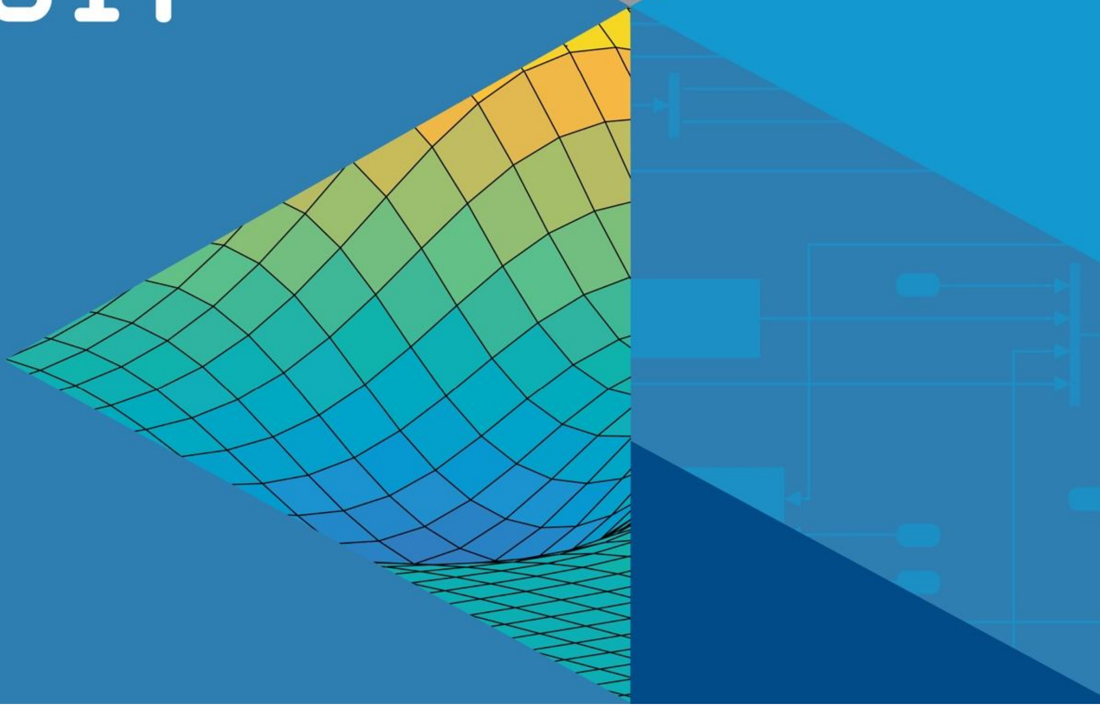
MATLAB EXPO 2017

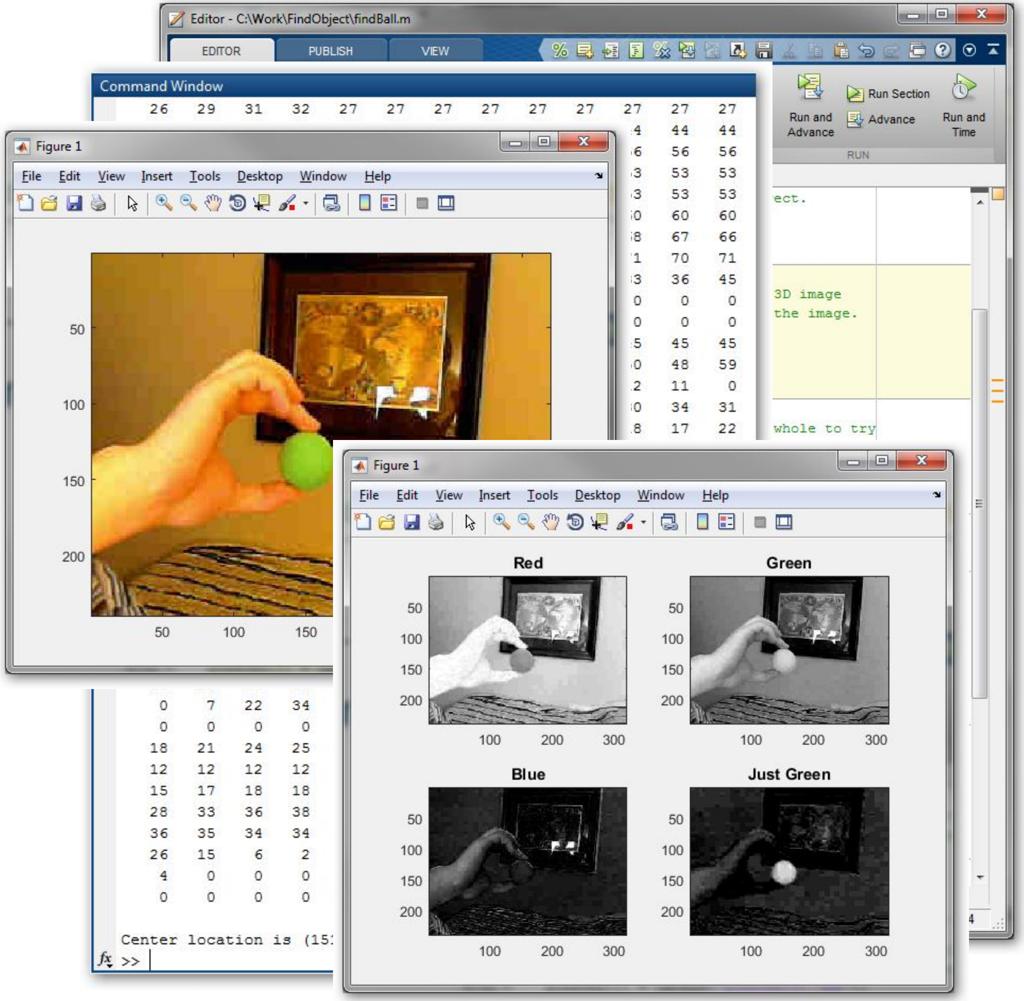
Teaching with the MATLAB Live Editor

Dr. Oliver Kluge



Editing and Running MATLAB Code without the MATLAB Live Editor

- Plain-text editing
- Output goes to Command Window
- Multiple figure windows appear
- Equations, images, and hyperlinks only appear if published



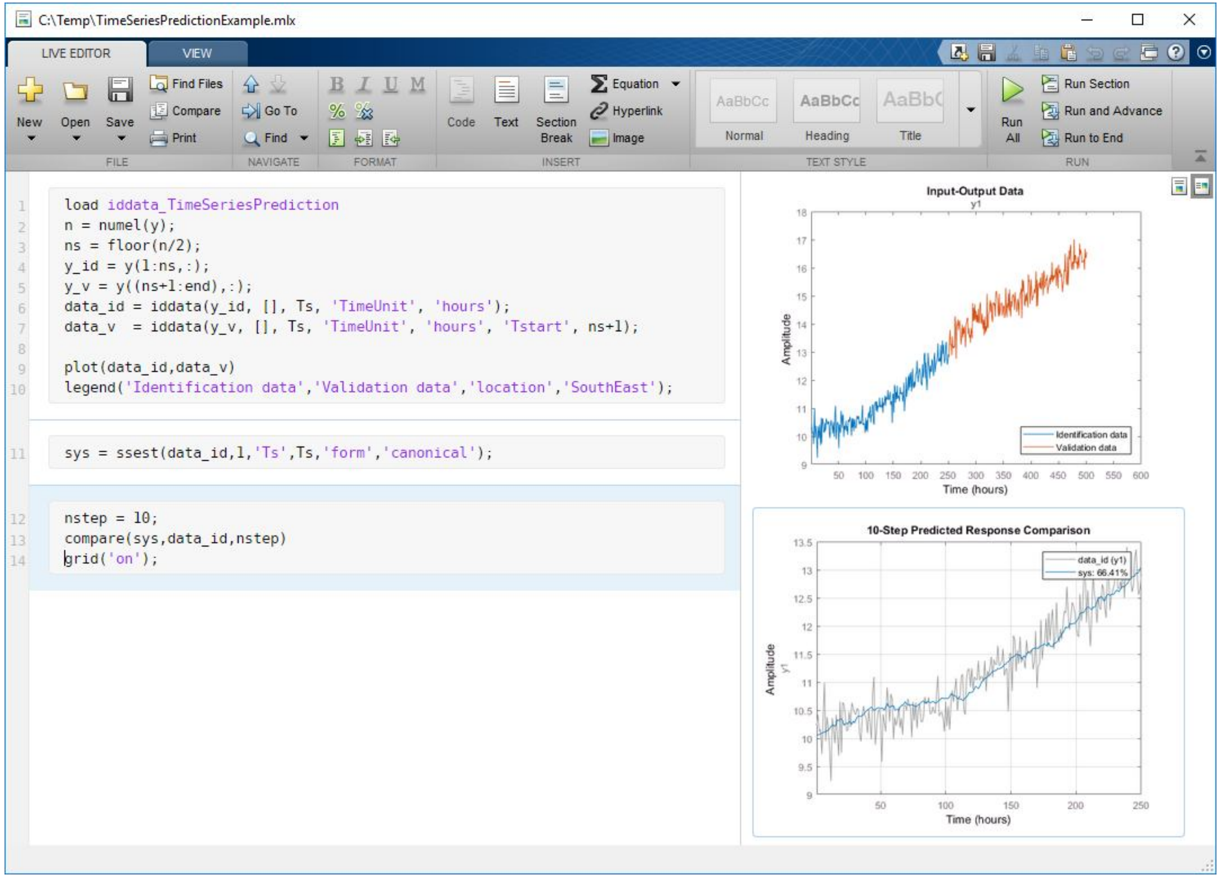
```
   26   29   31   32   27   27   27   27   27   27   27   27   27
                                                          4   44   44
                                                          6   56   56
                                                          3   53   53
                                                          3   53   53
                                                          0   60   60
                                                          8   67   66
                                                          1   70   71
                                                          3   36   45
                                                          0    0    0
                                                          0    0    0
                                                          5   45   45
                                                          0   48   59
                                                          2   11    0
                                                          0   34   31
                                                          8   17   22
```

```
    0    7   22   34
    0    0    0    0
   18   21   24   25
   12   12   12   12
   15   17   18   18
   28   33   36   38
   36   35   34   34
   26   15    6    2
    4    0    0    0
    0    0    0    0

Center location is (15
>>
```

```
ect.

3D image
the image.

whole to try
```

MATLAB Live Editor

The Live Editor provides a new way to create, edit and run MATLAB scripts.



