structures on other data (e.g., Fortran arrays) and has a number of other facilities to support construction of large systems and applications. These facilities are incompletely described in the current document, which provides a basic introduction.

2 Class/Function Lookup Table

SubscriptRange – a support class	Description int i, n SubscriptRange: R, R1, R2
Constructors	Notation: i, n are int
SubscriptRange (I, n)	Defines subscript range [i,...,n]
SubscriptRange (R)	Copy constructor
Members	
R.Min; R.Max	Minimum and maximum index values
R.Length ()	Max-Min+1
R.Includes (i)	Min<=i && i<=Max
R1.Includes (R2)	R1.Includes(R.Min)&&R1.Includes(R1.Max)
Overloaded operators and functions	
R1==R2	R1.Min==R2.Min && R2.Max==R2.Max
R+n	SubscriptRange(Min+n,Max+n)
R-n	SubscriptRange(Min-n,Max-n)
R1=R2	Assignment operator
R = Intersection (R1, R2)	R = SubscriptRange of all indices in both R1&R2
BasicVector<scalar>	Description
intVector floatVector doubleVector Vec3Vector	<scalar> is basic element type scalar s; SubscriptRange I, I1,I2; int i0,i1,n; BasicVector<scalar> V, V1, V2,V3;
Constructors	
BasicVector<scalar> (n)	Create a vector on n scalars
BasicVector<scalar> (V)	Copy constructor
BasicVector<scalar> (MakeCopy, V)	Constructs a copy of V
BasicVector<scalar> (MakeCopy, V, I) ;	
BasicVector<scalar> (MakeReference, V)	Constructs a vector referencing the identical storage as V
BasicVector<scalar> (MakeReference, &s, I)	Creates a vector referring to the storage elements { s[I.Min], s[I.Min+1],...,s[I.Max]}
intVector (n)	Create a vector of n ints
floatVector (n)	Create a vector of n floats
DoubleVector (n)	Create a vector on n doubles
Vec3Vector (n)	Create a vector on n Vec3'

Note: for most user code, the derived forms should be used.

Overloaded operators

V(i)	Produces reference to i'th element of V
V[i]	Produces reference to V(i) with no bounds check
V(I)	Produces BasicVector(MakeReference,V,I)
V1=V2	Copies V2 into V2 (sizes must agree)
V=s	Sets all elements of V to s
V1+V2	Element-wise addition
V1-V2	Element-wise subtraction
V1+s	Adds s to each element of V1
V1-s	Subtracts s from each element of V1
V1*s	Multiplies each element of V1 by s
V+=s	Increments each element of V by s
V-=s	Decrements each element of V by s
V*=s	Scales each element of V by s
V1+=V2	Same as V1=V1+V2
V1-=V2	Same as V1=V1-V2
V1*V2	Vector inner product

Functions

V.Length()	Number of elements in V
V.MaxIndex()	Maximum valid index (V.Length()-1)
V.MinIndex()	Returns 0

BasicMatrix<scalar>

intVector
floatVector
doubleVector

Description

<scalar> is basic element type
scalar s, s1,s2;
SubscriptRange I, J, I1,I2;
int i,j,k,m,n;
BasicVector<scalar> V, V1, V2,V3;
BasicMatrix<scalar> M,M1,M2,M3

Constructors

BasicMatrix<scalar>(m,n)	Create a matrix on n columns of m scalars
BasicMatrix<scalar>(M)	Copy constructor
BasicMatrix<scalar>(MakeCopy,M)	Constructs a copy of M
BasicMatrix<scalar>(MakeCopy,M,I,J);	Copies the block M(I,J)
BasicMatrix<scalar>(MakeReference,M)	Constructs a matrix referencing the identical storage as M
BasicMatrix<scalar>(MakeReference, m,n,&s)	Creates a matrix referring to the storage elements $\begin{bmatrix} s[0] & \cdots & s[m*(n-1)+0] \\ \vdots & \ddots & \vdots \\ s[m-1] & \cdots & s[m*(n-1)+m-1] \end{bmatrix}$

intMatrix(m,n)	Create a matrix of n ints
floatMatrix(m,n)	Create a matrix of n floats
DoubleMatrix(m,n)	Create a matrix on n doubles

Note: for most user code, the derived forms should be used.

