RMIT University

OENG-1116: MODELLING & SIMUALTION OF ENGINEERING SYSTEMS

Week 1, Part 2

MATLAB in Simulating Systems, described with Ordinary Differential Equations

Prof Pavel M.Trivailo

pavel.trivailo@rmit.edu.au

FEEDBACK:

QUESTIONS
HOME WORK
OBSERVATIONS
GENERAL DISCUSSIONS

REVIEW:

MATLAB
AS CALCULATOR

STARTING MATLAB, EXPLORING CONTENT

STARTING MATLAB. EXPLORING CONTEXT

Command Window

New to MATLAB? See resources for Getting Started.

```
>> ver
-----------------------------------------------------------------------------
MATLAB Version: 8.5.0.197613 (R2015a)
MATLAB License Number: xxxxxxxx
Operating System: Microsoft Windows 8.1 Pro Version 6.3 (Build 9600)
Java Version: Java 1.7.0_60-b19 with Oracle Corporation Java HotSpot(TM) 64-Bit Ser
-----------------------------------------------------------------------------
MATLAB                                                Version 8.5       (R2015a)
Simulink                                              Version 8.5       (R2015a)
Aerospace Blockset                                    Version 3.15      (R2015a)
Aerospace Toolbox                                     Version 2.15      (R2015a)
Bioinformatics Toolbox                                Version 4.5.1     (R2015a)
Communications System Toolbox                         Version 6.0       (R2015a)
Computer Vision System Toolbox                        Version 6.2       (R2015a)
Control System Toolbox                                Version 9.9       (R2015a)
Curve Fitting Toolbox                                 Version 3.5.1     (R2015a)
DSP System Toolbox                                    Version 9.0       (R2015a)
```

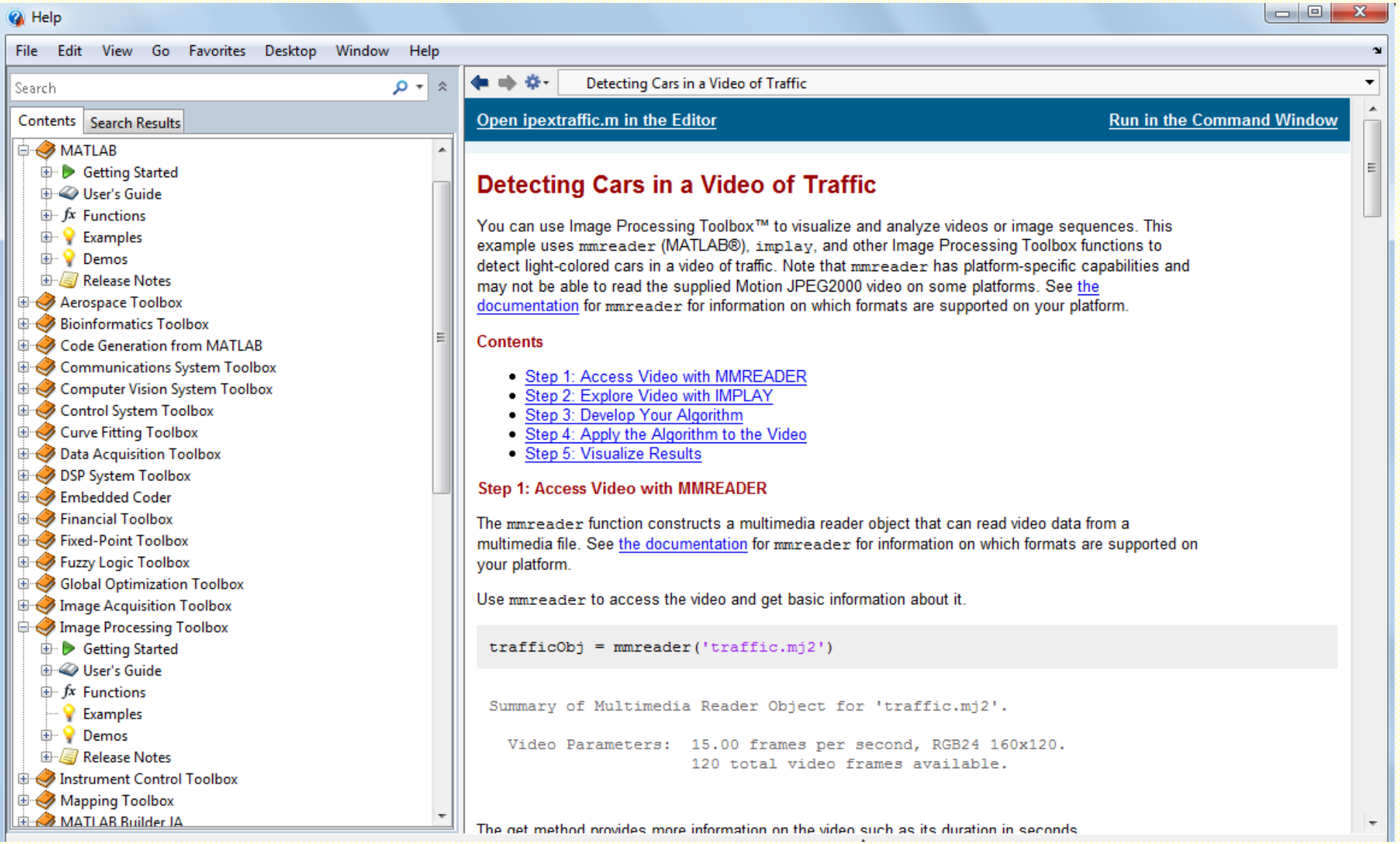
INSPECTING LICENCE & CONTENT:

MATLAB
Simulink
Aerospace Blockset
Aerospace Toolbox
Bioinformatics Toolbox
Communications System Toolbox
Computer Vision System Toolbox
Control System Toolbox
Curve Fitting Toolbox
DSP System Toolbox
Data Acquisition Toolbox
Datafeed Toolbox
Embedded Coder
Financial Instruments Toolbox
Financial Toolbox
Fixed-Point Designer
Fuzzy Logic Toolbox
Global Optimization Toolbox
HDL Coder

Image Acquisition Toolbox
Image Processing Toolbox
Instrument Control Toolbox
LTE System Toolbox
MATLAB Coder
MATLAB Compiler
MATLAB Compiler SDK
MATLAB Report Generator
Mapping Toolbox
Model Predictive Control Toolbox
Neural Network Toolbox
Optimization Toolbox
Parallel Computing Toolbox
Partial Differential Equation Toolbox
RF Toolbox
Robust Control Toolbox
Signal Processing Toolbox
SimElectronics
SimEvents
SimHydraulics

SimMechanics
SimPowerSystems
SimRF
Simscape
Simulink 3D Animation
Simulink Coder
Simulink Control Design
Simulink Design Optimization
Simulink Desktop Real-Time
Simulink Real-Time
Simulink Report Generator
Simulink Verification and Validation
Spreadsheet Link EX
Stateflow
Statistics and Machine Learning Toolbox
Symbolic Math Toolbox
System Identification Toolbox
Trading Toolbox
Wavelet Toolbox

PLEASE, EXPLORE DEMOS!



