

<https://www.halvorsen.blog>



LabVIEW MathScript

Hans-Petter Halvorsen

Contents

1. LabVIEW MathScript

- Basic Examples
- Plotting Examples
- Simulation Examples
- Create Functions

2. MathScript Node

LabVIEW MathScript

- You need to install an additional module called **LabVIEW MathScript Module**.
- You should also install **LabVIEW Control Design and Simulation Module** because it adds Control and Simulation features to the MathScript Module
- This module can be used in 2 different ways:
 - **LabVIEW MathScript** – A separate Application similar to MATLAB (But you need to have LabVIEW installed)
 - **MathScript Node** integrated in LabVIEW Code

LabVIEW MathScript Module

- Add-on Module for LabVIEW where we can do text-based programming and simulations
- GUI and syntax are identical with MATLAB
- You can run MATLAB scripts in LabVIEW MathScript with almost no changes needed (assuming you use the core functionality or the MATLAB Control Toolbox)
- LabVIEW MathScript don't have the same speed, flexibility and toolboxes as MATLAB
- If you know MATLAB, you know LabVIEW MathScript

How do you start using MathScript?

- You need to install LabVIEW and the LabVIEW MathScript Module.
- When necessary software is installed, start MathScript by open LabVIEW
- From the LabVIEW menu, select Tools
-> MathScript Window...

Output Window

```

For help, enter 'help classes'
>>
x =
    2
y =
    8

```

Output Window

Here you can see the results of the calculations

Command Window

```


help plot

```

Command Window

You can use the Command Window to enter single commands

Variables Script History



```

1 x = 2
2
3 y = 3*x + 2

```

Run your Script and the results are available in the Output Window

Script Window

This is the Editor where you create your program (script). The Script can be saved as a .m file

