An NDoc Documented Class Library
### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CholeskyDecomposition</strong></td>
<td>Cholesky Decomposition. For a symmetric, positive definite matrix A, the Cholesky decomposition is an lower triangular matrix L so that A = L*L'. If the matrix is not symmetric or positive definite, the constructor returns a partial decomposition and sets an internal flag that may be queried by the isSPD() method.</td>
</tr>
<tr>
<td><strong>EigenvalueDecomposition</strong></td>
<td>Eigenvalues and eigenvectors of a real matrix. If A is symmetric, then A = V<em>D</em>V' where the eigenvalue matrix D is diagonal and the eigenvector matrix V is orthogonal. I.e. A = V.Multiply(D.Multiply(V.Transpose())) and V.Multiply(V.Transpose()) equals the identity matrix. If A is not symmetric, then the eigenvalue matrix D is block diagonal with the real eigenvalues in 1-by-1 blocks and any complex eigenvalues, lambda + i<em>mu, in 2-by-2 blocks, [lambda, mu; -mu, lambda]. The columns of V represent the eigenvectors in the sense that A</em>V = V<em>D, i.e. A.Multiply(V) equals V.Multiply(D). The matrix V may be badly conditioned, or even singular, so the validity of the equation A = V</em>D*Inverse(V) depends upon V.cond().</td>
</tr>
<tr>
<td>GeneralMatrix</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>---</td>
</tr>
<tr>
<td>LUDecomposition</td>
<td></td>
</tr>
<tr>
<td>Maths</td>
<td></td>
</tr>
<tr>
<td>QRDecomposition</td>
<td>QR Decomposition. For an m-by-n matrix $A$ with $m \geq n$, the QR decomposition is an m-by-n orthogonal matrix $Q$ and an n-by-n upper triangular matrix $R$ so that $A = QR$. The QR decomposition always exists, even if the matrix does not have full rank, so the constructor will never fail. The primary use of the QR decomposition is in the least squares solution of nonsquare systems of simultaneous linear equations. This will fail if <code>IsFullRank()</code> returns false.</td>
</tr>
<tr>
<td>SingularValueDecomposition</td>
<td></td>
</tr>
</tbody>
</table>
An NDoc Documented Class Library
Cholesky Decomposition. For a symmetric, positive definite matrix A, the Cholesky decomposition is an lower triangular matrix L so that A = L*L'. If the matrix is not symmetric or positive definite, the constructor returns a partial decomposition and sets an internal flag that may be queried by the isSPD() method.

For a list of all members of this type, see CholeskyDecomposition Members.

See Also

CholeskyDecomposition Members | DotNetMatrix Namespace
An NDoc Documented Class Library
### CholeskyDecomposition Members

#### CholeskyDecomposition overview

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Cholesky algorithm for symmetric and positive definite matrix.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CholeskyDecomposition Constructor</code></td>
<td></td>
</tr>
</tbody>
</table>

#### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SPD</code></td>
<td>Is the matrix symmetric and positive definite?</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals</code> (inherited from <code>Object</code>)</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>GetHashCode</code> (inherited from <code>Object</code>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><code>GetL</code></td>
<td>Return triangular factor.</td>
</tr>
<tr>
<td><code>GetType</code> (inherited from <code>Object</code>)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><code>Solve</code></td>
<td>Solve A*X = B</td>
</tr>
<tr>
<td><code>ToString</code> (inherited from <code>Object</code>)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

#### Private Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isspd</code></td>
<td>Symmetric and positive definite flag. @serial is symmetric and positive definite flag.</td>
</tr>
<tr>
<td><code>L</code></td>
<td>Array for internal storage of</td>
</tr>
<tr>
<td>Decomposition</td>
<td>@serial internal array storage.</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>n</td>
<td>Row and column dimension (square matrix). @serial matrix dimension.</td>
</tr>
</tbody>
</table>

**Explicit Interface Implementations**

`ISerializable.GetObjectData`

**See Also**

[CholeskyDecomposition Class] | [DotNetMatrix Namespace]
An NDoc Documented Class Library
**CholeskyDecomposition Constructor**

Cholesky algorithm for symmetric and positive definite matrix.

```visualbasic
Public Sub New(ByVal Arg As GeneralMatrix)
```

```csharp
public CholeskyDecomposition(GeneralMatrix Arg);
```

**Parameters**

*Arg*
Square, symmetric matrix.

**Return Value**
Structure to access L and isspd flag.

**See Also**

CholeskyDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
The fields of the **CholeskyDecomposition** class are listed below. For a complete list of **CholeskyDecomposition** class members, see the [CholeskyDecomposition Members](#) topic.

### Private Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isspd</td>
<td>Symmetric and positive definite flag. @serial is symmetric and positive definite flag.</td>
</tr>
<tr>
<td>L</td>
<td>Array for internal storage of decomposition. @serial internal array storage.</td>
</tr>
<tr>
<td>n</td>
<td>Row and column dimension (square matrix). @serial matrix dimension.</td>
</tr>
</tbody>
</table>

See Also

[CholeskyDecomposition Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
CholeskyDecomposition.isspd Field

Symmetric and positive definite flag. @serial is symmetric and positive definite flag.

[Visual Basic] Private isspd As Boolean

[C#] private bool isspd;

See Also

CholeskyDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
**CholeskyDecomposition.L Field**

Array for internal storage of decomposition. @serial internal array storage.

[Visual Basic]  
Private L As Double()[]

[C#]  
private double[][] L;

See Also  
[CholeskyDecomposition Class] | [DotNetMatrix Namespace]
An NDoc Documented Class Library
CholeskyDecomposition.n Field

Row and column dimension (square matrix). @serial matrix dimension.

[Visual Basic] Private n As Integer

[C#]
private int n;

See Also

CholeskyDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
The properties of the `CholeskyDecomposition` class are listed below. For a complete list of `CholeskyDecomposition` class members, see the `CholeskyDecomposition Members` topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SPD</code></td>
<td>Is the matrix symmetric and positive definite?</td>
</tr>
</tbody>
</table>

**See Also**

- `CholeskyDecomposition Class` | `DotNetMatrix Namespace`
An NDoc Documented Class Library
CholeskyDecomposition.SPD Property

Is the matrix symmetric and positive definite?

[Visual Basic] Overridable Public ReadOnly Property SPD

[C#]
public virtual bool SPD {get;}

See Also

CholeskyDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
CholeskyDecomposition Methods

The methods of the CholeskyDecomposition class are listed below. For a complete list of CholeskyDecomposition class members, see the CholeskyDecomposition Members topic.

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetL</strong></td>
<td>Return triangular factor.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>Solve</strong></td>
<td>Solve A*X = B</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ISerializable.GetObjectData</strong></td>
<td></td>
</tr>
</tbody>
</table>

See Also

CholeskyDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
**CholeskyDecomposition.GetL Method**

Return triangular factor.

**[Visual Basic]**

```vbnet
Overridable Public Function GetL()
```

**[C#]**

```csharp
public virtual GeneralMatrix GetL();
```

**Return Value**

L

**See Also**

[CholeskyDecomposition Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
### CholeskyDecomposition.Solve Method

Solve \( A*X = B \)

**[Visual Basic]**

```vbnet
Overridable Public Function Solve(ByVal B As GeneralMatrix) As GeneralMatrix
```

**[C#]**

```csharp
public virtual GeneralMatrix Solve(GeneralMatrix B);
```

**Parameters**

*B*

A Matrix with as many rows as \( A \) and any number of columns.

**Return Value**

\( X \) so that \( L*L'X = B \)

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Matrix row dimensions must agree.</td>
</tr>
<tr>
<td>SystemException</td>
<td>Matrix is not symmetric positive definite.</td>
</tr>
</tbody>
</table>

**See Also**

- [CholeskyDecomposition Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
CholeskyDecomposition.ISerializable.GetObjectData Method

[Visual Basic] Sub GetObjectData( ByVal info As SerializationInfo, ByVal context As StreamingContext ) Implements ISerializable.GetObjectData

[C#] void ISerializable.GetObjectData( SerializationInfo info, StreamingContext context );

Implements
ISerializable.GetObjectData

See Also
CholeskyDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
Eigenvalues and eigenvectors of a real matrix. If $A$ is symmetric, then $A = V^*D^*V'$ where the eigenvalue matrix $D$ is diagonal and the eigenvector matrix $V$ is orthogonal. I.e. $A = V.Multiply(D.Multiply(V.Transpose()))$ and $V.Multiply(V.Transpose())$ equals the identity matrix. If $A$ is not symmetric, then the eigenvalue matrix $D$ is block diagonal with the real eigenvalues in 1-by-1 blocks and any complex eigenvalues, $\lambda + i*\mu$, in 2-by-2 blocks, $[\lambda, \mu; -\mu, \lambda]$. The columns of $V$ represent the eigenvectors in the sense that $A^*V = V^*D$, i.e. $A.Multiply(V)$ equals $V.Multiply(D)$. The matrix $V$ may be badly conditioned, or even singular, so the validity of the equation $A = V^*D*Inverse(V)$ depends upon $V.cond()$.

For a list of all members of this type, see EigenvalueDecomposition Members.

System.Object    EigenvalueDecomposition

[Visual Basic]
Public Class EigenvalueDecomposition
Implements ISerializable

[C#]
public class EigenvalueDecomposition : ISerializable

Requirements

Namespace: DotNetMatrix

Assembly: GeneralMatrix (in GeneralMatrix.dll)

See Also

EigenvalueDecomposition Members  |  DotNetMatrix Namespace
An NDoc Documented Class Library
EigenvalueDecomposition Members

EigenvalueDecomposition overview

Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EigenvalueDecomposition Constructor</td>
<td>Check for symmetry, then construct the eigenvalue decomposition</td>
</tr>
</tbody>
</table>

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Return the block diagonal eigenvalue matrix</td>
</tr>
<tr>
<td>ImagEigenvalues</td>
<td>Return the imaginary parts of the eigenvalues</td>
</tr>
<tr>
<td>RealEigenvalues</td>
<td>Return the real parts of the eigenvalues</td>
</tr>
</tbody>
</table>

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>GetV</td>
<td>Return the eigenvector matrix</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

Private Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdivi</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Arrays for internal storage of eigenvalues. @serial internal storage of eigenvalues.</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
</tr>
<tr>
<td>Arrays for internal storage of eigenvalues. @serial internal storage of eigenvalues.</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
</tr>
<tr>
<td>Array for internal storage of nonsymmetric Hessenberg form. @serial internal storage of nonsymmetric Hessenberg form.</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
</tr>
<tr>
<td>Symmetry flag. @serial internal symmetry flag.</td>
<td></td>
</tr>
<tr>
<td>issymmetric</td>
<td></td>
</tr>
<tr>
<td>Row and column dimension (square matrix). @serial matrix dimension.</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
</tr>
<tr>
<td>Working storage for nonsymmetric algorithm. @serial working storage for nonsymmetric algorithm.</td>
<td></td>
</tr>
<tr>
<td>ort</td>
<td></td>
</tr>
<tr>
<td>Array for internal storage of eigenvectors. @serial internal storage of eigenvectors.</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>

**Private Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cdv</td>
<td></td>
</tr>
<tr>
<td>hqr2</td>
<td></td>
</tr>
<tr>
<td>orthes</td>
<td></td>
</tr>
<tr>
<td>tql2</td>
<td></td>
</tr>
<tr>
<td>tred2</td>
<td></td>
</tr>
</tbody>
</table>

**Explicit Interface Implementations**
See Also

EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
**EigenvalueDecomposition Constructor**

Check for symmetry, then construct the eigenvalue decomposition

```visual-basic
Public Sub New( _
    ByVal Arg As GeneralMatrix _
)
```

```csharp
public EigenvalueDecomposition(
    GeneralMatrix Arg
);
```

**Parameters**

*Arg*
Square matrix

**Return Value**
Structure to access D and V.

**See Also**

EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
EigenvalueDecomposition Fields

The fields of the `EigenvalueDecomposition` class are listed below. For a complete list of `EigenvalueDecomposition` class members, see the `EigenvalueDecomposition Members` topic.