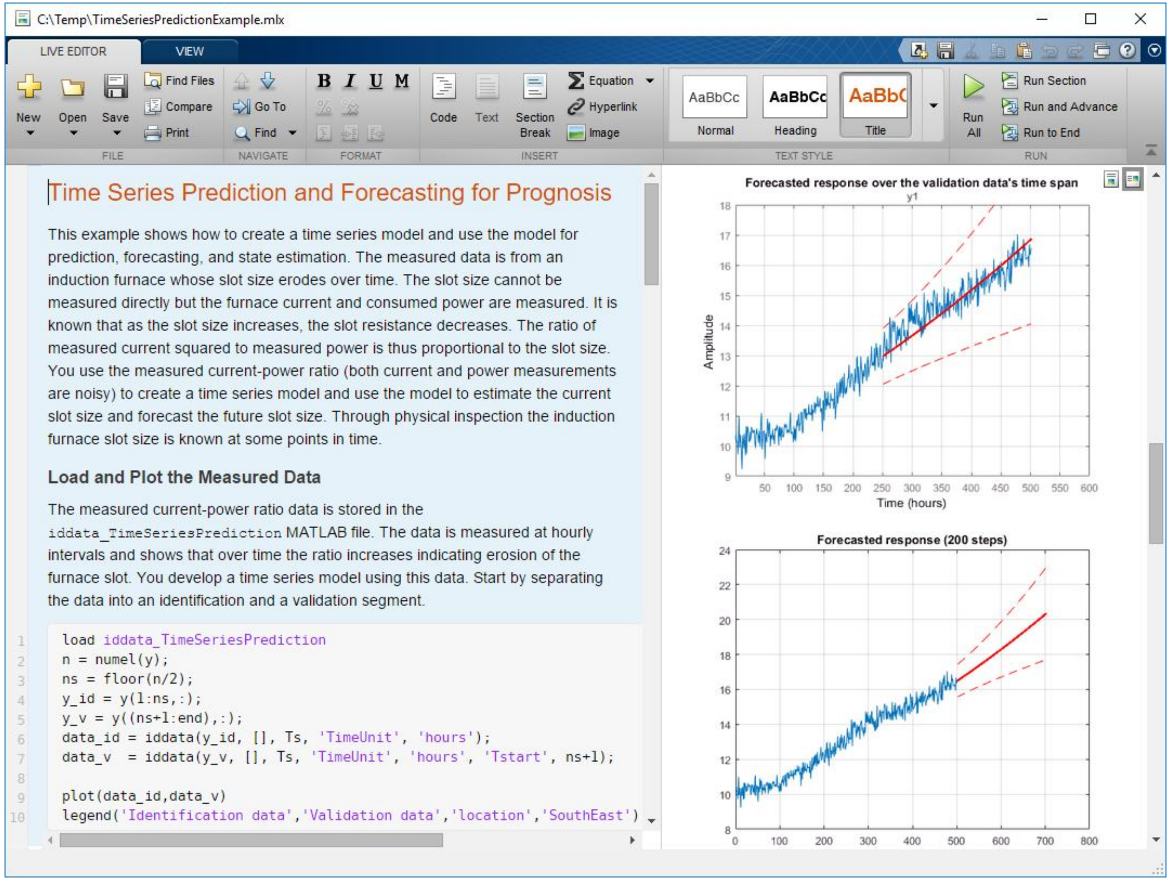
```
load iddata_TimeSeriesPrediction
n = numel(y);
ns = floor(n/2);
y_id = y(1:ns,:);
y_v = y((ns+1:end),:);
data_id = iddata(y_id, [], Ts, 'TimeUnit', 'hours');
data_v  = iddata(y_v, [], Ts, 'TimeUnit', 'hours', 'Tstart', ns+1);

plot(data_id,data_v)
legend('Identification data','Validation data','location','SouthEast');
```

```
sys = ssest(data_id,1,'Ts',Ts,'form','canonical');
```

```
nstep = 10;
compare(sys,data_id,nstep)
grid('on');
```

MATLAB Live Editor

Turn script into an Interactive Narrative for Exploratory Learning and for Teaching.



Time Series Prediction and Forecasting for Prognosis

This example shows how to create a time series model and use the model for prediction, forecasting, and state estimation. The measured data is from an induction furnace whose slot size erodes over time. The slot size cannot be measured directly but the furnace current and consumed power are measured. It is known that as the slot size increases, the slot resistance decreases. The ratio of measured current squared to measured power is thus proportional to the slot size. You use the measured current-power ratio (both current and power measurements are noisy) to create a time series model and use the model to estimate the current slot size and forecast the future slot size. Through physical inspection the induction furnace slot size is known at some points in time.

Load and Plot the Measured Data

The measured current-power ratio data is stored in the `iddata_TimeSeriesPrediction` MATLAB file. The data is measured at hourly intervals and shows that over time the ratio increases indicating erosion of the furnace slot. You develop a time series model using this data. Start by separating the data into an identification and a validation segment.

```
load iddata_TimeSeriesPrediction
n = numel(y);
ns = floor(n/2);
y_id = y(1:ns,:);
y_v = y((ns+1:end),:);
data_id = iddata(y_id, [], Ts, 'TimeUnit', 'hours');
data_v  = iddata(y_v, [], Ts, 'TimeUnit', 'hours', 'Tstart', ns+1);

plot(data_id,data_v)
legend('Identification data','Validation data','location','SouthEast')
```

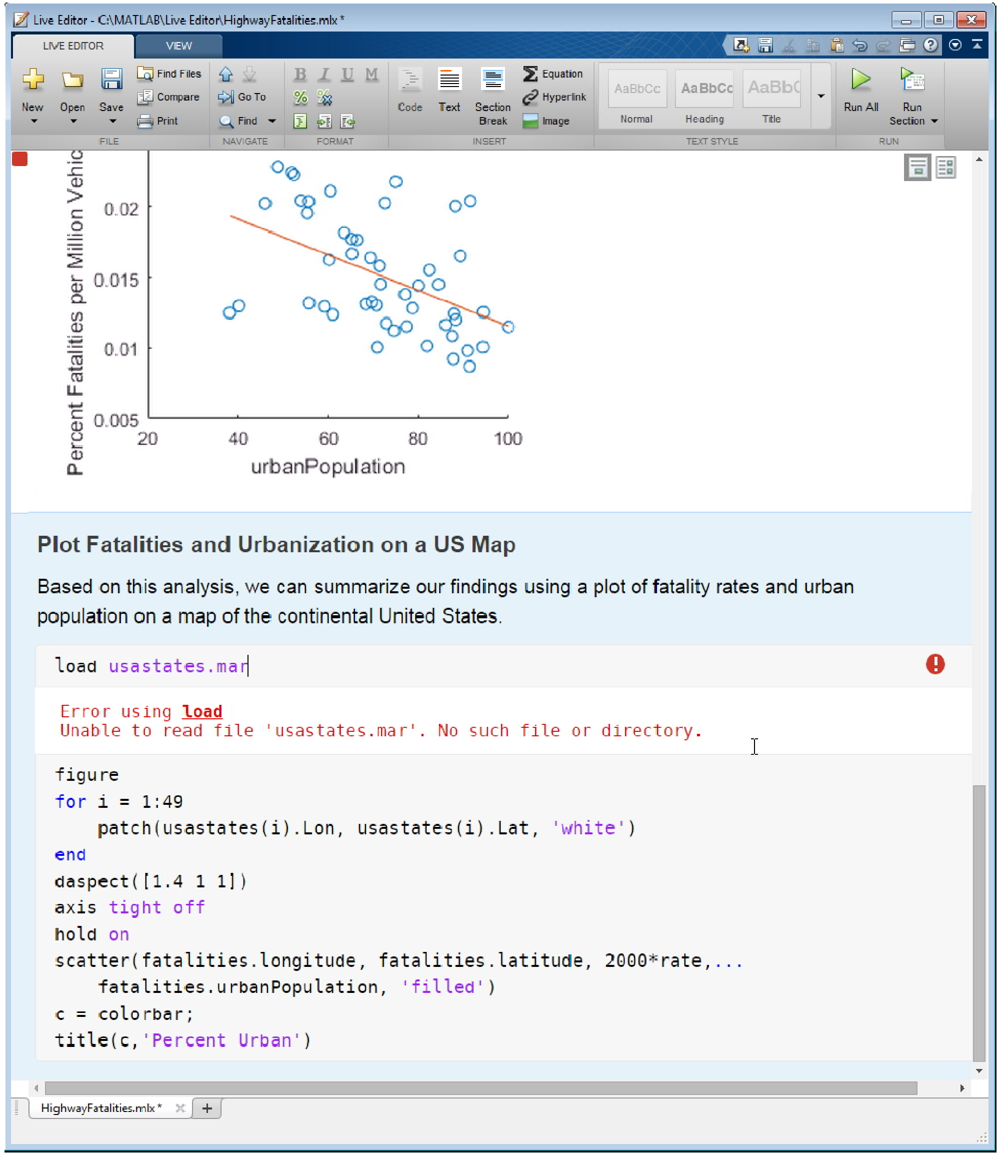
Live Editor – Areas of Application

- Exploratory Programming and Learning
- Create an Interactive Narrative
- Teach with Live Scripts

Live Editor

Exploratory Programming and Learning

- Write, execute, and test code in a single interactive environment
- Generate results and graphics alongside the code that produced them
- Run blocks of code individually or run the whole file
- Find errors at the location in the file where they occur



Plot Fatalities and Urbanization on a US Map

Based on this analysis, we can summarize our findings using a plot of fatality rates and urban population on a map of the continental United States.

```
load usastates.mar
```

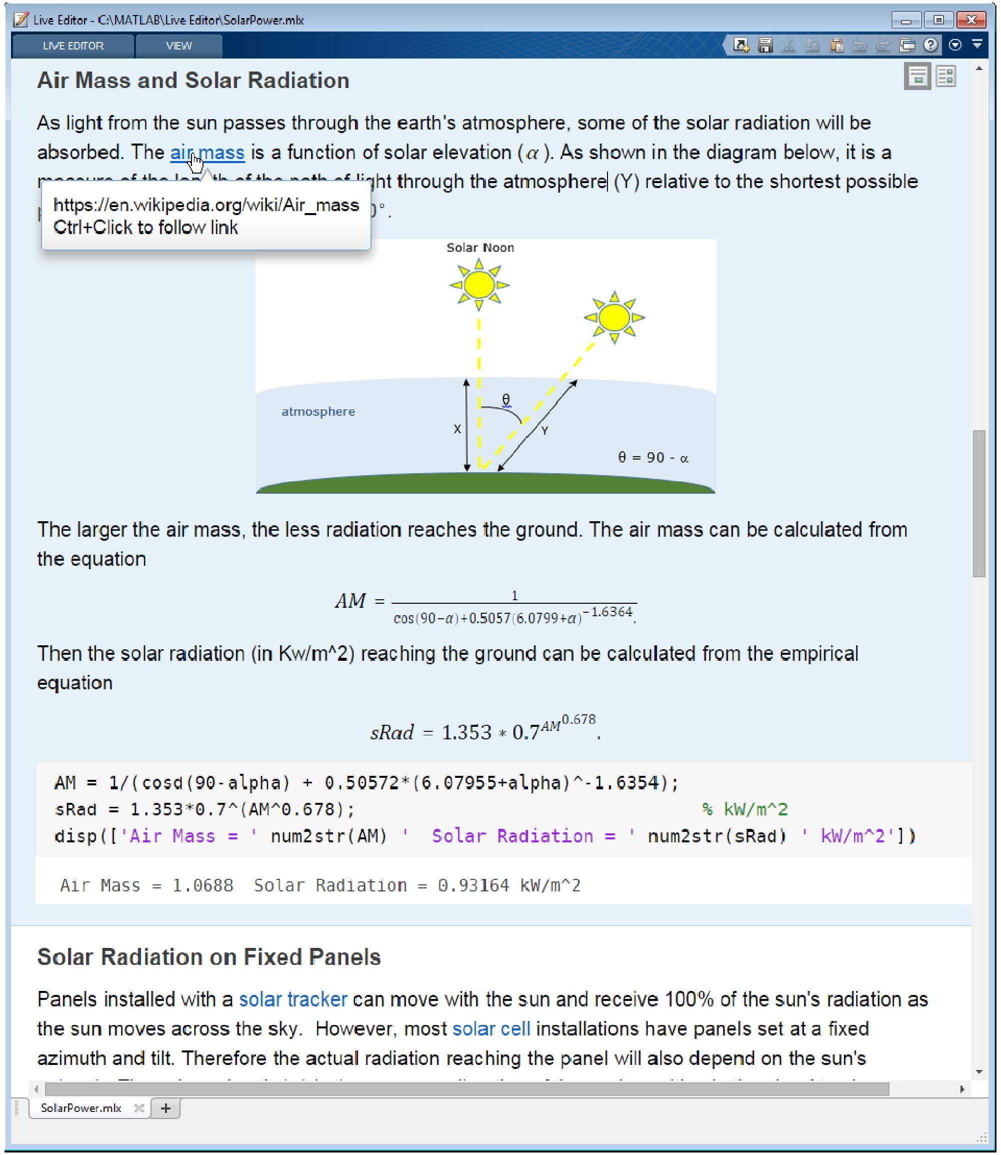
```
Error using load
Unable to read file 'usastates.mar'. No such file or directory.
```

```
figure
for i = 1:49
    patch(usastates(i).Lon, usastates(i).Lat, 'white')
end
daspect([1.4 1 1])
axis tight off
hold on
scatter(fatalities.longitude, fatalities.latitude, 2000*rate,...
    fatalities.urbanPopulation, 'filled')
c = colorbar;
title(c,'Percent Urban')
```