<https://www.halvorsen.blog>

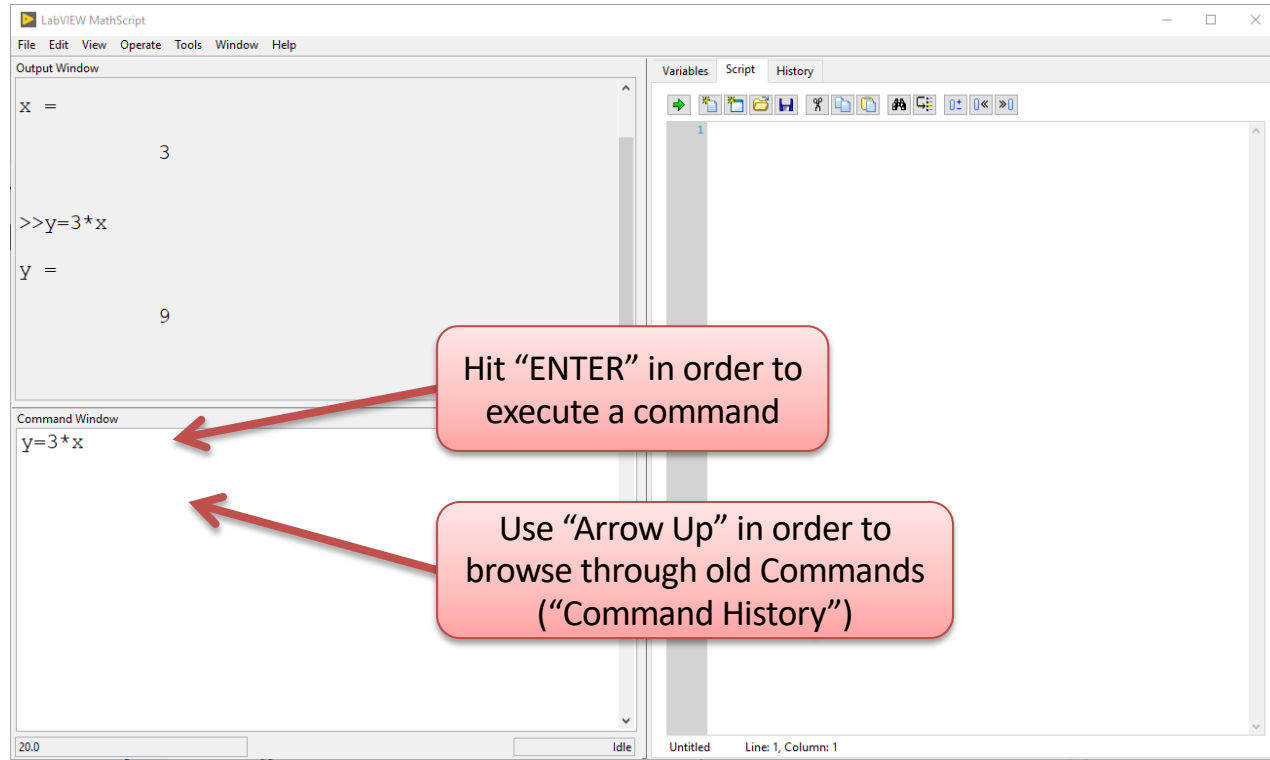


Basic Examples

Hans-Petter Halvorsen

Command Window

The Command Window is the main window in MathScript. Use the Command Window to enter variables and to run functions and M-files scripts (more about m-files later). Its like an advanced calculator!



The screenshot displays the LabVIEW MathScript environment. The main window is titled "LabVIEW MathScript" and contains several panes. On the left, the "Output Window" shows the results of a calculation: $x = 3$, followed by the command $>>y=3*x$, and the result $y = 9$. Below this is the "Command Window" where the command $y=3*x$ has been entered. On the right, there are panes for "Variables", "Script", and "History". The "Script" pane shows a single line of code: 1 . Two red callout boxes with arrows point to the Command Window. The top callout box says "Hit 'ENTER' in order to execute a command" and points to the command $y=3*x$. The bottom callout box says "Use 'Arrow Up' in order to browse through old Commands ('Command History')" and points to the Command Window area.

Hit "ENTER" in order to execute a command


Use "Arrow Up" in order to browse through old Commands ("Command History")

Case Sensitive Variables

MathScript/MATLAB is **case sensitive**! The variables `x` and `X` are not the same.

```
>> x=5;  
>> X=6;  
>> x+X  
  
ans =  
    11
```

```
>> x=3  
x =  
    3  
  
>> y=4;  
>>
```



Unlike many other languages, where the semicolon is used to terminate commands, in MathScript/MATLAB the semicolon serves to suppress the output of the line that it concludes.

clear/clc

```
>> clear
```

```
>> clc
```

The “clear” command deletes all existing variables” from the memory

The “clc” command removes everything from the Command Window
clc – clear command window

Built-in Constants

Name	Description
<code>i, j</code>	Used for complex numbers, e.g., $z=2+4i$
<code>pi</code>	π
<code>inf</code>	∞ , Infinity
<code>NaN</code>	Not A Number. If you, e.g., divide by zero, you get NaN

```
>> r=5;
>> A=pi*r^2

A =
    78.5398
```

```
>> z1=3+3i;
>> z2=3+5i;
>> z = z1+z2
z =
    6.0000 + 8.0000i
```

```
>> a=2;
>> b=0;
>> a/b
```

Mathematical Expressions

$$y(x) = \frac{3x + 2}{2}$$
$$y(2) = ?$$

```
>> x=2;  
>> y=3*x+2/2  
y =  
    7  
>> y=(3*x+2)/2  
y =  
    4
```

Which are correct?

Calculate this expression, try with different values for x and y

	MATLAB
$\ln(x)$	<code>log(x)</code>
$\log_{10}(x)$	<code>log10(x)</code>
\sqrt{x}	<code>sqrt(x)</code>
e^x	<code>exp(x)</code>
x^2	<code>x^2</code>

