# Function Expressions

## Contents

# 1.0 Overview of Function Expressions

A function expression sets the value of a quantity that is used in your RecurDyn model to define the magnitude of a motion input (displacement, velocity or acceleration), force and so forth. A function expression consists of elementary functions such as Fortran functions, functions that report the status of model entities (such as displacements, forces, etc), and other mathematical functions.

Please note that a function expression should not contain operator symbols that may be ambiguous. For example, the double operator in the expression: sin (time)*-5.0 is invalid.

Rather, the valid expression is: sin (time)*(-5.0).

## 1.1 Expression Creation

**Steps to Create a Function Expression**

Your PLM software will provide a method for created and modifying function expressions. In this section we will consider the method provided in the standalone RecurDyn/Professional software.

The user can choose the **Expression** command in the RecurDyn **Subentity** menu. The **Expression List** dialog box will be shown, as shown in Figure 1. When using RecurDyn a list of expressions can be defined and the user selects out of the list of expressions the particular expression that will be used for each entity. In your PLM system if is possible that a single expression will automatically be associated with each entity.

Click the **Create** button within the **Expression List** dialog box, and then **Expression** dialog box will appear, as shown in Figure 2.

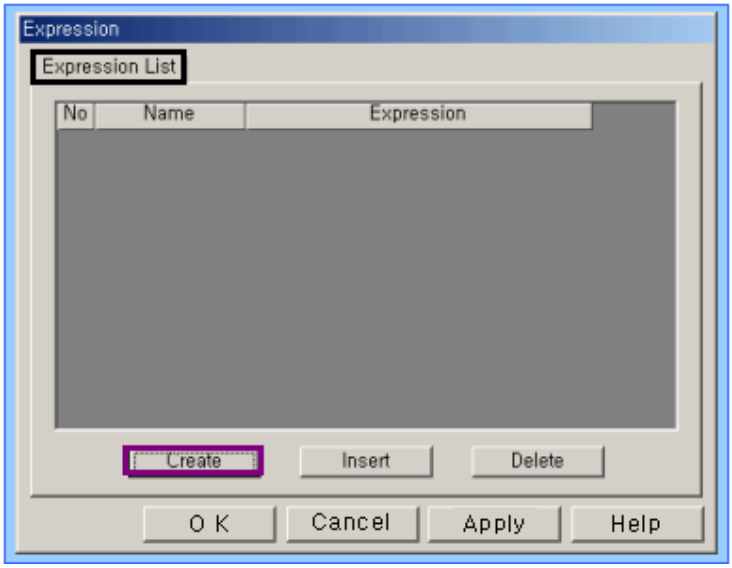


Figure 1. **Expression List** dialog box

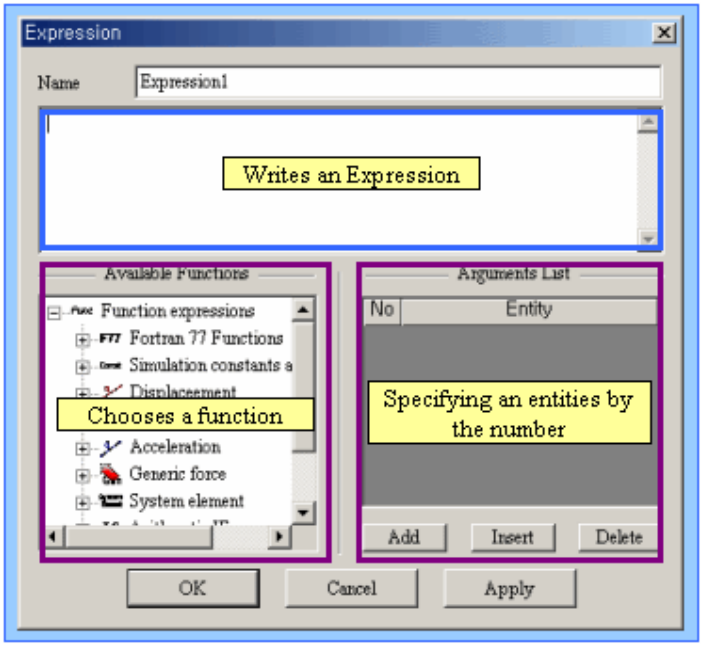


Figure 2. **Expression** dialog box

Specify the **Name** of the expression in the top field and write out the **Expression** in the text field below. When you define an expression, you can use any of the functions that are listed in the **Available Functions** section of the dialog box (or are listed in this document).

