

Release Notes for Symbolic Math Toolbox™

How to Contact MathWorks



www.mathworks.com Web
comp.soft-sys.matlab Newsgroup
www.mathworks.com/contact_TS.html Technical Support



suggest@mathworks.com Product enhancement suggestions
bugs@mathworks.com Bug reports
doc@mathworks.com Documentation error reports
service@mathworks.com Order status, license renewals, passcodes
info@mathworks.com Sales, pricing, and general information



508-647-7000 (Phone)



508-647-7001 (Fax)



The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

Release Notes for Symbolic Math Toolbox™

© COPYRIGHT 2004–2013 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

R2013a

Linear algebra functions for computing matrix factorizations (<code>lu</code> , <code>chol</code>), pseudoinverse, orthogonal basis, and adjoint	2
Verification of solutions of systems of equations and arbitrary symbolic function substitution in <code>subs</code> function	3
Simplification for more types of trigonometric and hyperbolic expressions and expressions with nested roots	4
Special functions for computing polar angle, <code>atan2</code> function, imaginary error function, and exponential and elliptic integrals	5
<code>toeplitz</code> function for creating Toeplitz matrices	6
<code>sqrtn</code> function for computing square roots of matrices ...	7
<code>sign</code> function for computing signs of numbers	8
Real option of the <code>linalg::orthog</code> function for avoiding complex conjugates	9
Real option of the <code>linalg::factorCholesky</code> function for avoiding complex conjugates	10
New arguments of the <code>svd</code> function for computing the “economy size” singular value decomposition	11
<code>isequaln</code> function for testing equality of symbolic objects	12
Control over the order in which <code>solve</code> and <code>vpasolve</code> functions return solutions	13
Functionality Being Removed or Changed	14

R2012b

MATLAB symbolic matrix analysis functions for characteristic (<code>charpoly</code>) and minimal (<code>minpoly</code>) polynomials and for norm (<code>norm</code>) and condition (<code>cond</code>) number	20
<code>poles</code> function for determining the poles of an expression	21

vpsolve function for solving equations and systems using variable precision arithmetic	22
Functions for converting linear systems of equations to matrix form AX=B (equationsToMatrix) and solving matrix equations (linsolve)	23
MATLAB symbolic functions for describing pulses: rectangularPulse and triangularPulse	24
MuPAD functions for computing integral and Z-transforms	25
MuPAD Pref::fourierParameters function for specifying Fourier parameters	26
MuPAD functions for adding transform patterns	27
noFlatten option of the MuPAD proc function for preventing sequence flattening	28
testtype uses testtypeDom slot for overloading by the second argument	29
New upper limit on the number of digits in double	30
New definition for real and imag	31
Functionality Being Removed or Changed	32

R2012a

New Special Functions	38
New Vector Analysis Functions	39
Computations with Symbolic Functions	40
Assumptions on Variables	41
New Relational Operators Create Equations, Inequalities, and Relations	42
New Logical Operators Create Logical Expressions	43
New Functions Test Validity of Symbolic Equations, Inequalities, and Relations	44
New Functions Manipulate Symbolic Expressions	45
New odeToVectorField Function Converts Higher-Order Differential Equations to Systems of First-Order Differential Equations	46
New Calling Syntax for the taylor Function	47
New MuPAD Functions Compute Rectangular and Triangular Pulse Functions	48
MuPAD det, linalg::det, inverse, linsolve, and linalg::matlinsolve Functions Accept the New Normal Option	49

MuPAD linalg::matlinsolve Function Accepts the New ShowAssumptions Option	50
Enhanced MuPAD pdivide Function	51
Improved MuPAD prog::remember Function	52
Functionality Being Removed or Changed	53

R2011b

MATLAB Editor Now Supports MuPAD Program Files ...	56
dsolve, expand, int, simple, simplify, and solve Accept More Options	57
New read Function Reads MuPAD Program Files in MATLAB	58
New symprod Function Computes Products of Series	59
New hessian Function Computes Hessian Matrices	60
New gradient Function Computes Vector Gradients	61
New erfc Function Computes the Complementary Error Function	62
New psi Function Computes the Digamma and Polygamma Functions	63
New wrightOmega Function Computes the Wright omega Function	64
New simplifyFraction Function Simplifies Expressions ...	65
New MuPAD simplifyRadical Function Simplifies Radicals in Arithmetical Expressions	66
pretty Function Now Uses Abbreviations in Long Output Expressions for Better Readability	67
MuPAD normal Function Accepts the New Expand Option	68
Modified MuPAD groebner Library	69
MuPAD groebner::gbasis Function Accepts the New Factor and IgnoreSpecialCases Options	70
New MuPAD Functions for Computing Logarithms	71
Functionality Being Removed or Changed	72

R2011a

Expression Wrapping of Math Output in the MuPAD Notebook Interface	74
---	----

Symbolic Solver Handles More Non-Algebraic Equations	75
Improved Performance in the Ordinary Differential Equation Solver	76
Improved Performance for Polynomial Arithmetic Operations	77
New MuPAD polylib::subresultant Function Computes Subresultants of Polynomials	78
MuPAD partfrac Function Accepts the New List Option ..	79
New MuPAD inverf and inverfc Functions Compute the Inverses of Error Functions	80
New MuPAD numlib::checkPrimalityCertificate Function Tests Primality Certificates	81
New Demos	82
Functionality Being Removed or Changed	83

R2010b

sym Function Creates Matrices of Symbolic Variables	86
generate::Simscape Function Generates Simscape Equations from MuPAD Expressions	87
MuPAD Code Generation Functions Accept the New NoWarning Option	88
Improved MuPAD Hyperlink Dialog Box	89
MuPAD Notebook Highlights Matched and Unmatched Delimiters	90
Improved Performance When Solving Linear Systems in a Matrix Form	91
MuPAD Solver for Ordinary Differential Equations Handles More Equation Types	92
New Syntax for the MuPAD prog::getOptions Function ...	93
New Syntax for the MuPAD prog::trace Function	94
Improved Interface for Arithmetical Operations on Polynomials	95
MuPAD igcd Function Now Accepts Complex Numbers as Arguments	96
Enhanced Solver For Factorable Polynomial Systems	97
MuPAD Now Evaluates Large Sums with Subtractions Faster	98
MuPAD freeIndets Function Accepts the New All Option	99
Functionality Being Removed or Changed	100

R2010a

When Opening Notebook, MuPAD Can Jump to Particular Locations	102
simscapeEquation Function Generates Simscape Equations from Symbolic Expressions	103
New Calling Syntax for the sort Function	104
Changes in the symengine Function	105
64-Bit GUI Support for Macintosh	106
New MuPAD Print Preview Dialog	107
Improved Configure MuPAD Dialog Box	108
MuPAD Support for Basic Arithmetic Operations for Lists	109
Improved Performance When Operating on Matrices with Symbolic Elements	110
Enhanced MuPAD divide Function	111
Improved Performance for Operations on Polynomials ...	112
MuPAD coeff Function Accepts the New All Option	113
MuPAD expand Function Accepts the New ArithmeticOnly Option	114
MuPAD expand Function Now Expands Powers of Products	115
New Calling Syntax for MuPAD rationalize Function ...	116
Enhanced MuPAD simplify and Simplify Functions	117
MuPAD subs Function Accepts the New EvalChanges Option	118
MuPAD Solver for Ordinary Differential Equations Handles More Equation Types	119
Functionality Being Removed or Changed	120

R2009b

Support for Windows x64 and 64-Bit Macintosh	122
sym and syms Use Reserved Words as Variable Names ..	123
Toolbox Now Displays Floating-Point Results with Their Original Precision	124
New MuPAD Preference Pref::outputDigits Controls Floating-Point Outputs	125
Solver for Ordinary Differential Equations Handles More Equation Types	126
MuPAD limit Function Supports Limits for Incomplete Gamma Function and Exponential Integral Function ..	127

Enhanced Simplification Routines for MuPAD Special Functions	128
Enhanced MuPAD combine Function for Logarithms	129
MuPAD normal Function Accepts New Options	130
Functionality Being Removed or Changed	131

R2009a

dsolve Accepts the New Option	
IgnoreAnalyticConstraints	134
emlBlock Function Generates Embedded MATLAB	
Function Blocks from Symbolic Objects	135
matlabFunction Improves Control over Input and Output	
Parameters	136
Enhancements to Object-Oriented Programming	
Capabilities	137
generate::MATLAB Function Converts MuPAD Expressions	
to MATLAB Code	138
MuPAD IgnoreAnalyticConstraints Option Specifies That	
Core Functions Apply Common Algebraic Assumptions	
to Simplify Results	139
MuPAD Outputs Contain Abbreviations for Better	
Readability	140
MuPAD Solver for Ordinary Differential Equations Handles	
More Equation Types	141
MuPAD limit Function Now Can Compute Limits for	
Piecewise Functions	142
New and Improved MuPAD Special Functions	143
New Calling Syntax for Test Report Function prog::tcov ..	144
New Demos	145

R2008b

R2008a+

R2007b+

MuPAD Engine Replaces Maple Engine	152
New MuPAD Language and Libraries Supplant Extended Symbolic Math Toolbox Software	157
New MuPAD Help Viewer (GUI)	158
New MuPAD Notebook Interface (GUI)	159
New MuPAD Editor and Debugger (GUI)	160
New Functionality for Communication Between MATLAB Workspace and MuPAD	161
New symengine Command for Choosing a Maple Engine ..	162
New matlabFunction Generates MATLAB Functions	163

R2008a

R2007b

R2007a

Maple10 Access Added for Linux 64-bit Processors and Intel Macintosh Platforms	170
---	-----

Change in call to code generation package using the maple function	172
---	------------

R2013a

Version: 5.10
New Features: Yes
Bug Fixes: Yes

Linear algebra functions for computing matrix factorizations (`lu`, `chol`), pseudoinverse, orthogonal basis, and adjoint

- `lu` computes the LU factorization of a matrix. Permutation information can be returned as a matrix or as a row vector.
- `chol` computes the Cholesky factorization of a matrix. The result can be returned as an upper or lower triangular matrix. Permutation information can be returned as a matrix or as a row vector.
- `pinv` computes the Moore-Penrose pseudoinverse of a matrix.
- `orth` computes an orthonormal basis for the range of a symbolic matrix.
- `adjoint` computes the adjoint of a symbolic square matrix.