REVIEW
WITH
ADDED NEW ELEMENTS:
MATLAB
AS CALCULATOR

MATLAB as "basic" Calculator

$$2^5+1=?$$

1. **Calculation with MATLAB**

```
Command Window
New to MATLAB? See resources for Getting Started.
>> 2^5+1

ans =

     33
```

$$\left[2^5\ 3^5\right]+1=?$$

1. **Calculation with MATLAB**

```
Command Window
New to MATLAB? See resources for Getti
>> [2^5 3^5]+1

ans =

     33    244
```

2. **Calculation with MATLAB**

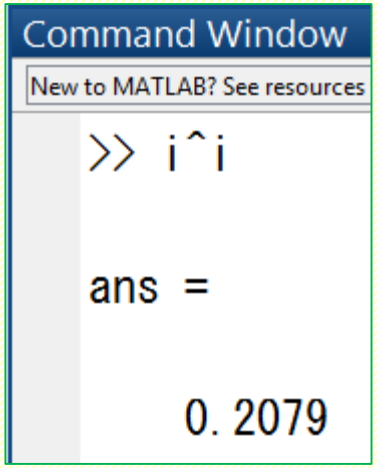
```
Command Window
New to MATLAB? See resources for Ge
>> [2 3].^5+1

ans =

     33    244
```

"PSEUDO" OPERATION !!!

MATLAB as very "smart" Calculator: Example-1

$$i^i = ?$$

1. Calculation with MATLAB



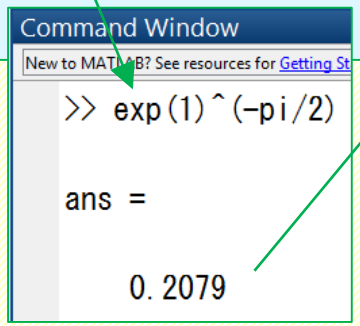
!!!!?!!!!?!!!!?

2. Analytical Solution

$$i^i = e^{\ln(i^i)} = e^{i\,\ln(i)}$$

$$e^{i\theta} = \cos\theta + i\sin\theta$$

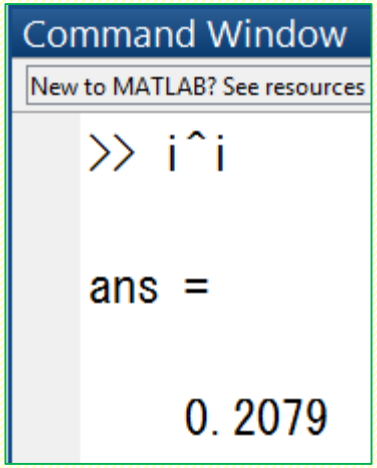
$$i^i = e^{\ln(i^i)} = e^{i\,\ln(i)} = e^{i\times i\frac{\pi}{2}} = e^{-\frac{\pi}{2}}$$



MATLAB as very "smart" Calculator: Example-1

$$i^i = ?$$

1. Calculation with MATLAB



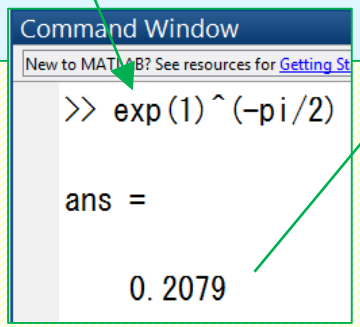
!!!!?!!!!?!!!!?

2. Analytical Solution

$$i^i = e^{\ln(i^i)} = e^{i\,\ln(i)}$$

$$e^{i\theta} = \cos\theta + i\sin\theta \quad \Rightarrow \quad e^{i\frac{\pi}{2}} = i \quad \Rightarrow \quad \ln(i) = i\frac{\pi}{2}$$

$$i^i = e^{\ln(i^i)} = e^{i\,\ln(i)} = e^{i\times i\frac{\pi}{2}} = e^{-\frac{\pi}{2}} \approx 0.2079$$



MATLAB as very "smart" Calculator: Example-2

$$(-1)^{\pi} = ?$$

1. Calculation with MATLAB

Command Window

New to MATLAB? See resources for Getting Started.

```
>> (-1)^pi

ans =

  -0.9027 - 0.4303i
```

2. Analytical Solution

$$e^{i\theta} = \cos\theta + i\sin\theta$$

$$e^{i\pi} = \cos\pi + i\sin\pi$$

$$e^{i\pi} = (-1) + i(0)$$

$$e^{i\pi} = (-1)$$

$$(-1) = e^{i\pi}$$

$$(-1)^{\pi} = \left(e^{i\pi}\right)^{\pi}$$

$$(-1)^{\pi} = e^{i\pi^2}$$

$$(-1)^{\pi} = \cos\pi^2 + i\sin\pi^2$$

$$(-1)^{\pi} \approx -0.9027 - (0.4303)i$$

Anonymous Functions in MATLAB

$$[-1 \quad 0 \quad 1 \quad 2]^{\pi} = ?$$

Command Window

New to MATLAB? See resources for Getting Started.

```
>> y = @(x) x.^pi

y =

    @(x)x.^pi

>> y([-1 0 1 2])

ans =

  -0.9027 - 0.4303i   0.0000 + 0.0000i   1.0000 + 0.0000i   8.8250 + 0.0000i

>> Y = y([-1 0 1 2]);
>> Y(1)

ans =

  -0.9027 - 0.4303i
```

1. Calculation with MATLAB

MATLAB: CALCULATOR

TRY to calculate in your memory:

$$10^2 + 11^2 + 12^2 = ?$$

Check it against MATLAB:

```
>> 10^2 + 11^2 + 12^2
```

Alternatively, try in MATLAB:

```
>> sum([10:12].^2)
```

ans =

365

year ?

MATLAB: CALCULATOR

What about this?

$$10^2 + 11^2 + 12^2 + 13^2 + 14^2 = ?$$

Check it against MATLAB:

```
>> 10^2 + 11^2 + 12^2 + 13^2 + 14^2
```

Alternatively, try in MATLAB:

```
>> sum([10:14].^2)
```

```
ans =

   730
```

730 = 365 x 2

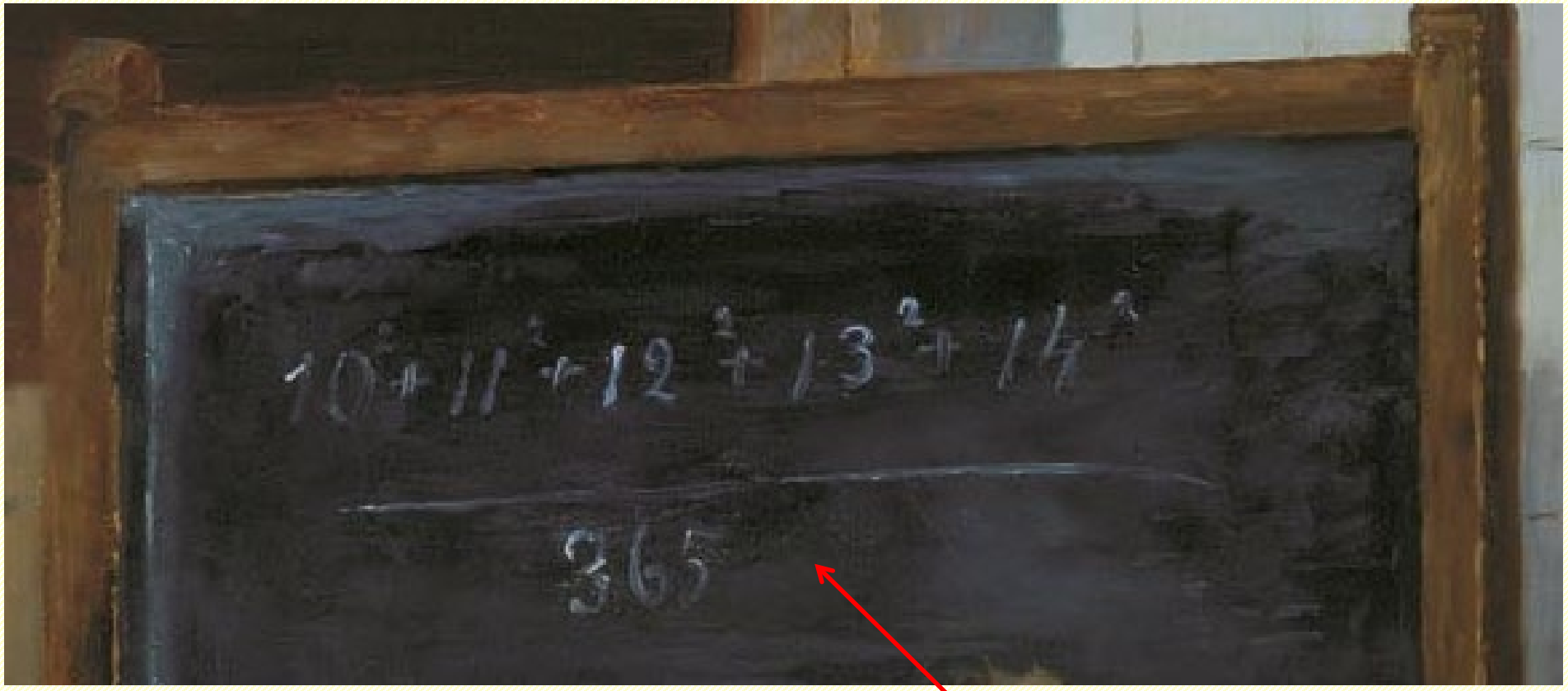
2 years ?