> ***Note:*** *In RecurDyn you can use the **'-'** sign as first character in Expression Edit box, also you can use the **"Enter"** key.*

In RecurDyn you can indicate the entity that you are working with by adding its name to the Arguments List section of the dialog box. To see a list of the entities that are available you can open the Database window by clicking the database button in the upper icon toolbar. Add the necessary number of fields to the Arguments List area by clicking of the add button. You can select the name of the entity in the Database Window and drag the name to the field in the Arguments List section. The name of the entity will remain in the field. In your PLM software you may refer to an entity by ID of the appropriate marker (coordinate system).

## 1.2 Available Functions

A complete list of available functions is provided and explained in Sections 2.0 through 9.0 of this document. You can add a **Function** to the text field by double clicking on the function name, as shown in Figure 3.

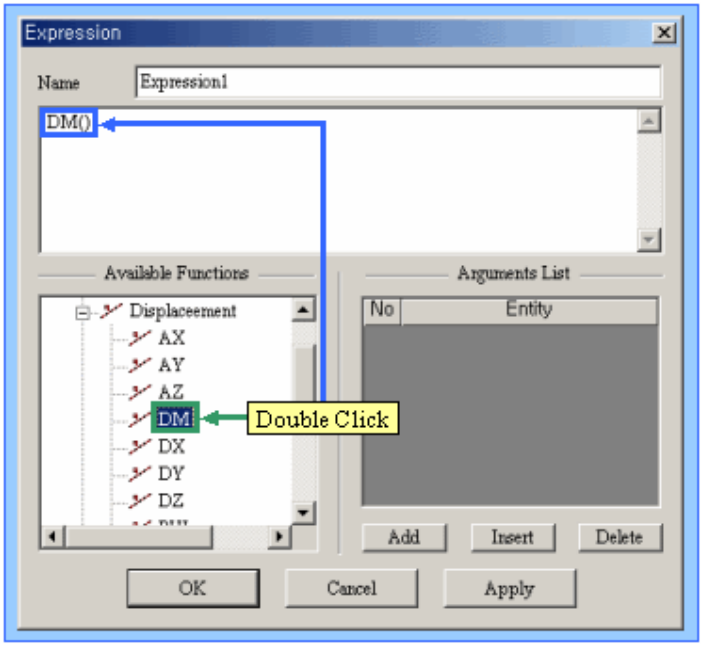


Figure 3. **Function Expression** dialog box

Each function has input argument such as in **Function** (**Marker1**, **Marker2**). The marker name must be the format of **bodyname.markername** for the **DM** function. A marker name in a subsystem may be referred as **bodyname.markername@subsystemname**. An example of a DM function is **DM(Body1.Marker1,Ground.Marker1)**.

Your PLM software documentation will tell you how to define the marker names that are used in Function Expressions.

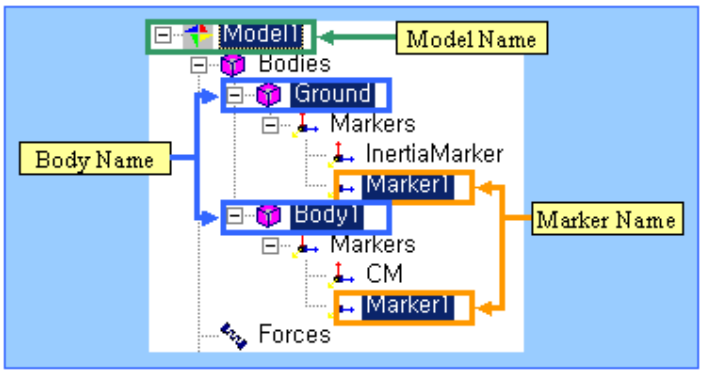


Figure 4. **Marker Name** in the **Database window**

## 1.3 Argument List

In RecurDyn, you can register a marker name and the corresponding ID may be used in place of the marker name. This will significantly reduce the string length of the expression. As an example, **Function( Body1.Maker1, Ground.Marker2 )** is equivalent to **Function(1,2)** if **Body1.Marker1** and **Ground.Marker1** are registered to be 1 and 2.