$$z = 3x^2 + \sqrt{x^2 + y^2} + e^{\ln(x)}$$

Mathematical Expressions

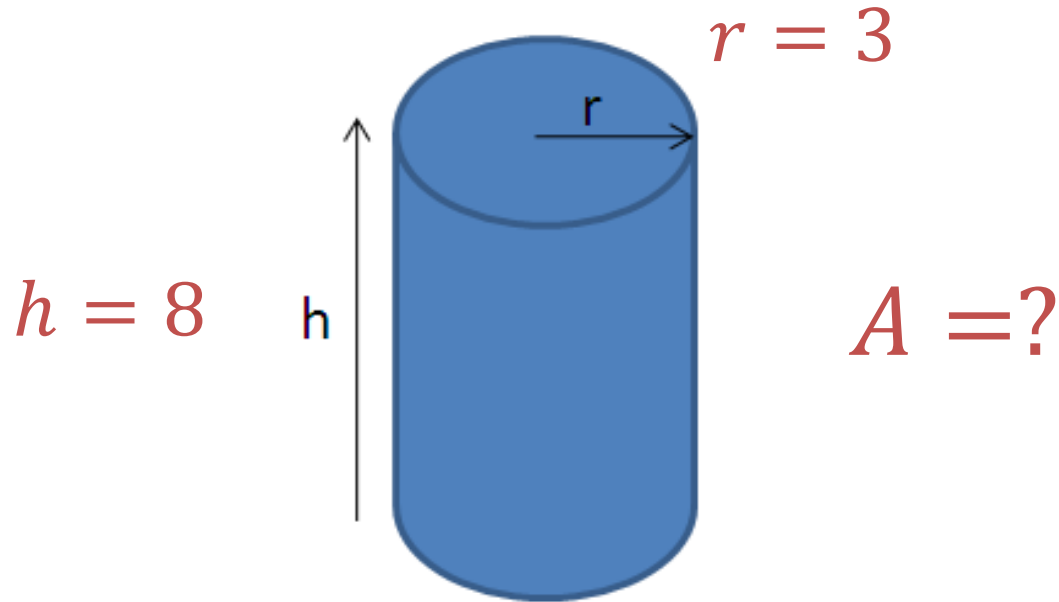
$$z = 3x^2 + \sqrt{x^2 + y^2} + e^{\ln(x)}$$

```
>> x=2;, y=2
>> z = 3*x^2 + sqrt(x^2 + y^2) + exp(log(x))

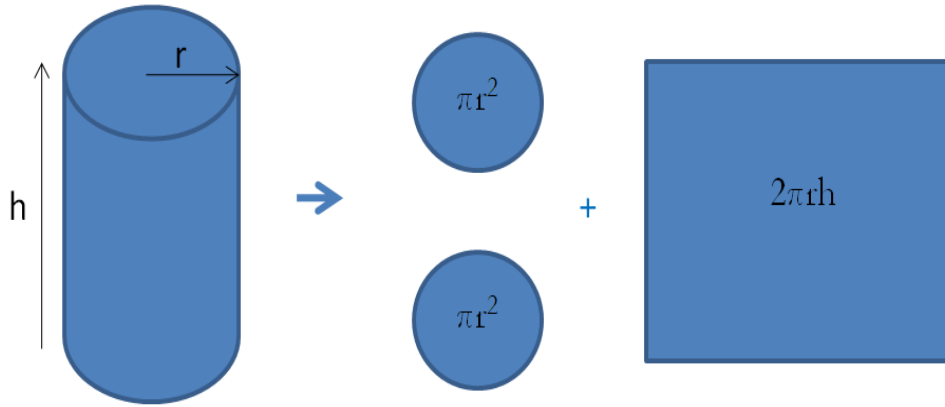
ans =
    16.8284
...
```

Solving Mathematical Problems

We will use MathScript in order to find the surface area of a cylinder based on the height (h) and the radius (r) of the cylinder



Solving Mathematical Problems



MathScript Code:

```
>> h=8
>> r=3
>> A = 2*pi*r^2 +2*pi*r*h;
A =
    207.3451
```

<https://www.halvorsen.blog>



Plotting

Hans-Petter Halvorsen

Plotting

Example:

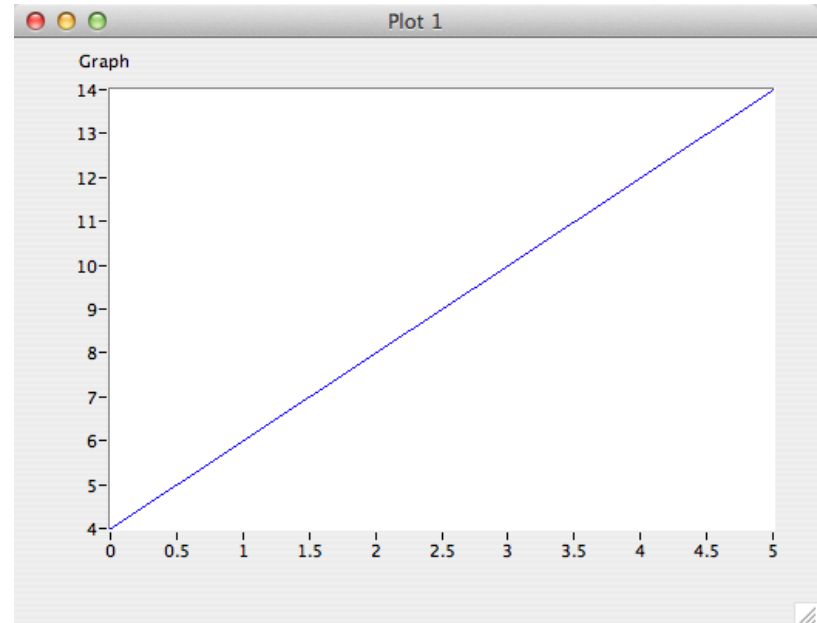
$$y(t) = 2x + 4$$

interval on x axis

```
x = 0:5;
```

```
y = 2*x + 4;
```

```
plot(x, y)
```



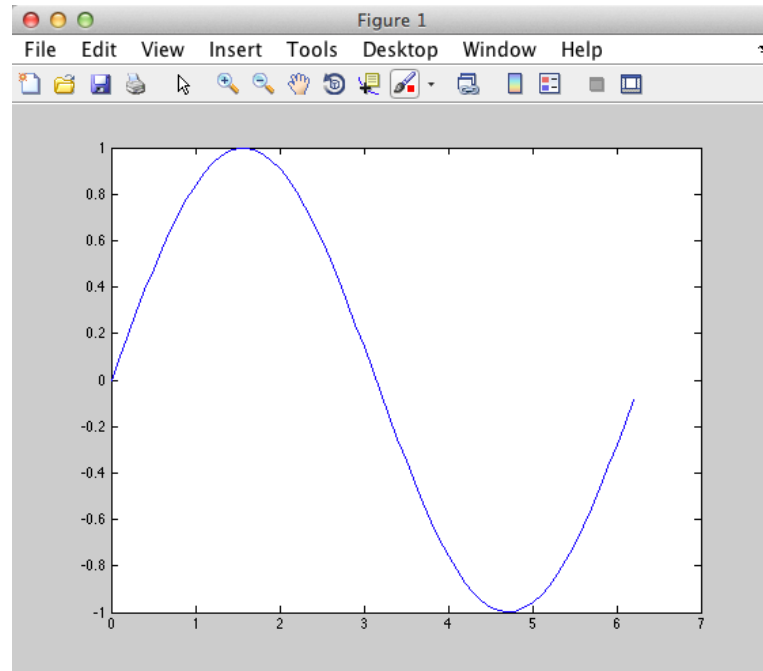
Useful MathScript functions for plotting: `plot()`, `xlabel()`, `ylabel()`, `title()`, `grid()`

Some Examples

```
>> x = 0:0.1:2*pi;  
>> y = sin(x);  
>> plot(x,y)
```

```
>> x = 0:0.1:2*pi;  
>> y = sin(x);  
>> y2 = cos(x);  
>> plot(x,y, x,y2)
```

```
...  
>> plot(x,y,'r*', x,y2,'g+')
```



Plotting Functions

Plotting functions:

Name	Description
plot	Create a Plot
figure	Define a new Figure/Plot window
grid on/off	Create Grid lines in a plot
title	Add Title to current plot
xlabel	Add a Label on the x-axis
ylabel	Add a Label on the x-axis
axis	Set xmin,xmax,ymin,ymax
hold on/off	Add several plots in the same Figure
legend	Create a legend in the corner (or at a specified position) of the plot
subplot	Divide a Figure into several Subplots

Examples:

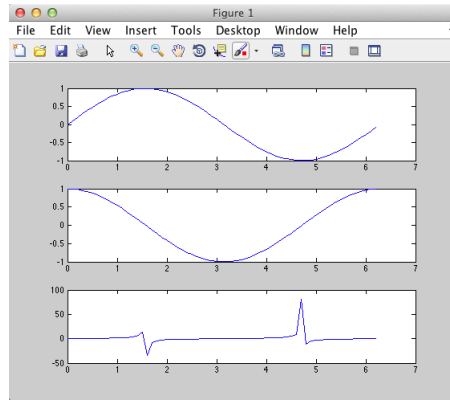
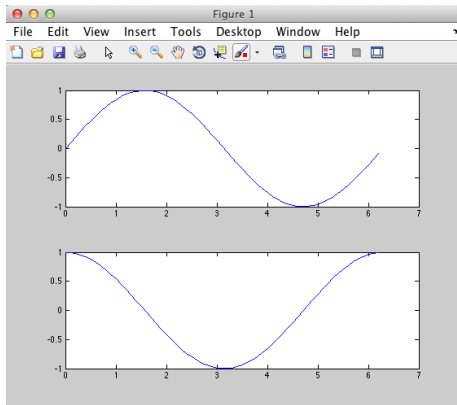
```
>> x=0:0.1:2*pi;  
>> y=sin(x);  
>> plot(x,y)  
>> title('Plot Example')  
>> xlabel('x')  
>> ylabel('y=sin(x)')  
>> grid on  
>> axis([0,2*pi,-1,1])  
>> legend('Temperature')
```