Live Editor

Create an Interactive Narrative

- Add titles, headings, and formatted text
- Include equations
- Add images, and hyperlinks as background material
- Save your narrative with code, results, images, and text in a single file
- Others can use your narrative to validate and extend your results
- Convert interactive documents to HTML or PDF for publication



Air Mass and Solar Radiation

As light from the sun passes through the earth's atmosphere, some of the solar radiation will be absorbed. The air mass is a function of solar elevation (α). As shown in the diagram below, it is a measure of the length of the path of light through the atmosphere (Y) relative to the shortest possible [illegible] °.

https://en.wikipedia.org/wiki/Air_mass
Ctrl+Click to follow link

The larger the air mass, the less radiation reaches the ground. The air mass can be calculated from the equation

$$AM = \frac{1}{\cos(90-\alpha)+0.5057(6.0799+\alpha)^{-1.6364}}.$$

Then the solar radiation (in Kw/m^2) reaching the ground can be calculated from the empirical equation

$$sRad = 1.353 * 0.7^{AM^{0.678}}.$$

```
AM = 1/(cosd(90-alpha) + 0.50572*(6.07955+alpha)^-1.6354);
sRad = 1.353*0.7^(AM^0.678);                                    % kW/m^2
disp(['Air Mass = ' num2str(AM) '  Solar Radiation = ' num2str(sRad) ' kW/m^2'])
```

```
Air Mass = 1.0688  Solar Radiation = 0.93164 kW/m^2
```

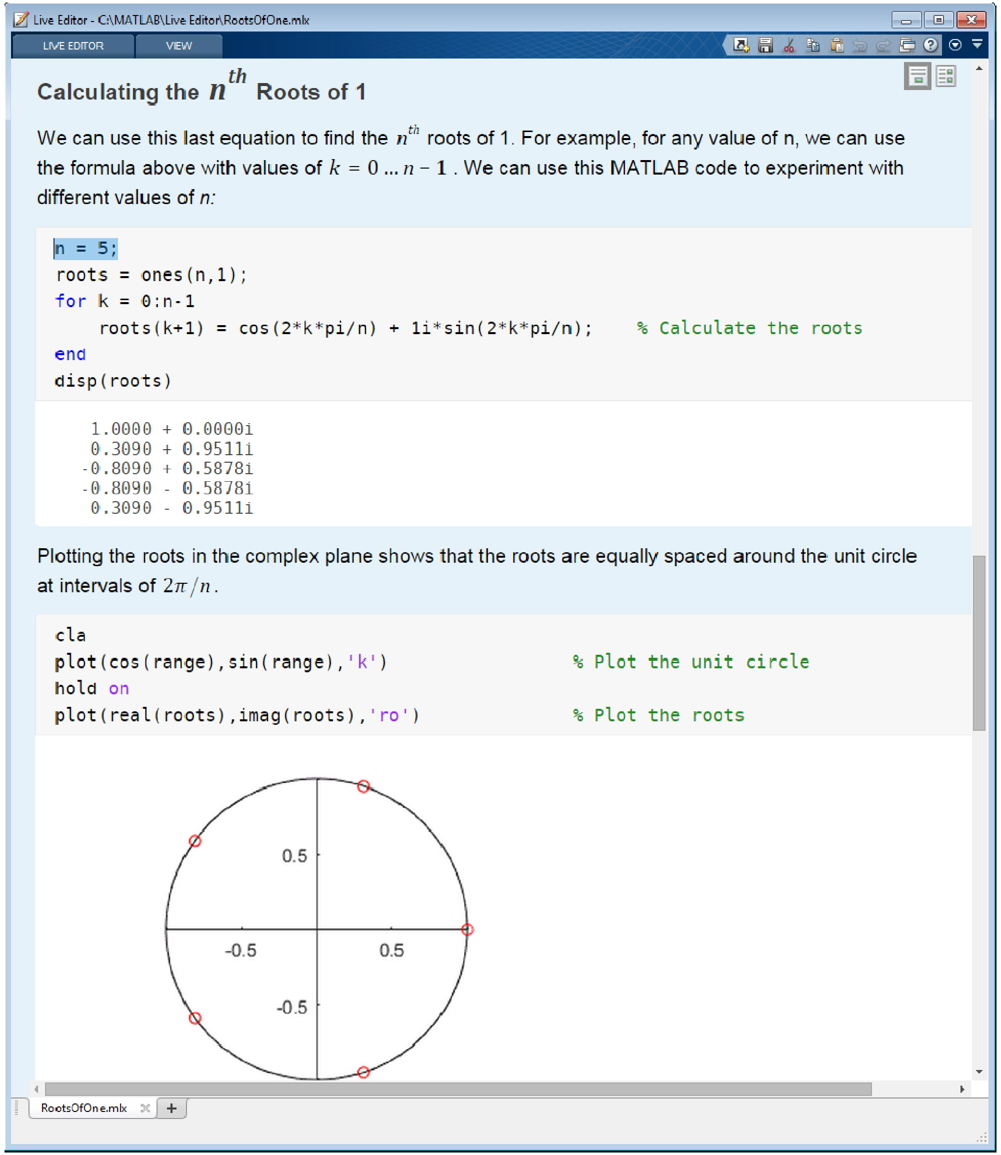
Solar Radiation on Fixed Panels

Panels installed with a solar tracker can move with the sun and receive 100% of the sun's radiation as the sun moves across the sky. However, most solar cell installations have panels set at a fixed azimuth and tilt. Therefore the actual radiation reaching the panel will also depend on the sun's

Live Editor

Teach with Live Scripts

- Create training materials that combine code and results with formatted text and mathematical equations
- Include images, and links to supporting materials
- Modify and run code on the fly to answer questions or explore related topics
- Share as interactive documents or in hardcopy format.
- Create partially completed files for individual assignments or team projects



Calculating the n^{th} Roots of 1

We can use this last equation to find the n^{th} roots of 1. For example, for any value of n, we can use the formula above with values of $k = 0 \ldots n-1$. We can use this MATLAB code to experiment with different values of *n:*

```
n = 5;
roots = ones(n,1);
for k = 0:n-1
    roots(k+1) = cos(2*k*pi/n) + 1i*sin(2*k*pi/n);    % Calculate the roots
end
disp(roots)
```

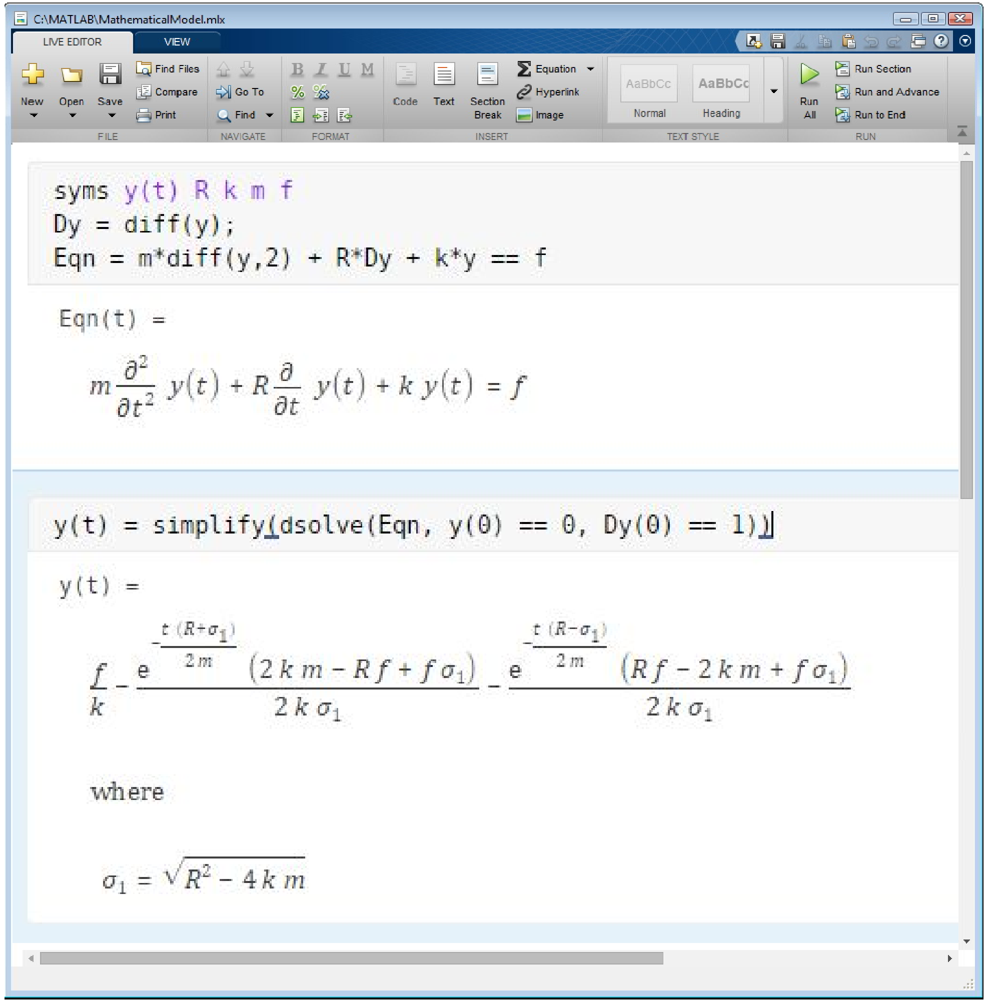
```
   1.0000 + 0.0000i
   0.3090 + 0.9511i
  -0.8090 + 0.5878i
  -0.8090 - 0.5878i
   0.3090 - 0.9511i
```