**Steps to register a marker into the Argument List**


Figure 5. **Arguments List** box

Click **Add** button in the **Arguments List Box**.

Choose a maker by dragging and dropping from the database window. Also you can write the name in the edit box.

Again, your PLM software documentation will tell you how to define the marker names that are used in Function Expressions.

## 2.0 How to Use the Fortran Functions

**RecurDyn** supports the following functions of the **FORTRAN77** language:

### Usage

| Function | Usage |
|---|---|
| **ABS(X)** | The absolute value of X. |
| **ACOS(X)** | The arc cosine(in radians) of X. |
| **AINT(X)** | Truncates a value to a whole number.<br>**Example** : AINT(3.678) = 3.0 |
| **ASIN(X)** | The arc sine(in radians) of X. |
| **ATAN(X)** | The arc tangent (in radians) of X. |
| **ATAN2(Y, X)** | The inverse arc tangent (in radians) of X, Y, where:<br>ATAN2( Y, X ) = ATAN ( Y / X ),<br>for the 1st and 4th Quadrant (X> 0)<br>ATAN2( Y, X ) = ATAN ( Y / X ) - PI ,<br>for the 3rd. Quadrant (X<0 and Y<0)<br>ATAN2( Y, X ) = ATAN ( Y / X ) + PI ,<br>for the 2nd. Quadrant (X<0) and Y>0) |
| **COS(X)** | The cosine (in radians) of X. |
| **COSH(X)** | The hyperbolic cosine of X. |
| **DIM(X,Y)** | The positive difference between X and Y. |
| **EXP(X)** | The exponential value for X. |
| **LOG(X)** | The natural logarithm of X . |
| **LOG10(X)** | The common logarithm (base10) of X. |
| **MAX(A1,A2,{A3})** | The maximum value in the set of arguments:<br>A1, A2, A3 (optional).<br>A1, A2, and A3 (optional) must all be of the same type (either integer or real). |

| | |
|---|---|
| **MIN(A1,A2,{A3}** | The minimum value in the set of arguments: A1, A2, and A3 (optional).<br>A1, A2, and A3 (optional) must all be of the same type (either integer or real). |
| **MOD(A, P)** | The remainder of the arguments has the sign of the first argument. |
| **SIGN(A,B)** | A value with the sign transferred from its second argument. |
| **SINH(X)** | The hyperbolic sine of X. |
| **SQRT(X)** | The square root of X. |
| **TAN(X)** | The tangent (in radian) of X. |
| **TANH(X)** | The hyperbolic tangent (in degrees) of X. |

# 3.0 How to use Displacement Functions

A marker name in brackets ("{ }") is optional. You can define the marker names as follows:

**(m1, m2,{m3})** : (bodyname.markername, bodyname.markername, {bodyname.markername}).

**m1** is the action marker and **m2** is the base marker. If **m3** is defined, **m3** becomes the reference marker. You will obtain the displacement of **m1** with respect to **m2** in the coordinate system of **m3** (**m3** defaults to the global coordinate system if not defined).

## 3.1 Displacement Functions (DM, DX, DY and DZ)

### Magnitude of Displacement (DM)

| Syntax | DM(m1{,m2}) |
|---|---|
| Formulation | $DM = ([R_{m1} - R_{m2}] \cdot [R_{m1} - R_{m2}])^{1/2}$ |

### X-Component Displacement (DX)

| Syntax | DX(m1{,m2}{,m3}) |
|---|---|
| Formulation | $DX = [R_{m1} - R_{m2}] \cdot \hat{x}_{m3}$ |

### Y-Component Displacement (DY)

| Syntax | DY(m1{,m2}{,m3}) |
|---|---|
| Formulation | $DY = [R_{m1} - R_{m2}] \cdot \hat{y}_{m3}$ |

### Z-Component Displacement (DZ)

| Syntax | DZ(m1{,m2}{,m3}) |
|---|---|
| Formulation | $DZ = [R_{m1} - R_{m2}] \cdot \hat{z}_{m3}$ |

## 3.2 Rotational Value Functions (AX, AY and AZ)

### Rotational Value about the X-Axis (AX)

| Syntax | AX(m1{,m2}) |
|---|---|
| Formulation | $AX = ATAN2(\vec{y}_{m1} \bullet \vec{z}_{m2}, \vec{y}_{m1} \bullet \vec{y}_{m2})$ |