Subplots

```
>> x=0:0.1:2*pi;  
>> y=sin(x);  
>> y2=cos(x);
```

```
>> subplot(2,1,1)  
>> plot(x,y)
```

```
>> subplot(2,1,2)  
>> plot(x,y2)
```

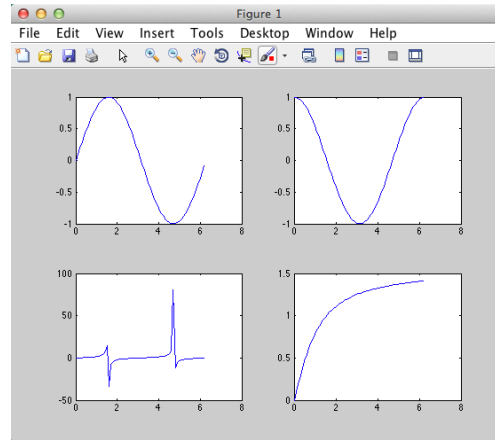


```
>> x=0:0.1:2*pi;  
>> y=sin(x);  
>> y2=cos(x);  
>> y3=tan(x);
```

```
>> subplot(3,1,1)  
>> plot(x,y)
```

```
>> subplot(3,1,2)  
>> plot(x,y2)
```

```
>> subplot(3,1,3)  
>> plot(x,y3)
```



```
>> x=0:0.1:2*pi;  
>> y=sin(x);  
>> y2=cos(x);  
>> y3=tan(x);  
>> y4=atan(x);
```

```
>> subplot(2,2,1)  
>> plot(x,y)
```

```
>> subplot(2,2,2)  
>> plot(x,y2)
```

```
>> subplot(2,2,3)  
>> plot(x,y3)
```

```
>> subplot(2,2,4)  
>> plot(x,y4)
```

<https://www.halvorsen.blog>



Simulation Example

Hans-Petter Halvorsen

Simulation Example

Assume the following model (Differential Equation):

$$\dot{x} = -ax + bu$$

We start by setting $a = 0.25$ and $b = 2$

Then we get:

$$\frac{x(k+1) - x(k)}{T_s} = -ax(k) + bu(k)$$

Finally, we get:

$$x(k+1) = (1 - T_s a)x(k) + T_s bu(k)$$

This is the discrete version of the differential equation

In order to simulate this system in LabVIEW MathScript we typically need to find the discrete differential equation.

We can use e.g., the **Euler Approximation**:

$$\dot{x} \approx \frac{x(k+1) - x(k)}{T_s}$$

Where T_s is the Sampling Time

Code

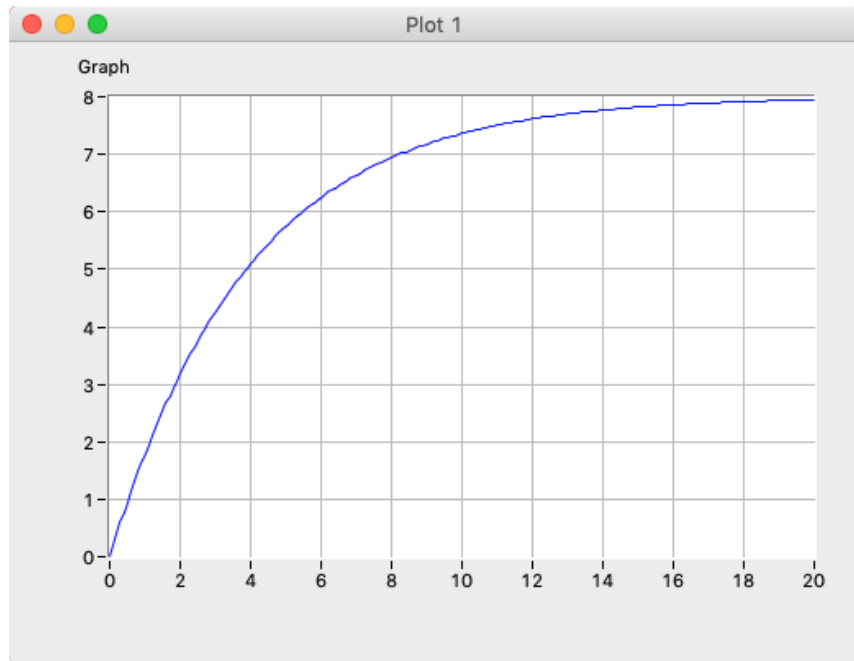
```
% Simulation of discrete model
clear, clc

% Model Parameters
a = 0.25;b = 2;

% Simulation Parameters
Ts = 0.1; %s
Tstop = 20; %s
uk = 1; % Step Response
x(1) = 0;

% Simulation
for k=1:(Tstop/Ts)
    x(k+1) = (1-a*Ts).*x(k) + Ts*b*uk;
end

% Plot the Simulation Results
k=0:Ts:Tstop;
plot(k,x)
grid on
```