### Private Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdivi</td>
<td>Arrays for internal storage of eigenvalues. @serial internal storage of eigenvalues.</td>
</tr>
<tr>
<td>cdivr</td>
<td>Arrays for internal storage of eigenvalues. @serial internal storage of eigenvalues.</td>
</tr>
<tr>
<td>d</td>
<td>Array for internal storage of nonsymmetric Hessenberg form. @serial internal storage of nonsymmetric Hessenberg form.</td>
</tr>
<tr>
<td>e</td>
<td>Symmetry flag. @serial internal symmetry flag.</td>
</tr>
<tr>
<td>H</td>
<td>Row and column dimension (square matrix). @serial matrix dimension.</td>
</tr>
<tr>
<td>issymmetric</td>
<td>Working storage for nonsymmetric algorithm. @serial working storage for nonsymmetric algorithm.</td>
</tr>
<tr>
<td>V</td>
<td>Array for internal storage of eigenvectors. @serial internal storage of eigenvectors.</td>
</tr>
</tbody>
</table>

See Also
EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
EigenvalueDecomposition.cdivi Field

[Visual Basic] Private cdivi As Double

[C#] private double cdivi;

See Also

EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
### EigenvalueDecomposition.cdivr Field

**[Visual Basic]**

```vbnet
Private cdivr As Double
```

**[C#]**

```csharp
private double cdivr;
```

See Also

[ EigenvalueDecomposition Class ] | [ DotNetMatrix Namespace ]
An NDoc Documented Class Library
EigenvalueDecomposition.D Property

Return the block diagonal eigenvalue matrix

[Visual Basic] Overridable Public ReadOnly Property D

[C#]
public virtual GeneralMatrix D {get;}

See Also

EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
Arrays for internal storage of eigenvalues. @serial internal storage of eigenvalues.

[Visual Basic] Private e As Double()

[C#]
private double[] e;

See Also
EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
EigenvalueDecomposition.H Field

Array for internal storage of nonsymmetric Hessenberg form.
@serial internal storage of nonsymmetric Hessenberg form.

[Visual Basic] Private H As Double()[[]]

[C#] private double[][] H;

See Also
EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
**EigenvalueDecomposition.issymmetric Field**

Symmetry flag. @serial internal symmetry flag.

**[Visual Basic]**

```
Private issymmetric As Boolean
```

**[C#]**

```
private bool issymmetric;
```

See Also

[ EigenvalueDecomposition Class ] | [ DotNetMatrix Namespace ]
An NDoc Documented Class Library
**EigenvalueDecomposition.n Field**

Row and column dimension (square matrix). @serial matrix dimension.

**Visual Basic**
```
Private n As Integer
```

**C#**
```
private int n;
```

See Also

[ EigenvalueDecomposition Class ](https://dotnetmatrix.org/)
[ DotNetMatrix Namespace ](https://dotnetmatrix.org/)
An NDoc Documented Class Library
EigenvalueDecomposition.ort Field

Working storage for nonsymmetric algorithm.

@serial working storage for nonsymmetric algorithm.

**[Visual Basic]**

```vbnet
Private ort As Double()
```

**[C#]**

```csharp
private double[] ort;
```

See Also

EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
EigenvalueDecomposition.V Field

Array for internal storage of eigenvectors. @serial internal storage of eigenvectors.

[Visual Basic] Private V As Double[,][]

[C#] private double[][] V;

See Also

EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
### EigenvalueDecomposition Properties

The properties of the `EigenvalueDecomposition` class are listed below. For a complete list of `EigenvalueDecomposition` class members, see the [EigenvalueDecomposition Members](#) topic.

#### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>D</code></td>
<td>Return the block diagonal eigenvalue matrix</td>
</tr>
<tr>
<td><code>ImagEigenvalues</code></td>
<td>Return the imaginary parts of the eigenvalues</td>
</tr>
<tr>
<td><code>RealEigenvalues</code></td>
<td>Return the real parts of the eigenvalues</td>
</tr>
</tbody>
</table>

#### See Also

[ EigenvalueDecomposition Class ](#) | [ DotNetMatrix Namespace ](#)
An NDoc Documented Class Library
EigenvalueDecomposition.ImagEigenvalues Property

Return the imaginary parts of the eigenvalues

[Visual Basic]
Overridable Public ReadOnly Property ImagEigenvalues

[C#]
public virtual double[] ImagEigenvalues {get;}

See Also
EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
EigenvalueDecomposition.RealEigenvalues Property

Return the real parts of the eigenvalues

[Visual Basic] Overridable Public ReadOnly Property RealEigenvalues As Double(

[C#] public virtual double[] RealEigenvalues {get;}

See Also

EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
The methods of the **EigenvalueDecomposition** class are listed below. For a complete list of **EigenvalueDecomposition** class members, see the [EigenvalueDecomposition Members](#) topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <a href="#">Object</a> is equal to the current <a href="#">Object</a>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <a href="#">Type</a> of the current instance.</td>
</tr>
<tr>
<td><strong>GetV</strong></td>
<td>Return the eigenvector matrix</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a <a href="#">String</a> that represents the current <a href="#">Object</a>.</td>
</tr>
</tbody>
</table>

### Private Instance Methods

- `cdiv`
- `hqr2`
- `orthes`
- `tql2`
- `tred2`

### Explicit Interface Implementations

- [ISerializable.GetObjectData](#)

### See Also

[DotNetMatrix Namespace](#)
An NDoc Documented Class Library
EigenvalueDecomposition.cdiv Method

[Visual Basic]
Private Sub cdiv( _
    ByVal xr As Double, _
    ByVal xi As Double, _
    ByVal yr As Double, _
    ByVal yi As Double _
)

[C#]
private void cdiv(
    double xr,
    double xi,
    double yr,
    double yi
);
An NDoc Documented Class Library
EigenvalueDecomposition.GetV Method

Return the eigenvector matrix

[Visual Basic] Overridable Public Function GetV()

[C#]
public virtual GeneralMatrix GetV();

Return Value
V

See Also
EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
EigenvalueDecomposition.hqr2 Method

[Visual Basic] Private Sub hqr2()

[C#]
private void hqr2();

See Also

EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
**EigenvalueDecomposition.orthes Method**

<table>
<thead>
<tr>
<th>[Visual Basic]</th>
<th>Private Sub orthes()</th>
</tr>
</thead>
</table>

| [C#] | private void orthes(); |

See Also

[ EigenvalueDecomposition Class ] | [ DotNetMatrix Namespace ]
An NDoc Documented Class Library
EigenvalueDecomposition.ISerializable.GetObjectData Method

[Visual Basic] Sub GetObjectData( _
    ByVal info As SerializationInfo, _
    ByVal context As StreamingContext _
) Implements _
    ISerializable.GetObjectData

[C#] void ISerializable.GetObjectData(  
    SerializationInfo info,  
    StreamingContext context
);

Implements  
    ISerializable.GetObjectData

See Also  
    EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
EigenvalueDecomposition.tql2 Method

[Visual Basic] Private Sub tql2()

[C#]
private void tql2();

See Also

EigenvalueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
**EigenvalueDecomposition.tred2 Method**

**[Visual Basic]**

```vbnet
Private Sub tred2()
```

**[C#]**

```csharp
private void tred2();
```

See Also

- [EigenvalueDecomposition Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
GeneralMatrix Class

For a list of all members of this type, see GeneralMatrix Members.

System.Object   GeneralMatrix

|Visual Basic|
Public Class GeneralMatrix
Implements ICloneable, ISerializable, IDisposable

|C#|
public class GeneralMatrix : ICloneable, ISerializable, IDisposable

Requirements

Namespace: DotNetMatrix

Assembly: GeneralMatrix (in GeneralMatrix.dll)

See Also

GeneralMatrix Members | DotNetMatrix Namespace
An NDoc Documented Class Library
### GeneralMatrix Members

#### GeneralMatrix overview

### Public Static (Shared) Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>Construct a matrix from a copy of a 2-D array.</td>
</tr>
<tr>
<td>Identity</td>
<td>Generate identity matrix</td>
</tr>
<tr>
<td>Random</td>
<td>Generate matrix with random elements</td>
</tr>
</tbody>
</table>

### Public Static (Shared) Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition Operator</td>
<td>Addition of matrices</td>
</tr>
<tr>
<td>Multiplication Operator</td>
<td>Multiplication of matrices</td>
</tr>
<tr>
<td>Subtraction Operator</td>
<td>Subtraction of matrices</td>
</tr>
</tbody>
</table>

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeneralMatrix</td>
<td>Overloaded. Initializes a new instance of the GeneralMatrix class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>Access the internal two-dimensional array.</td>
</tr>
<tr>
<td>ArrayCopy</td>
<td>Copy the internal two-dimensional array.</td>
</tr>
<tr>
<td>ColumnDimension</td>
<td>Get column dimension.</td>
</tr>
<tr>
<td>ColumnPackedCopy</td>
<td>Make a one-dimensional column packed copy of the internal array.</td>
</tr>
<tr>
<td>RowDimension</td>
<td>Get row dimension.</td>
</tr>
<tr>
<td>RowPackedCopy</td>
<td>Make a one-dimensional row</td>
</tr>
</tbody>
</table>
packed copy of the internal array.