Verification of solutions of systems of equations and arbitrary symbolic function substitution in subs function

Compatibility Considerations: Yes

The MATLAB® symbolic subs function lets you:

- Verify solutions of systems of equations by substituting the solutions returned by solve back into the systems
- Substitute elements of a symbolic expression with arbitrary symbolic functions

Compatibility Considerations

subs(s,old,new,0) will not accept 0 in a future release. Replace all instances of subs(s,old,new,0) with subs(s,old,new). The subs function does not switch old and new anymore.

subs does not return double-precision floating-point results anymore. Instead, it consistently returns symbolic results. To convert such results to double-precision numbers, use double.

Simplification for more types of trigonometric and hyperbolic expressions and expressions with nested roots

Compatibility Considerations: Yes

The MATLAB symbolic `simplify` function and the MuPAD® `simplify` function achieve better simplification of trigonometric expressions and expressions with nested roots.

The MATLAB symbolic `simplify` function accepts the new `Criterion` option. This option lets you discourage `simplify` from returning results containing complex numbers.

The MuPAD `simplify` function accepts two new options:

- `Steps` specifies the number of internal simplification steps.
- `Seconds` limits the time allowed for the internal simplification process.

Compatibility Considerations

The default number of simplification steps used by the MATLAB symbolic `simplify` function and the MuPAD `simplify` function changed from 100 to 1.

The `FinalValuation` option used in MuPAD `Simplify` function calls is renamed. The new name is `Criterion`.

Special functions for computing polar angle, atan2 function, imaginary error function, and exponential and elliptic integrals

- `angle` computes the polar angle of a complex value.
- `atan2` computes the four-quadrant inverse tangent (arctangent).
- `erfi` computes the imaginary error function.
- `ei` computes the one-argument exponential integral.
- `expint` computes the two-argument exponential integral.

The following new MATLAB symbolic functions compute elliptic integrals:

- `ellipticK` computes the complete elliptic integral of the first kind.
- `ellipticF` computes the incomplete elliptic integral of the first kind.
- `ellipticE` computes the complete and incomplete elliptic integrals of the second kind.
- `ellipticCK` computes the complementary complete elliptic integral of the first kind.
- `ellipticCE` computes the complementary complete elliptic integral of the second kind.
- `ellipticPi` computes the complete and incomplete elliptic integrals of the third kind.
- `ellipticCPi` computes the complementary complete elliptic integral of the third kind.
- `ellipke` computes the complete elliptic integrals of the first and second kinds simultaneously.

toeplitz function for creating Toeplitz matrices

The new MATLAB symbolic `toeplitz` function generates a symbolic Toeplitz matrix from two vectors that specify its first column and first row. This function can also generate a symmetric Toeplitz matrix from one vector.

The MuPAD `linalg::toeplitz` function now generates a Toeplitz matrix from two vectors that specify its first column and first row. (In MuPAD, vectors are created as 1-by- n or n -by-1 matrices.) `linalg::toeplitz` accepts the new syntaxes along with the existing syntaxes.

sqrtm function for computing square roots of matrices

The MATLAB symbolic sqrtm function computes the square root of a symbolic matrix.

sign function for computing signs of numbers

The MATLAB symbolic `sign` function returns signs of symbolic real and complex values. The sign of a complex value z is defined as $z/\text{abs}(z)$.

Real option of the `linalg::orthog` function for avoiding complex conjugates

The MuPAD `linalg::orthog` function accepts the new `Real` option. This option lets you avoid using a complex scalar product in the orthogonalization process.

Real option of the `linalg::factorCholesky` function for avoiding complex conjugates

Compatibility Considerations: Yes

The MuPAD `linalg::factorCholesky` function accepts the new `Real` option. When you use this option, `linalg::factorCholesky` assumes that the input matrix is real and symmetric, and does not apply complex conjugation in the course of the algorithm.

Compatibility Considerations

`linalg::factorCholesky` can now compute the Cholesky factorization of a complex Hermitian positive definite matrix. In previous releases, `linalg::factorCholesky` required the input matrix to be symmetric even when working with complex entries. To get the same results as in previous releases for symmetric matrices, use the `Real` option.

New arguments of the svd function for computing the “economy size” singular value decomposition

svd accepts the new arguments 0 and 'econ' that let you compute the “economy size” singular value decomposition of a matrix.

isequaln function for testing equality of symbolic objects

The MATLAB `isequaln` function tests symbolic objects for equality, treating NaN values as equal.

Control over the order in which solve and vpsolve functions return solutions

Compatibility Considerations: Yes

The MATLAB symbolic `solve` and `vpsolve` functions now let you control the order in which they return solutions. To ensure the order of the returned solutions, explicitly specify the independent (input) variables. For example, the syntax `[b,a] = solve(eqns,b,a)` guarantees the order of the returned solutions, while the syntax `[b,a] = solve(eqns)` does not.

Functionality Being Removed or Changed

Compatibility Considerations: Yes

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
simple	Still runs	simplify	Replace all instances of <code>simple(S)</code> with <code>simplify(S)</code> . There is no replacement for <code>[r, how] = simple(S)</code> .
emlBlock	Warns	matlabFunctionBlock	Replace all instances of <code>emlBlock</code> with <code>matlabFunctionBlock</code> .
MuPAD factoring functions <code>numlib::mpqs</code> , <code>numlib::pollard</code> , and <code>numlib::ecm</code>	Warns	ifactor	Replace all instances of <code>numlib::mpqs</code> , <code>numlib::pollard</code> , and <code>numlib::ecm</code> with <code>ifactor</code> .
MuPAD <code>Dom::SparseMatrixF2</code> domain	Warns	<code>Dom::Matrix(Dom::IntegerMod(2))</code>	Replace all instances of <code>Dom::SparseMatrixF2</code> with <code>Dom::Matrix(Dom::IntegerMod(2))</code> .
MuPAD <code>userinfo</code>	Warns	<code>print</code>	Use <code>print</code> instead of <code>userinfo</code> .
MuPAD <code>setuserinfo</code>	Warns	<code>prog::trace</code> or <code>debug</code>	Try using <code>prog::trace</code> or <code>debug</code> instead of <code>setuserinfo</code> .
<code>poly</code>	Errors	<code>charpoly</code>	Replace all instances of <code>poly</code> with <code>charpoly</code> .
<code>sqrt</code> target of the MuPAD <code>simplify</code> function	Errors	MuPAD <code>radsimp</code> or <code>simplifyRadical</code>	Replace all instances of <code>simplify</code> function calls involving the <code>sqrt</code> target with <code>radsimp</code> or <code>simplifyRadical</code> . Alternatively, replace these calls with <code>simplify</code> function calls without targets.

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
cos, sin, exp, and ln targets of the MuPAD simplify function	Errors	MuPAD simplify without targets	Replace all instances of simplify function calls involving these targets with simplify function calls without targets. This can lead to a better simplification for some expressions.
MuPAD transform::fourier	Errors	MuPAD fourier	Replace all instances of transform::fourier with fourier.
MuPAD transform::invfourier	Errors	MuPAD ifourier	Replace all instances of transform::invfourier with ifourier.
MuPAD transform::laplace	Errors	MuPAD laplace	Replace all instances of transform::laplace with laplace.
MuPAD transform::invlaplace	Errors	MuPAD ilaplace	Replace all instances of transform::invlaplace with ilaplace.
MuPAD transform::ztrans	Errors	MuPAD ztrans	Replace all instances of transform::ztrans with ztrans.
MuPAD transform::invztrans	Errors	MuPAD iztrans	Replace all instances of transform::invztrans with iztrans.
MuPAD transform::fourier::addpattern	Errors	MuPAD fourier::addpattern	Replace all instances of transform::fourier::addpattern with fourier::addpattern.