### Rotational Value about the Y-Axis (AY)

| Syntax | AY (m1{,m2}) |
|---|---|
| Formulation | $AY = ATAN2(\vec{z}_{m1} \bullet \vec{x}_{m2}, \vec{z}_{m1} \bullet \vec{z}_{m2})$ |

### Rotational Value about the Z-Axis (AZ)

| Syntax | AZ(m1{,m2}) |
|---|---|
| Formulation | $AZ = ATAN2(\vec{x}_{m1} \bullet \vec{y}_{m2}, \vec{x}_{m1} \bullet \vec{x}_{m2})$ |

## 3.3 Components of the 3-1-3 Euler Angles

### PSI

| Syntax | PSI (m1{,m2}) |
|---|---|
| Content | First rotational components of **3-1-3 Euler angle** |

### THETA

| Syntax | THETA(m1{,m2}) |
|---|---|
| Content | Second rotational component of the **3-1-3 Euler angle** |

### PHI

| Syntax | PHI(m1{,m2}) |
|---|---|
| Content | Third rotational component of the **3-1-3 Euler angle** |

## 3.4 Components of 3-2-1 Yaw-Pitch-Roll

### YAW

| Syntax | Yaw(m1{,m2}) |
|---|---|
| Content | First rotational components of **3-2-1 Yaw-Pitch-Roll** |

### PITCH

| Syntax | Pitch(m1{,m2}) |
|---|---|
| Content | Second rotational components of **3-2-1 Yaw-Pitch-Roll** |

### ROLL

| Syntax | Roll(m1{,m2}) |
|---|---|
| Content | Third rotational components of **3-2-1 Yaw-Pitch-Roll** |

# 4.0 How to use Velocity Functions

A Marker in brackets ("{ }") is optional. The marker names are defined as follows:

**(m1, m2,{m3})** : (bodyname.markername, bodyname.markername, {bodyname.markername}).

**m1** becomes the action marker and **m2** becomes base marker. If **m3** is defined, **m3** becomes reference marker. You will obtain the velocity of **m1** with respect to **m2** in the coordinate system of **m3** (**m3** defaults to the global coordinate system if not defined).

## 4.1 Magnitude of Velocity (VM) and Relative Velocity (VR)

### Magnitude of Velocity (VM)

| Syntax | VM( m1 {,m2}) |
|---|---|
| Formulation | $VM = ([V_{m1} - V_{m2}] \cdot [V_{m1} - V_{m2}])^{1/2}$ |

### Relative Velocity (VR)

| Syntax | VR(m1{,m2}) |
|---|---|
| Formulation | $VR = \frac{([V_{m1} - V_{m2}] \cdot [R_{m1} - R_{m2}])}{DM(m1, m2)}$ |

## 4.2 Translational Velocity (VX, VY, and VZ)

### Transitional Velocity along the X-Axis (VX)

| Syntax | VX(m1{,m2}{,m3}) |
|---|---|
| Formulation | $VX = [V_{m1} - V_{m2}] \cdot \hat{x}_{m3}$ |

### Transitional Velocity along the Y-Axis (VY)

| Syntax | VY(m1{,m2}{,m3} ) |
|---|---|
| Formulation | $VY = [V_{m1} - V_{m2}] \cdot \hat{y}_{m3}$ |

### Transitional Velocity along the Z-Axis (VZ)

| Syntax | VZ(m1{,m2} {,m3}) |
|---|---|

| Formulation | $VZ = [V_{m1} - V_{m2}] \cdot \hat{z}_{m3}$ |
|---|---|

## 4.3 Angular Velocity (WM, WX, WY and WZ)

### Magnitude of Angular Velocity (WM)

| Syntax | WM(m1{,m2}) |
|---|---|
| Formulation | $WM = ([\omega_{m1} - \omega_{m2}] \cdot [\omega_{m1} - \omega_{m2}])^{1/2}$ |

### Angular Velocity about the X-Axis (WX)

| Syntax | WX(m1{,m2}{,m3}) |
|---|---|
| Formulation | $WX = [\omega_{m1} - \omega_{m2}] \ \hat{x}_{m3}$ |