**Public Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>C = A + B</td>
</tr>
<tr>
<td>AddEquals</td>
<td>A = A + B</td>
</tr>
<tr>
<td>ArrayLeftDivide</td>
<td>Element-by-element left division, C = A \ B</td>
</tr>
<tr>
<td>ArrayLeftDivideEquals</td>
<td>Element-by-element left division in place, A = A \ B</td>
</tr>
<tr>
<td>ArrayMultiply</td>
<td>Element-by-element multiplication, C = A.*B</td>
</tr>
<tr>
<td>ArrayMultiplyEquals</td>
<td>Element-by-element multiplication in place, A = A.*B</td>
</tr>
<tr>
<td>ArrayRightDivide</td>
<td>Element-by-element right division, C = A ./ B</td>
</tr>
<tr>
<td>ArrayRightDivideEquals</td>
<td>Element-by-element right division in place, A = A ./ B</td>
</tr>
<tr>
<td>chol</td>
<td>Cholesky Decomposition</td>
</tr>
<tr>
<td>Clone</td>
<td>Clone the GeneralMatrix object.</td>
</tr>
<tr>
<td>Condition</td>
<td>Matrix condition (2 norm)</td>
</tr>
<tr>
<td>Copy</td>
<td>Make a deep copy of a matrix</td>
</tr>
<tr>
<td>Determinant</td>
<td>GeneralMatrix determinant</td>
</tr>
<tr>
<td>Dispose</td>
<td>Overloaded. Do not make this method virtual. A derived class should not be able to override this method.</td>
</tr>
<tr>
<td>Eigen</td>
<td>Eigenvalue Decomposition</td>
</tr>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Method</strong></td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>GetElement</strong></td>
<td>Get a single element.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetMatrix</strong></td>
<td>Overloaded. Get a submatrix.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>(inherited from Object) Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>Inverse</strong></td>
<td>Matrix inverse or pseudoinverse</td>
</tr>
<tr>
<td><strong>LUD</strong></td>
<td>LU Decomposition</td>
</tr>
<tr>
<td><strong>Multiply</strong></td>
<td>Overloaded. Linear algebraic matrix multiplication, A * B</td>
</tr>
<tr>
<td><strong>MultiplyEquals</strong></td>
<td>Multiply a matrix by a scalar in place, A = s*A</td>
</tr>
<tr>
<td><strong>Norm1</strong></td>
<td>One norm</td>
</tr>
<tr>
<td><strong>Norm2</strong></td>
<td>Two norm</td>
</tr>
<tr>
<td><strong>NormF</strong></td>
<td>Frobenius norm</td>
</tr>
<tr>
<td><strong>NormInf</strong></td>
<td>Infinity norm</td>
</tr>
<tr>
<td><strong>QRD</strong></td>
<td>QR Decomposition</td>
</tr>
<tr>
<td><strong>Rank</strong></td>
<td>GeneralMatrix rank</td>
</tr>
<tr>
<td><strong>SetElement</strong></td>
<td>Set a single element.</td>
</tr>
<tr>
<td><strong>SetMatrix</strong></td>
<td>Overloaded. Set a submatrix.</td>
</tr>
<tr>
<td><strong>Solve</strong></td>
<td>Solve A*X = B</td>
</tr>
<tr>
<td><strong>SolveTranspose</strong></td>
<td>Solve X*A = B, which is also A'*X' = B'</td>
</tr>
<tr>
<td><strong>Subtract</strong></td>
<td>C = A - B</td>
</tr>
<tr>
<td><strong>SubtractEquals</strong></td>
<td>A = A - B</td>
</tr>
<tr>
<td><strong>SVD</strong></td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Trace</strong></td>
<td>Matrix trace.</td>
</tr>
<tr>
<td><strong>Transpose</strong></td>
<td>Matrix transpose.</td>
</tr>
<tr>
<td><strong>UnaryMinus</strong></td>
<td>Unary minus</td>
</tr>
</tbody>
</table>

**Private Instance Fields**

| **A** | Array for internal storage of elements. @serial internal array storage. |
| **m** | Row and column dimensions. @serial row dimension. @serial column dimension. |
| **n** | Row and column dimensions. @serial row dimension. @serial column dimension. |

**Private Instance Methods**

| **CheckMatrixDimensions** | Check if size(A) == size(B) * |
| **Dispose** | Overloaded. Dispose(bool disposing) executes in two distinct scenarios. If disposing equals true, the method has been called directly or indirectly by a user's code. Managed and unmanaged resources can be disposed. If disposing equals false, the method has been called by the runtime from inside the finalizer and you should not reference other objects. Only unmanaged resources can be disposed. |
Explicit Interface Implementations

| `ISerializable.GetObjectData` | A method called when serializing this class |

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix Constructor

Construct an m-by-n matrix of zeros.

Overload List

Construct a matrix from a one-dimensional packed array

```
public GeneralMatrix(double[],int);
```

Construct a matrix from a 2-D array.

```
public GeneralMatrix(double[][]);
```

Construct a matrix quickly without checking arguments.

```
public GeneralMatrix(double[][],int,int);
```

Construct an m-by-n matrix of zeros.

```
public GeneralMatrix(int,int);
```

Construct an m-by-n constant matrix.

```
public GeneralMatrix(int,int,double);
```

See Also

GeneralMatrix Class  |  DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix Constructor (Int32, Int32)

Construct an m-by-n matrix of zeros.

[Visual Basic] Overloads Public Sub New( ByVal m As Integer, ByVal n As Integer )

[C#]
public GeneralMatrix( int m, int n );

Parameters

m
  Number of rows.

n
  Number of columns.

See Also

GeneralMatrix Class | DotNetMatrix Namespace | GeneralMatrix Constructor Overload List
An NDoc Documented Class Library
GeneralMatrix Constructor (Int32, Int32, Double)

Construct an m-by-n constant matrix.

[Visual Basic] Overloads Public Sub New( _
  ByVal m As Integer, _
  ByVal n As Integer, _
  ByVal s As Double _
)

[C#]
public GeneralMatrix(
  int m,
  int n,
  double s
);

Parameters

m
  Number of rows.

n
  Number of columns.

s
  Fill the matrix with this scalar value.

See Also

GeneralMatrix Class | DotNetMatrix Namespace | GeneralMatrix Constructor Overload List
An NDoc Documented Class Library
GeneralMatrix Constructor (Double[][])

Construct a matrix from a 2-D array.

[Visual Basic] Overloads Public Sub New( _
    ByVal A As Double[][] _
)

[C#]
public GeneralMatrix(
    double[][] A
);

Parameters

A
Two-dimensional array of doubles.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>All rows must have the same length</td>
</tr>
</tbody>
</table>

See Also

GeneralMatrix Class | DotNetMatrix Namespace | GeneralMatrix Constructor Overload List | Create
An NDoc Documented Class Library
GeneralMatrix Constructor (Double[][], Int32, Int32)

Construct a matrix quickly without checking arguments.

[Visual Basic] Overloads Public Sub New ( _
    ByVal A As Double[][]), _
    ByVal m As Integer, _
    ByVal n As Integer _
)

[C#]
public GeneralMatrix (  
double[][] A,
    int m,
    int n
);

Parameters

A
   Two-dimensional array of doubles.

m
   Number of rows.

n
   Number of columns.

See Also

GeneralMatrix Class | DotNetMatrix Namespace | GeneralMatrix Constructor Overload List
An NDoc Documented Class Library
GeneralMatrix Constructor (Double[], Int32)

Construct a matrix from a one-dimensional packed array

[Visual Basic]
Overloads Public Sub New( _
    ByVal vals As Double(), _
    ByVal m As Integer _
)

[C#]
public GeneralMatrix(  
    double[] vals,  
    int m
);

Parameters
vals
One-dimensional array of doubles, packed by columns (ala Fortran).

m
Number of rows.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Array length must be a multiple of m.</td>
</tr>
</tbody>
</table>

See Also

GeneralMatrix Class | DotNetMatrix Namespace | GeneralMatrix Constructor Overload List
An NDoc Documented Class Library
GeneralMatrix Fields

The fields of the GeneralMatrix class are listed below. For a complete list of GeneralMatrix class members, see the GeneralMatrix Members topic.

Private Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Array for internal storage of elements. @serial internal array storage.</td>
</tr>
<tr>
<td>m</td>
<td>Row and column dimensions. @serial row dimension. @serial column dimension.</td>
</tr>
<tr>
<td>n</td>
<td>Row and column dimensions. @serial row dimension. @serial column dimension.</td>
</tr>
</tbody>
</table>

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.A Field

Array for internal storage of elements. @serial internal array storage.

[Visual Basic] Private A As Double()[]

[C#] private double[][] A;

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.m Field

Row and column dimensions. @serial row dimension. @serial column dimension.

[Visual Basic] Private m As Integer

[C#]
private int m;

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.n Field

Row and column dimensions. @serial row dimension. @serial column dimension.

[Visual Basic] Private n As Integer

[C#]
private int n;

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
The properties of the `GeneralMatrix` class are listed below. For a complete list of `GeneralMatrix` class members, see the `GeneralMatrix Members` topic.

**Public Instance Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Array</code></td>
<td>Access the internal two-dimensional array.</td>
</tr>
<tr>
<td><code>ArrayCopy</code></td>
<td>Copy the internal two-dimensional array.</td>
</tr>
<tr>
<td><code>ColumnDimension</code></td>
<td>Get column dimension.</td>
</tr>
<tr>
<td><code>ColumnPackedCopy</code></td>
<td>Make a one-dimensional column packed copy of the internal array.</td>
</tr>
<tr>
<td><code>RowDimension</code></td>
<td>Get row dimension.</td>
</tr>
<tr>
<td><code>RowPackedCopy</code></td>
<td>Make a one-dimensional row packed copy of the internal array.</td>
</tr>
</tbody>
</table>

See Also

- [GeneralMatrix Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
GeneralMatrix.Array Property

Access the internal two-dimensional array.

[Visual Basic] Overridable Public ReadOnly Property Array As ()

[C#]
public virtual double[][] Array {get;}

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
Copy the internal two-dimensional array.

[Visual Basic] Overridable Public ReadOnly Property

[C#] public virtual double[][] ArrayCopy {get;}

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.ColumnDimension Property

Get column dimension.

[Visual Basic] Overridable Public ReadOnly Property ColumnDimension As Integer

[C#]
public virtual int ColumnDimension {get;}

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.ColumnPackedCopy Property

Make a one-dimensional column packed copy of the internal array.

[Visual Basic] Overridable Public Readonly Property ColumnPackedCopy

[C#] public virtual double[] ColumnPackedCopy {get;}

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.RowDimension Property

Get row dimension.

[Visual Basic] Overridable Public ReadOnly Property RowDimension

[C#] public virtual int RowDimension {get;}

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.RowPackedCopy Property

Make a one-dimensional row packed copy of the internal array.