Prompted by this old Painting of the Country School, 1895

Painting by
Н. П. Богданов-Бельский
(1868—1945):
“Устный счёт.
В народной школе
С. А. Рачинского”,
1895

Courtesy: Wikipedia

https://ru.wikipedia.org/wiki/Устный_счёт._В_народной_школе_С._А._Рачинского



$$\frac{10^2 + 11^2 + 12^2 + 13^2 + 14^2}{365}$$

DISCOVERIES by P.M.TRIVAILO:

This is just INCREDIBLE !!!

Command Window

```
 1^2 + ... +  72^2 =  127020 = 365 x 348
 3^2 + ... +  45^2 =   31390 = 365 x 86
 3^2 + ... +  75^2 =  143445 = 365 x 393
 5^2 + ... +  77^2 =  155125 = 365 x 425
 6^2 + ... +  37^2 =   17520 = 365 x 48
 8^2 + ... +  80^2 =  173740 = 365 x 476
10^2 + ... +  12^2 =     365 = 365 x 1
10^2 + ... +  14^2 =     730 = 365 x 2
10^2 + ... +  82^2 =  186880 = 365 x 512
10^2 + ... +  85^2 =  208050 = 365 x 570
10^2 + ... +  87^2 =  223015 = 365 x 611
13^2 + ... +  14^2 =     365 = 365 x 1
13^2 + ... +  82^2 =  186515 = 365 x 511
13^2 + ... +  85^2 =  207685 = 365 x 569
13^2 + ... +  87^2 =  222650 = 365 x 610
15^2 + ... +  82^2 =  186150 = 365 x 510
15^2 + ... +  85^2 =  207320 = 365 x 568
15^2 + ... +  87^2 =  222285 = 365 x 609
16^2 + ... +  22^2 =    2555 = 365 x 7
16^2 + ... +  95^2 =  289080 = 365 x 792
```

Command Window

```
18^2 + ... +  90^2 =  245280 = 365 x 672
20^2 + ... +  92^2 =  261340 = 365 x 716
21^2 + ... +  47^2 =   32850 = 365 x 90
23^2 + ... +  95^2 =  286525 = 365 x 785
25^2 + ... +  97^2 =  304045 = 365 x 833
26^2 + ... +  52^2 =   42705 = 365 x 117
28^2 + ... +  70^2 =  109865 = 365 x 301
28^2 + ... +  84^2 =  194180 = 365 x 532
28^2 + ... + 100^2 =  331420 = 365 x 908
36^2 + ... +  67^2 =   87600 = 365 x 240
36^2 + ... +  79^2 =  152570 = 365 x 418
46^2 + ... +  75^2 =  112055 = 365 x 307
51^2 + ... +  57^2 =   20440 = 365 x 56
51^2 + ... +  74^2 =   94900 = 365 x 260
58^2 + ... +  74^2 =   74460 = 365 x 204
59^2 + ... +  63^2 =   18615 = 365 x 51
68^2 + ... +  79^2 =   64970 = 365 x 178
71^2 + ... +  84^2 =   84315 = 365 x 231
71^2 + ... + 100^2 =  221555 = 365 x 607
83^2 + ... +  85^2 =   21170 = 365 x 58
83^2 + ... +  87^2 =   36135 = 365 x 99
85^2 + ... + 100^2 =  137240 = 365 x 376
86^2 + ... +  87^2 =   14965 = 365 x 41
```

DISCOVERIES by P.M.TRIVAILO (extract):

$$\frac{16^2+17^2+18^2+19^2+20^2+21^2+22^2}{365}=7$$

I am really surprised !!!

year ?

week?

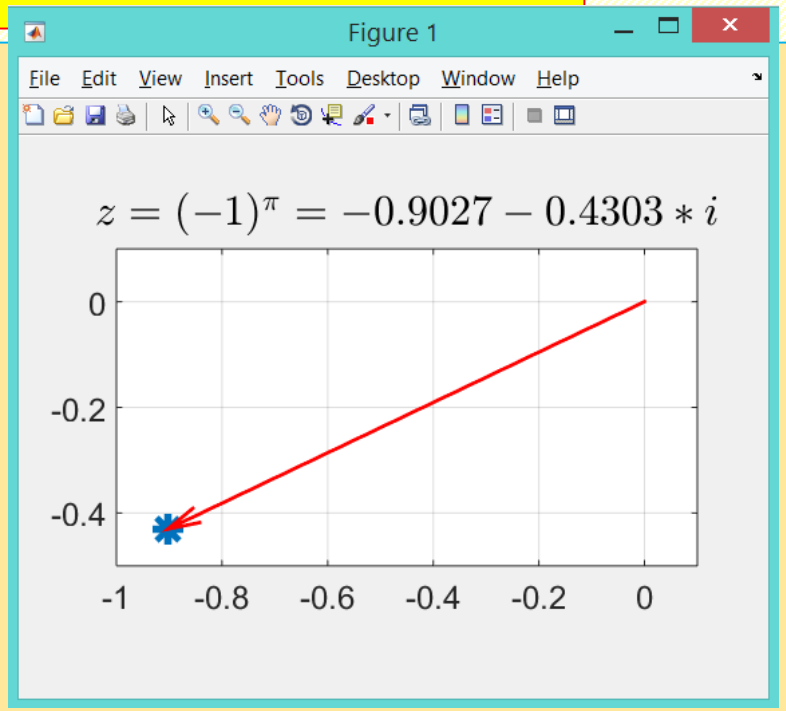
REVIEW:

PLOTTING WITH MATLAB

Plotting Complex Numbers in MATLAB

$$z = -0.9027 + (-0.4303)i$$

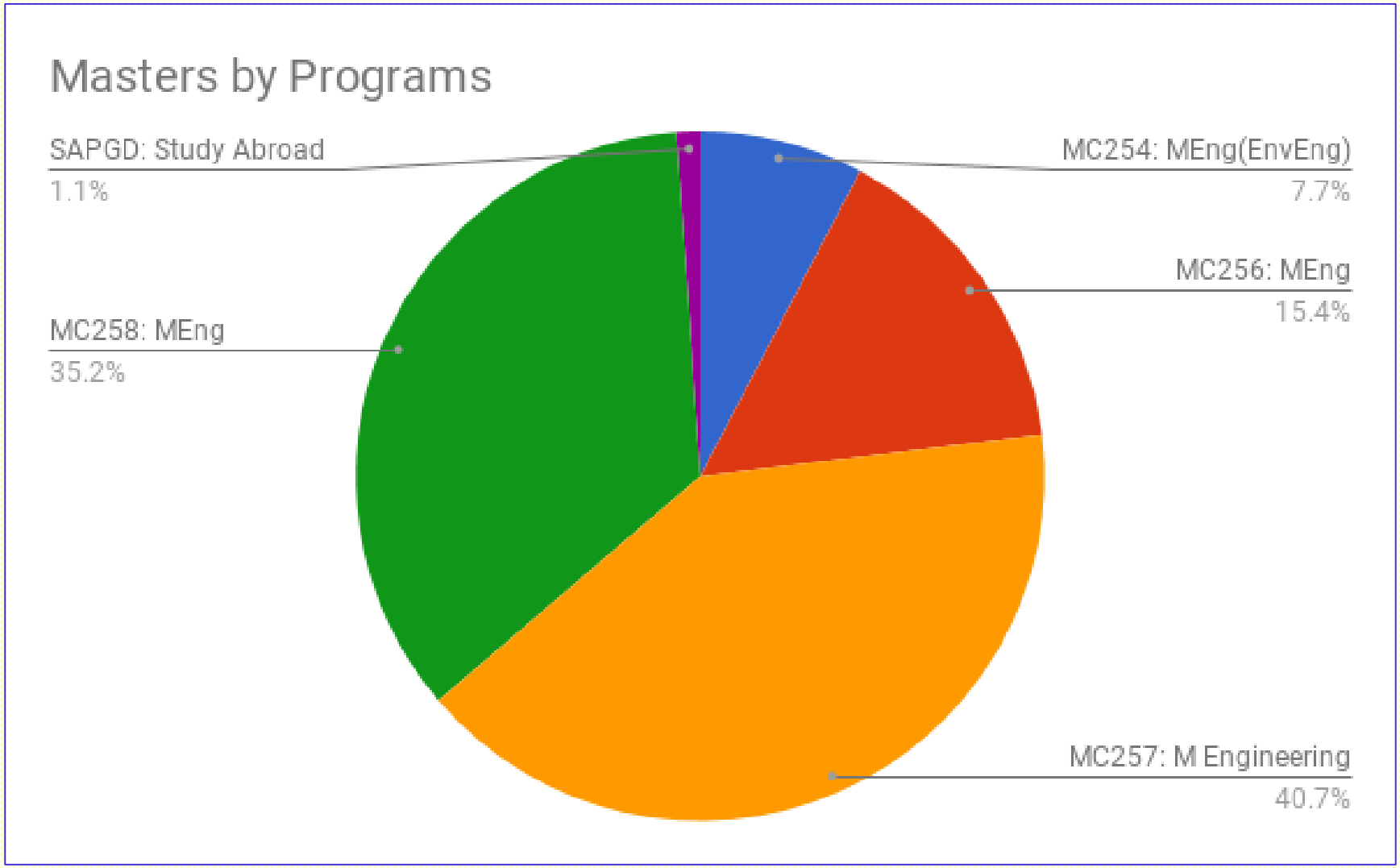
```
%%
% Designed by Prof P.M.Trivailo (C)2019
% -------------------------
clc; clear; close('all');
z=(-1)^pi; %z=-0.9027 - 0.4303*i;

figure;
plot(z,'*','MarkerSize',16, ...
       'LineWidth',4); hold on;
quiver(0,0,real(z),imag(z),0,...
    'Color','r','LineWidth',2);

title('$z=(-1)^\pi=-0.9027 - 0.4303*i$',...
      'Interpreter','LaTeX','FontSize',24);
axis equal; axis([-1 0.1 -0.5 0.1]); grid on;
set(gca,'FontSize',18);
```



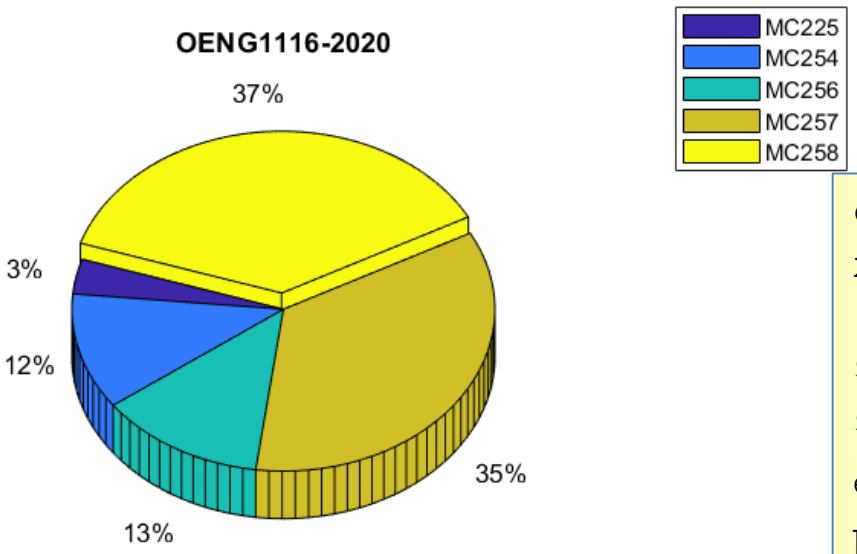
OENG1116
2020 ENROLMENT NUMBERS
BY PROGRAMS

TimeStamp of STAT	SEMESTER	PROGRAM CODE	PROGRAM NAME	OENG1116	TOTAL
05/03/2020	1810	MC225	MEng (Aerospace)	6	**6**
05/03/2020	1810	MC254	MEng(EnvEng)	20	**20**
05/03/2020	1810	MC256	MEng(RoboMechEng)	22	**22**
05/03/2020	1810	MC257	M Engineering (Civil Eng)	60	**60**
05/03/2020	1810	MC258	MEng (Mechanical Engineering)	64	**64**
GRAND TOTAL		**5**		**172**	**172**

Distribution Plot by GOOGLE



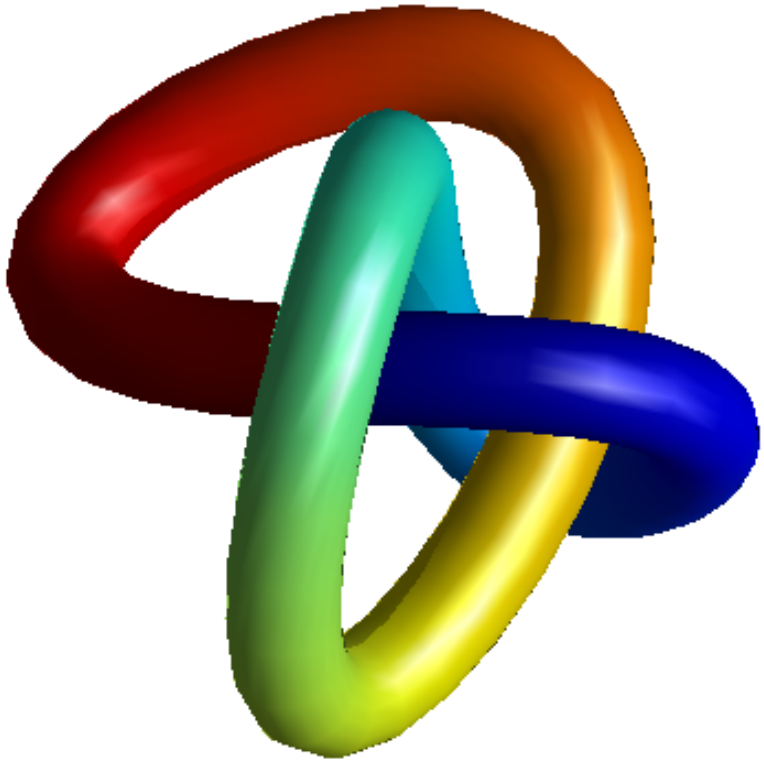
PIE PLOTS: Class Exercises (Cont'd)




```
clc; close('all'); figure;
x = [6 20 22 60 64];

% Create a 3D pie chart
% using the pie3 function
explode = [0 0 0 0 1 ];
pie3(x, explode)
view([-72 50])
rotate3d on

% Add a title & legend
title('OENG1116-2020')

legend('MC225','MC254','MC256',...
       'MC257','MC258')
```