<https://www.halvorsen.blog>



Creating Functions

Hans-Petter Halvorsen

Create Function

The screenshot displays the LabVIEW MathScript environment. The main window is titled "LabVIEW MathScript" and contains several panes:

- Output Window:** Shows the execution of commands: `x =` followed by `2`, `>>y=4`, `y =` followed by `4`, `>>add(x,y)`, and `ans =` followed by `6`.
- Command Window:** Contains the command `add(x,y)`.
- Script Editor:** Shows the definition of the `add` function:

```
1 function answer = add(x,y)
2
3 answer = x + y;
```

The status bar at the bottom indicates the current file is `add.m` and the cursor is at `Line: 3, Column: 16`.

Create Functions in MathScript

The screenshot shows the LabVIEW MathScript environment with several callouts:

- 1** Create your function in the Script window: A callout box points to the Script window containing the code:

```
function total = add(x,y)
% this function add 2 numbers
total = x+y;
```
- 2** Save your function as a .m file: A callout box points to the Save icon in the Script window toolbar.
- 3** Open MathScript Properties: A callout box points to the 'LabVIEW MathScript Properties' menu item in the File menu.
- 4** Add Search Folder for your Code: A callout box points to the 'Add Folder' button in the 'MathScript: Search Paths' dialog box.
- 5** Test your function in the Command window: A callout box points to the Command Window containing the code:

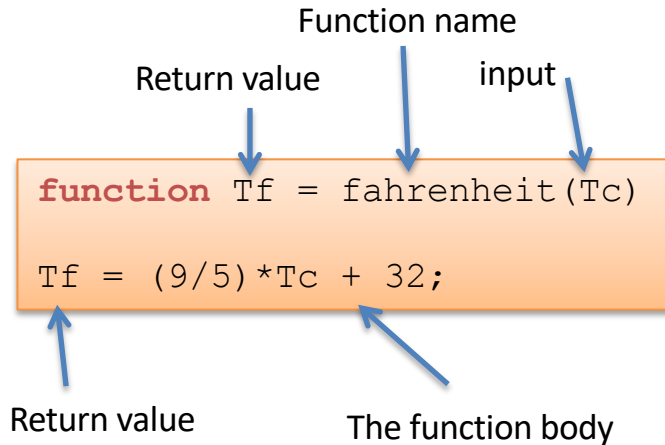
```
add(3,5)
```

The 'LabVIEW MathScript Properties' dialog box is open, showing the 'MathScript: Search Paths' section with the path 'C:\tmp\MathScript' and an 'Add Folder' button. The 'Working directory' is also set to 'C:\tmp\MathScript'.

Celsius to Fahrenheit

$$T_F = \frac{9}{5}T_C + 32$$

Step 1: Create the Function



The function needs to be saved as **“fahrenheit.m”** on your hard drive

Step 2: Execute the Function

```
Tc = 23;  
Tf = fahrenheit(Tc)
```

This can be done from Command window or Script window

<https://www.halvorsen.blog>



Tips and Tricks

Hans-Petter Halvorsen

Tips and Tricks

Use Comments (%)

```
% This is a comment  
x=2; % Comment2  
y=3*x % Comment3
```

- but that have to make sense!

DO NOT use "spaces" in Filename or names that are similar to built-in functions in MathScript/MATLAB!

Decimal sign: Use "." – NOT "," !
i.e. $y=3.2$ – not $y=3,2$

Use english names on variables, functions, files, etc. This is common practice in programming!
Use always variables – Do not use numbers directly in the expressions!