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
MuPAD <code>transform::invfourier::addpattern</code>	Errors	MuPAD <code>ifourier::addpattern</code>	Replace all instances of <code>transform::invfourier::addpattern</code> with <code>ifourier::addpattern</code> .
MuPAD <code>transform::laplace::addpattern</code>	Errors	MuPAD <code>laplace::addpattern</code>	Replace all instances of <code>transform::laplace::addpattern</code> with <code>laplace::addpattern</code> .
MuPAD <code>transform::invlaplace::addpattern</code>	Errors	MuPAD <code>ilaplace::addpattern</code>	Replace all instances of <code>transform::invlaplace::addpattern</code> with <code>ilaplace::addpattern</code> .
MuPAD <code>transform::ztrans::addpattern</code>	Errors	MuPAD <code>ztrans::addpattern</code>	Replace all instances of <code>transform::ztrans::addpattern</code> with <code>ztrans::addpattern</code> .
MuPAD <code>transform::invztrans::addpattern</code>	Errors	MuPAD <code>iztrans::addpattern</code>	Replace all instances of <code>transform::invztrans::addpattern</code> with <code>iztrans::addpattern</code> .
MuPAD <code>prog::calledFrom</code>	Errors	<code>context(hold(procname))</code>	Replace all instances of <code>prog::calledFrom()</code> with <code>context(hold(procname))</code> .
MuPAD <code>prog::calltree</code>	Errors	<code>prog::trace</code>	Use <code>prog::trace</code> instead of <code>prog::calltree</code> .
MuPAD <code>prog::error</code>	Errors	<code>getlasterror</code>	Use <code>getlasterror</code> instead of <code>prog::error</code> .
MuPAD <code>prog::memuse</code>	Errors	<code>prog::trace(Mem)</code> or <code>bytes()</code>	Use <code>prog::trace(Mem)</code> or <code>bytes()</code> instead of <code>prog::memuse</code> .
MuPAD <code>prog::testfunc</code>	Errors	<code>print(Unquoted, "...")</code>	Use <code>print(Unquoted, "...")</code> instead of <code>prog::testfunc</code> .

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
MuPAD <code>prog::testmethod</code>	Errors	<code>prog::test(..., Method = myTestMethod)</code>	Use <code>prog::test(..., Method = myTestMethod)</code> instead of <code>prog::testmethod</code> .
MuPAD <code>prog::testnum</code>	Errors	Nothing	No replacement
Dynamic modules for MuPAD, including the module, <code>external</code> , and <code>Pref::unloadableModules</code> functions and all functions of the module library	Errors	Nothing	No replacement

R2012b

Version: 5.9
New Features: Yes
Bug Fixes: Yes

MATLAB symbolic matrix analysis functions for characteristic (`charpoly`) and minimal (`minpoly`) polynomials and for norm (`norm`) and condition (`cond`) number

`charpoly` computes the characteristic polynomial of a matrix.

`minpoly` computes the minimal polynomial of a matrix.

`norm` computes the 2-norm (default), 1-norm, Frobenius norm, and infinity norm of a symbolic matrix. It also computes the P-norm, Frobenius norm, infinity norm, and negative infinity norm of a symbolic vector.

`cond` computes the corresponding condition numbers of a matrix.

poles function for determining the poles of an expression

The MATLAB `poles` function determines the poles of a symbolic expression or function. The `poles` function is also implemented in MuPAD.

vpasolve function for solving equations and systems using variable precision arithmetic

The MATLAB `vpasolve` function solves equations and systems of equations numerically.

Functions for converting linear systems of equations to matrix form $AX=B$ (`equationsToMatrix`) and solving matrix equations (`linsolve`)

The MATLAB `equationsToMatrix` function converts a linear system of equations to the matrix form $AX = B$. The function returns the coefficient matrix A and the vector B that contains the right sides of the equations.

The MATLAB `linsolve` function solves linear systems of equations represented in the matrix form $AX = B$. The function also returns the reciprocal of the condition number of the square coefficient matrix A . If A is rectangular, `linsolve` returns the rank of A .

MATLAB symbolic functions for describing pulses: rectangularPulse and triangularPulse

`rectangularPulse` and `triangularPulse` compute the rectangular and triangular pulse functions, respectively.

In MuPAD, the new `rectangularPulse` and `triangularPulse` functions are equivalent to `rectpulse` and `tripulse`, respectively.

MuPAD functions for computing integral and Z-transforms

These new MuPAD functions compute integral and Z-transforms:

- `fourier` computes the Fourier transform. You can specify the parameters of the Fourier transform using the new `Pref::fourierParameters` function.
- `ifourier` computes the inverse Fourier transform. You can specify the parameters of the inverse Fourier transform using the new `Pref::fourierParameters` function.
- `laplace` computes the Laplace transform.
- `ilaplace` computes the inverse Laplace transform.
- `ztrans` computes the Z-transform.
- `iztrans` computes the inverse Z-transform.

MuPAD `Pref::fourierParameters` function for specifying Fourier parameters

The MuPAD `Pref::fourierParameters` function lets you specify parameters for Fourier and inverse Fourier transforms.

MuPAD functions for adding transform patterns

These new MuPAD functions add new patterns for integral and Z-transforms:

- `fourier::addpattern` adds new patterns for the Fourier transform.
- `ifourier::addpattern` adds new patterns for the inverse Fourier transform.
- `laplace::addpattern` adds new patterns for the Laplace transform.
- `ilaplace::addpattern` adds new patterns for the inverse Laplace transform.
- `ztrans::addpattern` adds new patterns for the Z-transform.
- `iztrans::addpattern` adds new patterns for the inverse Z-transform.

MuPAD does not save custom patterns permanently. The new patterns are available in the *current* MuPAD session only.

noFlatten option of the MuPAD proc function for preventing sequence flattening

The MuPAD proc function accepts the new noFlatten option. This option prevents flattening of sequences passed as arguments of the procedure.

testtype uses testtypeDom slot for overloading by the second argument

Compatibility Considerations: Yes

If in the call `testtype(object, T)` the argument `T` is a domain, then the method `testtypeDom` of `T` is called with the arguments `object, T`. If `T` is not a domain, then the method `testtypeDom` of `T::dom` is called with the arguments `object, T`.

Compatibility Considerations

In previous releases, `testtype` used the `testtype` slot for overloading by the second argument.

New upper limit on the number of digits in double

Compatibility Considerations: Yes

By default, the working precision for `double` is now limited to at most by 664 digits. You can explicitly specify a larger precision using `digits`.

Compatibility Considerations

Some results returned by `double` can differ from previous releases. For example, in previous releases `double` approximated the expression

```
x = sym('400!*((exp(2000)+1)/(exp(2000) - 1) - 1)')
```

by 3.2997. Now it approximates this expression by 0.

To get the same result as in previous releases, increase the precision of computations:

```
digits(1000)
double(x)

ans =
    3.2997
```


New definition for real and imag

Compatibility Considerations: Yes

Starting in R2012a, `real` and `imag` are no longer defined via `conj`. They use the MuPAD `Re` and `Im` functions instead.

Compatibility Considerations

In R2011b and earlier, `real` and `imag` are defined via the `conj` function:

```
syms z
real(z)
imag(z)

ans =
z/2 + conj(z)/2

ans =
- (z*i)/2 + (conj(z)*i)/2
```

Therefore, `real` and `imag` can return results in a different form. Results returned by `real` and `imag` now are mathematically equivalent to the results returned in previous releases.

Functionality Being Removed or Changed

Compatibility Considerations: Yes

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
Old syntax of <code>taylor</code>	Errors	New calling syntax	Update all instances of <code>taylor</code> function calls using the new syntax.
<code>char(A,d)</code>	Errors	<code>char(A)</code>	Replace all instances of <code>char(A,d)</code> with <code>char(A)</code> .
<code>poly</code>	Warns	<code>charpoly</code>	Replace all instances of <code>poly</code> with <code>charpoly</code> .
<code>emlBlock</code>	Warns	<code>matlabFunctionBlock</code>	Replace all instances of <code>emlBlock</code> with <code>matlabFunctionBlock</code> .
Ability to create links from MuPAD notebooks to MuPAD documentation pages	Not available	Nothing	No replacement
<code>openmuphlp</code>	Errors	Nothing	No replacement
MuPAD Help Browser	Not available	Documentation Center	MuPAD documentation is now available in Documentation Center.
MuPAD Editor	Not available	MATLAB Editor	Open and edit MuPAD program files (.mu files) in the MATLAB Editor. The MATLAB Editor supports syntax highlighting and smart indenting for these files.

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
<code>psi(k0:k1,X)</code>	Errors	<code>psi(k,X)</code> , where k is a scalar specifying the k th derivative of <code>psi</code> at the elements of X .	Replace all instances of <code>psi(k0:k1,X)</code> with <code>psi(k,X)</code> , where k is a scalar. To modify your code, loop through the values $k0:k1$. For example: <pre>for k = k0:k1 Y(:,k) = psi(k,X); end</pre> In a future release, <code>size(Y)</code> will be <code>size(X)</code> . Modify any code that depends on <code>size(Y)</code> .
<code>sqrt</code> target of the MuPAD <code>simplify</code> function	Errors	MuPAD <code>radsimp</code> or <code>simplifyRadical</code>	Replace all instances of <code>simplify</code> function calls involving the <code>sqrt</code> target with <code>radsimp</code> or <code>simplifyRadical</code> . Alternatively, replace these calls with <code>simplify</code> function calls without targets.
<code>cos</code> , <code>sin</code> , <code>exp</code> , and <code>ln</code> targets of the MuPAD <code>simplify</code> function	Errors	MuPAD <code>simplify</code> without targets	Replace all instances of <code>simplify</code> function calls involving these targets with <code>simplify</code> function calls without targets. This can lead to a better simplification for some expressions.
MuPAD <code>transform::fourier</code>	Warns	MuPAD <code>fourier</code>	Replace all instances of <code>transform::fourier</code> with <code>fourier</code> .