Plotting the roots in the complex plane shows that the roots are equally spaced around the unit circle at intervals of $2\pi/n$.

```
cla
plot(cos(range),sin(range),'k')                 % Plot the unit circle
hold on
plot(real(roots),imag(roots),'ro')              % Plot the roots
```

Live Editor – Symbolic Math

- **Math –** Create, manipulate, substitute and solve equations in a familiar mathematical typeset.



```
syms y(t) R k m f
Dy = diff(y);
Eqn = m*diff(y,2) + R*Dy + k*y == f
```

Eqn(t) =

$$m \frac{\partial^2}{\partial t^2} y(t) + R \frac{\partial}{\partial t} y(t) + k\, y(t) = f$$

```
y(t) = simplify(dsolve(Eqn, y(0) == 0, Dy(0) == 1))
```

y(t) =

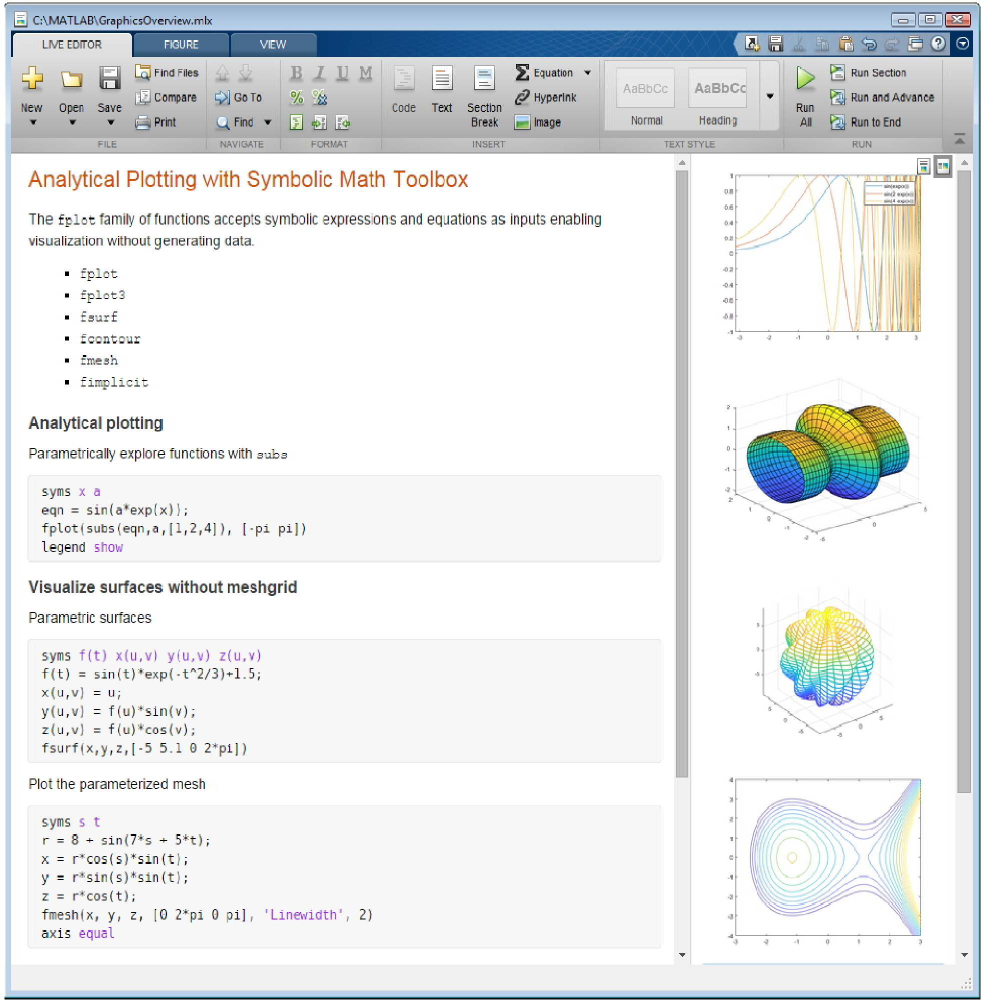
$$\frac{f}{k} - \frac{e^{-\frac{t\,(R+\sigma_1)}{2\,m}}\left(2\,k\,m - R\,f + f\,\sigma_1\right)}{2\,k\,\sigma_1} - \frac{e^{-\frac{t\,(R-\sigma_1)}{2\,m}}\left(R\,f - 2\,k\,m + f\,\sigma_1\right)}{2\,k\,\sigma_1}$$

where

$$\sigma_1 = \sqrt{R^2 - 4\,k\,m}$$

Live Editor – Symbolic Math

- **Math –** Create, manipulate, substitute and solve equations in a familiar mathematical typeset.
- **Visualize** – Plot expressions and equations without generating discrete data.



Analytical Plotting with Symbolic Math Toolbox

The `fplot` family of functions accepts symbolic expressions and equations as inputs enabling visualization without generating data.

- `fplot`
- `fplot3`
- `fsurf`
- `fcontour`
- `fmesh`
- `fimplicit`

Analytical plotting

Parametrically explore functions with `subs`

```
syms x a
eqn = sin(a*exp(x));
fplot(subs(eqn,a,[1,2,4]), [-pi pi])
legend show
```

Visualize surfaces without meshgrid

Parametric surfaces

```
syms f(t) x(u,v) y(u,v) z(u,v)
f(t) = sin(t)*exp(-t^2/3)+1.5;
x(u,v) = u;
y(u,v) = f(u)*sin(v);
z(u,v) = f(u)*cos(v);
fsurf(x,y,z,[-5 5.1 0 2*pi])
```

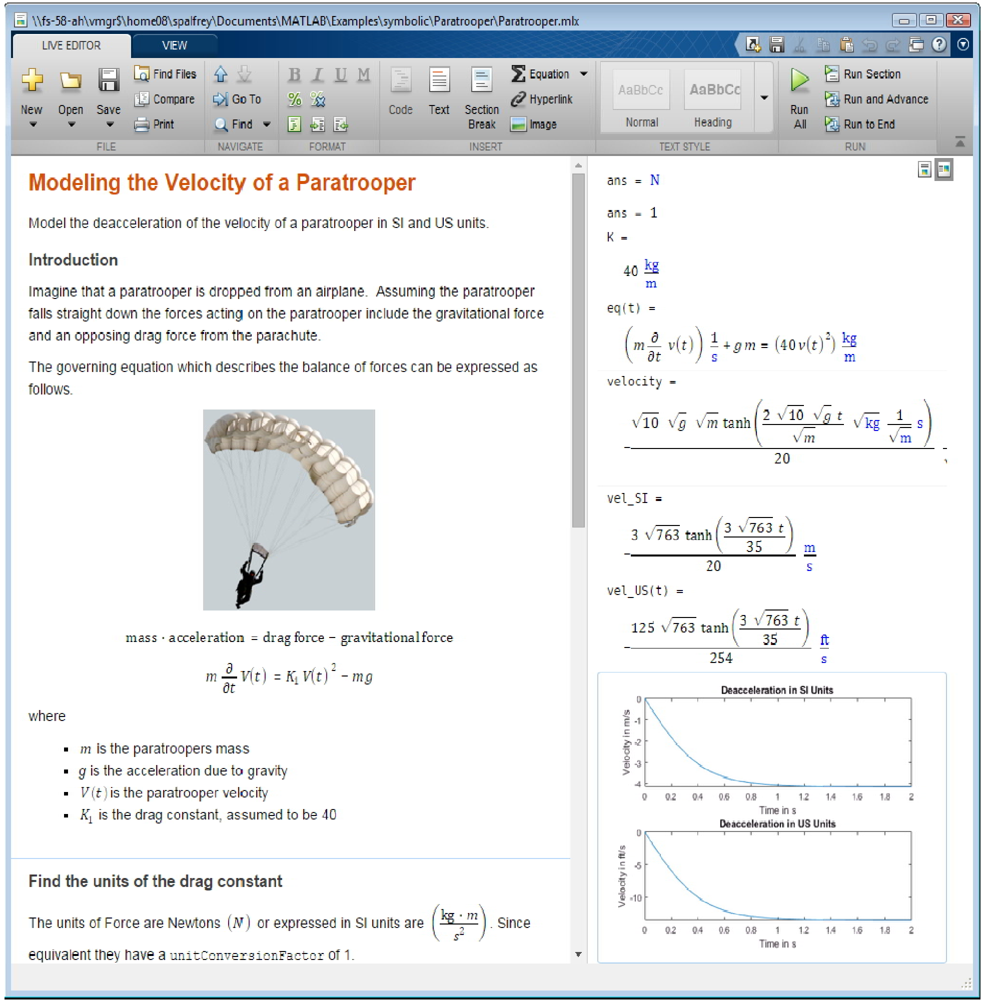
Plot the parameterized mesh

```
syms s t
r = 8 + sin(7*s + 5*t);
x = r*cos(s)*sin(t);
y = r*sin(s)*sin(t);
z = r*cos(t);
fmesh(x, y, z, [0 2*pi 0 pi], 'Linewidth', 2)
axis equal
```

MATLAB EXPO 2017

Live Editor – Symbolic Math

- **Math –** Create, manipulate, substitute and solve equations in a familiar mathematical typeset.
- **Visualize** – Plot expressions and equations without generating discrete data.
- **Units** – Work with dimensioned physical quantities.



Modeling the Velocity of a Paratrooper

Model the deacceleration of the velocity of a paratrooper in SI and US units.

Introduction

Imagine that a paratrooper is dropped from an airplane. Assuming the paratrooper falls straight down the forces acting on the paratrooper include the gravitational force and an opposing drag force from the parachute.