Yes:

```
a=2;  
b=4;  
y=a+  
b
```

No:

```
y=2+  
4
```

Functions:

- Only ONE function in each File!
- The Filename (.m) AND the Name of the Function MUST be the same!

```
clear  
clc  
close all  
...
```

Always include these lines in your Script

Tips and Tricks

Greek letters: In math and control theory it is common to use Greek letters in formulas, etc. These cannot be used directly in MathScript/MATLAB, so you need to find other good alternatives.

Examples:

ω_0 – $w0$

ζ – zeta or just z

etc.

A Golden Rule: One Task – one file, i.e. DONT put all the Tasks in one single file!!

$$z = 3x^2 + \sqrt{x^2 + y^2} + e^{\ln(x)}$$

$z(2,2) = ?$

```
x = 2;  
y = 2;  
z = 3*x^2 + sqrt(x^2 + y^2) + exp(log(x))
```

Use help in order to find out how to use a function in MathScript/MATLAB. In order to get help for the `tf` function, type the following in the Command window:
`help tf`

Mathematical expressions:

The following applies in MathScript/MATLAB

x^2	<code>x^2</code>
\sqrt{x}	<code>sqrt(x)</code>
$\ln(x)$	<code>log(x)</code>
$\log(x)$	<code>log10(x)</code>
e^x	<code>exp(x)</code>
π	<code>pi</code>

<https://www.halvorsen.blog>



MathScript Node

Hans-Petter Halvorsen

MathScript Node

With MathScript Node you can create and use MathScript/MATLAB code within LabVIEW

MathScript Node

```
1 SumA = eye(size(A));
2 for i = 1:n
3 SumA = SumA + A^i/factorial(i);
4 end
5 Delta = SumA - expm(A);
```

input variable (optional) **A**

input variable (optional) **n**

output variable (optional) **Delta**

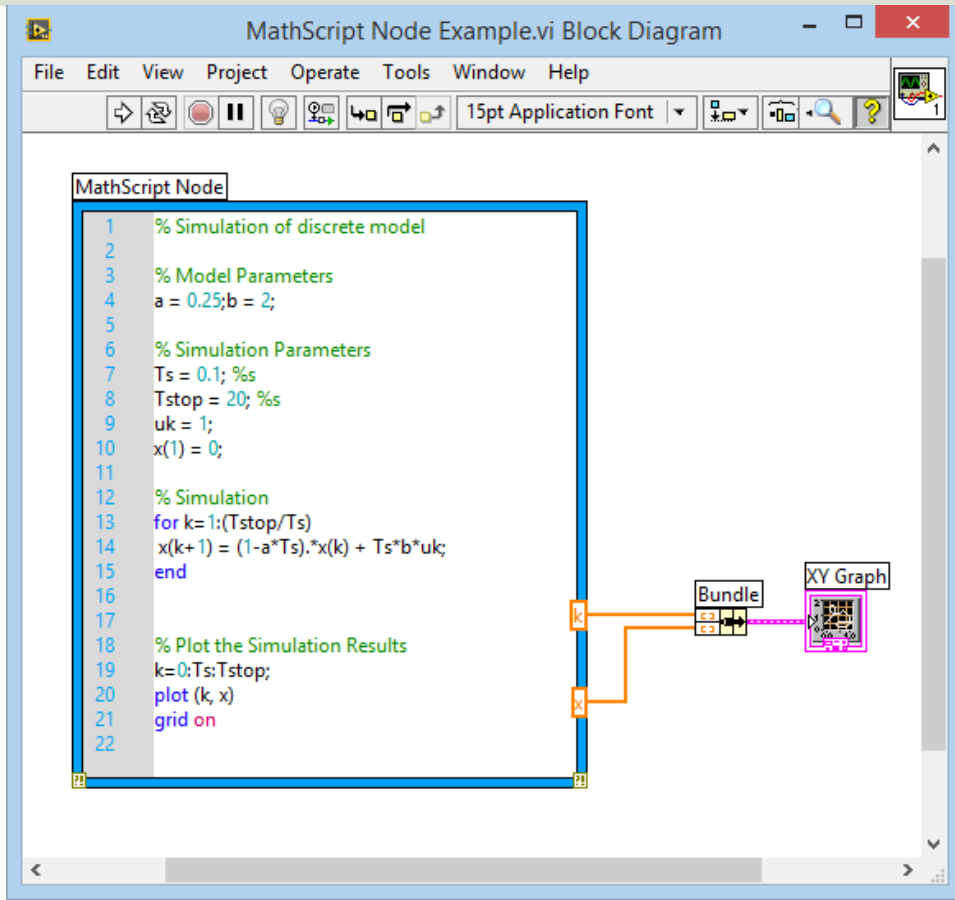
error in error out

Executes LabVIEW MathScripts and your other text-based scripts using the MathScript RT Module engine. You can use the MathScript Node to evaluate scripts that you create in the LabVIEW MathScript Window.