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
MuPAD <code>transform::invfourier</code>	Warns	MuPAD <code>ifourier</code>	Replace all instances of <code>transform::invfourier</code> with <code>ifourier</code> .
MuPAD <code>transform::laplace</code>	Warns	MuPAD <code>laplace</code>	Replace all instances of <code>transform::laplace</code> with <code>laplace</code> .
MuPAD <code>transform::invlaplace</code>	Warns	MuPAD <code>ilaplace</code>	Replace all instances of <code>transform::invlaplace</code> with <code>ilaplace</code> .
MuPAD <code>transform::ztrans</code>	Warns	MuPAD <code>ztrans</code>	Replace all instances of <code>transform::ztrans</code> with <code>ztrans</code> .
MuPAD <code>transform::invztrans</code>	Warns	MuPAD <code>iztrans</code>	Replace all instances of <code>transform::invztrans</code> with <code>iztrans</code> .
MuPAD <code>transform::fourier::addpattern</code>	Warns	MuPAD <code>fourier::addpattern</code>	Replace all instances of <code>transform::fourier::addpattern</code> with <code>fourier::addpattern</code> .
MuPAD <code>transform::invfourier::addpattern</code>	Warns	MuPAD <code>ifourier::addpattern</code>	Replace all instances of <code>transform::invfourier::addpattern</code> with <code>ifourier::addpattern</code> .
MuPAD <code>transform::laplace::addpattern</code>	Warns	MuPAD <code>laplace::addpattern</code>	Replace all instances of <code>transform::laplace::addpattern</code> with <code>laplace::addpattern</code> .
MuPAD <code>transform::invlaplace::addpattern</code>	Warns	MuPAD <code>ilaplace::addpattern</code>	Replace all instances of <code>transform::invlaplace::addpattern</code> with <code>ilaplace::addpattern</code> .

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
MuPAD <code>transform::ztrans::addpattern</code>	Warns	MuPAD <code>ztrans::addpattern</code>	Replace all instances of <code>transform::ztrans::addpattern</code> with <code>ztrans::addpattern</code> .
MuPAD <code>transform::invztrans::addpattern</code>	Warns	MuPAD <code>iztrans::addpattern</code>	Replace all instances of <code>transform::invztrans::addpattern</code> with <code>iztrans::addpattern</code> .
MuPAD <code>prog::calledFrom</code>	Warns	<code>context(hold(procname))</code>	Replace all instances of <code>prog::calledFrom()</code> with <code>context(hold(procname))</code> .
MuPAD <code>prog::calltree</code>	Warns	<code>prog::trace</code>	Use <code>prog::trace</code> instead of <code>prog::calltree</code> .
MuPAD <code>prog::error</code>	Warns	<code>getlasterror</code>	Use <code>getlasterror</code> instead of <code>prog::error</code> .
MuPAD <code>prog::memuse</code>	Warns	<code>prog::trace(Mem)</code> or <code>bytes()</code>	Use <code>prog::trace(Mem)</code> or <code>bytes()</code> instead of <code>prog::memuse</code> .
MuPAD <code>prog::testfunc</code>	Warns	<code>print(Unquoted, "...")</code>	Use <code>print(Unquoted, "...")</code> instead of <code>prog::testfunc</code> .
MuPAD <code>prog::testmethod</code>	Warns	<code>prog::test(..., Method = myTestMethod)</code>	Use <code>prog::test(..., Method = myTestMethod)</code> instead of <code>prog::testmethod</code> .

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
MuPAD prog::testnum	Warns	Nothing	No replacement
Dynamic modules for MuPAD, including the module, external, and Pref::unloadableModules functions and all functions of the module library	Warns	Nothing	No replacement

R2012a

Version: 5.8
New Features: Yes
Bug Fixes: Yes

New Special Functions

The following special functions are available:

- `airy` computes the Airy functions of the first and the second kinds. It also computes the first derivatives of the Airy functions.
- `beta` computes the beta function.
- `erfinv` and `erfcinv` compute the inverse and inverse complementary error functions.
- `factorial` computes the factorial function.
- `nchoosek` computes binomial coefficients.
- `whittakerM` and `whittakerW` compute the Whittaker M and Whittaker W functions.

New Vector Analysis Functions

The following vector analysis functions are available:

- `curl` computes the curl of a vector field.
- `divergence` computes the divergence of a vector field.
- `laplacian` computes the laplacian of a scalar function.
- `potential` computes the scalar potential of a vector field.
- `vectorPotential` computes the vector potential of a three-dimensional vector field.

Computations with Symbolic Functions

The toolbox lets you create symbolic functions. For details, see [Creating Symbolic Functions](#).

`dsolve`, `ezplot`, the new `odeToVectorField` function, and other Symbolic Math Toolbox™ functions now support computations with symbolic functions.

The toolbox also provides the following functions to support common operations on symbolic functions:

- `argnames` returns a symbolic array of all input variables of a symbolic function.
- `formula` returns a mathematical expression that defines the symbolic function.

Assumptions on Variables

You can set assumptions on symbolic variables by using these functions:

- `assume` sets assumptions on symbolic variables.
- `assumeAlso` adds assumptions on symbolic variables without erasing the previous assumptions.
- `assumptions` shows assumptions set on symbolic variables.

New Relational Operators Create Equations, Inequalities, and Relations

Compatibility Considerations: Yes

Use these relational operators to create symbolic equations, inequalities, and relations:

- `==` and its functional form `eq` create a symbolic equation. You can solve these equations using `solve` or `dsolve`, plot them using `ezplot`, set assumptions using `assume` or `assumeAlso`, or use them in logical expressions.
- `~=` and its functional form `ne` create a symbolic inequality. You can use inequalities in assumptions and logical expressions.
- `>`, `>=`, `<`, `<=`, and their functional forms `ge`, `gt`, `le`, and `lt` create symbolic relations. You can use relations in assumptions and logical expressions.

Compatibility Considerations

In previous releases, `eq` evaluated equations and returned logical 1 or 0. Now it returns unevaluated equations letting you create equations that you can pass to `solve`, `assume`, and other functions. To obtain the same results as in previous releases, wrap equations in `logical` or `isAlways`. For example, use `logical(A == B)`.

New Logical Operators Create Logical Expressions

Use these logical operations let you create logical expressions with symbolic subexpressions:

- `&` or its functional form `and` defines the logical conjunction (the logical AND) for symbolic expressions.
- `|` or its functional form `or` defines the logical disjunction (the logical OR) for symbolic expressions.
- `~` or its functional form `not` defines the logical negation (the logical NOT) for symbolic expressions.
- `xor` defines the logical exclusive disjunction (the logical XOR) for symbolic expressions.

If logical expressions are elements of a symbolic array, you can use these new functions to test the logical expressions:

- `all` tests whether all equations and inequalities represented as elements of a symbolic array are valid.
- `any` tests whether at least one of equations and inequalities represented as elements of a symbolic array is valid.

New Functions Test Validity of Symbolic Equations, Inequalities, and Relations

Use these functions to test symbolic equations, inequalities, and relations, including logical statements:

- `isAlways` checks whether an equation, inequality, or relation holds for all values of its variables.
- `logical` checks the validity of an equation, inequality, or relation. This function does not simplify or mathematically transform expressions that form an equation, inequality, or relation. It also typically ignores assumptions on variables.

New Functions Manipulate Symbolic Expressions

These functions provide more flexible options for manipulating symbolic expressions:

- The `rewrite` function rewrites expressions in terms of target functions. It returns a mathematically equivalent form of an expression using the specified target functions. For example, it can rewrite trigonometric expressions using the exponential function.
- `children` returns child subexpressions, or terms, of a symbolic expression.

New `odeToVectorField` Function Converts Higher-Order Differential Equations to Systems of First-Order Differential Equations

`odeToVectorField` converts second- and higher-order differential equations to systems of first-order differential equations. It returns a symbolic vector representing the resulting system of first-order differential equations. With `matlabFunction` you can generate a MATLAB function from this vector, and then use it as an input for the MATLAB numerical solvers `ode23` and `ode45`.

In MuPAD, the new `numeric::odeToVectorField` function is equivalent to `numeric::ode2vectorfield`.

New Calling Syntax for the `taylor` Function

Compatibility Considerations: Yes

The `taylor` function that computes the Taylor series expansions has a new syntax and set of options.

Compatibility Considerations

The new syntax is not valid before Version 5.8. The old syntax is still supported, but will be removed in a future release. To update existing code that relies on the old syntax, make the following changes to the `taylor` function calls:

- Specify the truncation order using the name-value pair argument `Order`.
- Specify the expansion point using the name-value pair argument `ExpansionPoint`.

Alternatively, specify the expansion point as a third input argument. In this case, you must also specify the independent variable or the vector of variables as the second input argument.

For details and examples, see `taylor`.

New MuPAD Functions Compute Rectangular and Triangular Pulse Functions

The MuPAD `rectpulse` and `tripulse` functions compute the rectangular and triangular pulse functions, respectively.