The governing equation which describes the balance of forces can be expressed as follows.

$$\text{mass} \cdot \text{acceleration} = \text{drag force} - \text{gravitational force}$$

$$m \frac{\partial}{\partial t} V(t) = K_1 V(t)^2 - mg$$

where

- m is the paratroopers mass
- g is the acceleration due to gravity
- $V(t)$ is the paratrooper velocity
- K_1 is the drag constant, assumed to be 40

Find the units of the drag constant

The units of Force are Newtons (N) or expressed in SI units are $\left(\frac{\mathrm{kg} \cdot m}{s^2}\right)$. Since equivalent they have a `unitConversionFactor` of 1.

ans = N

ans = 1

K =

$$40 \, \frac{\mathrm{kg}}{\mathrm{m}}$$

eq(t) =

$$\left(m \frac{\partial}{\partial t} v(t)\right) \frac{1}{\mathrm{s}} + g\,m = \left(40\,v(t)^2\right) \frac{\mathrm{kg}}{\mathrm{m}}$$

velocity =

$$-\frac{\sqrt{10}\,\sqrt{g}\,\sqrt{m}\,\tanh\left(\frac{2\sqrt{10}\,\sqrt{g}\,t}{\sqrt{m}}\,\sqrt{\mathrm{kg}}\,\frac{1}{\sqrt{\mathrm{m}}}\,\mathrm{s}\right)}{20}$$

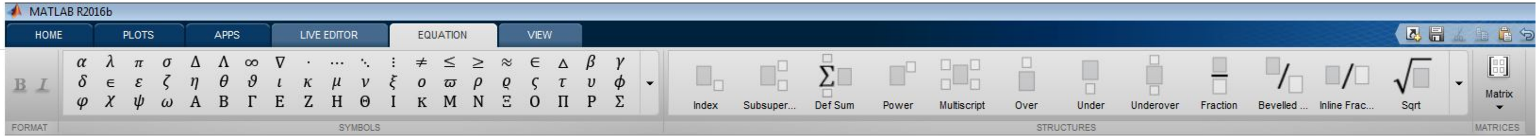
vel_SI =

$$-\frac{3\sqrt{763}\,\tanh\left(\frac{3\sqrt{763}\,t}{35}\right)}{20}\,\frac{\mathrm{m}}{\mathrm{s}}$$

vel_US(t) =

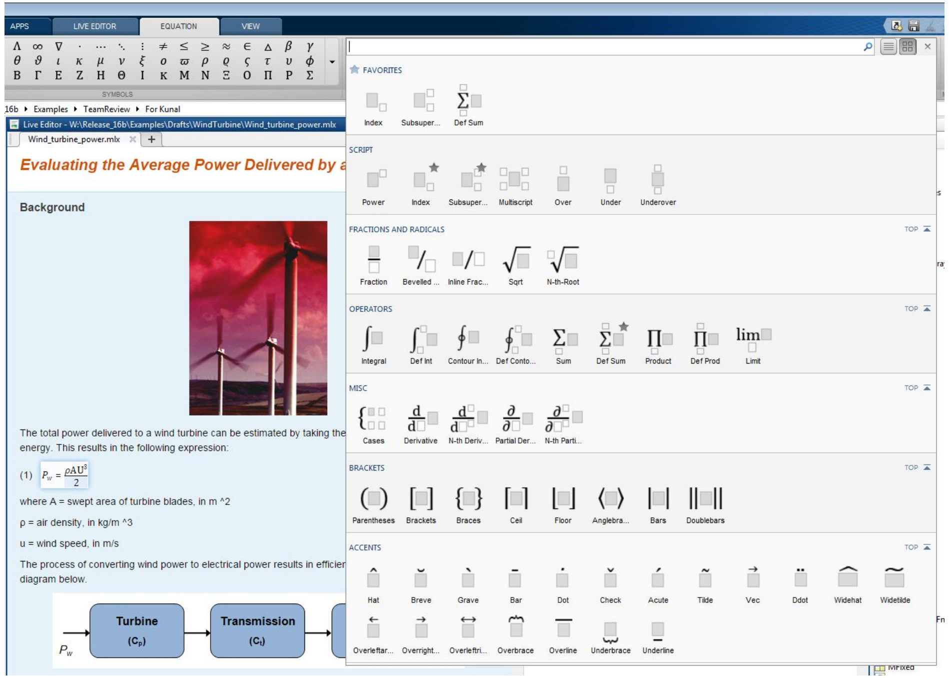
$$-\frac{125\sqrt{763}\,\tanh\left(\frac{3\sqrt{763}\,t}{35}\right)}{254}\,\frac{\mathrm{ft}}{\mathrm{s}}$$

Live Editor – Equation Editing



Create equations

- Integrated equation editor
- Easy authoring of mathematics.
- Shortkeys
- Copy equation as LaTeX or MathML



Evaluating the Average Power Delivered by a

Background

The total power delivered to a wind turbine can be estimated by taking the energy. This results in the following expression:

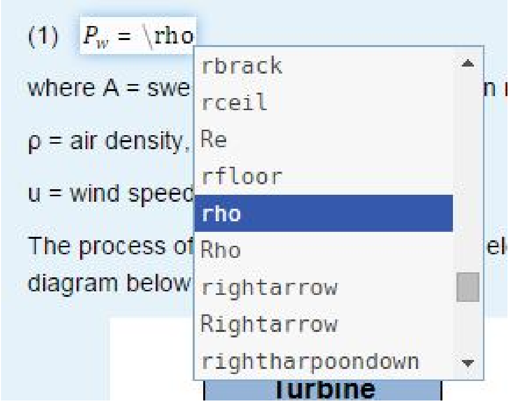
(1) $P_w = \frac{\rho A U^3}{2}$

where A = swept area of turbine blades, in m ^2

ρ = air density, in kg/m ^3

u = wind speed, in m/s

The process of converting wind power to electrical power results in efficien diagram below.



(1) P_w = \rho

where A = swe

ρ = air density,

u = wind speed

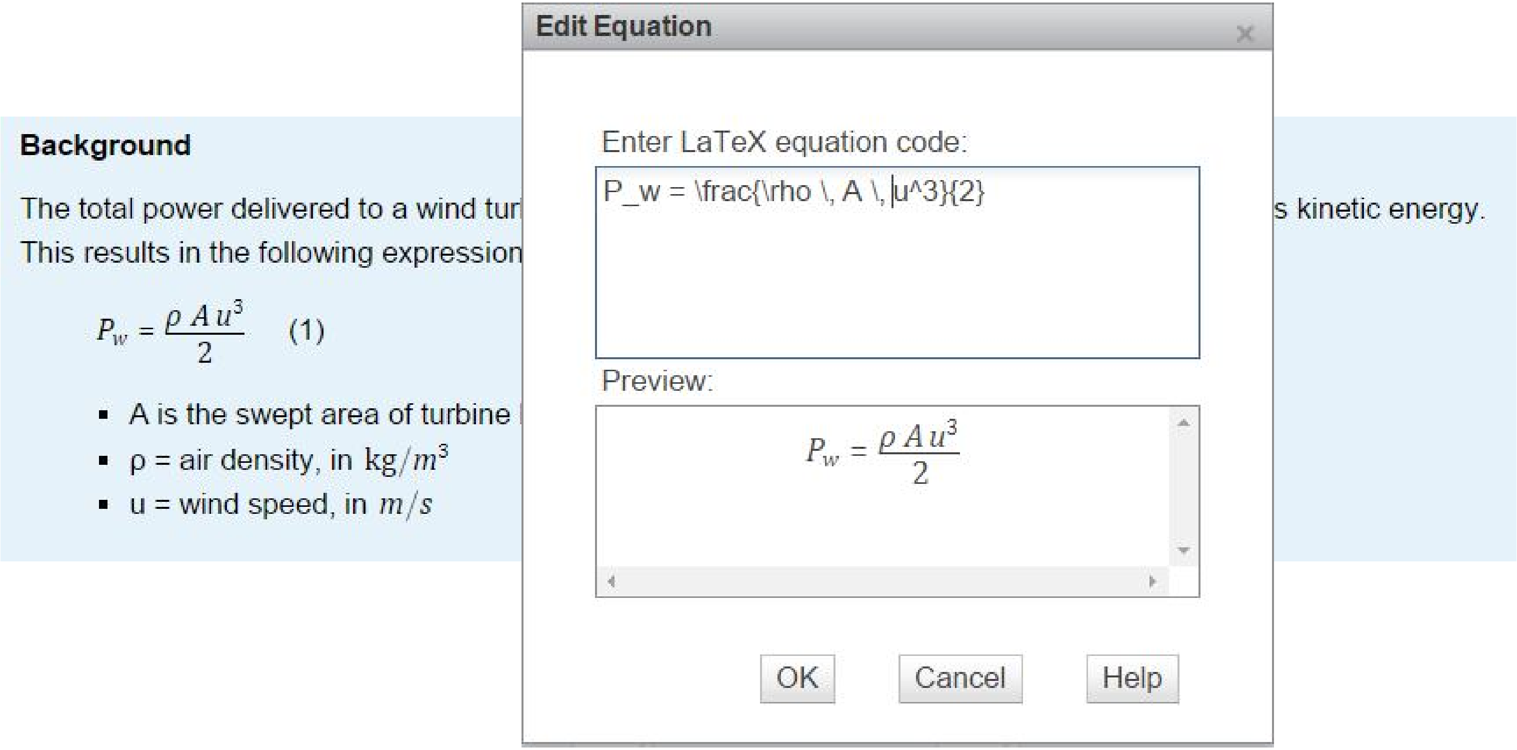
The process of

diagram below

Live Editor – Equation Editing

Create equations

- LaTeX input.



Background

The total power delivered to a wind tur s kinetic energy.

This results in the following expression

$$P_w = \frac{\rho\, A\, u^3}{2} \quad (1)$$

- A is the swept area of turbine
- ρ = air density, in kg/m^3
- u = wind speed, in m/s

Edit Equation

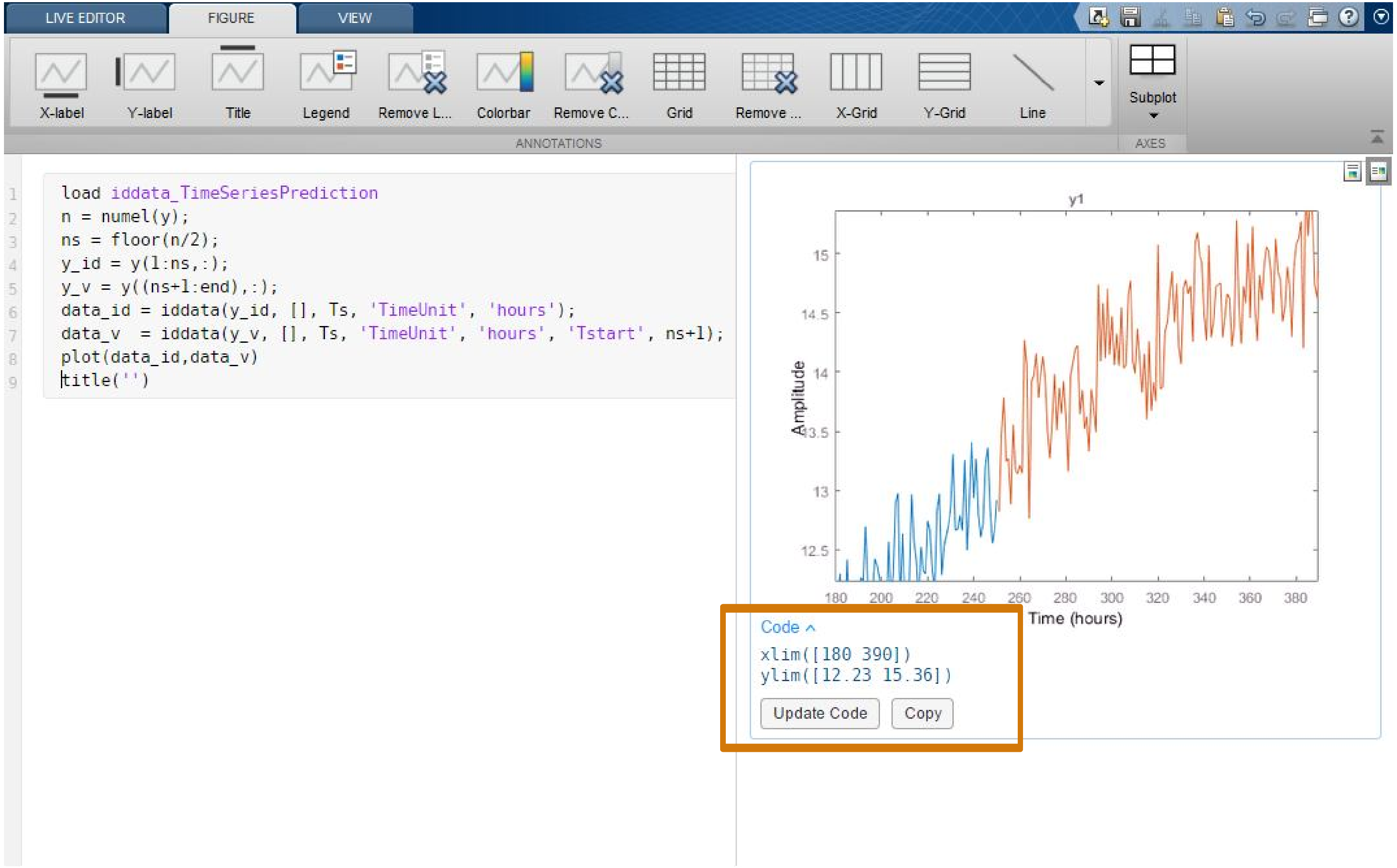
Enter LaTeX equation code:

```
P_w = \frac{\rho \, A \, u^3}{2}
```