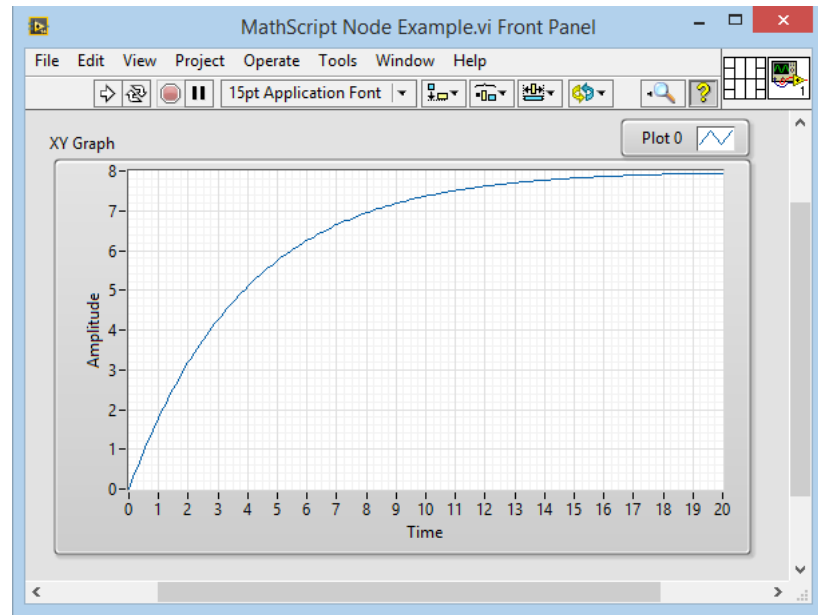
If a MathScript Node contains a warning glyph, LabVIEW operates with slower run-time performance for the node. You can modify your script to remove the warning glyph from the MathScript Node and improve run-time performance.

[Detailed help](#)

Example



```
1 % Simulation of discrete model
2
3 % Model Parameters
4 a = 0.25;b = 2;
5
6 % Simulation Parameters
7 Ts = 0.1; %s
8 Tstop = 20; %s
9 uk = 1;
10 x(1) = 0;
11
12 % Simulation
13 for k=1:(Tstop/Ts)
14 x(k+1) = (1-a*Ts).*x(k) + Ts*b*uk;
15 end
16
17 % Plot the Simulation Results
18 k=0:Ts:Tstop;
19 plot(k, x)
20 grid on
21
22
```

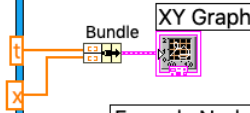


Alternative: Formula Node

MathScript Node

```
1 % Simulation of discrete model
2
3 % Model Parameters
4 a = 0.25;b = 2;
5
6 % Simulation Parameters
7 Ts = 0.1; %s
8 Tstop = 20; %s
9 uk = 1; % Step Response
10 x(1) = 0;
11
12 % Simulation
13 for k=1:(Tstop/Ts)
14 x(k+1) = (1-a*Ts).*x(k) + Ts*b*uk;
15 end
16
17 % Plot the Simulation Results
18 t=0:Ts:Tstop;
19 plot(t,x)
20 grid on
```

The MathScript Node uses MATLAB syntax



Formula Node

```
//Model Parameters
float a = 0.25;
float b = 2;

//Simulation Parameters
float Ts = 0.1;
float Tstop = 20;
float uk = 1;
float xk = 0;
float xk1 = 0;
int k;
int N = Tstop/Ts;

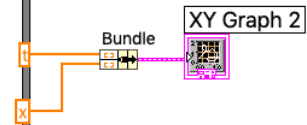
float t[200];
float x[200];

for (k=1; k<N; k++)
{
  xk1 = (1-a*Ts) * xk + Ts*b*uk;

  xk = xk1;

  t[k] = Ts*k;
  x[k] = xk1;
}
```

The Formula Node uses C syntax



Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: <https://www.halvorsen.blog>