[Visual Basic] Overridable Public ReadOnly Property RowPackedCopy

[C#]
public virtual double[] RowPackedCopy {get;}

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
**GeneralMatrix Methods**

The methods of the `GeneralMatrix` class are listed below. For a complete list of `GeneralMatrix` class members, see the `GeneralMatrix Members` topic.

<table>
<thead>
<tr>
<th>Public Static (Shared) Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create</strong></td>
<td>Construct a matrix from a copy of a 2-D array.</td>
</tr>
<tr>
<td><strong>Identity</strong></td>
<td>Generate identity matrix</td>
</tr>
<tr>
<td><strong>Random</strong></td>
<td>Generate matrix with random elements</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Public Instance Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>C = A + B</td>
</tr>
<tr>
<td><strong>AddEquals</strong></td>
<td>A = A + B</td>
</tr>
<tr>
<td><strong>ArrayLeftDivide</strong></td>
<td>Element-by-element left division, C = A ./ B</td>
</tr>
<tr>
<td><strong>ArrayLeftDivideEquals</strong></td>
<td>Element-by-element left division in place, A = A ./ B</td>
</tr>
<tr>
<td><strong>ArrayMultiply</strong></td>
<td>Element-by-element multiplication, C = A .* B</td>
</tr>
<tr>
<td><strong>ArrayMultiplyEquals</strong></td>
<td>Element-by-element multiplication in place, A = A .* B</td>
</tr>
<tr>
<td><strong>ArrayRightDivide</strong></td>
<td>Element-by-element right division, C = A ./ B</td>
</tr>
<tr>
<td><strong>ArrayRightDivideEquals</strong></td>
<td>Element-by-element right division in place, A = A ./ B</td>
</tr>
<tr>
<td><strong>chol</strong></td>
<td>Cholesky Decomposition</td>
</tr>
<tr>
<td><strong>Clone</strong></td>
<td>Clone the GeneralMatrix object.</td>
</tr>
<tr>
<td><strong>Condition</strong></td>
<td>Matrix condition (2 norm)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Copy</strong></td>
<td>Make a deep copy of a matrix</td>
</tr>
<tr>
<td><strong>Determinant</strong></td>
<td>GeneralMatrix determinant</td>
</tr>
<tr>
<td><strong>Dispose</strong></td>
<td>Overloaded. Do not make this method virtual. A derived class should not be able to override this method.</td>
</tr>
<tr>
<td><strong>Eigen</strong></td>
<td>Eigenvalue Decomposition</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetElement</strong></td>
<td>Get a single element.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetMatrix</strong></td>
<td>Overloaded. Get a submatrix.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>Inverse</strong></td>
<td>Matrix inverse or pseudoinverse</td>
</tr>
<tr>
<td><strong>LUD</strong></td>
<td>LU Decomposition</td>
</tr>
<tr>
<td><strong>Multiply</strong></td>
<td>Overloaded. Linear algebraic matrix multiplication, A * B</td>
</tr>
<tr>
<td><strong>MultiplyEquals</strong></td>
<td>Multiply a matrix by a scalar in place, A = s*A</td>
</tr>
<tr>
<td><strong>Norm1</strong></td>
<td>One norm</td>
</tr>
<tr>
<td><strong>Norm2</strong></td>
<td>Two norm</td>
</tr>
<tr>
<td><strong>NormF</strong></td>
<td>Frobenius norm</td>
</tr>
<tr>
<td><strong>NormInf</strong></td>
<td>Infinity norm</td>
</tr>
<tr>
<td><strong>QRD</strong></td>
<td>QR Decomposition</td>
</tr>
<tr>
<td><strong>Rank</strong></td>
<td>GeneralMatrix rank</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SetElement</strong></td>
<td>Set a single element.</td>
</tr>
<tr>
<td><strong>SetMatrix</strong></td>
<td>Overloaded. Set a submatrix.</td>
</tr>
<tr>
<td><strong>Solve</strong></td>
<td>Solve $A \times X = B$.</td>
</tr>
<tr>
<td><strong>SolveTranspose</strong></td>
<td>Solve $X \times A = B$, which is also $A^*X' = B'$.</td>
</tr>
<tr>
<td><strong>Subtract</strong></td>
<td>$C = A - B$</td>
</tr>
<tr>
<td><strong>SubtractEquals</strong></td>
<td>$A = A - B$</td>
</tr>
<tr>
<td><strong>SVD</strong></td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td><strong>Trace</strong></td>
<td>Matrix trace.</td>
</tr>
<tr>
<td><strong>Transpose</strong></td>
<td>Matrix transpose.</td>
</tr>
<tr>
<td><strong>UnaryMinus</strong></td>
<td>Unary minus</td>
</tr>
</tbody>
</table>

**Private Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CheckMatrixDimensions</strong></td>
<td>Check if size(A) == size(B) *</td>
</tr>
<tr>
<td><strong>Dispose</strong></td>
<td>Overloaded. Dispose(bool disposing) executes in two distinct scenarios. If disposing equals true, the method has been called directly or indirectly by a user's code. Managed and unmanaged resources can be disposed. If disposing equals false, the method has been called by the runtime from inside the finalizer and you should not reference other objects. Only unmanaged resources can be disposed.</td>
</tr>
</tbody>
</table>

**Explicit Interface Implementations**
| `Serializable.GetObjectData` | A method called when serializing this class |

See Also

[GeneralMatrix Class] | [DotNetMatrix Namespace]
An NDoc Documented Class Library
GeneralMatrix/Add Method

\[ C = A + B \]

### Visual Basic

```vbnet
Overridable Public Function Add(ByVal B As GeneralMatrix) As GeneralMatrix
```

### C#

```csharp
public virtual GeneralMatrix Add(GeneralMatrix B);
```

**Parameters**

- \( B \) another matrix

**Return Value**

\( A + B \)

**See Also**

- [GeneralMatrix Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
A = A + B

[Visual Basic] Overridable Public Function AddEquals(
    ByVal B As GeneralMatrix
) As GeneralMatrix

[C#]
public virtual GeneralMatrix AddEquals(
    GeneralMatrix B
);

Parameters

B
    another matrix

Return Value

A + B

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.ArrayLeftDivide Method

Element-by-element left division, \( C = A \div B \)

[Visual Basic] Overridable Public Function ArrayLeftDivide(ByVal B As GeneralMatrix) As GeneralMatrix

[C#]
public virtual GeneralMatrix ArrayLeftDivide(GeneralMatrix B)

Parameters

\( B \)
another matrix

Return Value

\( A \div B \)

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
Element-by-element left division in place, A = A \ B

Parameters

B

another matrix

Return Value

A \ B

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
Element-by-element multiplication, $C = A \cdot B$

**Parameters**

- $B$
  another matrix

**Return Value**

- $A \cdot B$

**See Also**

- [GeneralMatrix Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
Element-by-element multiplication in place, \( A = A \cdot B \)

**[Visual Basic]**

```vbnet
Overridable Public Function ArrayMultiplyEquals(
    ByVal B As GeneralMatrix
) As GeneralMatrix
```

**[C#]**

```csharp
public virtual GeneralMatrix ArrayMultiplyEquals(
    GeneralMatrix B
);
```

**Parameters**

- **B**
  - another matrix

**Return Value**

- \( A \cdot B \)

**See Also**

- [GeneralMatrix Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
Element-by-element right division, $C = A./B$

**Parameters**

- $B$  
  another matrix

**Return Value**

$A./B$

**See Also**

- GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.ArrayRightDivideEquals Method

Element-by-element right division in place, \( A = A./B \)

[Visual Basic] Overridable Public Function ArrayRightDivideEquals(ByVal B As GeneralMatrix) As GeneralMatrix

[C#]
public virtual GeneralMatrix ArrayRightDivideEquals(GeneralMatrix B);

Parameters

- \( B \)
  another matrix

Return Value

- \( A./B \)

See Also

- GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.CheckMatrixDimensions Method

Check if size(A) == size(B) *

[Visual Basic] Private Sub CheckMatrixDimensions(
   ByVal B As GeneralMatrix
)

[C#] private void CheckMatrixDimensions(
   GeneralMatrix B
);

See Also
   GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.chol Method

Cholesky Decomposition

[Visual Basic] Overridable Public Function chol()

[C#] public virtual CholeskyDecomposition chol();

Return Value

CholeskyDecomposition

See Also

GeneralMatrix Class | DotNetMatrix Namespace | CholeskyDecomposition
An NDoc Documented Class Library
GeneralMatrix.Clone Method

Clone the GeneralMatrix object.

[Visual Basic] NotOverridable Public Function Implements _
ICloneable.Clone

[C#] public object Clone();

Implements
ICloneable.Clone

See Also
GeneralMatrix Class | DotNetMatrix Namespace
GeneralMatrix.Condition Method

Matrix condition (2 norm)

[Visual Basic] Overridable Public Function Condition()

[C#]
public virtual double Condition();

Return Value
ratio of largest to smallest singular value.

See Also
GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.Copy Method

Make a deep copy of a matrix

[Visual Basic] Overridable Public Function Copy()

[C#]
public virtual GeneralMatrix Copy();

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.Create Method

Construct a matrix from a copy of a 2-D array.

[Visual Basic] Public Shared Function Create(
    ByVal A As Double[][]
) As GeneralMatrix

[C#]
public static GeneralMatrix Create(
    double[][] A
);

Parameters

A
    Two-dimensional array of doubles.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>All rows must have the same length</td>
</tr>
</tbody>
</table>

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.Determinant Method

GeneralMatrix determinant

[Visual Basic] Overridable Public Function Determinant()

[C#]
public virtual double Determinant();

Return Value
determinant

See Also
GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.Dispose Method

Do not make this method virtual. A derived class should not be able to override this method.

Overload List

Do not make this method virtual. A derived class should not be able to override this method.

public void Dispose();

Dispose(bool disposing) executes in two distinct scenarios. If disposing equals true, the method has been called directly or indirectly by a user’s code. Managed and unmanaged resources can be disposed. If disposing equals false, the method has been called by the runtime from inside the finalizer and you should not reference other objects. Only unmanaged resources can be disposed.

private void Dispose(bool);

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.Dispose Method ()

Do not make this method virtual. A derived class should not be able to override this method.

[Visual Basic] NotOverridable Overloads Public Implements _
    IDisposable.Dispose

[C#]
public void Dispose();

Implements
    IDisposable.Dispose

See Also
    GeneralMatrix Class | DotNetMatrix Namespace | GeneralMatrix.Dispose Overload List
An NDoc Documented Class Library
Dispose(bool disposing) executes in two distinct scenarios. If disposing equals true, the method has been called directly or indirectly by a user's code. Managed and unmanaged resources can be disposed. If disposing equals false, the method has been called by the runtime from inside the finalizer and you should not reference other objects. Only unmanaged resources can be disposed.