Plot the Surface

```
surf(x1,x2,x3,color);
shading interp;
light
lighting gouraud % 'lighting phong' will use zbuffer, slower
view(2)
axis equal off
axis vis3d % for smooth rotate3d
```



PYTHAGOREAN TRIPLES:

PMT SOLUTION

PIERRE de FERMAT

Courtesy: Wikipedia https://en.wikipedia.org/wiki/Pierre_de_Fermat

Pierre de Fermat (between 31 October and 6 December 1607 – 12 January 1665) was a French lawyer at the Parlement of Toulouse, France, and a **mathematician** who is given credit for early developments that led to infinitesimal calculus, including his technique of adequality. In particular, he is recognized for his discovery of an original method of finding the greatest and the smallest ordinates of curved lines, which is analogous to that of differential calculus, then unknown, and his research into number theory. He made notable contributions to analytic geometry, probability, and optics. He is best known for his Fermat's principle for light propagation and his **Fermat's Last Theorem** in number theory, which he described in a note at the margin of a copy of Diophantus' Arithmetica.

Fermat's Last Theorem

Courtesy: Wikipedia https://en.wikipedia.org/wiki/Fermat%27s_Last_Theorem

In number theory, **Fermat's Last Theorem** (sometimes called Fermat's conjecture, especially in older texts) states that no three positive integers *a*, *b*, and *c* satisfy the equation

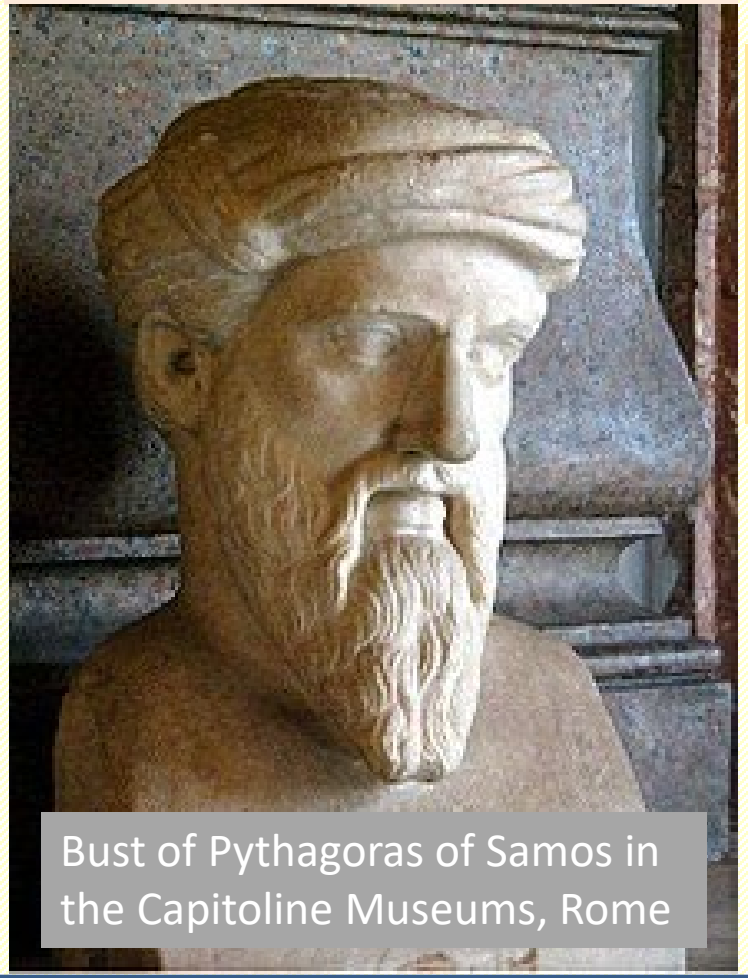
$$a^n + b^n = c^n$$

for any integer value of n greater than 2.

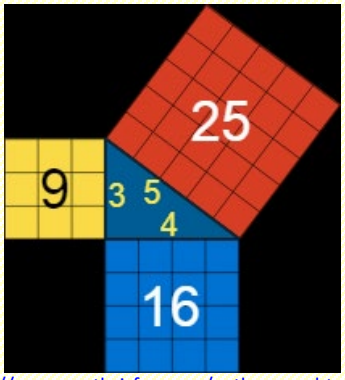
The cases n = 1 and n = 2 have been known to have infinitely many solutions since antiquity.

PYTHAGORAS of SAMOS

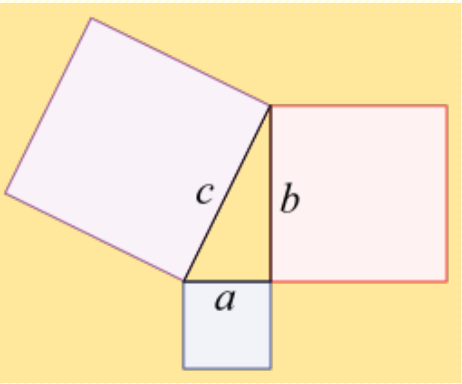
Courtesy: Wikipedia
https://en.wikipedia.org/wiki/Pythagoras



Bust of Pythagoras of Samos in the Capitoline Museums, Rome



https://www.mathsisfun.com/pythagoras.html



$$a^2 + b^2 = c^2$$

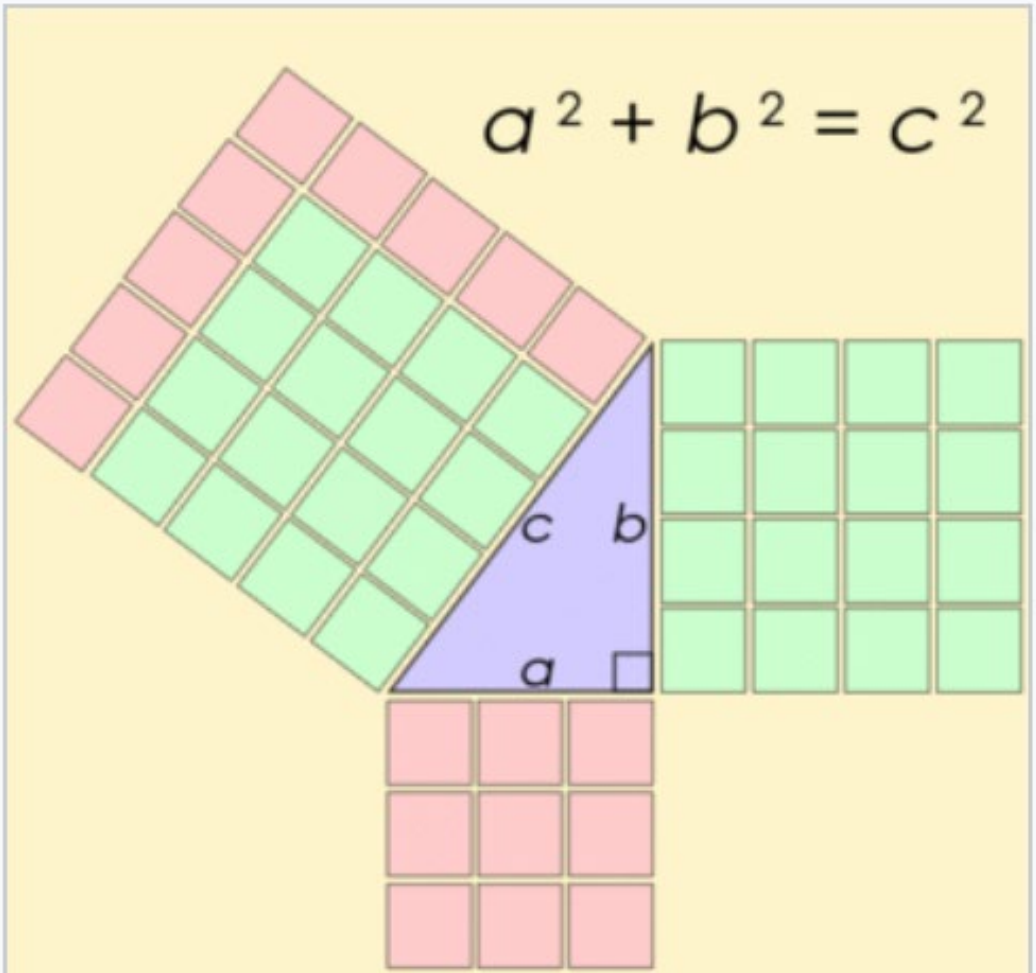
Pythagorean theorem
The sum of the areas of the two squares on the legs (*a* and *b*) equals the area of the square on the hypotenuse (*c*).