### Angular Velocity about the Y-Axis (WY)

| Syntax | WY(m1{,m2}{,m3}) |
|---|---|
| Formulation | $WY = [\omega_{m1} - \omega_{m2}] \cdot \hat{y}_{m3}$ |

### Angular Velocity about the Z-Axis (WZ)

| Syntax | WZ(m1[,m2] [,i3]) |
|---|---|
| Formulation | $WZ = [\omega_{m1} - \omega_{m2}] \cdot \hat{z}_{m3}$ |

# 5.0 How to use Acceleration Functions

A marker name in brackets{ } is optional. The marker names are defined as follow:

**(m1, m2,{m3})** : (bodyname.markername, bodyname.markername, {bodyname.markername}).

**m1** is the action marker and **m2** is base marker. If **m3** is defined, **m3** becomes the reference marker. You will obtain the acceleration of **m1** with respect to **m2** in the coordinate system of **m3** (**m3** defaults to the global coordinate system if not defined).

## 5.1 Translational Accelerations (ACCM, ACCX, ACCY and ACCZ)

### Magnitude of Acceleration (ACCM)

| Syntax | ACCM(m1{,m2}) |
|---|---|
| Formulation | $ACCM = ([a_{m1} - a_{m2}] \cdot [a_{m1} - a_{m2}])^{\frac{1}{2}}$ |

### Transitional Acceleration along X-Axis (ACCX)

| Syntax | ACCX(m1{,m2}{,m3}) |
|---|---|
| Formulation | $ACCX = [a_{m1} - a_{m2}] \cdot \hat{x}_{m3}$ |

### Transitional Acceleration along Y-Axis (ACCY)

| Syntax | ACCY(m1{,m2}{,m3}) |
|---|---|
| Formulation | $ACCY = [a_{m1} - a_{m2}] \cdot \hat{y}_{m3}$ |

### Transitional Acceleration along Z-Axis (ACCZ)

| Syntax | ACCZ(m1{,m2}{,m3}) |
|---|---|
| Formulation | $ACCZ = [a_{m1} - a_{m2}] \cdot \hat{z}_{m3}$ |

## 5.2 Angular Accelerations (WDTM, WDTX, WDTY and WDTZ)

### Magnitude of Angular Acceleration (WDTM)

| Syntax | WDTM(m1[,m2]) |
|---|---|
| Formulation | $WDTM = ([\dot{\omega}_{m1} - \dot{\omega}_{m2}] \cdot [\dot{\omega}_{m1} - \dot{\omega}_{m2}])^{1/2}$ |

### Angular Acceleration about the X-axis (WDTX)

| Syntax | WDTX(m1{,m2}{,m3}) |
|---|---|
| Formulation | $WDTX = [\dot{\omega}_{m1} - \dot{\omega}_{m2}] \cdot \hat{x}_{m3}$ |

### Angular Acceleration about the Y-Axis (WDTY)

| Syntax | WDTY(m1{,m2}{,m3}) |
|---|---|
| Formulation | $WDTY = [\dot{\omega}_{m1} - \dot{\omega}_{m2}] \cdot \hat{y}_{m3}$ |

### Angular Acceleration about the Z-Axis (WDTZ)

| Syntax | WDTZ(m1{,m2}{,m3}) |
|---|---|
| Formulation | $WDTZ = [\dot{\omega}_{m1} - \dot{\omega}_{m2}] \cdot \hat{z}_{m3}$ |

# 6.0 How to use Force and Torque Functions

A marker name in brackets ("{ }") is optional. The marker names are defined as follow:

**(m1, m2,{m3})** : (bodyname.markername, bodyname.markername, {bodyname.markername}).

**m1** is the action marker and **m2** is base marker. If **m3** is defined, **m3** becomes the reference marker. You will obtain the force acting on **m1** with respect to **m2** in the coordinate system of **m3** (**m3** defaults to the global coordinate system if not defined).