Overloaded operators

$M(i, j)$	Reference to the (i, j) 'th element, of M (I.e., to the I 'th element of the j 'th column of M)
$M(i)$	Produces a vector reference to the i 'th column
$M(I, J)$	Produces a matrix referring to the elements $\begin{bmatrix} M(I.Min, J.Min) & \cdots & M(I.Min, J.Max) \\ \vdots & \ddots & \vdots \\ M(I.Max, J.Min) & \cdots & M(I.Max, J.Max) \end{bmatrix}$
$M(I, j)$	Produces a vector referring to the a subvector of the elements of $M(j)$ $[M(I.Min, j) \ \cdots \ M(I.Max, j)]^T$
$M1=M2$	Copy elements of $M2$ into $M1$ (sizes must match)
$M=s$	Sets every element of M to s
$M1+M2$	Element-wise addition
$M1-M2$	Element-wise subtraction
$M*s$	Element-wise scaling
$M*V$	Matrix-vector inner product $M(0)*V(0)+\dots M(n-1)*V(n-1)$
$V*M$	Vector-Matrix inner product $[V*M(0), \dots, V*M(n-1)]^T$
$M1*M2$	Matrix-Matrix inner product

Member Functions

$M.diag()$	Returns a vector containing a copy of the diagonal elements of M
$M.Rows()$	Returns the number of rows of M
$M.Cols()$	Returns the number of columns of M
$M.Size()$	Returns $M.Rows()*M.Cols()$
$M.MaxRow()$	Returns $M.Rows()-1$
$M.MaxCol()$	Returns $M.Cols()-1$
$M.RowRange()$	Returns $SubscriptRange(0, M.MaxRow())$
$M.ColRange()$	Returns $SubscriptRange(0, M.MaxRow())$
$M.Fill(s);$	Sets every element of M to s
$M.FillDiag(s1, s2);$	Sets all diagonal elements of M to $s1$ and all off-diagonal elements to $s2$
$M.FillDiag(V,s)$	Sets $M(k,k)=V(k)$ for $k = 0..\min(M.Rows(), M.Cols())-1$ Sets $M(i,j) = s$ for all $i \neq j$
$M.Transpose()$	Returns a new matrix whose value is the transpose of M

Non-member functions

Inner (M1, M2, M)

Sets M = M1*M2

Inner (V1, M2, V)

Sets V=V1*M

Inner (M1, V2, V)

Sets V=M1*V

MtMProduct (M1, M2, M)

Sets M=M1.Transpose()*M2

MMtProduct (M1, M2, M)

Sets M=M1*M2.Transpose()

Outer (V1, V2, M)

Sets M to

$$\begin{bmatrix} V1[0]*V2[0] & \cdots & V1[n-1]*V2[0] \\ \vdots & \ddots & \vdots \\ V1[0]*V2[n-1] & \cdots & V1[n-1]*V2[n-1] \end{bmatrix}$$

3 Coding Examples

```
#include <ArithVectors.h>
#include <ArithMatrices.h>

intVector AllIndices(SubscriptRange R)
{ intVector ret;
  for (int k=R.Min();k<=R.Max();k++)
    ret(k)=k;
  return ret;
}

main()
{ doubleVector V(100), V1(10), V2(10);
  doubleMatrix M1(10,100);
  doubleMatrix M2(100,10);
  doubleMatrix M(10,10)
  SubscriptRange Z9(0,9);

  V1=AllIndices(Z9); // V1 = [0.0,...,9.0]
  V2=V1*3.0;        // V2 = [0.0,...,27.0]
  V=V1+V2;          // V = [0.0,...,36.0]

  double s = V1*V2; // s = 0+1*3+2*6+...9*27

  M.FillDiag(V,10.0); // sets M to
                        //
                        // 
$$\begin{bmatrix} 0.0 & 10.0 & 10.0 & \dots & 10.0 \\ 10.0 & 4.0 & 10.0 & \dots & 10.0 \\ 10.0 & 10.0 & 8.0 & \vdots & 10.0 \\ 10.0 & 10.0 & 10.0 & \ddots & 10.0 \\ 10.0 & 10.0 & 10.0 & \dots & 36.0 \end{bmatrix}$$

                        //

  for (int i=0;i<100;i++)
    for (int j=0;j<10;j++) M2(i,j)=some_random_function();
  M1=M2.Transpose();

  M = M1*M2; // matrix inner product
  V2 = M2*M1;
  V = V1*M2;

  V = M2(0); // sets V to 0th column of M2
  M(0) = V1; // sets 0th column of M to V1
  M1(0) = -10.; // sets 0th column of M1 to -10.0;

}
```

4 Contact Information

Document Maintainer Russ Taylor (rht@cs.jhu.edu)
Software Maintainer Russ Taylor (rht@cs.jhu.edu)
Repository CIS Lab, JHU

Authors Russ Taylor (rht@cs.jhu.edu)
Acknowledgments