MuPAD `det`, `linalg::det`, `inverse`, `linsolve`, and `linalg::matlinsolve` Functions Accept the New Normal Option

The MuPAD `det`, `linalg::det`, `inverse`, `linsolve`, and `linalg::matlinsolve` functions accept the new `Normal` option that guarantees normalization of the returned results. The `_invert` methods of the MuPAD `Dom::Matrix(R)` and `Dom::DenseMatrix(R)` domains also accept `Normal`.

MuPAD `linalg::matlinsolve` Function Accepts the New `ShowAssumptions` Option

The MuPAD `linalg::matlinsolve` function accepts the new `ShowAssumptions` option. This option lets you see internal assumptions on symbolic parameters that `linalg::matlinsolve` makes while solving a system of equations.

Enhanced MuPAD pdivide Function

Enhanced MuPAD pdivide function now performs pseudo-division of multivariate polynomials.

Improved MuPAD prog::remember Function

Improved MuPAD prog::remember function, which lets you use the remember mechanism in procedures streamlines such processes as debugging, profiling, and argument checking.

Functionality Being Removed or Changed

Compatibility Considerations: Yes

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
Old syntax of <code>taylor</code>	Warns	New syntax	Update all instances of <code>taylor</code> function calls using the new syntax.
Default number of simplification steps in <code>simplify</code> has changed from 50 to 100.	Uses the new default setting	<code>simplify(S, 'Steps', 50)</code>	To terminate algebraic simplification after 50 steps, call <code>simplify</code> with the name-value pair argument <code>'Steps', 50</code> .
<code>char(A,d)</code>	Warns	<code>char(A)</code>	Replace all instances of <code>char(A,d)</code> with <code>char(A)</code> .
<code>emlBlock</code>	Warns	<code>matlabFunctionBlock</code>	Replace all instances of <code>emlBlock</code> with <code>matlabFunctionBlock</code> .
<code>psi(k0:k1,X)</code>	Warns	<code>psi(k,X)</code> where <code>k</code> is a scalar specifying the <code>k</code> th derivative of <code>psi</code> at the elements of <code>X</code> .	<p>Replace all instances of <code>psi(k0:k1,X)</code> with <code>psi(k,X)</code>, where <code>k</code> is a scalar. To modify your code, loop through the values <code>k0:k1</code>. For example:</p> <pre>for k = k0:k1 Y(:,k) = psi(k,X); end</pre> <p>In the future, <code>size(Y)</code> will be <code>size(X)</code>. Modify any code that depends on <code>size(Y)</code>.</p>

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
<code>sqrt</code> target of the MuPAD <code>simplify</code> function	Warns	MuPAD <code>radsimp</code> or <code>simplifyRadical</code>	Replace all instances of <code>simplify</code> function calls involving the <code>sqrt</code> target with <code>radsimp</code> or <code>simplifyRadical</code> . Alternatively, replace these calls with <code>simplify</code> function calls without targets.
<code>cos</code> , <code>sin</code> , <code>exp</code> , and <code>ln</code> targets of the MuPAD <code>simplify</code> function	Warns	MuPAD <code>simplify</code> without targets	Replace all instances of <code>simplify</code> function calls involving these targets with <code>simplify</code> function calls without targets. This can lead to a better simplification for some expressions.
MuPAD <code>frame</code> function	Errors	Nothing	No replacement

R2011b

Version: 5.7
New Features: Yes
Bug Fixes: Yes

MATLAB Editor Now Supports MuPAD Program Files

You can open and edit MuPAD program files (.mu files) in the MATLAB Editor. MATLAB Editor supports syntax highlighting and smart indenting for these files.

dsolve, expand, int, simple, simplify, and solve Accept More Options

dsolve now accepts the IgnoreAnalyticConstraints and MaxDegree options.

expand now accepts the ArithmeticOnly and IgnoreAnalyticConstraints options.

int now accepts the IgnoreAnalyticConstraints, IgnoreSpecialCases, and PrincipalValue options.

simple now accepts the IgnoreAnalyticConstraints option.

simplify now accepts the IgnoreAnalyticConstraints, Seconds, and Steps options.

solve now accepts the IgnoreAnalyticConstraints, IgnoreProperties, MaxDegree, PrincipalValue, and Real options.

new read Function Reads MuPAD Program Files in MATLAB

`read` simplifies using your own MuPAD procedures in MATLAB. See [Before Calling a Procedure](#) for details.

New symprod Function Computes Products of Series

symprod computes definite and indefinite products of symbolic series.

New `hessian` Function Computes Hessian Matrices

`hessian` computes the Hessian matrix of a scalar function.

New gradient Function Computes Vector Gradients

`gradient` computes the vector gradient of a scalar function in Cartesian coordinates. In MuPAD, the new `linalg::gradient` function is equivalent to `linalg::grad`.

New erfc Function Computes the Complementary Error Function

`erfc` computes the complementary error function.

New psi Function Computes the Digamma and Polygamma Functions

psi computes the digamma and polygamma functions.

New wrightOmega Function Computes the Wright omega Function

wrightOmega computes the Wright omega function.

New simplifyFraction Function Simplifies Expressions

`simplifyFraction` returns a simplified form of a fraction where both numerator and denominator are polynomials and their greatest common divisor is 1. In MuPAD, the new `simplifyFraction` function is equivalent to `normal`.

New MuPAD `simplifyRadical` Function Simplifies Radicals in Arithmetical Expressions

The new MuPAD `simplifyRadical` function is equivalent to the MuPAD `radsimp` function.

pretty Function Now Uses Abbreviations in Long Output Expressions for Better Readability

pretty uses abbreviations when presenting symbolic results in the MATLAB Command Window. This new format of presenting symbolic results enhances readability of long output expressions.

MuPAD normal Function Accepts the New Expand Option

Compatibility Considerations: Yes

The MuPAD `normal` function accepts the new `Expand` option that determines whether numerators and denominators of fractions are expanded.

Compatibility Considerations

In previous releases, `normal` returned a fraction with the expanded numerator and denominator by default. Now the default setting is that `normal` can return factored expressions in numerators and denominators. In explicit calls to `normal`, you can use the `Expand` option to get the same behavior as in previous releases.

If a function calls `normal` internally, then that function can return its results in a different form. These new results are mathematically equivalent to the results that you get in previous releases. Many MuPAD library functions can call `normal`.

Modified MuPAD groebner Library

All functions of the MuPAD groebner library now can accept and return polynomials with arbitrary arithmetical expressions.

MuPAD `groebner::gbasis` Function Accepts the New `Factor` and `IgnoreSpecialCases` Options

With the `Factor` option, `groebner::gbasis` returns a set of lists, such that:

- Each list is the Groebner basis of an ideal.
- The union of these ideals is a superset of the ideal given as input, and a subset of the radical of that ideal.

With the `IgnoreSpecialCases` option, `groebner::gbasis` handles all coefficients in all intermediate results as nonzero unless these coefficients are equal to 0 for all parameter values.

New MuPAD Functions for Computing Logarithms

The new MuPAD `log2` and `log10` functions compute logarithms to the bases 2 and 10, respectively. Also, in MuPAD `log(x)` is now an alias for `ln(x)`.

Functionality Being Removed or Changed

Compatibility Considerations: Yes

Functionality	What Happens When You Use It?	Use This Instead	Compatibility Considerations
<code>emlBlock</code>	Warns	<code>matlabFunctionBlock</code>	Replace all instances of <code>emlBlock</code> with <code>matlabFunctionBlock</code> .
Real and <code>IgnoreProperties</code> options in MuPAD <code>ode::solve</code>	Warns	<code>IgnoreSpecialCases</code> or <code>IgnoreAnalyticConstraints</code>	Try using <code>IgnoreSpecialCases</code> or <code>IgnoreAnalyticConstraints</code> instead.

R2011a

Version: 5.6
New Features: Yes
Bug Fixes: Yes

Expression Wrapping of Math Output in the MuPAD Notebook Interface

The new default format of presenting results enhances readability by wrapping long output expressions, including long numbers, fractions and matrices.

Symbolic Solver Handles More Non-Algebraic Equations

The enhanced `rationalize` function in MuPAD helps the symbolic solver to handle more systems of non-algebraic equations. In particular, this improvement enables the toolbox to solve more systems of trigonometric equations.

Improved Performance in the Ordinary Differential Equation Solver

The ordinary differential equation solver demonstrates better performance.

Improved Performance for Polynomial Arithmetic Operations

The MuPAD functions `gcdex`, `partfrac`, `polylib::resultant`, and `solveLib::pdioe` now demonstrate better performance.

New MuPAD `polylib::subresultant` Function Computes Subresultants of Polynomials

`polylib::subresultant` computes subresultants of two polynomials or polynomial expressions.

MuPAD partfrac Function Accepts the New List Option

With the new `List` option, `partfrac` returns a list consisting of the numerators and denominators of the partial fraction decomposition.

New MuPAD `inverf` and `inverfc` Functions Compute the Inverses of Error Functions

The `inverf` function computes the inverse of the error function.

The `inverfc` function computes the inverse of the complementary error function.

New MuPAD numlib::checkPrimalityCertificate Function Tests Primality Certificates

`numlib::checkPrimalityCertificate` tests primality certificates returned by `numlib::proveprime`. For information about proving primality of numbers, see “Proving Primality” in the MuPAD documentation.