Pythagoras of Samos (c570–495 BC) was an Ionian Greek philosopher. He exerted a profound impact on the philosophies of Plato, Aristotle, and, through them, Western philosophy. In antiquity, Pythagoras was credited with many mathematical and scientific discoveries, including the **Pythagorean theorem**, Pythagorean tuning, the five regular solids, the Theory of Proportions, the sphericity of the Earth, and the identity of the morning and evening stars as the planet Venus.

PYTHAGOREAN TRIPLE

A **Pythagorean triple** consists of three positive integers a, b, and c, such that $a^2 + b^2 = c^2$.
Such a triple is commonly written (a, b, c), and a well-known example is (3, 4, 5). If (a, b, c) is a Pythagorean triple, then so is (ka, kb, kc) for any positive integer k.
A **primitive Pythagorean triple** is one in which a, b and c are coprime (that is, they have no common divisor larger than 1).
A triangle whose sides form a Pythagorean triple is called a **Pythagorean triangle**, and is necessarily a right triangle.

PYTHAGOREAN TRIPLE (Cont'd)



Courtesy: https://en.wikipedia.org/wiki/Pythagorean_triple

Animation demonstrating the simplest Pythagorean triple, $3^2 + 4^2 = 5^2$.

Refer to animation demo on Wikipedia

DISCOVERIES by P.M.TRIVAILO (extract):

Script

Solutions for selected range

```
% Designed by Prof P.M.Trivailo (C)2019
% Demo on Fermat Theorem
clc; str='a=%3i  b=%3i  c=%3i';
for a=300:400
   for b=a:400
      c=sqrt(a^2+b^2);
      if rem(c,1)==0,
         disp(sprintf(str,a,b,c))
      end
   end
end

commandwindow
```

```
%-------------------------
% a=300  b=315  c=435
% a=300  b=400  c=500
% a=319  b=360  c=481
% a=320  b=336  c=464
% a=325  b=360  c=485
% a=336  b=377  c=505
% a=336  b=385  c=511
% a=340  b=357  c=493
% a=357  b=360  c=507
% a=360  b=378  c=522
% a=380  b=399  c=551
%-------------------------
% [380^2+399^2 551^2]
% ans =
%       303601       303601
```

check

DISCOVERIES by P.M.TRIVAILO (extract):

Script can be modified to mark solutions with prime “c” values

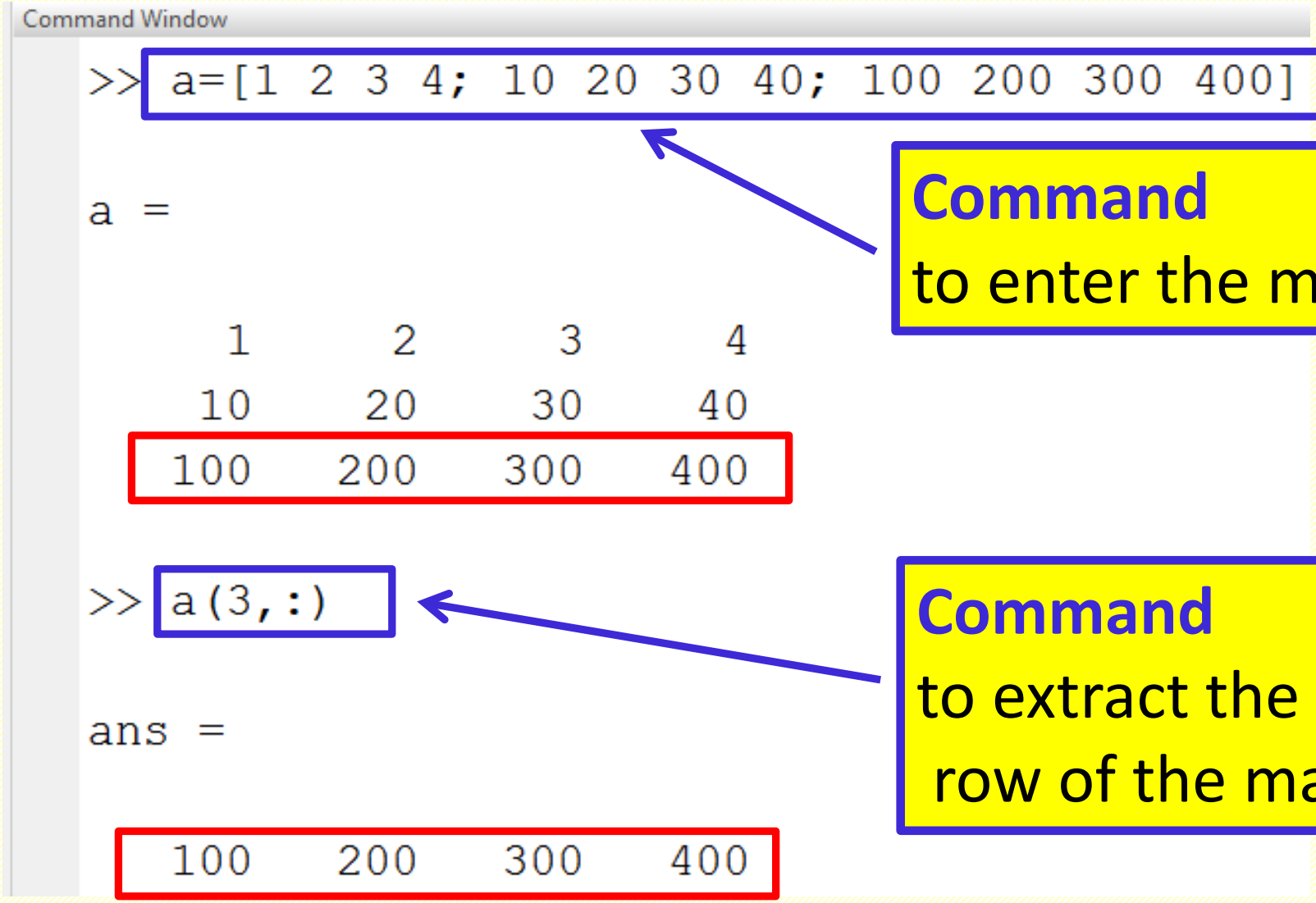
```
a=360   b=378   c=522
a=368   b=465   c=593! (c-prime number)
a=380   b=399   c=551
a=384   b=440   c=584
a=390   b=432   c=582
```

```
a=15015   b=16048   c=21977! (c-prime number)
a=15169   b=16440   c=22369! (c-prime number)
a=15405   b=15892   c=22133! (c-prime number)
a=15721   b=16680   c=22921! (c-prime number)
a=15792   b=16745   c=23017! (c-prime number)
a=16119   b=16520   c=23081! (c-prime number)
```

