



Matrix Library

Define matrices of arbitrary dimensions. Execute basic operations (addition, multiplication, linear equations solving, inverting matrices and computing determinants) on matrices.

(A 7 day demo license is available for testing.)

Product description

Licensing:

Workstation License

The matrix library offers a data type to define matrices of arbitrary dimension and functions to perform basic operations on them.

Matrices are defined through the `mtx.Matrix` data type. `Mtx.Matrix` saves a matrix as `ARRAY` of `LREAL`. The array is in row-major form.

Basic matrix operations are provided as functions that expect matrices as `VAR_IN_OUT` argument. E.g. the function for addition:

```
(* Adds two matrices : C := A + B.
 * A, B, and C must have identical dimensions.
 * Note: A, B, and C may all be the same matrix. *)
FUNCTION AddM : ResultCode
VAR_IN_OUT
  C : Matrix ; (* The result *)
  A : Matrix ; (* The first summand *)
  B : Matrix ; (* The second summand *)
END_VAR
```

The library offers following operations:

- Addition: `AddM`
- Subtraction: `SubM`
- Multiplication: `MultM`
- Multiplication (element-wise): `TimesM`
- Division (element-wise): `RDivideM`
- Scalar multiplication: `MultMS`
- Transposition: `TransposeM`

Additionally there are some help functions:

- Initialize a matrix with an array of values: `InitMatrix`
- Copy array elements to matrix: `CopyElems`
- Copy matrices with same dimension: `CopyMatrix`
- Initialize as identity matrix: `IdentityMatrix`
- Initialize as zero matrix: `ZeroMatrix`
- Read and write elements: `Elem`, `SetElem`

Also more complex operations are provided:

- Solve a linear equation ($A \cdot X = B$): `SolveLU`
- Invert a quadratic matrix: `InvertLU`
- Compute determinant of a quadratic matrix: `DeterminantLU`
- Compute the LU factorization: `DecomposeLU`

Memory Management

The user is responsible for memory management. Matrices will be initialized with a pointer to the memory. In some cases it is possible to provide a suitable memory.

Furthermore matrices can be initialized with arrays of constant size. Therefore `MatrixS`, `ColVectorS` and

RowVectorS can be used. All three implement the IMatrixAllocator interface.

Matrix Examples

The Matrix Examples show the usage of the different functions in detail. For illustration there is also an example of a linear equation with its solution.

For further information please also have a look at "MatrixTests.project" project.

General information

Supplier:

CODESYS GmbH
 Memminger Strasse 151
 87439 Kempten
 Germany

Support:

<https://support.codesys.com>

Item:

Matrix Library

Item number:

2111000003

Sales:

CODESYS Store

<https://store.codesys.com>

Included in delivery:

- CODESYS software and / or license key with billing information
- For training courses and events: Booking confirmation

System requirements and restrictions

Programming System	CODESYS Development System Version 3.5.14.0 or higher
Runtime System	CODESYS Control Version 3.5.2.0
Supported Platforms/ Devices	All
Additional Requirements	-
Restrictions	-
Licensing	<p>Workstation License: The license can be used on the workstation on which the CODESYS Development System is installed and executed.</p> <p>Licenses are activated on a software-based license container (soft container), which is permanently connected to the workstation. Alternatively the license can be stored on a CODESYS Key (USB-Dongle). By replugging the CODESYS Key, the license can be used on any other workstation.</p>
Required Accessories	CODESYS Key for CODESYS < 3.5.14.0

Note: Not all CODESYS features are available in all territories. For more information on geographic restrictions, please contact sales@codesys.com.

Note: Technical specifications are subject to change. Errors and omissions excepted. The content of the current online version of this document applies.