Preview:

$$P_w = \frac{\rho\, A\, u^3}{2}$$

Live Editor – Interactive Figures

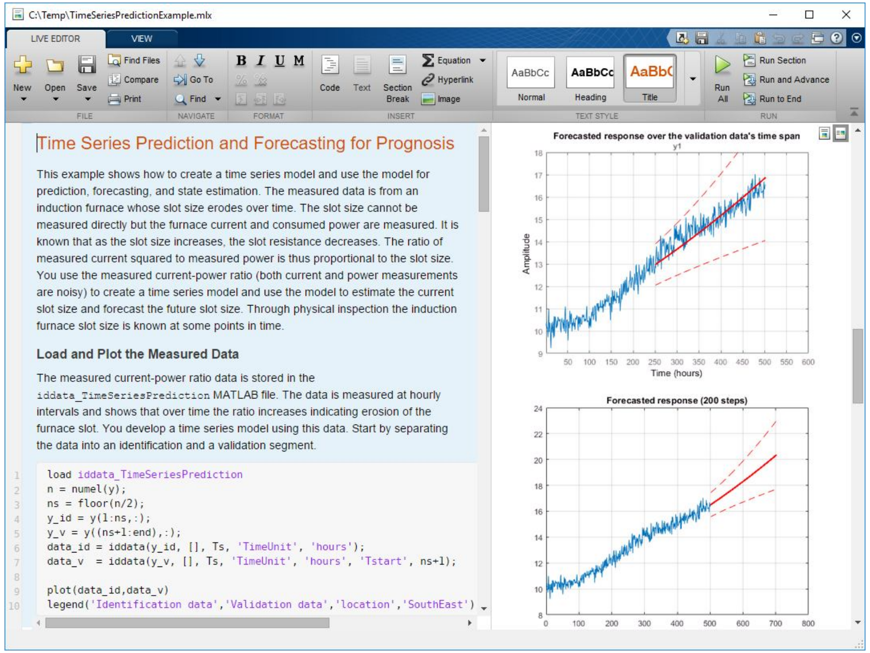


```
load iddata_TimeSeriesPrediction
n = numel(y);
ns = floor(n/2);
y_id = y(1:ns,:);
y_v = y((ns+1:end),:);
data_id = iddata(y_id, [], Ts, 'TimeUnit', 'hours');
data_v  = iddata(y_v, [], Ts, 'TimeUnit', 'hours', 'Tstart', ns+1);
plot(data_id,data_v)
title('')
```

```
xlim([180 390])
ylim([12.23 15.36])
```

Live Editor – Availability

Desktop MATLAB



Time Series Prediction and Forecasting for Prognosis

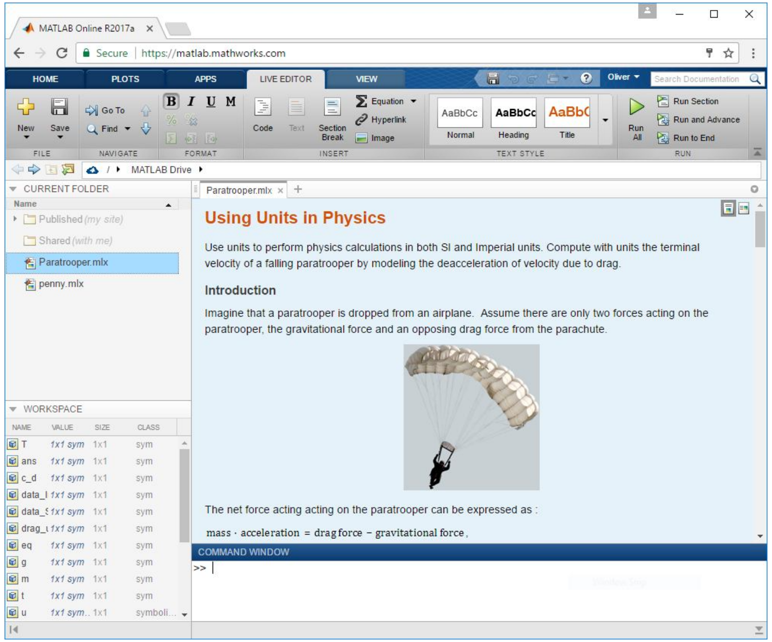
This example shows how to create a time series model and use the model for prediction, forecasting, and state estimation. The measured data is from an induction furnace whose slot size erodes over time. The slot size cannot be measured directly but the furnace current and consumed power are measured. It is known that as the slot size increases, the slot resistance decreases. The ratio of measured current squared to measured power is thus proportional to the slot size. You use the measured current-power ratio (both current and power measurements are noisy) to create a time series model and use the model to estimate the current slot size and forecast the future slot size. Through physical inspection the induction furnace slot size is known at some points in time.

Load and Plot the Measured Data

The measured current-power ratio data is stored in the `iddata_TimeSeriesPrediction` MATLAB file. The data is measured at hourly intervals and shows that over time the ratio increases indicating erosion of the furnace slot. You develop a time series model using this data. Start by separating the data into an identification and a validation segment.

```
load iddata_TimeSeriesPrediction
n = numel(y);
ns = floor(n/2);
y_id = y(1:ns,:);
y_v = y((ns+1:end),:);
data_id = iddata(y_id, [], Ts, 'TimeUnit', 'hours');
data_v  = iddata(y_v, [], Ts, 'TimeUnit', 'hours', 'Tstart', ns+1);

plot(data_id,data_v)
legend('Identification data','Validation data','location','SouthEast')
```

MATLAB Online



NAME	VALUE	SIZE	CLASS
T	1x1 sym	1x1	sym
ans	1x1 sym	1x1	sym
c_d	1x1 sym	1x1	sym
data_l	1x1 sym	1x1	sym
data_!	1x1 sym	1x1	sym
drag_	1x1 sym	1x1	sym
eq	1x1 sym	1x1	sym
g	1x1 sym	1x1	sym
m	1x1 sym	1x1	sym
t	1x1 sym	1x1	sym
u	1x1 sym..	1x1	symboli...

Using Units in Physics

Use units to perform physics calculations in both SI and Imperial units. Compute with units the terminal velocity of a falling paratrooper by modeling the deacceleration of velocity due to drag.

Introduction

Imagine that a paratrooper is dropped from an airplane. Assume there are only two forces acting on the paratrooper, the gravitational force and an opposing drag force from the parachute.

The net force acting acting on the paratrooper can be expressed as :

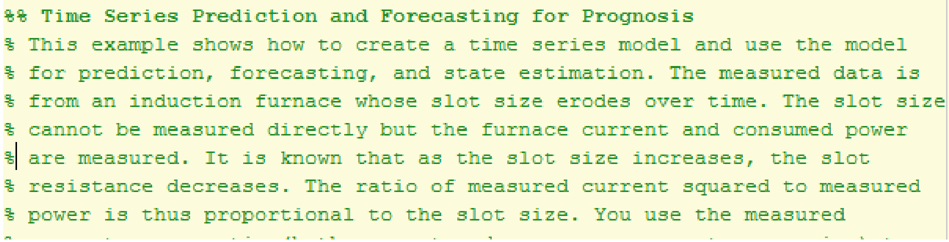
$$\mathrm{mass} \cdot \mathrm{acceleration} = \mathrm{drag\,force} - \mathrm{gravitational\,force},$$

```
>> 
```

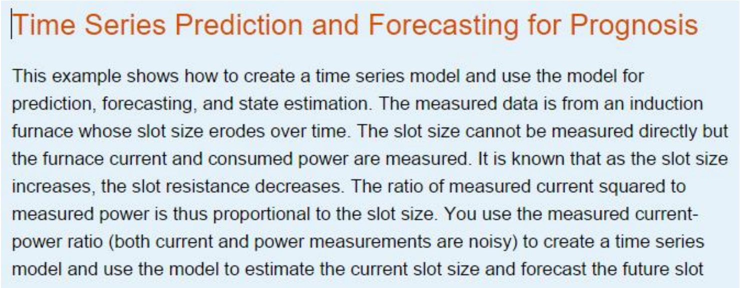
Live Scripts – Interoperability

Plain Scripts (.m scripts) can be opened as Live Scripts

Live Scripts can be saved as Plain Scripts



```
%% Time Series Prediction and Forecasting for Prognosis
% This example shows how to create a time series model and use the model
% for prediction, forecasting, and state estimation. The measured data is
% from an induction furnace whose slot size erodes over time. The slot size
% cannot be measured directly but the furnace current and consumed power
% are measured. It is known that as the slot size increases, the slot
% resistance decreases. The ratio of measured current squared to measured
% power is thus proportional to the slot size. You use the measured
```

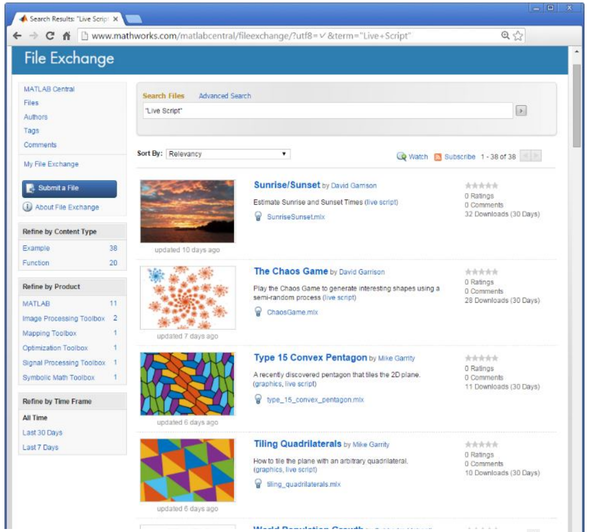


Time Series Prediction and Forecasting for Prognosis

This example shows how to create a time series model and use the model for prediction, forecasting, and state estimation. The measured data is from an induction furnace whose slot size erodes over time. The slot size cannot be measured directly but the furnace current and consumed power are measured. It is known that as the slot size increases, the slot resistance decreases. The ratio of measured current squared to measured power is thus proportional to the slot size. You use the measured current-power ratio (both current and power measurements are noisy) to create a time series model and use the model to estimate the current slot size and forecast the future slot