## 6.1 Translational Force (FM, FX, FY and FZ)

### Magnitude of Force (FM)

| **Syntax** | FM( m1, m2) |
|---|---|

### X - component Force (FX)

| **Syntax** | FX( m1, m2 {,m3}) |
|---|---|

### Y - component Force (FY)

| **Syntax** | FY( m1, m2{,m3}) |
|---|---|

### Z - component Force (FZ)

| **Syntax** | FZ( m1, m2 {,m3}) |
|---|---|

**NOTE:** The markers used must be part of a Force entity

6.2 Torque (TM, TX, TY and TZ)

## Magnitude of Torque (TM)

| Syntax | TM( m1, m2 ) |
|---|---|

## X - component Torque (TX)

| Syntax | TX( m1, m2 {,m3} ) |
|---|---|

## Y - component Torque (TY)

| Syntax | TY( m1, m2 {,m3} ) |
|---|---|

## Z - component Torque (TZ)

| Syntax | TZ( m1, m2 {,m3} ) |
|---|---|

# 7.0 How to use Specific Force Functions

## 7.1 Contact Force Function

| Syntax | CONTACT( name, flag, comp, rm ) |
|---|---|

## 7.2 Driving Force Function of Motion

| Syntax | MOTION( name, flag, comp, rm ) |
|---|---|

## 7.3 Friction Force Function of Joint

| Syntax | JFRICTION( name, flag, comp, rm ) |
|---|---|

## 7.4 Driving or Driven Force Function of Coupler

| Syntax | COUPLER( name, flag, comp, rm ) |
|---|---|

## 7.5 Driving or Driven Force Function of Gear

| Syntax | GEAR( name, flag, comp, rm ) |
|---|---|

### Arguments:

name: The name of specific force entity such as contact force, driving force of joint and joint friction force.

flag: When the value is 0, the force applied on an action (or driving) body is returned. When the value is 1, the force applied on a base (or driven) body is returned.

comp: A component of the returned force.

| | |
|---|---|
| 1: FM | 5: TM |
| 2: FX | 6: TX |
| 3: FY | 7: TY |
| 4: FZ | 8: TZ |

rm: A reference marker name. The returned force is a component of a vector measured in RM.

# 8.0 How to use the Arithmetic IF Function

The arithmetic **IF** statement conditionally defines a function expression.

## Arithmetic IF Function

| **Syntax** | IF(Expression1: Expression2, Expression3, Expression4) |
|---|---|

If the value of Expression1 < 0, IF calculates and returns the value of Expression2.

If the value of Expression1 = 0, IF calculates and returns the value of Expression3.

If the value of Expression1 >0, IF calculates and returns the value of Expression4.

### Example

The following example conditionally defines the different values according to the value of the time.

**IF(TIME-2.0: -1.0, 0.0, 1.0)**

Expression1 = TIME - 2.0

Expression2 = -1.0

Expression3 = 0

Expression4 = 1.0

TIME < 2.0 Result: IF = -1.0

TIME = 2.0 Result: IF = 0.0

TIME > 2.0 Result: IF = 1.0

# 9.0 How to use the Interpolation Functions

## Interpolation Value

| Syntax | AKISPL( x, z, modelname.curvename, order) |
|---|---|

## Higher Order Derivative of Interpolated Value

| Syntax | AKISPL( x, z, modelname.curvename, order) |
|---|---|

## Arguments:

x: independent variable of x axis

z: interpolated value of z axis, but you must put the 0 as a z value.

Modelname.curvename: identifier of the curve or spline that is being used.

order: optional differential order

## 10.0 How to use Variable and Differential Equations

### Return Value of an Algebraic Variable

| Syntax | VARVAL( model name. variable equation name) |
|---|---|

### Return Value of a State Variable

| Syntax | DIF(model name, differential equation name) |
|---|---|

### Return Value of the Time Derivative of a State Variable

| Syntax | DIF1(model name, Differential equation name) |
|---|---|

# 11.0 How to use Predefined Functions

## CHEBY

| Function name | CHEBY |
|---|---|
| Usage | Define the Chebyshev Polynomial |
| Formulation and usage form | $CHEBY(x, x_0, c_0, c_1, \cdots, c_{20})$ |
| | $C(x) = \sum_{j=0}^{n} c_j * T_j(x - x_0) \quad 0 \le j \le n$<br>where<br>$T_j(x - x_0) = 2*(x - x_0)*T_{j-1}(x - x_0) - T_{j-2}(x - x_0)$<br>$T_0(x - x_0) = 1, \quad T_1(x - x_0) = x - x_0$ |