MATRICES

IN MATLAB

ENTER THE MATRIX & EXTRACT ROW:



Command
to enter the matrix

Command
to extract the 3rd row of the matrix

SYMBOLIC OPERATIONS,

USING MATLAB

MATLAB: SYMBOLIC TOOLBOX

To declare variables x and y as symbolic objects use the syms command:

Command Window

```
>> syms x y
>> x+x+y

ans =

2*x + y

>> simplify(x+x+y+(x+y)*(x-y))

ans =

x^2 + 2*x - y^2 + y
```

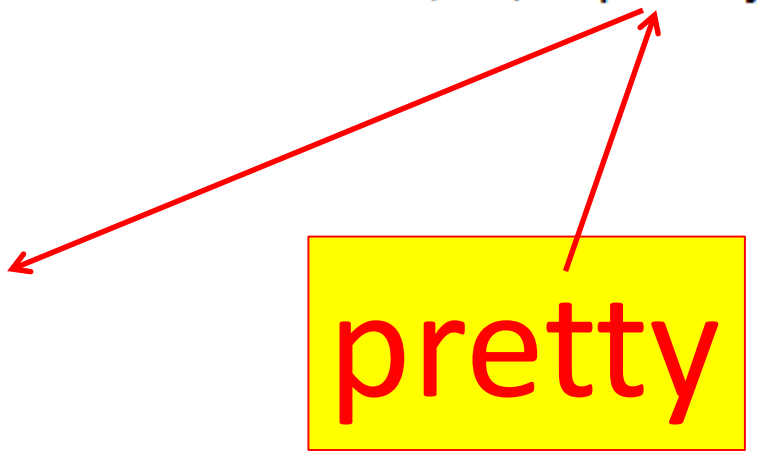
MATLAB: SYMBOLIC TOOLBOX

To declare variables x and y as symbolic objects use the syms command:

Command Window

New to MATLAB? See resources for Getting Started.

```
>> syms x a b c; s = solve(a*x^2 + b*x + c, x); pretty(s);
/                2                \
|    b + sqrt(b  - 4 a c)         |
| - --------------------          |
|             2 a                 |
|                                 |
|                2                |
|    b - sqrt(b  - 4 a c)         |
| - --------------------          |
\             2 a                 /
```



Symbolic Summation (Example-1)

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \frac{1}{6^2} + \frac{1}{7^2} + \dots$$

```
>> clear all, close('all'); clc;
>> syms k;
>> symsum(1/k^2,k,1,inf)
```

The Basel Problem

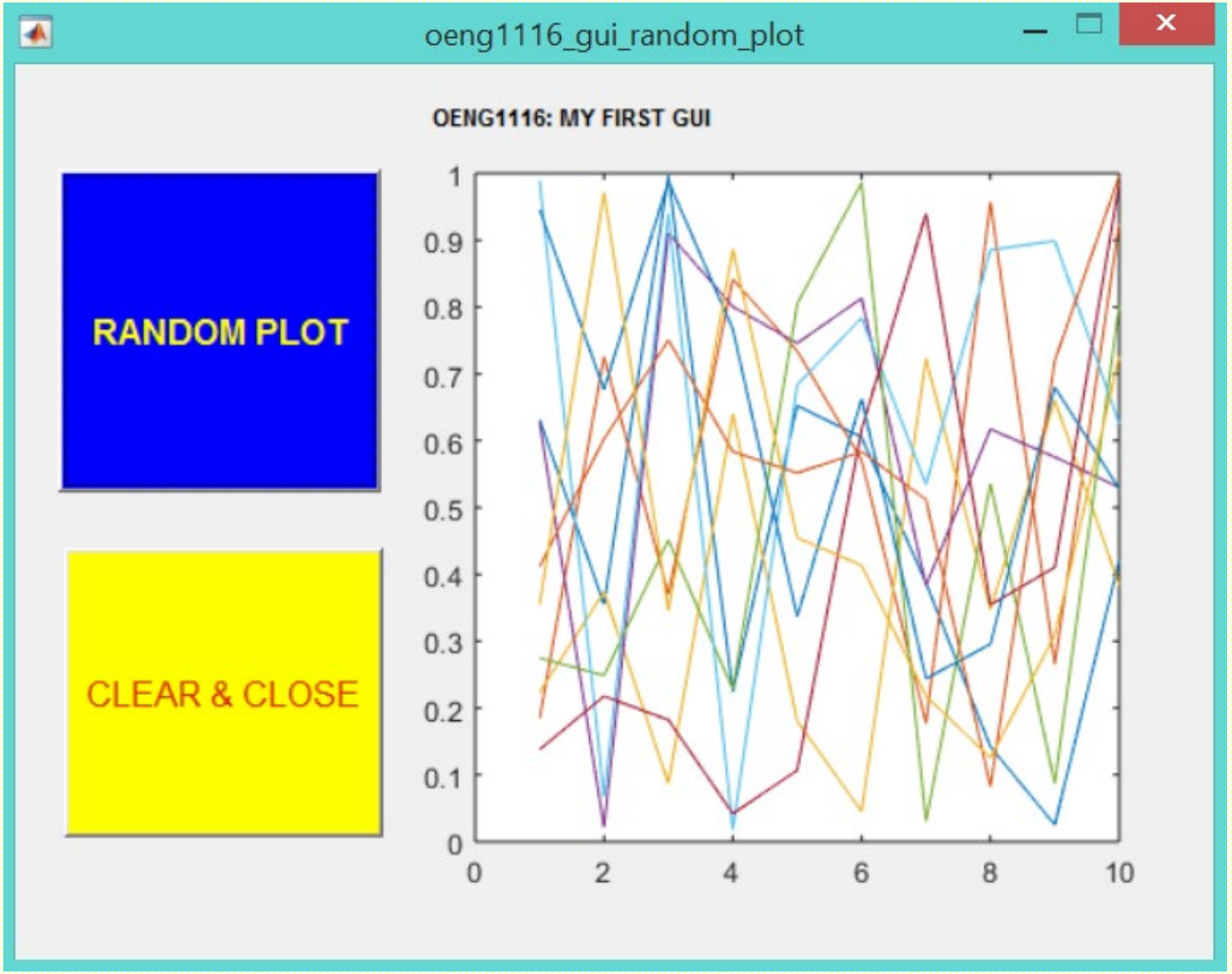
```
ans =

pi^2/6
```

ANSWER

MATLAB: GUIs

GUIs

EXAMPLE-1: Testing GUI



EXAMPLES OF ANIMATIONS,

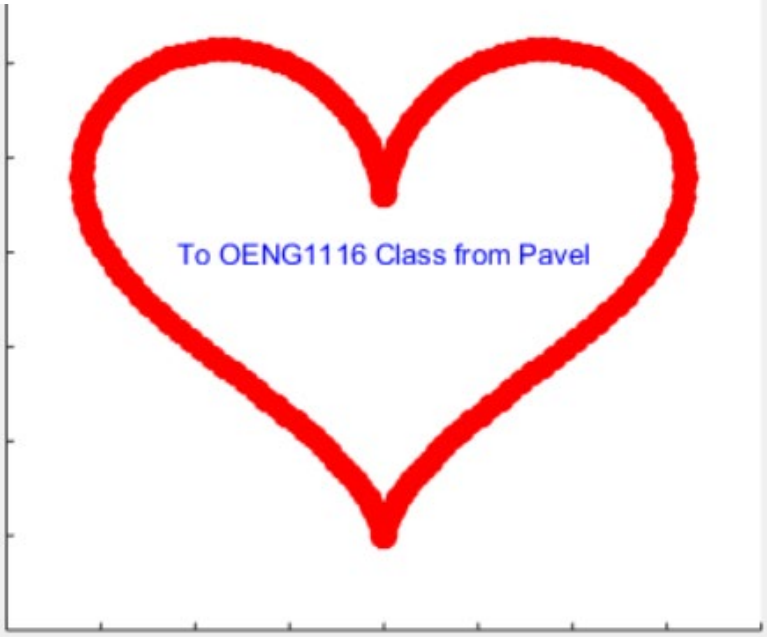
USING MATLAB

Introduction into MATLAB Animations

Command Window

New to MATLAB? See resources for Getting Started.

```
===== Prof Trivailo's Heart Equations: =====
x=16*sin(t).^3;
y=11*cos(t)-5*cos(2*t)-2*cos(3*t)-cos(4*t);
```