Learn More

- MATLAB Live Editor website
- Live Editor Webinar
- Documentation Examples
- Live scripts on File Exchange
- Symbolic Math Toolbox website



Search Results: "Live Scrip

www.mathworks.com/matlabcentral/fileexchange/?utf8=✓&term="Live+Script"

File Exchange

MATLAB Central
Files
Authors
Tags
Comments

My File Exchange

Submit a File

About File Exchange

Search Files   Advanced Search

"Live Script"

Sort By: Relevancy

Watch   Subscribe   1 - 38 of 38

Refine by Content Type	
Example	38
Function	20

Refine by Product	
MATLAB	11
Image Processing Toolbox	2
Mapping Toolbox	1
Optimization Toolbox	1
Signal Processing Toolbox	1
Symbolic Math Toolbox	1

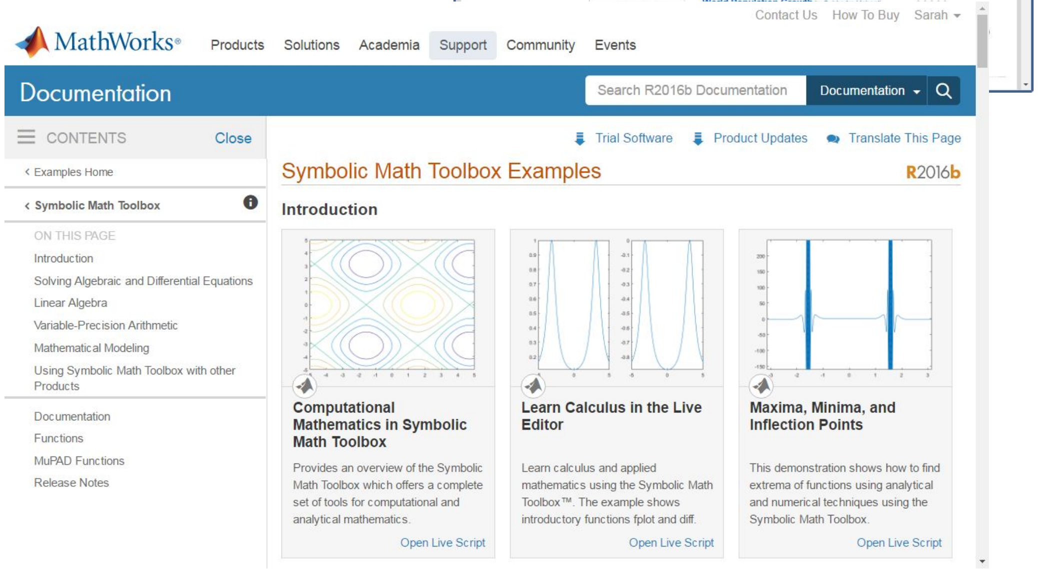
Refine by Time Frame
All Time
Last 30 Days
Last 7 Days

Sunrise/Sunset by David Garrison
Estimate Sunrise and Sunset Times (live script)
SunriseSunset.mlx
updated 10 days ago
0 Ratings
0 Comments
32 Downloads (30 Days)

The Chaos Game by David Garrison
Play the Chaos Game to generate interesting shapes using a semi-random process (live script)
ChaosGame.mlx
updated 7 days ago
0 Ratings
0 Comments
28 Downloads (30 Days)

Type 15 Convex Pentagon by Mike Garrity
A recently discovered pentagon that tiles the 2D plane. (graphics, live script)
type_15_convex_pentagon.mlx
updated 6 days ago
0 Ratings
0 Comments
11 Downloads (30 Days)

Tiling Quadrilaterals by Mike Garrity
How to tile the plane with an arbitrary quadrilateral. (graphics, live script)
tiling_quadrilaterals.mlx
updated 6 days ago
0 Ratings
0 Comments
10 Downloads (30 Days)



Contact Us   How To Buy   Sarah

MathWorks®   Products   Solutions   Academia   Support   Community   Events

Documentation   Search R2016b Documentation   Documentation

CONTENTS   Close

‹ Examples Home

‹ Symbolic Math Toolbox

ON THIS PAGE
Introduction
Solving Algebraic and Differential Equations
Linear Algebra
Variable-Precision Arithmetic
Mathematical Modeling
Using Symbolic Math Toolbox with other Products

Documentation
Functions
MuPAD Functions
Release Notes

Trial Software   Product Updates   Translate This Page

Symbolic Math Toolbox Examples   R2016b

Introduction

Computational Mathematics in Symbolic Math Toolbox
Provides an overview of the Symbolic Math Toolbox which offers a complete set of tools for computational and analytical mathematics.
Open Live Script

Learn Calculus in the Live Editor
Learn calculus and applied mathematics using the Symbolic Math Toolbox™. The example shows introductory functions fplot and diff.
Open Live Script

Maxima, Minima, and Inflection Points
This demonstration shows how to find extrema of functions using analytical and numerical techniques using the Symbolic Math Toolbox.
Open Live Script

www.mathworks.com/products/matlab/live-editor
www.mathworks.com/products/symbolic/

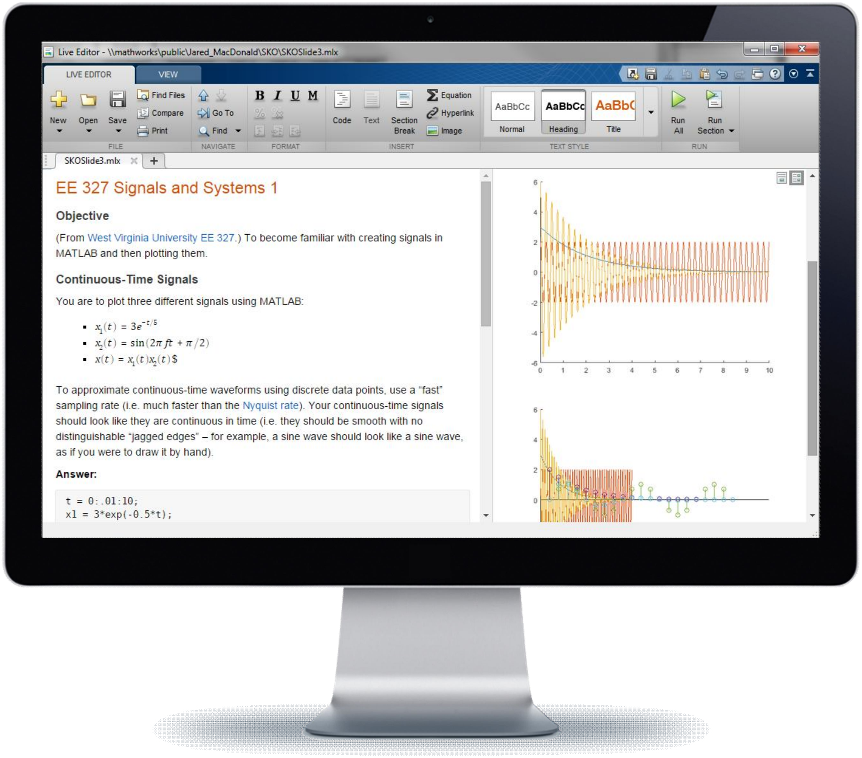
MATLAB EXPO 2017

Live Editor – Additional Information

On the following slides additional information can be found:

- Sharing Live Scripts
- Cross-Locale Sharing
- Functions in Scripts

Live Editor – Sharing



EE 327 Signals and Systems 1

Objective

(From West Virginia University EE 327.) To become familiar with creating signals in MATLAB and then plotting them.

Continuous-Time Signals

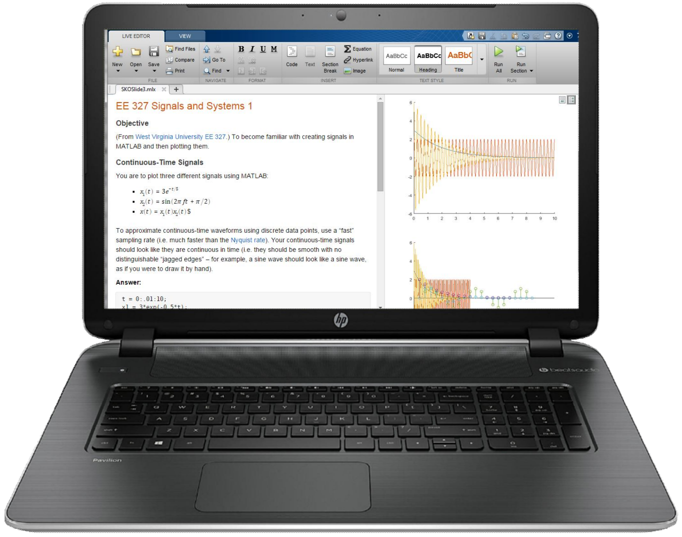
You are to plot three different signals using MATLAB:

- $x_1(t) = 3e^{-t/5}$
- $x_2(t) = \sin(2\pi ft + \pi/2)$
- $x(t) = x_1(t)x_2(t)$$

To approximate continuous-time waveforms using discrete data points, use a "fast" sampling rate (i.e. much faster than the Nyquist rate). Your continuous-time signals should look like they are continuous in time (i.e. they should be smooth with no distinguishable "jagged edges" – for example, a sine wave should look like a sine wave, as if you were to draw it by hand).

Answer:

```
t = 0:.01:10;
x1 = 3*exp(-0.5*t);
```



EE 327 Signals and Systems 1

Objective

(From West Virginia University EE 327.) To become familiar with creating signals in MATLAB and then plotting them.

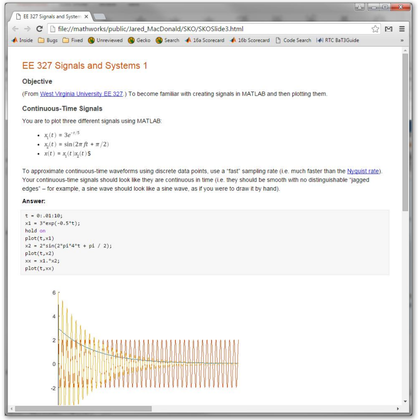
Continuous-Time Signals

You are to plot three different signals using MATLAB:

To approximate continuous-time waveforms using discrete data points, use a "fast" sampling rate (i.e. much faster than the Nyquist rate). Your continuous-time signals should look like they are continuous in time (i.e. they should be smooth with no distinguishable "jagged edges" – for example, a sine wave should look like a sine wave, as if you were to draw it by hand).

Answer:

Colleague with MATLAB



EE 327 Signals and Systems 1

Objective

(From West Virginia University EE 327.) To become familiar with creating signals in MATLAB and then plotting them.