### Example

CHEBY(time, 1, 1, 1, 1)

x = time: Independent variable

$x_0$ = 1: Shift in the Chebyshev polynomial

$c_0$, $c_1$, $c_2$ : The coefficients for the Chebyshev polynomial

$c_0$ = 1

$c_1$ = 1

$c_2$ = 1

C(x) =1*1 +1*(time-1)+1*(2*(time-1)$^2$-1)

=time+2*time$^2$-4time+2-1

=2*time$^2$-3time+1

## FORCOS

| Function name | FORCOS |
|---|---|
| Usage | Define the Fourier Cosine series |
| Formulation and usage form | FORCOS(x, $x_0$, $\omega$, $c_0$, $c_1$, ···, $c_{30}$) |
| | $F(x) = c_0 + \sum_{j=1}^{n} c_j * T_j(x - x_0), \quad 0 \le j \le n$ where $T_j(x - x_0) = \cos[j * \omega * (x - x_0)]$ |

### Example

FORCOS (time, 0, 360D, 1, 2, 3)

x = time: Independent variable

$x_0$ = 0: Shift in the Fourier cosine series

$\omega$ = 360D: Frequency of the Fourier cosine series

$c_0$, $c_1$, $c_2$: Define coefficient for the Fourier series

$c_0$ = 1

$c_1$ = 2

$c_2$ = 3

F(x) = 1 + 2*cos(1*360D*time) + 3*cos(2*360D*time)

## STEP

| Function name | STEP |
|---|---|
| Usage | Define the function that smoothly approximates a step transition from one value to another value, using a Cubic Polynomial |
| Formulation and usage form | $STEP(x, x_0, h_0, x_1, h_1)$ |
| | $STEP = h_0$, when $x \le x_0$<br>=<br>$h_0 + (h_1 - h_0) * [(x - x_0)/(x_1 - x_0)]^2 * (3 - 2 * [(x - x_0)/$<br>,when $x_0 \le x \le x_1$<br>$= h_1$, wher $x \ge x_1$ |

### Example

**STEP(time, 1.0, 0.0, 2.0, 1.0)**

x = time: Independent variable

$x_0$ = 1.0: X-value at which the **STEP** function begins

$h_0$ =0.0: Initial value of the **STEP** function

$x_1$ = 2.0: X-value at which the **STEP** functions ends

$h_1$ = 1.0: Final value of the **STEP** function

## IMPACT

| Function name | IMPACT |
|---|---|
| Usage | Define collision between two rigid bodies. Ref. **BISTOP** |
| Formulation and usage form | IMPACT( x, $\dot{x}$, $x_1$, k. exp, $c_{max}$, d) |
| | IMPACT= $k(x_1 - x)^{exp} - STEP(x, x_1 - d, c_{max}, x_1, 0) * \dot{x}$ when $x < x_1$<br><br>IMPACT = 0 when, $x \geq x_1$ |

### Example

**IMPACT(DZ,VZ,1.0,100,1.5,25,0.1)**

x = DZ: Distance variable

$\dot{x}$ =VZ: The time derivative of x

$x_1$ =1.0: Free length of x

k =100: Stiffness

exp = 1.5: Exponent of the force

$c_{max}$ =25: Maximum damping coefficient

d =0.1: Boundary penetration

## POLY

| Function name | POLY |
|---|---|
| Usage | Define the POLYNOMIAL function |
| Formulation and usage form | POLY(x, $x_0$, $a_0$, $a_1$, ···, $a_{30}$) |
| | $P(x) = \sum_{j=0}^{n} a_j (x - x_0)^j$ = $a_0 + a_1 * (x - x_0) + a_2 * (x - x_0)^2 + \cdots + a_n * (x - x_0)^n$ where, $0 < j < n$ |

### Example

**POLY (time, 0, 1,1,1)**

x = time: Independent variable

$x_0$ = 0: Shift in the polynomial

$a_0$, $a_1$, $a_2$: Define coefficient for the polynomial series

$a_0 = 1$

$a_1 = 1$

$a_2 = 1$

P (x) = 1+1*time+1*time$^2$

= time$^2$+time+1

## SHF

| Function name | SHF |
|---|---|
| Usage | Define the Simple Harmonic function |
| Formulation and usage form | $SHF(x, x_0, a, \omega, \phi, b)$ |
| | $SHF = a * \sin(\omega * (x - x_0) - \phi) + b$ |

### Example

**SHF(time, 10D, PI,360D,0,3)**

x = time: Independent variable in the function

$x_0$ = 10D: Offset in the independent variable x

a = PI: Amplitude of the harmonic function

$\omega$ = 360D: The frequency of the harmonic function.