### Parameters

- **disposing**

### See Also

- [GeneralMatrix Class](#) | [DotNetMatrix Namespace](#) | [GeneralMatrix.Dispose Overload List](#)
An NDoc Documented Class Library
GeneralMatrix.Eigen Method

Eigenvalue Decomposition

[Visual Basic] Overridable Public Function Eigen()

[C#] public virtual EigenvalueDecomposition Eigen();

Return Value

EigenvalueDecomposition

See Also

GeneralMatrix Class | DotNetMatrix Namespace | EigenvalueDecomposition
An NDoc Documented Class Library
GeneralMatrix.GetElement Method

Get a single element.

[Visual Basic] Overridable Public Function GetElement( _
    ByVal i As Integer, _
    ByVal j As Integer _
) As Double

[C#]
public virtual double GetElement(  
    int i,  
    int j  
);  

Parameters

i
Row index.

j
Column index.

Return Value

A(i,j)

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexOutOfRangeException</td>
<td></td>
</tr>
</tbody>
</table>

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.GetMatrix Method

Get a submatrix.

Overload List

Get a submatrix.

public virtual GeneralMatrix GetMatrix(int,int,int,int);

Get a submatrix.

public virtual GeneralMatrix GetMatrix(int,int,int[]);

Get a submatrix.

public virtual GeneralMatrix GetMatrix(int[],int,int);

Get a submatrix.

public virtual GeneralMatrix GetMatrix(int[],int[]);

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
Get a submatrix.

```visual-basic
Overridable Overloads Public Function GetMatrix(
    ByVal i0 As Integer, _
    ByVal i1 As Integer, _
    ByVal j0 As Integer, _
    ByVal j1 As Integer _
) As GeneralMatrix
```

```csharp
public virtual GeneralMatrix GetMatrix(
    int i0,
    int i1,
    int j0,
    int j1
);
```

**Parameters**

- **i0**
  - Initial row index

- **i1**
  - Final row index

- **j0**
  - Initial column index

- **j1**
  - Final column index

**Return Value**

\[ A(i0:i1,j0:j1) \]

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
</table>


| IndexOutOfRangeException | Submatrix indices |

See Also

GeneralMatrix Class | DotNetMatrix Namespace | GeneralMatrix.GetMatrix Overload List
An NDoc Documented Class Library
Get a submatrix.

**[Visual Basic]**
```vbnet
Overridable Overloads Public Function GetMatrix(
    ByVal i0 As Integer, _
    ByVal i1 As Integer, _
    ByVal c As Integer() _
) As GeneralMatrix
```

**[C#]**
```csharp
public virtual GeneralMatrix GetMatrix(
    int i0,
    int i1,
    int[] c
);
```

**Parameters**
- **i0**
  Initial row index
- **i1**
  Final row index
- **c**
  Array of column indices.

**Return Value**

$A(i0:i1,c(\cdot))$

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexOutOfRangeException</td>
<td>Submatrix indices</td>
</tr>
</tbody>
</table>

**See Also**

GeneralMatrix Class | DotNetMatrix Namespace |
GeneralMatrix.GetMatrix Overload List
An NDoc Documented Class Library
Get a submatrix.

[Visual Basic] Overrideable Overloads Public Function GetMatrix(_
    ByVal r As Integer(), _
    ByVal j0 As Integer, _
    ByVal j1 As Integer _
) As GeneralMatrix

[C#]
public virtual GeneralMatrix GetMatrix(
    int[] r,
    int j0,
    int j1
);

Parameters

  r
  Array of row indices.

  j0
  Initial column index

  j1
  Final column index

Return Value

  A(r(:,j0:j1))

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexOutOfRangeException</td>
<td>Submatrix indices</td>
</tr>
</tbody>
</table>

See Also

  GeneralMatrix Class | DotNetMatrix Namespace |
GeneralMatrix.GetMatrix Overload List
An NDoc Documented Class Library
Get a submatrix.

**Visual Basic**

```vbnet
Overridable Overloads Public Function GetMatrix("
    ByVal r As Integer(), 
    ByVal c As Integer() 
) As GeneralMatrix
```

**C#**

```csharp
public virtual GeneralMatrix GetMatrix(
    int[] r,
    int[] c
);
```

### Parameters

- `r`  
  Array of row indices.

- `c`  
  Array of column indices.

### Return Value

- `A(r:), c(:)`

### Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IndexOutOfRangeException</code></td>
<td>Submatrix indices</td>
</tr>
</tbody>
</table>

### See Also

- [GeneralMatrix Class](#) | [DotNetMatrix Namespace](#) |
- [GeneralMatrix.GetMatrix Overload List](#)
An NDoc Documented Class Library
GeneralMatrix.Identity Method

Generate identity matrix

[Visual Basic] Public Shared Function Identity(  
   ByVal m As Integer, _  
   ByVal n As Integer _  
) As GeneralMatrix

[C#]
public static GeneralMatrix Identity(  
   int m,  
   int n  
);

Parameters

   m
   Number of rows.

   n
   Number of columns.

Return Value

   An m-by-n matrix with ones on the diagonal and zeros elsewhere.

See Also

   GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.Inverse Method

Matrix inverse or pseudoinverse

[Visual Basic] Overridable Public Function Inverse()

[C#]
public virtual GeneralMatrix Inverse();

Return Value

inverse(A) if A is square, pseudoinverse otherwise.

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
**GeneralMatrix.LUD Method**

LU Decomposition

[Visual Basic] **Overridable Public Function LUD()**

[C#]
```csharp
public virtual LUDecomposition LUD();
```

**Return Value**

LUDecomposition

**See Also**

[GeneralMatrix Class](#) | [DotNetMatrix Namespace](#) | [LUDecomposition](#)
An NDoc Documented Class Library
**GeneralMatrix.Multiply Method**

Linear algebraic matrix multiplication, A * B

**Overload List**

Linear algebraic matrix multiplication, A * B

```csharp
public virtual GeneralMatrix Multiply(GeneralMatrix);  
```

Multiply a matrix by a scalar, C = s*A

```csharp
public virtual GeneralMatrix Multiply(double);  
```

**See Also**

[GeneralMatrix Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
GeneralMatrix.Multiply Method (GeneralMatrix)

Linear algebraic matrix multiplication, A * B

[Visual Basic] Overridable Overloads Public Function Multiply( _
    ByVal B As GeneralMatrix _
) As GeneralMatrix

[C#]
public virtual GeneralMatrix Multiply( _
    GeneralMatrix B
);

Parameters

B
another matrix

Return Value

Matrix product, A * B

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Matrix inner dimensions must agree.</td>
</tr>
</tbody>
</table>

See Also

GeneralMatrix Class | DotNetMatrix Namespace | GeneralMatrix.Multiply Overload List
An NDoc Documented Class Library
Multiply a matrix by a scalar, $C = s \times A$

**Parameters**

$s$

Scalar

**Return Value**

$s \times A$

**See Also**


---

[Visual Basic] `Overridable Overloads Public Function Multiply(ByVal s As Double) As GeneralMatrix`
An NDoc Documented Class Library
GeneralMatrix.MultiplyEquals Method

Multiply a matrix by a scalar in place, \( A = s \cdot A \)

**[Visual Basic]**

```vbnet
Overridable Public Function MultiplyEquals(
    ByVal s As Double
) As GeneralMatrix
```

**[C#]**

```csharp
public virtual GeneralMatrix MultiplyEquals(
    double s
);
```

**Parameters**

- \( s \)  
  scalar

**Return Value**

replace \( A \) by \( s \cdot A \)

**See Also**

[GeneralMatrix Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
GeneralMatrix.Norm1 Method

One norm

[Visual Basic] Overridable Public Function Norm1()

[C#]
public virtual double Norm1();

Return Value
maximum column sum.

See Also
GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.Norm2 Method

Two norm

[Visual Basic] Overridable Public Function Norm2()

[C#] public virtual double Norm2();

Return Value

maximum singular value.

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.NormF Method

Frobenius norm

[Visual Basic] Overridable Public Function NormF()

[C#]
public virtual double NormF();

Return Value

sqrt of sum of squares of all elements.

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.NormInf Method

Infinity norm

[Visual Basic] Overridable Public Function NormInf()

[C#]
public virtual double NormInf();

Return Value
    maximum row sum.

See Also
    GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.QRD Method

QR Decomposition

[Visual Basic] Overridable Public Function QRD()

[C#]
public virtual QRDecomposition QRD();

Return Value

QRDecomposition

See Also

GeneralMatrix Class | DotNetMatrix Namespace | QRDecomposition
An NDoc Documented Class Library
GeneraGeneralMatrix.Random Method

Generate matrix with random elements

**[Visual Basic]**
*Public Shared Function Random*(
  ByVal m As *Integer*,
  ByVal n As *Integer* )
  As *GeneralMatrix*

**[C#]**
*public static* *GeneralMatrix* *Random*(
  *int* *m*,
  *int* *n* )
);

**Parameters**

- **m**
  Number of rows.

- **n**
  Number of columns.

**Return Value**

An m-by-n matrix with uniformly distributed random elements.

**See Also**

*GeneralMatrix Class* | *DotNetMatrix Namespace*
An NDoc Documented Class Library
GeneralMatrix.Rank Method

GeneralMatrix rank

[Visual Basic] Overridable Public Function Rank()

[C#]
public virtual int Rank();

Return Value

effective numerical rank, obtained from SVD.

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.SetElement Method

Set a single element.

[Visual Basic] Overridable Public Sub SetElement(  ByVal i As Integer, _  ByVal j As Integer, _  ByVal s As Double _)  

[C#]  
public virtual void SetElement(  int i,  int j,  double s  );  

Parameters

i  
Row index.

j  
Column index.

s  
A(i,j).

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexOutOfRangeException</td>
<td></td>
</tr>
</tbody>
</table>

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.SetMatrix Method

Set a submatrix.

Overload List

Set a submatrix.

```csharp
public virtual void SetMatrix(int,int,int,int,GeneralMatrix);
```

Set a submatrix.

```csharp
public virtual void SetMatrix(int,int,int[],GeneralMatrix);
```

Set a submatrix.

```csharp
public virtual void SetMatrix(int[],int,int,GeneralMatrix);
```

Set a submatrix.

```csharp
public virtual void SetMatrix(int[],int[],GeneralMatrix);
```

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
Set a submatrix.