New Demos

There are three new demos that show how to solve equations and compute derivatives and integrals:

- Solving Algebraic and Differential Equations
- Differentiation
- Integration

To run the new demos, enter `symeqndemo`, `syndiffdemo`, or `symintdemo` in the MATLAB Command Window.

Functionality Being Removed or Changed

Compatibility Considerations: Yes

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
MuPAD <code>matchlib::analyze</code>	Errors	MuPAD <code>prog::exprtree</code>	To visualize expressions, use <code>prog::exprtree</code> .
MuPAD <code>prog::testcall</code>	Errors	Nothing	No replacement
MuPAD <code>prog::testerrors</code>	Errors	Nothing	No replacement
Old syntax of MuPAD <code>prog::getOptions</code>	Errors	The new syntax	Update all instances of <code>prog::getOptions</code> calls using the new syntax.
Old syntax of MuPAD <code>prog::trace</code>	Errors	The new syntax	Update all instances of <code>prog::trace</code> calls using the new syntax.

R2010b

Version: 5.5
New Features: Yes
Bug Fixes: Yes

sym Function Creates Matrices of Symbolic Variables

The `sym` function now provides a shortcut for creating vectors and matrices of symbolic variables.

For more information, see [Creating a Matrix of Symbolic Variables](#).

generate::Simscape Function Generates Simscape Equations from MuPAD Expressions

The new MuPAD function `generate::Simscape` converts MuPAD expressions to Simscape™ equations.

MuPAD Code Generation Functions Accept the New NoWarning Option

MuPAD functions `generate::C`, `generate::fortran`, `generate::MATLAB`, and `generate::Simscape` accept the new `NoWarning` option. The option suppresses all warnings issued by these functions.

Improved MuPAD Hyperlink Dialog Box

Creating and editing links in MuPAD has become easier with the improved Hyperlink dialog box.

MuPAD Notebook Highlights Matched and Unmatched Delimiters

MuPAD Notebook now can notify you about matched and unmatched delimiters such as parentheses, brackets, and braces.

Improved Performance When Solving Linear Systems in a Matrix Form

MuPAD `linalg::matlinsolve` function, which solves linear systems of equations in a matrix form, demonstrates better performance.

MuPAD Solver for Ordinary Differential Equations Handles More Equation Types

Enhanced MuPAD solver handles more first-order nonlinear and third-order linear ordinary differential equations. The solver demonstrates improved performance.

New Syntax for the MuPAD prog::getOptions Function

Compatibility Considerations: Yes

The prog::getOptions function that collects and verifies options within a procedure has the new syntax.

Compatibility Considerations

The new syntax is not valid in MuPAD versions earlier than 5.5. The old syntax is supported in MuPAD 5.5, but will be removed in a future release.

New Syntax for the MuPAD `prog::trace` Function

Compatibility Considerations: Yes

The `prog::trace` function used for debugging has the new syntax. The function observes entering and exiting the MuPAD functions.

Compatibility Considerations

The new syntax is not valid in MuPAD versions earlier than 5.5. The old syntax is not supported in MuPAD 5.5.

Improved Interface for Arithmetical Operations on Polynomials

Improved interface for arithmetical operations between polynomials and arithmetical expressions. In previous releases, to perform an arithmetical operation on a polynomial and an arithmetical expression, you must explicitly convert that expression to a polynomial of the corresponding type. Now, when you operate on a polynomial and an arithmetical expression, MuPAD internally converts the arithmetical expression to a polynomial and performs the calculation.

MuPAD igcd Function Now Accepts Complex Numbers as Arguments

The MuPAD `igcd` function, which computes the greatest common divisor of integers, now accepts complex numbers. Both real and imaginary parts of accepted complex numbers must be integers or arithmetic expressions that represent integers.

Enhanced Solver For Factorable Polynomial Systems

The MuPAD `solve` function performs better on factorable polynomial systems.

MuPAD Now Evaluates Large Sums with Subtractions Faster

Compatibility Considerations: Yes

MuPAD performs evaluations of large sums that contain subtractions faster than in previous releases.

Compatibility Considerations

In MuPAD, the difference operator ($-$) no longer invokes the `_subtract` function. Instead, it invokes the `_plus` and `_negate` functions. For example, `a - b` is equivalent to `_plus(a, _negate(b))`.

MuPAD freeIndets Function Accepts the New All Option

The `freeIndets` function accepts the new `All` option. With this option, `freeIndets` does not exclude the 0th operand from the list of free identifiers.

Functionality Being Removed or Changed

Compatibility Considerations: Yes

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
diff and int methods for inputs of the char type	Warns	sym	Use the sym method instead.
MuPAD matchlib::analyze	Warns	MuPAD prog::exptree	To visualize expressions, use prog::exptree.
MuPAD prog::testcall	Warns	None	No replacement
MuPAD prog::testerrors	Warns	None	No replacement
The following options in MuPAD prog::trace: <ul style="list-style-type: none"> • All • Backup • Force • Name • Proc • Plain • Width 	Errors	None	No replacement. These options are not supported in the current release.
Global properties in MuPAD	Errors	Assumptions on each variable	Make assumptions on each variable instead.

R2010a

Version: 5.4
New Features: Yes
Bug Fixes: Yes

When Opening Notebook, MuPAD Can Jump to Particular Locations

The `mupad` command that opens a MuPAD notebook now supports references to particular places inside a notebook. You can create a link target inside a notebook and refer to it when opening a notebook.

simscapeEquation Function Generates Simscape Equations from Symbolic Expressions

The new `simscapeEquation` command represents symbolic expressions in the form of Simscape equations. For more information, see [Generating Simscape Equations in the Symbolic Math Toolbox documentation](#).

New Calling Syntax for the sort Function

Compatibility Considerations: Yes

The sort function that sorts the element of symbolic arrays and polynomials has the new syntax and set of options.

Compatibility Considerations

In previous releases, the sort function flattened symbolic matrices to vectors before sorting the elements. Now the sort function sorts the elements of each column or each row of a symbolic matrix. If you want to obtain the same results as in the previous release, flatten the symbolic matrix before sorting it: `sort(A(:))`.

Changes in the symengine Function

The toolbox no longer supports the ability to choose an alternative symbolic engine.

64-Bit GUI Support for Macintosh

MuPAD now supports 64-bit graphical user interfaces (such as notebooks and Editor and Debugger windows) for a 64-bit Macintosh operating system.

New MuPAD Print Preview Dialog

Adjusting MuPAD documents for printing is easier with the new Print Preview dialog. You can view one or several pages, zoom in and out, switch between page orientations, adjust the page settings without closing the dialog, and print the page or save it to PDF format

Improved Configure MuPAD Dialog Box

Specifying the default settings for graphical user interfaces, such as notebooks and Editor and Debugger windows, has become easier with the improved configuration dialog box.

MuPAD Support for Basic Arithmetic Operations for Lists

Basic arithmetic operations now work for lists.

Improved Performance When Operating on Matrices with Symbolic Elements

MuPAD demonstrates better performance when handling some linear algebra operations on matrices containing symbolic elements.

Enhanced MuPAD divide Function

Enhanced MuPAD `divide` function computes the quotient and remainder for division of multivariate polynomials.

Improved Performance for Operations on Polynomials

Compatibility Considerations: Yes

Improved performance for conversions involving polynomials. Improved performance for operations on polynomials including evaluation, multiplication, and division.

Compatibility Considerations

If the coefficients of a polynomial contain the variables of the polynomial itself, the form of results returned by the MuPAD `poly` function can differ from previous releases. In previous releases, the `poly` function converted such coefficients to monomials. Now the `poly` function can return the coefficients of the original expression as coefficients in the resulting polynomial. To get the same behavior as in previous releases, use `expr` to convert an original polynomial into an expression, and then call the `poly` function. For example, the following call exercises the old behavior: `poly(expr(p), [y, x])`.

MuPAD coeff Function Accepts the New All Option

The `coeff` function accepts the `newAll` option. With this option, `coeff` returns all coefficients of a polynomial including those equal to 0.

MuPAD expand Function Accepts the New ArithmeticOnly Option

The `expand` function accepts the new `ArithmeticOnly` option. The option allows you to expand a sum without expanding trigonometric expressions and special functions in its terms. Technically, the option omits overloading the `expand` function for each term of the original expression.

MuPAD expand Function Now Expands Powers of Products

The expand function now expands powers of products such as $(xy)^n$ for positive x and y . When called with the `IgnoreAnalyticConstraints` option, the function expands the power of products for arbitrary terms.

New Calling Syntax for MuPAD rationalize Function

Compatibility Considerations: Yes

The `rationalize` function that transforms an arbitrary expression into a rational expression has the new syntax and set of options.

Compatibility Considerations

The new syntax is not valid in MuPAD versions earlier than 5.4. The old syntax is supported in MuPAD 5.4, but will be removed in a future release.

Enhanced MuPAD simplify and Simplify Functions

Enhanced simplification functions, `simplify` and `Simplify`, demonstrate better results for expressions involving trigonometric and hyperbolic functions, square roots, and sums over roots of unity.

MuPAD subs Function Accepts the New EvalChanges Option

The subs function now accepts the new EvalChanges option. By default, subs does not evaluate an expression after making substitutions. With this option, subs evaluates all subexpressions that contain substitutions.

MuPAD Solver for Ordinary Differential Equations Handles More Equation Types

Enhanced MuPAD solver handles more second-order linear and first-order nonlinear ordinary differential equations. The solver demonstrates improved performance.