$\phi$ =0: Phase shift in the harmonic function

b = 3: Average displacement of the harmonic function.

SHF = pi*sin (360D*(time-10D))+3

## BISTOP

| Function name | BISTOP |
|---|---|
| Usage | $BISTOP(x, \dot{x}, x_1, x_2, k, exp, c_{max}, d)$ |
| Formulation and usage form | $BISTOP = k(x_1 - x)^{exp} - STEP(x, x_1 - d, c_{max}, x_1, 0) * \dot{x}$ when $x < x_1$,<br>BISTOP =0, when $x_1 \le x \le x_2$,<br>BISTOP = $-k(x - x_2)^{exp} - STEP(x, x_2, 0, x_2 + d, c_{max}) * \dot{x}$ when $x \ge x_2$, |

### Example

**BISTOP** ( $x, \dot{x}, x_1, x_2, k, exp, c_{max}, d$)

x : Distance variable

$\dot{x}$ : Time derivative of x

$x_1$ : Lower bound of x

$x_2$ : Upper bound of x

k : Stiffness

exp: Exponent of force

$c_{max}$ : Maximum damping coefficient

d : Boundary penetration

## FORSIN

| Function name | FORSIN |
|---|---|
| Usage | Define the Fourier Sine series FORSIN |
| Formulation and usage form | $FORSIN(x, x_0, \omega, a_0, a_1, \cdots, a_{30})$ |
| | $F(x) = a_0 + \sum_{j=1}^{n} a_j * T_j(x - x_0) \quad 0 \le j \le n$<br>where $T_j(x - x_0) = \sin(j * \omega * (x - x_0))$ |

### Example

**FORSIN (time, 0.25, π, 0, 1, 2, 3)**

x = time: Independent variable

$x_0$ = 0.25: Shift in the Fourier sine series

$\omega$ = π: Frequency of the sine series

$a_0$, $a_1$, $a_2$: Define coefficient for the sine series

$a_0$ =0

$a_1$ =1

$a_2$ =2

$a_3$ =3

F(x) = sin ( π*( time+0.25) ) + 2 * sin (2 *π*( time+0.25))+ 3 * sin (3 *π*( time+0.25))

## HAVSIN

| Function name | HAVSIN |
|---|---|
| Usage | Define the HAVSIN function |
| Formulation and usage form | HAVSIN( $x, x_0, h_0, x_1, h_1$ ) |
| | $HAVSIN = h_0$, when $x < x_0$<br>$= (h_0 + h_1)/2 + (h_1 - h_0)/2 * \sin[\pi * (x - x_0)/(x_1 - x_0) - \pi/2]$, when $x_0 \le x \le x_1$<br>$= h_1$ when $x \ge x_1$ |

### Example

**HAVSIN** ( $x, x_0, h_0, x_1, h_1$ )

x : Independent variable

$x_0$ : The x-value at which the HAVSIN function begins

$h_0$ : Initial value of the HAVSIN function

$x_1$ : The x-value at which the HAVSIN function ends

$h_1$ : Final value of the HAVSIN function

HAVSIN $= h_0$ , when $x \le x_0$

$$= \frac{(h_0 + h_1)}{2} + \frac{(h_1 - h_0)}{2} * \sin\left(\frac{x - x_0}{x_1 - x_0} * \pi - \frac{\pi}{2}\right)$$ , when $x_0 \le x \le x_1$

$= h_1$, when $x \ge x_1$

First and Second Order Differential Functions for Predefined Functions

| Function Name | First Order Function Name | Second Order Function name |
|---|---|---|
| BISTOP | DBISTOP | DDBISTOP |
| CHEBY | DCHEBY | DDCHEBY |
| FORCOS | DFORCOS | DDFORCOS |
| FORSINE | DFORSINE | DDFORSINE |
| HAVSIN | DHAVSIN | DDHAVSIN |
| IMPACT | DIMPACT | DDIMPACT |
| POLY | DPOLY | DDPOLY |
| SHF | DSHF | DDSHF |
| STEP | DSTEP | DDSTEP |

# Index