**[Visual Basic]**

```vbnet
Overridable Overloads Public Sub SetMatrix(
    ByVal i0 As Integer, _
    ByVal i1 As Integer, _
    ByVal j0 As Integer, _
    ByVal j1 As Integer, _
    ByVal X As GeneralMatrix)
)
```

**[C#]**

```csharp
public virtual void SetMatrix(
    int i0,
    int i1,
    int j0,
    int j1,
    GeneralMatrix X
);
```

**Parameters**

- **i0**
  - Initial row index
- **i1**
  - Final row index
- **j0**
  - Initial column index
- **j1**
  - Final column index
- **X**
  - \(A(i0:i1,j0:j1)\)
## Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexOutOfRangeException</td>
<td>Submatrix indices</td>
</tr>
</tbody>
</table>

## See Also

- [GeneralMatrix Class](#) | [DotNetMatrix Namespace](#) |
- [GeneralMatrix.SetMatrix Overload List](#)
An NDoc Documented Class Library
GeneralMatrix.SetMatrix Method (Int32, Int32, Int32[], GeneralMatrix)

Set a submatrix.

[C#]
public virtual void SetMatrix(
    int i0,
    int i1,
    int[] c,
    GeneralMatrix X
);

[Visual Basic]
Overridable Overloads Public Sub SetMatrix(
    ByVal i0 As Integer, _
    ByVal i1 As Integer, _
    ByVal c As Integer(), _
    ByVal X As GeneralMatrix _
)

Parameters

    i0
    Initial row index

    i1
    Final row index

    c
    Array of column indices.

    X
    A(i0:i1,c(:))

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexOutOfRangeException</td>
<td>Submatrix indices</td>
</tr>
</tbody>
</table>
See Also

GeneralMatrix Class | DotNetMatrix Namespace | GeneralMatrix.SetMatrix Overload List
An NDoc Documented Class Library
GeneralMatrix.SetMatrix Method (Int32[], Int32, Int32, GeneralMatrix)

Set a submatrix.

[Visual Basic] Overridable Overloads Public Sub SetMatrix(
    ByVal r As Integer(), _
    ByVal j0 As Integer, _
    ByVal j1 As Integer, _
    ByVal X As GeneralMatrix)

[C#]
public virtual void SetMatrix(
    int[] r,
    int j0,
    int j1,
    GeneralMatrix X)

Parameters

$r$
Array of row indices.

$j0$
Initial column index

$j1$
Final column index

$X$
$A(r(:,j0:j1))$

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexOutOfRangeException</td>
<td>Submatrix indices</td>
</tr>
</tbody>
</table>
See Also

GeneralMatrix Class | DotNetMatrix Namespace | GeneralMatrix.SetMatrix Overload List
An NDoc Documented Class Library
GeneralMatrix.SetMatrix Method (Int32[], Int32[], GeneralMatrix)

Set a submatrix.

[Visual Basic] Overridable Overloads Public Sub SetMatrix(  ByVal r As Integer(),  ByVal c As Integer(),  ByVal X As GeneralMatrix)

[C#] public virtual void SetMatrix(  int[] r,  int[] c,  GeneralMatrix X)

Parameters

r
Array of row indices.

c
Array of column indices.

X
A(r(:,c()))

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexOutOfRangeException</td>
<td>Submatrix indices</td>
</tr>
</tbody>
</table>

See Also

GeneralMatrix Class | DotNetMatrix Namespace | GeneralMatrix.SetMatrix Overload List
An NDoc Documented Class Library
GeneralMatrix.Solve Method

Solve A*X = B

[Visual Basic] Overridable Public Function Solve(
    ByVal B As GeneralMatrix
) As GeneralMatrix

[C#]
public virtual GeneralMatrix Solve(
    GeneralMatrix B
);

Parameters

B
    right hand side

Return Value

solution if A is square, least squares solution otherwise

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.SolveTranspose Method

Solve X*A = B, which is also A'*X' = B'

[Visual Basic] Overridable Public Function SolveTranspose(ByVal B As GeneralMatrix) As GeneralMatrix

[C#]
public virtual GeneralMatrix SolveTranspose(GeneralMatrix B);

Parameters

B right hand side

Return Value

solution if A is square, least squares solution otherwise.

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
C = A - B

[Visual Basic] Overridable Public Function Subtract
    ByVal B As GeneralMatrix
) As GeneralMatrix

[C#]
public virtual GeneralMatrix Subtract(
    GeneralMatrix B
);

Parameters

B
   another matrix

Return Value

A - B

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.SubtractEquals Method

A = A - B

[Visual Basic] Overridable Public Function SubtractEquals( _
    ByVal B As GeneralMatrix _
) As GeneralMatrix

[C#] public virtual GeneralMatrix SubtractEquals( 
    GeneralMatrix B 
);  

Parameters

B
another matrix

Return Value

A - B

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.SVD Method

Singular Value Decomposition

**[Visual Basic]**

```vbnet
Overridable Public Function SVD()
```

**[C#]**

```csharp
public virtual SingularValueDecomposition SVD();
```

Return Value

SingularValueDecomposition

See Also

GeneralMatrix Class | DotNetMatrix Namespace | SingularValueDecomposition
An NDoc Documented Class Library
GeneralMatrix.ISerializable.GetObjectObjectData Method

A method called when serializing this class

[Visual Basic] Sub GetObjectData( _
    ByVal info As SerializationInfo, _
    ByVal context As StreamingContext _
) Implements _
    ISerializable.GetObjectData

[C#]
void ISerializable.GetObjectObjectData(
    SerializationInfo info,
    StreamingContext context
);

Parameters

  info
  context

Implements

  ISerializable.GetObjectObjectData

See Also

  GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
**GeneralMatrix.Trace Method**

Matrix trace.

**[Visual Basic]**
```
Overridable Public Function Trace()
```

**[C#]**
```
public virtual double Trace();
```

**Return Value**

sum of the diagonal elements.

**See Also**

[GeneralMatrix Class] | [DotNetMatrix Namespace]
An NDoc Documented Class Library
GeneralMatrix.Transpose Method

Matrix transpose.

[Visual Basic] Overridable Public Function Transpose()

[C#]
public virtual GeneralMatrix Transpose();

Return Value
A'

See Also
GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix.UnaryMinus Method

Unary minus

[Visual Basic] Overridable Public Function UnaryMinus()

[C#] public virtual GeneralMatrix UnaryMinus();

Return Value

-A

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
**GeneralMatrix Operators**

The operators of the `GeneralMatrix` class are listed below. For a complete list of `GeneralMatrix` class members, see the `GeneralMatrix Members` topic.

**Public Static (Shared) Operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Addition Operator</code></td>
<td>Addition of matrices</td>
</tr>
<tr>
<td><code>Multiplication Operator</code></td>
<td>Multiplication of matrices</td>
</tr>
<tr>
<td><code>Subtraction Operator</code></td>
<td>Subtraction of matrices</td>
</tr>
</tbody>
</table>

See Also

[GeneralMatrix Class](#) | [GeneralMatrix Members](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
GeneralMatrix Addition Operator

Addition of matrices

[Visual Basic] `returnValue = GeneralMatrix.op_Addition(m1, m2)`

[C#]
```csharp
public static GeneralMatrix operator +(GeneralMatrix m1, GeneralMatrix m2);
```

Parameters
- `m1`
- `m2`

Return Value

See Also
- GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
GeneralMatrix Multiplication Operator

Multiplication of matrices

[Visual Basic]\(\text{returnValue} = \text{GeneralMatrix}.\text{op_Multiply}(m1, m2)\)

[C#]
public static GeneralMatrix operator *(  
    GeneralMatrix m1,  
    GeneralMatrix m2  
);

Parameters

- \(m1\)
- \(m2\)

Return Value

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An *NDoc Documented Class Library*
GeneralMatrix Subtraction Operator

Subtraction of matrices

[Visual Basic]
returnValue = GeneralMatrix.op_Subtraction(m1, m2)

[C#]
public static GeneralMatrix operator -(  
    GeneralMatrix m1,  
    GeneralMatrix m2  
);

Parameters

m1
m2

Return Value

See Also

GeneralMatrix Class | DotNetMatrix Namespace
An NDoc Documented Class Library
LUDecomposition Class

For a list of all members of this type, see LUDecomposition Members.

System.Object LUDecomposition

[Visual Basic]
Public Class LUDecomposition
Implements ISerializable

[C#]
public class LUDecomposition : ISerializable

Requirements

Namespace: DotNetMatrix

Assembly: GeneralMatrix (in GeneralMatrix.dll)

See Also

LUDecomposition Members | DotNetMatrix Namespace
An NDoc Documented Class Library
LUDecomposition Members

**LUDecomposition overview**

Public Instance Constructors

<table>
<thead>
<tr>
<th>LUDecomposition Constructor</th>
<th>LU Decomposition</th>
</tr>
</thead>
</table>

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoublePivot</td>
<td>Return pivot permutation vector as a one-dimensional double array</td>
</tr>
<tr>
<td>IsNonSingular</td>
<td>Is the matrix nonsingular?</td>
</tr>
<tr>
<td>L</td>
<td>Return lower triangular factor</td>
</tr>
<tr>
<td>Pivot</td>
<td>Return pivot permutation vector</td>
</tr>
<tr>
<td>U</td>
<td>Return upper triangular factor</td>
</tr>
</tbody>
</table>

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determinant</td>
<td>Determinant</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>Solve</td>
<td>Solve A*X = B</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

Private Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>LU</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>m</td>
</tr>
<tr>
<td>n</td>
</tr>
<tr>
<td>piv</td>
</tr>
<tr>
<td>pivsign</td>
</tr>
</tbody>
</table>

**Explicit Interface Implementations**

| ISerializable.GetObjectData |

**See Also**

[LUDecomposition Class] | [DotNetMatrix Namespace]
An NDoc Documented Class Library
LU Decomposition Constructor

LU Decomposition

[Visual Basic]
Public Sub New( ByVal A As GeneralMatrix )

[C#]
public LUDecomposition( GeneralMatrix A );

Parameters

A
  Rectangular matrix

Return Value

Structure to access L, U and piv.

See Also

LUDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
LUDecomposition Fields

The fields of the LUDecomposition class are listed below. For a complete list of LUDecomposition class members, see the LUDecomposition Members topic.

Private Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LU</td>
<td>Array for internal storage of decomposition. @serial internal array storage.</td>
</tr>
<tr>
<td>m</td>
<td>Row and column dimensions, and pivot sign. @serial column dimension. @serial row dimension. @serial pivot sign.</td>
</tr>
<tr>
<td>n</td>
<td>Row and column dimensions, and pivot sign. @serial column dimension. @serial row dimension. @serial pivot sign.</td>
</tr>
<tr>
<td>piv</td>
<td>Internal storage of pivot vector. @serial pivot vector.</td>
</tr>
<tr>
<td>pivsign</td>
<td>Row and column dimensions, and pivot sign. @serial column dimension. @serial row dimension. @serial pivot sign.</td>
</tr>
</tbody>
</table>

See Also

LUDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
LUDecomposition.LU Field

Array for internal storage of decomposition. @serial internal array storage.