Functionality Being Removed or Changed

Compatibility Considerations: Yes

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
MuPAD Domain <code>Dom::Ideal</code>	Errors	<code>groebner</code>	Represent ideals as lists, and use functions of the <code>groebner</code> package instead.
MuPAD student library	Errors	<code>plot::Integral</code> and <code>linalg</code>	Use <code>plot::Integral</code> and the <code>linalg</code> package instead.
MuPAD relation option in <code>simplify</code>	Errors	None	No replacement
Global property	Warns	Assumptions on each variable	Make assumptions on each variable instead.
<code>digits</code> and <code>vpa</code> do not let you set the number of digits to 1.	Errors	Errors	It is no longer possible to set the number of digits to 1 when using the <code>digits</code> and <code>vpa</code> functions. The Symbolic Math Toolbox software version number 4.9 and lower allowed you to set the number of digits to 1.

R2009b

Version: 5.3
New Features: Yes
Bug Fixes: Yes

Support for Windows x64 and 64-Bit Macintosh

The toolbox now supports 64-bit Windows® and Macintosh operating systems. If you work in the MuPAD Notebook Interface on a 64-bit Macintosh operating system, MuPAD runs a 64-bit engine with 32-bit graphical user interfaces, such as notebooks and Editor and Debugger windows.

sym and syms Use Reserved Words as Variable Names

Compatibility Considerations: Yes

sym and syms commands now treat reserved MuPAD words, except pi, as variable names.

Compatibility Considerations

In previous releases, the reserved words returned MuPAD values. If your code uses the reserved words as MuPAD commands, modify your code and use the `evalin` command with the reserved word as a name. For example, use `evalin(symengine, 'beta')`.

Toolbox Now Displays Floating-Point Results with Their Original Precision

Compatibility Considerations: Yes

The toolbox now displays the floating-point results with the original precision with which the toolbox returned them.

Compatibility Considerations

In previous releases, the toolbox displayed floating-point results with the current precision. You must update the existing code that relies on the output precision for displaying floating-point numbers. Use `digits` to set the precision you need before computing such results. The toolbox displays the results with the same number of digits it used to compute the results. The toolbox also can increase the specified precision of calculations by several digits.

In previous releases, `sym(A, 'f')` represented numbers in the form $(2^e + N \cdot 2^{(e - 52)})$ or $-(2^e + N \cdot 2^{(e - 52)})$, with integers for N and e , and $N \neq 0$. Now `sym(A, 'f')` displays results in the rational form that actually represents the double-precision floating-point numbers.

New MuPAD Preference Pref::outputDigits Controls Floating-Point Outputs

New preference Pref::outputDigits controls the precision MuPAD uses to display floating-point results.

Solver for Ordinary Differential Equations Handles More Equation Types

Enhanced solvers handle more equation types of second-order homogeneous linear ordinary differential equations. The solver demonstrates improved performance.

MuPAD limit Function Supports Limits for Incomplete Gamma Function and Exponential Integral Function

Enhanced limit function now can compute limits for incomplete Gamma function and exponential integral function.

Enhanced Simplification Routines for MuPAD Special Functions

Enhanced simplification routines for MuPAD hypergeom, meyerG, and besse1 special functions.

Enhanced MuPAD combine Function for Logarithms

Enhanced combine function demonstrates better performance for logarithms.

MuPAD normal Function Accepts New Options

The normal command now accepts the options NoGcd, ToCancel, Rationalize, Recursive, and Iterations. The options control costly operations, such as recognizing greatest common divisors and algebraic dependencies.

Functionality Being Removed or Changed

Compatibility Considerations: Yes

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
MuPAD Domain <code>Dom::Ideal</code>	Warns	<code>groebner</code>	Represent ideals as lists, and use functions of the <code>groebner</code> package instead.
MuPAD student library	Warns	<code>plot::Integral</code> and <code>linalg</code>	Use <code>plot::Integral</code> and the <code>linalg</code> package instead.
<code>d in char(A, d)</code>	Warns	None	No replacement
MuPAD relation option in <code>simplify</code>	Warns	None	No replacement

R2009a

Version: 5.2
New Features: Yes
Bug Fixes: Yes

dsolve Accepts the New Option IgnoreAnalyticConstraints

Compatibility Considerations: Yes

The `dsolve` command now accepts the option `IgnoreAnalyticConstraints`. The option controls the level of mathematical rigor that the solver uses on the analytical constraints on the solution. By default, the solver ignores all analytical constraints.

Compatibility Considerations

The results of the `dsolve` command can differ from those returned in the previous release. If you want to obtain the same solutions as in the previous release, set the value of the option `IgnoreAnalyticConstraints` to `none`.

emlBlock Function Generates Embedded MATLAB Function Blocks from Symbolic Objects

The new `emlBlock` command converts symbolic expressions to Embedded MATLAB® Function Blocks. You can use these blocks in any Simulink® installation, even those without a Symbolic Math Toolbox license. For more information, see [Generating Embedded MATLAB Blocks in the Symbolic Math Toolbox documentation](#).

matlabFunction Improves Control over Input and Output Parameters

Compatibility Considerations: Yes

`matlabFunction` now accepts multiple expressions and cell arrays of symbolic arrays as input parameters. The function now allows you to specify the names of the output parameters.

Compatibility Considerations

In previous releases, the default name of an output variable was `RESULT`. Now the default names of the output variables coincide with the names you use to call `matlabFunction`. You must update existing code that relies on the default output name `RESULT`. You can change your code using any of these methods:

- Define the name of an output variable as `RESULT`.
- Change the name of an input variable to `RESULT`.
- Throughout your code change the variable name from `RESULT` to the input name.

Enhancements to Object-Oriented Programming Capabilities

Compatibility Considerations: Yes

The Symbolic Math Toolbox product uses some object-oriented programming features to implement symbolic objects. Major enhancements to object-oriented programming capabilities enable easier development and maintenance of large applications and data structures. For a full description of object-oriented features, see the MATLAB Object-Oriented Programming documentation.

Compatibility Considerations

It is no longer possible to add methods to `@sym` by creating a `@sym` directory containing custom methods.

For an empty `x`, `sym(x)` returns a symbolic object of the same size as `x`. In previous releases, `sym(x)` returned a symbolic object of size 0-by-0 for an empty `x`.

generate::MATLAB Function Converts MuPAD Expressions to MATLAB Code

The new `generate::MATLAB` command converts MuPAD expressions, equations, and matrices to MATLAB formatted strings.

MuPAD IgnoreAnalyticConstraints Option Specifies That Core Functions Apply Common Algebraic Assumptions to Simplify Results

The new IgnoreAnalyticConstraints option allows the use of a set of simplified mathematical rules when solving equations, simplifying expressions, or integrating. For example, this option applies practical, but not generally correct rules for combining logarithms: $\ln(a) + \ln(b) = \ln(a \cdot b)$

MuPAD Outputs Contain Abbreviations for Better Readability

The new default format of presenting results enhances readability of long output expressions by using abbreviations.

MuPAD Solver for Ordinary Differential Equations Handles More Equation Types

The solver now can handle more than 200 additional types of second-order ordinary differential equations. The solver demonstrates improved performance.

MuPAD limit Function Now Can Compute Limits for Piecewise Functions

The enhanced `limit` function computes limits of piecewise functions including bidirectional and one-sided limits.

New and Improved MuPAD Special Functions

MuPAD includes the following new special functions:

- `laguerreL` represents Laguerre's L function.
- `erfc(x,n)` returns iterated integrals of the complementary error function.
- `meijerG` represents the Meijer G function.

The hypergeom special function demonstrates better performance.

New Calling Syntax for Test Report Function

prog::tcov

Compatibility Considerations: Yes

The `prog::tcov` function that inspects the data collected during the code execution has the new syntax and set of options.

Compatibility Considerations

The new syntax is not valid in MuPAD versions earlier than 5.2. MuPAD 5.2 does not support the earlier syntax.

New Demos

To see new demos that use MuPAD Notebook Interface, type `mupadDemo` at the MATLAB command line or click [MuPAD Notebooks Demo](#).

R2008b

Version: 5.1
New Features: No
Bug Fixes: Yes

R2008a+

Version: 5.0
New Features: No
Bug Fixes: Yes

R2007b+

Version: 4.9
New Features: Yes
Bug Fixes: Yes

MuPAD Engine Replaces Maple Engine

Compatibility Considerations: Yes

The default Symbolic Math Toolbox engine is now the MuPAD engine. For more information, see the MuPAD in Symbolic Math Toolbox chapter in the Symbolic Math Toolbox User's Guide.

Compatibility Considerations

The new engine causes many computed results to differ from those returned by previous versions of Symbolic Math Toolbox software.