Continuous-Time Signals

You are to plot three different signals using MATLAB:

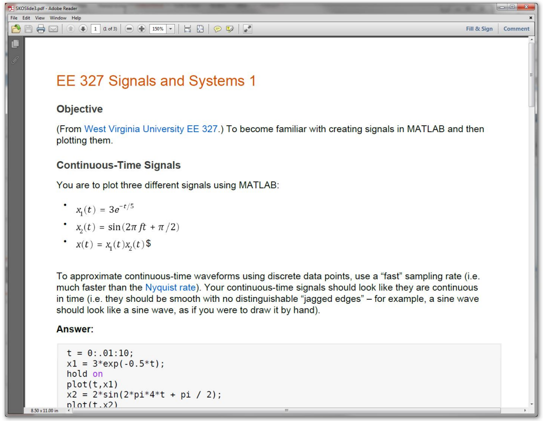
- $x_1(t) = 3e^{-t/5}$
- $x_2(t) = \sin(2\pi ft + \pi/2)$
- $x(t) = x_1(t)x_2(t)$$

To approximate continuous-time waveforms using discrete data points, use a "fast" sampling rate (i.e. much faster than the Nyquist rate). Your continuous-time signals should look like they are continuous in time (i.e. they should be smooth with no distinguishable "jagged edges" – for example, a sine wave should look like a sine wave, as if you were to draw it by hand).

Answer:

```
t = 0:.01:10;
x1 = 3*exp(-0.5*t);
hold on
plot(t,x1)
x2 = 2*sin(2*pi*4*t + pi / 2);
plot(t,x2)
xx = x1.*x2;
plot(t,xx)
```

HTML



EE 327 Signals and Systems 1

Objective

(From West Virginia University EE 327.) To become familiar with creating signals in MATLAB and then plotting them.

Continuous-Time Signals

You are to plot three different signals using MATLAB:

- $x_1(t) = 3e^{-t/5}$
- $x_2(t) = \sin(2\pi ft + \pi/2)$
- $x(t) = x_1(t)x_2(t)$$

To approximate continuous-time waveforms using discrete data points, use a "fast" sampling rate (i.e. much faster than the Nyquist rate). Your continuous-time signals should look like they are continuous in time (i.e. they should be smooth with no distinguishable "jagged edges" – for example, a sine wave should look like a sine wave, as if you were to draw it by hand).

Answer:

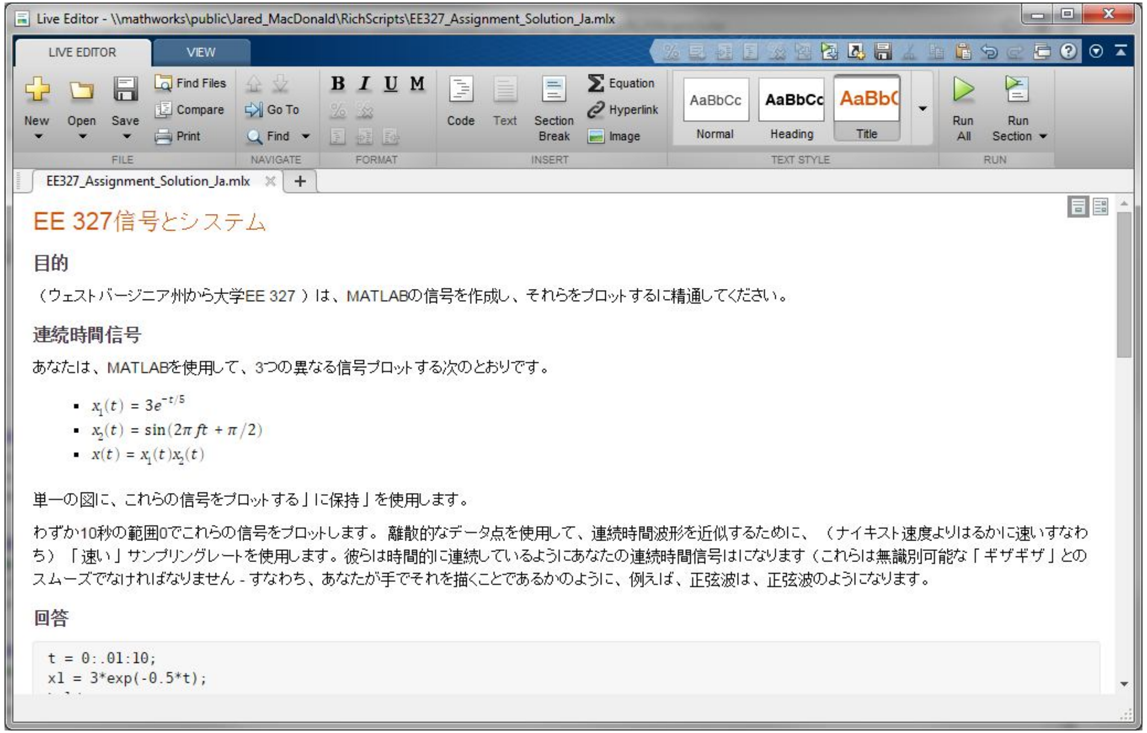
t = 0:.01:10;
x1 = 3*exp(-0.5*t);
hold on
plot(t,x1)
x2 = 2*sin(2*pi*4*t + pi / 2);
plot(t,x2)

PDF

Live Editor – Cross-Locale Sharing

Characters are correctly preserved across platforms and locales

- Share without loss of data with colleagues around the world
- Include symbols and special characters in your comments



EE 327信号とシステム

目的

（ウェストバージニア州から大学EE 327 ）は、MATLABの信号を作成し、それらをプロットするに精通してください。

連続時間信号

あなたは、MATLABを使用して、3つの異なる信号プロットする次のとおりです。

- $x_1(t) = 3e^{-t/5}$
- $x_2(t) = \sin(2\pi ft + \pi/2)$
- $x(t) = x_1(t)x_2(t)$

単一の図に、これらの信号をプロットする」に保持」を使用します。

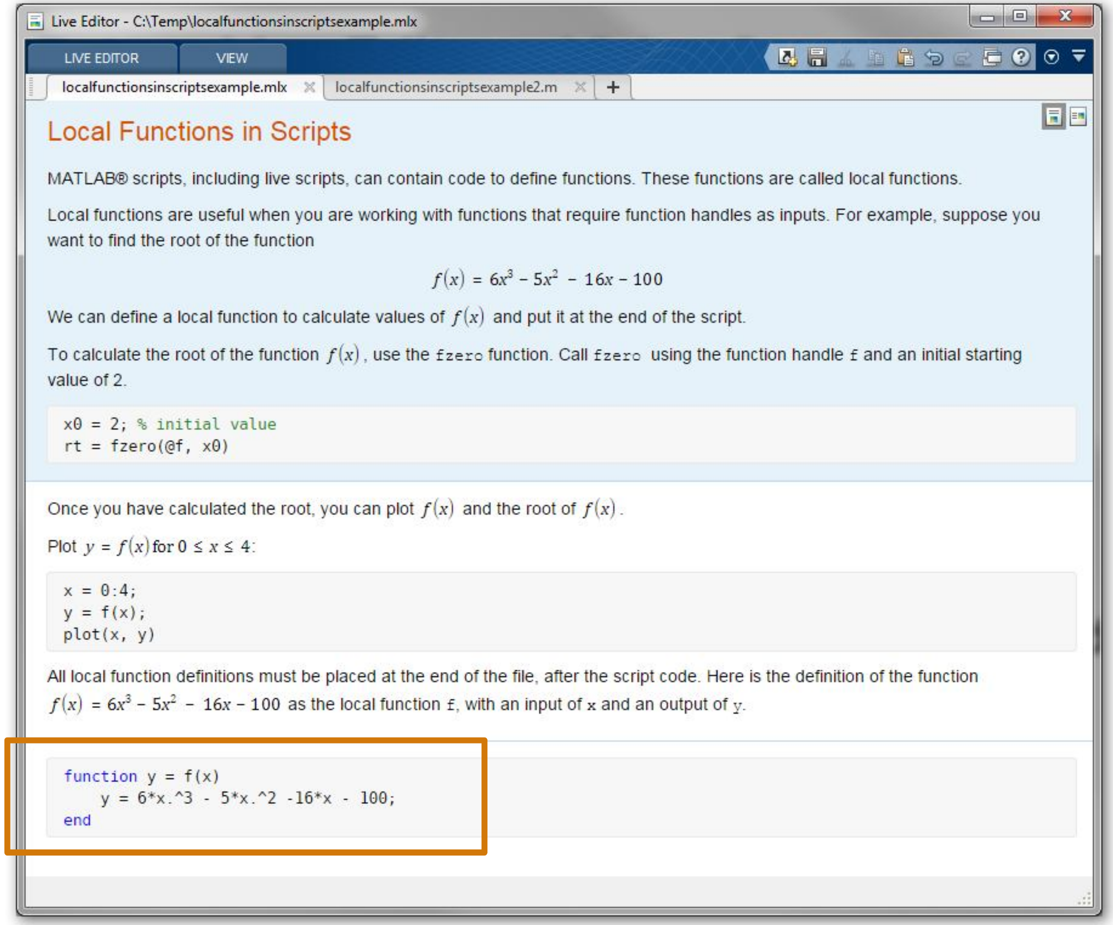
わずか10秒の範囲0でこれらの信号をプロットします。 離散的なデータ点を使用して、連続時間波形を近似するために、 （ナイキスト速度よりはるかに速いすなわち）「速い」サンプリングレートを使用します。彼らは時間的に連続しているようにあなたの連続時間信号はになります（これらは無識別可能な「ギザギザ」とのスムーズでなければなりません - すなわち、あなたが手でそれを描くことであるかのように、例えば、正弦波は、正弦波のようになります。

回答

```
t = 0:.01:10;
x1 = 3*exp(-0.5*t);
```

Live Editor – Functions in Scripts

Define and use functions from within a script, without needing to create a separate file


Live Editor - C:\Temp\localfunctionsinscriptsexample.mlx
LIVE EDITOR
VIEW
localfunctionsinscriptsexample.mlx
localfunctionsinscriptsexample2.m
+

Local Functions in Scripts

MATLAB® scripts, including live scripts, can contain code to define functions. These functions are called local functions.

Local functions are useful when you are working with functions that require function handles as inputs. For example, suppose you want to find the root of the function

$$f(x) = 6x^3 - 5x^2 - 16x - 100$$

We can define a local function to calculate values of $f(x)$ and put it at the end of the script.

To calculate the root of the function $f(x)$, use the `fzero` function. Call `fzero` using the function handle `f` and an initial starting value of 2.

```
x0 = 2; % initial value
rt = fzero(@f, x0)
```

Once you have calculated the root, you can plot $f(x)$ and the root of $f(x)$.

Plot $y = f(x)$ for $0 \le x \le 4$:

```
x = 0:4;
y = f(x);
plot(x, y)
```

All local function definitions must be placed at the end of the file, after the script code. Here is the definition of the function $f(x) = 6x^3 - 5x^2 - 16x - 100$ as the local function `f`, with an input of `x` and an output of `y`.

```
function y = f(x)
    y = 6*x.^3 - 5*x.^2 -16*x - 100;
end
```