[Visual Basic] Private LU As Double()[]

[C#] private double[][] LU;

See Also
LUDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
LUDecomposition.m Field

Row and column dimensions, and pivot sign. @serial column dimension. @serial row dimension. @serial pivot sign.

[Visual Basic] Private m As Integer

[C#] private int m;

See Also

LUDecomposition Class | DotNetMatrix Namespace
An *NDoc Documented Class Library*
LU Decomposition.n Field

Row and column dimensions, and pivot sign. @serial column dimension. @serial row dimension. @serial pivot sign.

[Visual Basic] Private n As Integer

[C#] private int n;

See Also

LU Decomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
LUDecomposition.piv Field

Internal storage of pivot vector. @serial pivot vector.

[Visual Basic] Private piv As Integer()

[C#] private int[] piv;

See Also

LUDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
LUDecomposition.pivsign Field

Row and column dimensions, and pivot sign. @serial column dimension. @serial row dimension. @serial pivot sign.

[Visual Basic] Private pivsign As Integer

[C#] private int pivsign;

See Also

LUDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
LUDecomposition Properties

The properties of the **LUDecomposition** class are listed below. For a complete list of **LUDecomposition** class members, see the **LUDecomposition Members** topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DoublePivot</code></td>
<td>Return pivot permutation vector as a one-dimensional double array</td>
</tr>
<tr>
<td><code>IsNonSingular</code></td>
<td>Is the matrix nonsingular?</td>
</tr>
<tr>
<td><code>L</code></td>
<td>Return lower triangular factor</td>
</tr>
<tr>
<td><code>Pivot</code></td>
<td>Return pivot permutation vector</td>
</tr>
<tr>
<td><code>U</code></td>
<td>Return upper triangular factor</td>
</tr>
</tbody>
</table>

### See Also

- [LUDecomposition Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
LUDecomposition.DoublePivot Property

Return pivot permutation vector as a one-dimensional double array

[Visual Basic] Overridable Public ReadOnly Property DoublePivot As DoublePivot

[C#]
public virtual double[] DoublePivot {get;}

See Also

LUDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
LU Decomposition.IsNonSingular Property

Is the matrix nonsingular?

[Visual Basic] Overridable Public ReadOnly Property IsNonSingular

[C#] public virtual bool IsNonSingular {get;}

See Also

LU Decomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
LU Decomposition.L Property

Return lower triangular factor

[Visual Basic] Overridable Public ReadOnly Property L As

[C#] public virtual GeneralMatrix L {get;}

See Also

LUDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
Return pivot permutation vector

[Visual Basic] Overridable Public ReadOnly Property Pivot

[C#]
public virtual int[] Pivot {get;}

See Also

LUDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
LUDecomposition.U Property

Return upper triangular factor

[Visual Basic]
Overridable Public ReadOnly Property U As U

[C#]
public virtual GeneralMatrix U {get;}

See Also
LUDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
LUDecomposition Methods

The methods of the LUDecomposition class are listed below. For a complete list of LUDecomposition class members, see the LUDecomposition Members topic.

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Determinant</strong></td>
<td>Determinant</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>Solve</strong></td>
<td>Solve A*X = B</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ISerializable.GetObjectData</strong></td>
<td></td>
</tr>
</tbody>
</table>

See Also

LUDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
LUDecomposition.Determinant Method

Determinant

[Visual Basic] Overridable Public Function Determinant()

[C#]
public virtual double Determinant();

Return Value

det(A)

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Matrix must be square</td>
</tr>
</tbody>
</table>

See Also

LUDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
Solve $A*X = B$

**Parameters**

$B$

A Matrix with as many rows as A and any number of columns.

**Return Value**

$X$ so that $L*U*X = B(piv,:)$

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Matrix row dimensions must agree.</td>
</tr>
<tr>
<td>SystemException</td>
<td>Matrix is singular.</td>
</tr>
</tbody>
</table>

**See Also**

[LUDEcomposition Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
LU Decomposition.IsSerializabe.GetObjectData Method

[Visual Basic] Sub GetObjectData( _
    ByVal info As SerializationInfo, _
    ByVal context As StreamingContext _
) Implements _
    ISerializable.GetObjectData

[C#]
void ISerializable.GetObjectData(
    SerializationInfo info,
    StreamingContext context
);

Implements

    ISerializable.GetObjectData

See Also

    LU Decomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
Maths Class

For a list of all members of this type, see Maths Members.

System.Object Maths

[Visual Basic]
Private Class Maths

[C#]
private class Maths

Requirements

Namespace: DotNetMatrix
Assembly: GeneralMatrix (in GeneralMatrix.dll)

See Also

Maths Members | DotNetMatrix Namespace
An NDoc Documented Class Library
## Maths Members

### Maths overview

### Public Static (Shared) Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypot</td>
<td>( \sqrt{a^2 + b^2} ) without under/overflow.</td>
</tr>
</tbody>
</table>

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maths Constructor</td>
<td>Initializes a new instance of the Maths class.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

### See Also

- Maths Class
- DotNetMatrix Namespace
An NDoc Documented Class Library
Maths Constructor

Initializes a new instance of the Maths class.

[Visual Basic] Public Sub New()

[C#]
public Maths();

See Also

Maths Class | DotNetMatrix Namespace
An NDoc Documented Class Library
Maths Methods

The methods of the Maths class are listed below. For a complete list of Maths class members, see the Maths Members topic.

Public Static (Shared) Methods

| Hypot | sqrt(a^2 + b^2) without under/overflow. |

Public Instance Methods

| Equals (inherited from Object) | Determines whether the specified Object is equal to the current Object. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |

See Also

Maths Class | DotNetMatrix Namespace
An NDoc Documented Class Library
sqrt(a^2 + b^2) without under/overflow.

**[Visual Basic]**
```vbnet
Public Shared Function Hypot(
    ByVal a As Double, _
    ByVal b As Double _
) As Double
```

**[C#]**
```csharp
public static double Hypot(
    double a,
    double b
);
```

**Parameters**
- `a`
- `b`

**Return Value**

**See Also**
- [Maths Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
QR Decomposition. For an m-by-n matrix A with m >= n, the QR decomposition is an m-by-n orthogonal matrix Q and an n-by-n upper triangular matrix R so that A = Q*R. The QR decomposition always exists, even if the matrix does not have full rank, so the constructor will never fail. The primary use of the QR decomposition is in the least squares solution of nonsquare systems of simultaneous linear equations. This will fail if IsFullRank() returns false.

For a list of all members of this type, see QRDecomposition Members.

System.Object QRDecomposition

[Visual Basic]
Public Class QRDecomposition
    Implements ISerializable

[C#]
public class QRDecomposition : ISerializable

Requirements

Namespace: DotNetMatrix
Assembly: GeneralMatrix (in GeneralMatrix.dll)

See Also

QRDecomposition Members | DotNetMatrix Namespace
An NDoc Documented Class Library
**QRDecomposition Members**

**QRDecomposition overview**

**Public Instance Constructors**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QRDecomposition Constructor</td>
<td>QR Decomposition, computed by Householder reflections.</td>
</tr>
</tbody>
</table>

**Public Instance Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FullRank</td>
<td>Is the matrix full rank?</td>
</tr>
<tr>
<td>H</td>
<td>Return the Householder vectors</td>
</tr>
<tr>
<td>Q</td>
<td>Generate and return the (economy-sized) orthogonal factor</td>
</tr>
<tr>
<td>R</td>
<td>Return the upper triangular factor</td>
</tr>
</tbody>
</table>

**Public Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td>Solve</td>
<td>Least squares solution of A*X = B</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**Private Instance Fields**
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>Row and column dimensions. @serial column dimension. @serial row dimension.</td>
</tr>
<tr>
<td>$n$</td>
<td>Row and column dimensions. @serial column dimension. @serial row dimension.</td>
</tr>
<tr>
<td>$QR$</td>
<td>Array for internal storage of decomposition. @serial internal array storage.</td>
</tr>
<tr>
<td>$Rdiag$</td>
<td>Array for internal storage of diagonal of R. @serial diagonal of R.</td>
</tr>
</tbody>
</table>

**Explicit Interface Implementations**

<table>
<thead>
<tr>
<th>Interface Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISerializable.GetObjectData</td>
</tr>
</tbody>
</table>

**See Also**

- QRDecomposition Class
- DotNetMatrix Namespace
An NDoc Documented Class Library
QR Decomposition Constructor

QR Decomposition, computed by Householder reflections.

[Visual Basic]
Public Sub New( _
   ByVal A As GeneralMatrix _
)

[C#]
public QRDecomposition(
   GeneralMatrix A
);

Parameters

A
   Rectangular matrix

Return Value

Structure to access R and the Householder vectors and compute Q.

See Also

QRDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
QRDecomposition Fields

The fields of the `QRDecomposition` class are listed below. For a complete list of `QRDecomposition` class members, see the `QRDecomposition Members` topic.

**Private Instance Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>Row and column dimensions. @serial column dimension. @serial row dimension.</td>
</tr>
<tr>
<td>n</td>
<td>Row and column dimensions. @serial column dimension. @serial row dimension.</td>
</tr>
<tr>
<td>QR</td>
<td>Array for internal storage of decomposition. @serial internal array storage.</td>
</tr>
<tr>
<td>Rdiag</td>
<td>Array for internal storage of diagonal of R. @serial diagonal of R.</td>
</tr>
</tbody>
</table>

See Also

[QRDecomposition Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
QRDecomposition.m Field

Row and column dimensions. @serial column dimension. @serial row dimension.

[Visual Basic] `Private m As Integer`

[C#] `private int m;`

See Also

QRDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
QRDecomposition.n Field

Row and column dimensions. @serial column dimension. @serial row dimension.

[Visual Basic] Private n As Integer

[C#] private int n;

See Also
QRDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
Array for internal storage of decomposition. @serial internal array storage.

**[Visual Basic]**
```
Private QR As Double()[]
```

**[C#]**
```
private double[][] QR;
```

See Also

[QRDecomposition Class] | [DotNetMatrix Namespace]
An NDoc Documented Class Library
Array for internal storage of diagonal of R. @serial diagonal of R.

**[Visual Basic]** `Private Rdiag As Double()`

**[C#]**
```csharp
private double[] Rdiag;
```

See Also

**QRDecomposition Class**  |  **DotNetMatrix Namespace**
An NDoc Documented Class Library
The properties of the **QRDecomposition** class are listed below. For a complete list of **QRDecomposition** class members, see the [QRDecomposition Members](#) topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FullRank</code></td>
<td>Is the matrix full rank?</td>
</tr>
<tr>
<td><code>H</code></td>
<td>Return the Householder vectors</td>
</tr>
<tr>
<td><code>Q</code></td>
<td>Generate and return the (economy-sized) orthogonal factor</td>
</tr>
<tr>
<td><code>R</code></td>
<td>Return the upper triangular factor</td>
</tr>
</tbody>
</table>

### See Also

- [QRDecomposition Class](#)
- [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
QRDecomposition.FullRank Property

Is the matrix full rank?