General Differences

- Many computations return in a permuted order (such as $a + b$ instead of $b + a$).
- Some computations return in a different, mathematically equivalent form (such as $(\cos(x))^2$ instead of $1 - (\sin(x))^2$).
- `diff(dirac(t))` returns `dirac(t,1)` instead of `dirac(1,t)`.
- `sym(x, 'f')` no longer produces strings of the form `hex digits*2^n`. Instead the strings have the form $(2^e + N*2^{(e-52)})$, where N and e are integers.
- For toolbox calculations, some symbols can only be used as symbolic variables, and not in strings: `E`, `I`, `D`, `O`, `beta`, `zeta`, `theta`, `psi`, `gamma`, `Ci`, `Si`, and `Ei`. This is because those symbols represent MuPAD reserved words, and are interpreted as the MuPAD word if you pass them as strings. The words `Ci`, `Si`, `Ei` represent special mathematical functions: the cosine integral, sine integral, and exponential integral respectively.
- Error and warning message IDs may have changed.
- Performance of numerical integration is slower than in previous versions.
- Subexpressions, calculated by the `subexpr` function, may be different than in previous versions.
- The `pretty` function no longer uses partial subexpressions (with syntax `%n`).

Calculus

- `Int` no longer evaluates some integrals, including many involving Bessel functions.
- `symsum(sin(k*pi)/k,0,n)` no longer evaluates to `pi`.

Linear Algebra

- The output of `colspace` may differ from previous versions, but it is mathematically equivalent.
- The `eig` function may return eigenvalues in a different order than previous versions. Expressions returned by `eig` may be larger than in previous versions.
- The `jordan` function may return diagonal subblocks in a different order than previous versions.
- `svd` may return singular values in a different order than previous versions.

Simplification

- The `coeffs` function may return multivariable terms in a different order than in previous versions.
- The `expand` function may return some trig and exponential expressions differently than in previous versions.
- The `simplify` function involving radicals and powers make fewer assumptions on unknown symbols than in previous versions.
- The `subexpr` function may choose a different subexpression to be the common subexpression than in previous versions.
- Subexpressions no longer have partial subexpressions (previous syntax `%n`).
- The `solve` function returns solutions with higher multiplicity only when solving a single polynomial.
- `acot(-x) = -acot(x)` instead of `pi - acot(x)` as in previous versions.
- `acoth(-x) = -acoth(x)` instead of `2*acoth(0) - acoth(x)` as in previous versions.
- The `simple` function has several differences:

- The 'how' value `combine(trig)` has been replaced with `combine(sincos)`, `combine(sinhcosh)`, and `combine(ln)`.
- The 'how' values involving `convert` have been replaced with `rewrite`.
- A new 'how' value of `mlsimplify(100)` indicates the MuPAD function `Simplify(...,Steps=100)` simplified the expression.
- Simplifications such as $(\sin(x)^2)^{1/2}$ to $\sin(x)$ are no longer performed, since the MuPAD language is careful not to make assumptions about the sign of $\sin(x)$.

Conversion

- Arithmetic involving the `vpa` function uses the current number of digits of precision. Variable precision arithmetic may have different rounding behaviors, and answers may differ in trailing digits (trailing zeros are now suppressed).
- The `char` function returns strings using MuPAD syntax instead of Maple™ syntax.
- Testing equality does not compare strings as in previous versions; the symbolic engine equality test is used.
- Saving and loading symbolic expressions is compatible with previous versions, except when the symbolic contents use syntax or functions that differ between Maple or MuPAD engines. For example, suppose you save the symbolic object `sym('transform::fourier(f,x,w)')`, which has MuPAD syntax. You get a MATLAB error if you try to open the object while using a Maple engine.
- LaTeX output from the `latex` function may look different than before.
- C and Fortran code generated with the `ccode` and `fortran` functions may be different than before. In particular, generated files have intermediate expressions as “optimized” code. For more information, see the Generating C or Fortran Code section of the User’s Guide.
- pretty output may look different than before.

Equation Solving

- `solve` returns solutions with higher multiplicity only when solving a single polynomial.
- `solve` may return a different number of solutions than before.
- Some calls to `dsolve` that used to return results involving `lambertw` now return no solution.
- `dsolve` can now use the variable `C`.
- Some `dsolve` results are more complete (more cases are returned).
- Some `dsolve` results are less complete (not all previous answers are found).
- `finverse` may be able to find inverses for different classes of functions than before.
- When `finverse` fails to find an explicit inverse, it produces different output than before.

Transforms

- Fourier and inverse Fourier transforms return the MuPAD form `transform::fourier` when they cannot be evaluated. For example,

```
h = sin(x)/exp(x^2);  
FF = fourier(h)
```

```
FF =  
transform::fourier(sin(x)/exp(x^2), x, -w)
```

The reason for this behavior is the MuPAD definition of Fourier transform and inverse Fourier transform differ from their Symbolic Math Toolbox counterparts by the sign in the exponent:

	Symbolic Math Toolbox definition	MuPAD definition
Fourier transform	$F(w) = \int_{-\infty}^{\infty} f(x)e^{-iwx} dx$	$F(w) = \int_{-\infty}^{\infty} f(x)e^{iwx} dx$
Inverse Fourier transform	$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(w)e^{iwx} dw$	$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(w)e^{-iwx} dw$

- Several Fourier transforms can no longer be calculated, especially those involving Bessel functions.
- ztrans and iztrans may return more complicated expressions than before.

Special Mathematical Functions

- The three-parameter Riemann Zeta function is no longer supported.
- heaviside(0) = 0.5; in previous versions it was undefined.

maple

- The maple, mhelp, and procread functions error, unless a Maple engine is installed and selected with symengine.

New MuPAD Language and Libraries Supplant Extended Symbolic Math Toolbox Software

The functionality of the MuPAD language, together with the included libraries, goes far beyond that of the previous Symbolic Math Toolbox software. However, it is not identical to that of the previous Extended Symbolic Math Toolbox™ software. The differences between these software packages are beyond the scope of these release notes.

You can access the MuPAD language in several ways:

- To learn the commands, syntax, and functionality of the language, use the MuPAD Help browser, or read the Tutorial.
- Use a MuPAD notebook, which contains an integrated help system for the language syntax.
- Use the new `evalin` function or `feval` function to access the MuPAD language at the MATLAB command line. More detail is available in the Calling Built-In MuPAD Functions from the MATLAB Command Window section of the User's Guide.

New MuPAD Help Viewer (GUI)

The MuPAD help viewer contains complete documentation of the MuPAD language, and of the MuPAD Notebook Interface. For more information, see the Getting Help for MuPAD section of the User's Guide.

New MuPAD Notebook Interface (GUI)

A MuPAD notebook is an interface for performing symbolic math computations with embedded math notation, graphics, animations, and text. It also enables you to share, document, and publish your calculations and graphics. For example, the MuPAD help viewer is essentially a special MuPAD notebook. For more information, see the Calculating in a MuPAD Notebook section of the User's Guide.

New MuPAD Editor and Debugger (GUI)

The MuPAD Editor GUI enables you to write custom symbolic functions and libraries in the MuPAD language. The Debugger enables you to test your code. For more information, consult the MuPAD help viewer.

New Functionality for Communication Between MATLAB Workspace and MuPAD

Function	Use
<code>doc(symengine,...)</code>	Access the MuPAD Help browser.
<code>evalin(symengine,...)</code>	Use MuPAD functionality in the MATLAB workspace.
<code>feval(symengine,...)</code>	Use MuPAD functionality in the MATLAB workspace.
<code>getVar</code>	Copy expressions residing in a MuPAD notebook into the MATLAB workspace.
<code>mupad</code>	Launch a MuPAD notebook .
<code>mupadwelcome</code>	Access MuPAD GUIs .
<code>reset(symengine,...)</code>	Clear the MuPAD engine for the MATLAB workspace.
<code>setVar</code>	Copy expressions residing in the MATLAB workspace into a MuPAD notebook.
<code>symvar</code>	Produce a list of symbolic objects in an expression.

For more information, see the Integration of MuPAD and MATLAB section of the User's Guide.

New `symengine` Command for Choosing a Maple Engine

If you own a compatible version of a Maple software, you can choose to have Symbolic Math Toolbox software use the Maple engine instead of a MuPAD engine. You might want to do this if you have existing Maple programs. Choose the engine by entering `symengine` at the MATLAB command line; this brings up a GUI for making your choice.

New matlabFunction Generates MATLAB Functions

The new `matlabFunction` generates MATLAB functions from symbolic expressions. `matlabFunction` writes the generated code to a file or creates a function handle. You can use the generated function handles and files in any MATLAB installation, even those without a Symbolic Math Toolbox license. For more information, see [Generating MATLAB Functions](#) in the User's Guide.

R2008a

Version: 3.2.3
New Features: No
Bug Fixes: Yes

R2007b

Version: 3.2.2
New Features: No
Bug Fixes: Yes

R2007a

Version: 3.2
New Features: Yes
Bug Fixes: Yes

Maple10 Access Added for Linux 64-bit Processors and Intel Macintosh Platforms

MATLAB now supports Maple Version 10 on 32-bit Windows, 32- and 64-bit Linux[®] platforms, and the Intel[®] and PowerPC[®] Macintosh platforms.

R2006b

Version: 3.1.5
New Features: Yes
Bug Fixes: Yes

Change in call to code generation package using the maple function

Compatibility Considerations: Yes

Calling a function in code generation package using Maple software now requires you to explicitly include the package name. For example,

```
maple('codegen[fortran](x^2-4)');
```

The generated code output using these methods is unaffected by this change.

Compatibility Considerations

In previous versions, functions in the code generation package of Maple software were made automatically available using the Maple with command, and did not require the package name. For example

```
maple('fortran(x^2-4)');
```

This sometimes caused a conflict when assigning to Maple variables having the same name as a function in the code generation package.