[Visual Basic] Overridable Public ReadOnly Property FullRank As Boolean

[C#]
public virtual bool FullRank {get;}

See Also

QRDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
QRDecomposition.H Property

Return the Householder vectors

[Visual Basic] Overridable Public ReadOnly Property

[C#]
public virtual GeneralMatrix H {get;}

See Also
QRDecomposition Class | DotNetMatrix Namespace
QRDecomposition.Q Property

Generate and return the (economy-sized) orthogonal factor

[Visual Basic] Overridable Public ReadOnly Property Q As GeneralMatrix

[C#] public virtual GeneralMatrix Q {get;}

See Also

QRDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
QR Decomposition.R Property

Return the upper triangular factor

[Visual Basic] Overridable Public ReadOnly Property R

[C#]
public virtual GeneralMatrix R {get;}

See Also

QR Decomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
QRDecomposition Methods

The methods of the QRDecomposition class are listed below. For a complete list of QRDecomposition class members, see the QRDecomposition Members topic.

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> <em>(inherited from Object)</em></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> <em>(inherited from Object)</em></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> <em>(inherited from Object)</em></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>Solve</strong></td>
<td>Least squares solution of A*X = B</td>
</tr>
<tr>
<td><strong>ToString</strong> <em>(inherited from Object)</em></td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Interface</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISerializable</td>
<td><strong>GetObjectData</strong></td>
</tr>
</tbody>
</table>

See Also

QRDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
QRDecomposition.Solve Method

Least squares solution of $A*X = B$

[Visual Basic] Overridable Public Function Solve(
    ByVal B As GeneralMatrix
) As GeneralMatrix

[C#]
public virtual GeneralMatrix Solve(
    GeneralMatrix B
);

Parameters

$B$

A Matrix with as many rows as $A$ and any number of columns.

Return Value

$X$ that minimizes the two norm of $Q*R*X-B$.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Matrix row dimensions must agree.</td>
</tr>
<tr>
<td>SystemException</td>
<td>Matrix is rank deficient.</td>
</tr>
</tbody>
</table>

See Also

QRDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
### QRDecomposition.ISerializable.GetObjectObjectData Method

[Visual Basic]
```vbnet
Sub GetObjectData( _
    ByVal info As SerializationInfo, _
    ByVal context As StreamingContext _
) Implements _
    ISerializable.GetObjectData
```

[C#]
```csharp
void ISerializable.GetObjectObjectData(
    SerializationInfo info,
    StreamingContext context
);
```

Implements

- ISerializable.GetObjectData

See Also

- [QRDecomposition Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
SingularValueDecomposition Class

For a list of all members of this type, see SingularValueDecomposition Members.

System.Object  SingularValueDecomposition

[Visual Basic]
Public Class SingularValueDecomposition
Implements ISerializable

[C#]
public class SingularValueDecomposition : ISerializable

Requirements

Namespace: DotNetMatrix

Assembly: GeneralMatrix (in GeneralMatrix.dll)

See Also

SingularValueDecomposition Members  |  DotNetMatrix Namespace
An NDoc Documented Class Library
## SingularValueDecomposition Members

### SingularValueDecomposition overview

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SingularValueDecomposition</strong></td>
<td>Construct the singular value decomposition</td>
</tr>
</tbody>
</table>

#### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Return the diagonal matrix of singular values</td>
</tr>
<tr>
<td><strong>SingularValues</strong></td>
<td>Return the one-dimensional array of singular values</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Condition</strong></td>
<td>Two norm condition number</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>GetU</strong></td>
<td>Return the left singular vectors</td>
</tr>
<tr>
<td><strong>GetV</strong></td>
<td>Return the right singular vectors</td>
</tr>
<tr>
<td><strong>Norm2</strong></td>
<td>Two norm</td>
</tr>
<tr>
<td><strong>Rank</strong></td>
<td>Effective numerical matrix rank</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>
Private Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>Row and column dimensions. @serial row dimension. @serial column dimension.</td>
</tr>
<tr>
<td>n</td>
<td>Row and column dimensions. @serial row dimension. @serial column dimension.</td>
</tr>
<tr>
<td>s</td>
<td>Array for internal storage of singular values. @serial internal storage of singular values.</td>
</tr>
<tr>
<td>U</td>
<td>Arrays for internal storage of U and V. @serial internal storage of U. @serial internal storage of V.</td>
</tr>
<tr>
<td>V</td>
<td>Arrays for internal storage of U and V. @serial internal storage of U. @serial internal storage of V.</td>
</tr>
</tbody>
</table>

Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Interface Implementation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISerializable.GetObjectData</td>
<td></td>
</tr>
</tbody>
</table>

See Also

[SingularValueDecomposition Class] | [DotNetMatrix Namespace]
SingularValueDecomposition Constructor

Construct the singular value decomposition

[Visual Basic] Public Sub New( _
    ByVal Arg As GeneralMatrix _
)  

[C#]
public SingularValueDecomposition(
    GeneralMatrix Arg
);

Parameters

Arg
Rectangular matrix

Return Value

Structure to access U, S and V.

See Also

SingularValueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
SingularValueDecomposition Fields

The fields of the **SingularValueDecomposition** class are listed below. For a complete list of **SingularValueDecomposition** class members, see the [SingularValueDecomposition Members](#) topic.

**Private Instance Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>Row and column dimensions. @serial row dimension. @serial column dimension.</td>
</tr>
<tr>
<td>n</td>
<td>Row and column dimensions. @serial row dimension. @serial column dimension.</td>
</tr>
<tr>
<td>s</td>
<td>Array for internal storage of singular values. @serial internal storage of singular values.</td>
</tr>
<tr>
<td>U</td>
<td>Arrays for internal storage of U and V. @serial internal storage of U. @serial internal storage of V.</td>
</tr>
<tr>
<td>V</td>
<td>Arrays for internal storage of U and V. @serial internal storage of U. @serial internal storage of V.</td>
</tr>
</tbody>
</table>

See Also

[SingularValueDecomposition Class](#) | [DotNetMatrix Namespace](#)
An NDoc Documented Class Library
SingularValueDecomposition.m Field

Row and column dimensions. @serial row dimension. @serial column dimension.

[Visual Basic] Private m As Integer

[C#] private int m;

See Also

SingularValueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
**SingularValueDecomposition.n Field**

Row and column dimensions. @serial row dimension. @serial column dimension.

[Visual Basic] `Private n As Integer`

[C#] `private int n;`

See Also

[SingularValueDecomposition Class] | [DotNetMatrix Namespace]
An NDoc Documented Class Library
Return the diagonal matrix of singular values

[Visual Basic] Overridable Public ReadOnly Property S

[C#]
public virtual GeneralMatrix S {get;}

See Also
SingularValueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
**SingularValueDecomposition.U Field**

Arrays for internal storage of U and V. @serial internal storage of U. @serial internal storage of V.

[Visual Basic] `Private U As Double[][]`

[C#] `private double[][] U;`

See Also

[SingularValueDecomposition Class] | [DotNetMatrix Namespace]
An NDoc Documented Class Library
Arrays for internal storage of U and V. @serial internal storage of U. @serial internal storage of V.

[Visual Basic] Private V As Double()[]

[C#]
private double[][] V;

See Also

SingularValueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
The properties of the SingularValueDecomposition class are listed below. For a complete list of SingularValueDecomposition class members, see the SingularValueDecomposition Members topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Return the diagonal matrix of singular values</td>
</tr>
<tr>
<td>SingularValues</td>
<td>Return the one-dimensional array of singular values</td>
</tr>
</tbody>
</table>

See Also

[SingularValueDecomposition Class] | [DotNetMatrix Namespace]
An NDoc Documented Class Library
SingularValueDecomposition.SingularValues Property

Return the one-dimensional array of singular values

[Visual Basic] Overridable Public ReadOnly Property

[C#] public virtual double[] SingularValues {get;}

See Also

SingularValueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
SingularValueDecomposition Methods

The methods of the SingularValueDecomposition class are listed below. For a complete list of SingularValueDecomposition class members, see the SingularValueDecomposition Members topic.

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition</td>
<td>Two norm condition number</td>
</tr>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>GetU</td>
<td>Return the left singular vectors</td>
</tr>
<tr>
<td>GetV</td>
<td>Return the right singular vectors</td>
</tr>
<tr>
<td>Norm2</td>
<td>Two norm</td>
</tr>
<tr>
<td>Rank</td>
<td>Effective numerical matrix rank</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

Explicit Interface Implementations

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISerializable.GetObjectData</td>
<td></td>
</tr>
</tbody>
</table>

See Also

SingularValueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
SingularValueDecomposition.Condition Method

Two norm condition number

[Visual Basic] Overridable Public Function Condition()

[C#]
public virtual double Condition();

Return Value
max(S)/min(S)

See Also
SingularValueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
SingularValueDecomposition.GetU Method

Return the left singular vectors

[Visual Basic] Overridable Public Function GetU()

[C#]
public virtual GeneralMatrix GetU();

Return Value

U

See Also

SingularValueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
SingularValueDecomposition.GetV Method

Return the right singular vectors

[Visual Basic] Overridable Public Function GetV()

[C#]
public virtual GeneralMatrix GetV();

Return Value

V

See Also

SingularValueDecomposition Class | DotNetMatrix Namespace
An NDoc Documented Class Library
**SingularValueDecomposition.Norm2 Method**

Two norm

**[Visual Basic]**

Overridable Public Function Norm2()

**[C#]**

public virtual double Norm2();

**Return Value**

max(S)

**See Also**

[SingularValueDecomposition Class] | [DotNetMatrix Namespace]
An NDoc Documented Class Library
Effective numerical matrix rank

**Visual Basic**

`Overridable Public Function Rank()`

**C#**

`public virtual int Rank();`

Return Value

Number of nonnegligible singular values.

See Also

[SingularValueDecomposition Class] | [DotNetMatrix Namespace]
An NDoc Documented Class Library
SingularValueDecomposition.ISerializable.GetObjectObjectData Method

[Visual Basic] Sub GetObjectData( _
    ByVal info As SerializationInfo, _
    ByVal context As StreamingContext _
) Implements _
    ISerializable.GetObjectData

[C#] void ISerializable.GetObjectData(
    SerializationInfo info,
    StreamingContext context
);

Implements

ISerializable.GetObjectData

See Also

SingularValueDecomposition Class | DotNetMatrix Namespace