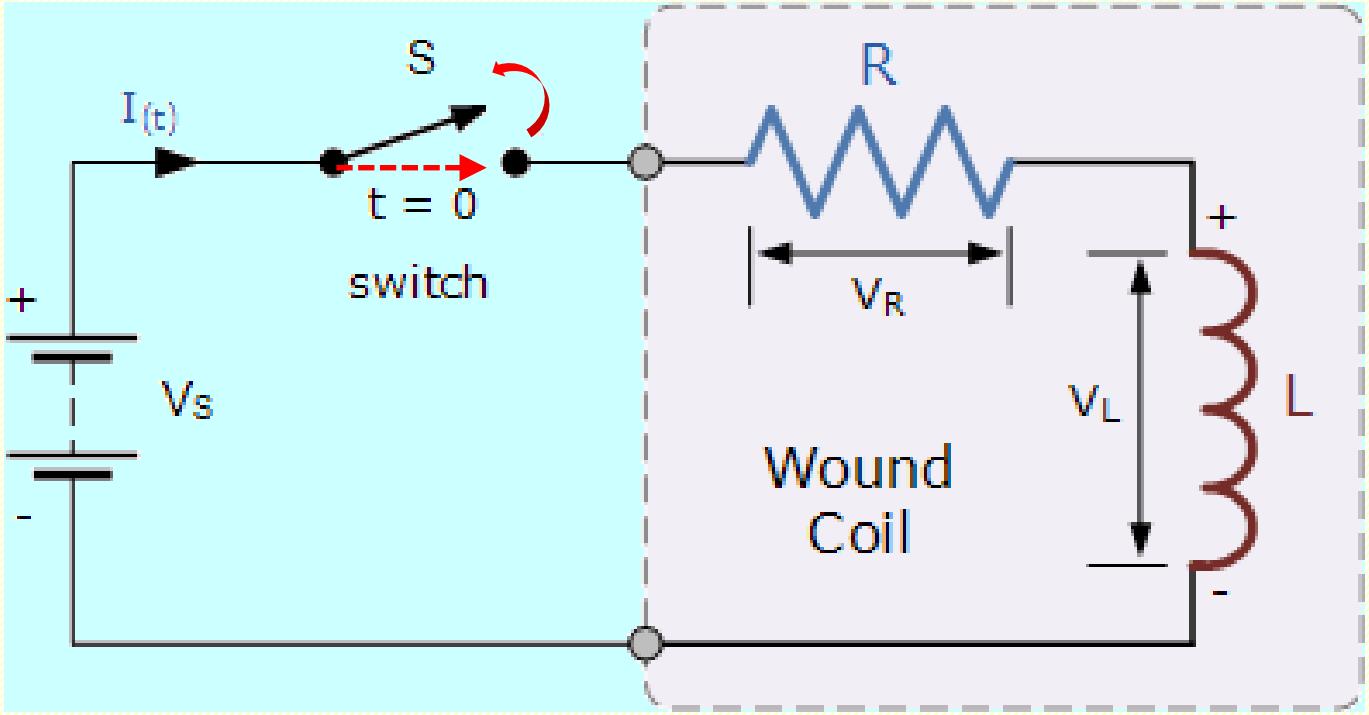


FIRST ORDER

DIFFERENTIAL EQUATIONS:

SERIES RESISTANCE-INDUCTANCE ELECTRICAL CIRCUIT EXAMPLE-1

SERIES RESISTANCE-INDUCTANCE DC ELECTRICAL CIRCUIT



When the electromotive force is removed from a circuit containing **inductance** and **resistance** but no capacitors, **the rate of decrease of current is proportional to the current.**

If the initial current is **30 Amps** but decays to **11 Amps** after 0.01 seconds,

find an expression for the current as a function of time.

Image (adapted by PMT) Courtesy : https://www.electronics-tutorials.ws/inductor/lr-circuits.html

When the electromotive force (emf) is removed from a circuit containing inductance and resistance but no capacitors, the rate of decrease of current is proportional to the current.

If the initial current is **30 Amps** but decays to **11 Amps** after 0.01 seconds,

find an expression for the current as a function of time.

Courtesy: http://www-personal.umich.edu/~wheeles/116Fall2012/diffmodextracredit.pdf

SOLUTION: Modelling

If $I(t)$ denotes current and t denotes time, the statement *"the rate of decrease of current is proportional to the current"* translates to:

$$\frac{dI}{dt} = -k\,I$$

where k denotes the constant of proportionality.

SOLUTION: Analytical Method

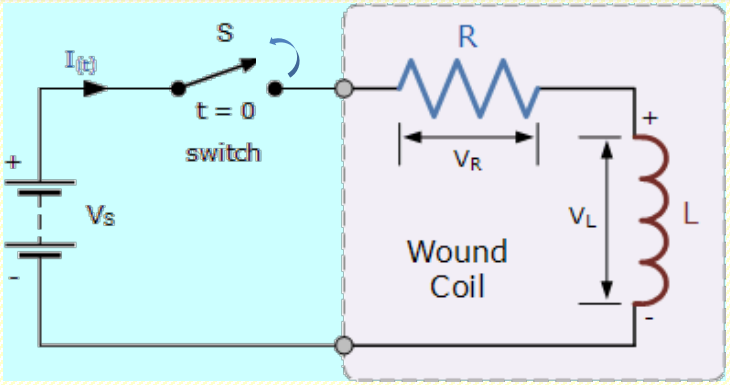


Image (adapted by PMT) Courtesy : https://www.electronics-tutorials.ws/inductor/lr-circuits.html

The differential equation is separable, this allows analytical solutions:

$$\frac{dI}{dt} = -k\,I; \qquad \frac{dI}{I} = -k\,dt; \qquad \int \frac{dI}{I} = -k \int dt$$

$$\ln|I| = -k\,t + \text{Constant}_1$$

$$I = (\text{Constant}_2) \times e^{-kt}; \quad (\text{the } e^{\text{Constant1}} \text{ and the } \pm \text{ all gets absorbed in the } \text{Constant}_2)$$

Initial zero balance enables us to find the value of the **Constant$_2$**:

$$I\big|_{t=0} = 30; \quad \Rightarrow \quad 30 = (\text{Constant}_2) \times e^{0}; \quad \Rightarrow \quad \text{Constant}_2 = 30$$

Analytical exact solution: $I(t) = 30 \times e^{-kt}$

Another condition in the task, saying that "if the current decays to 11 amps after 0.01 seconds", enables us to calculate value of k:

$$I\big|_{t=0.01} = 11; \quad \Rightarrow$$

$$11 = 30 \times e^{-0.01k}; \quad \Rightarrow$$

$$e^{-0.01k} = \frac{11}{30}; \quad \Rightarrow$$

$$-0.01k = \ln\left(\frac{11}{30}\right);$$

$$k = -100 \times \ln\left(\frac{11}{30}\right) = 100.3302$$

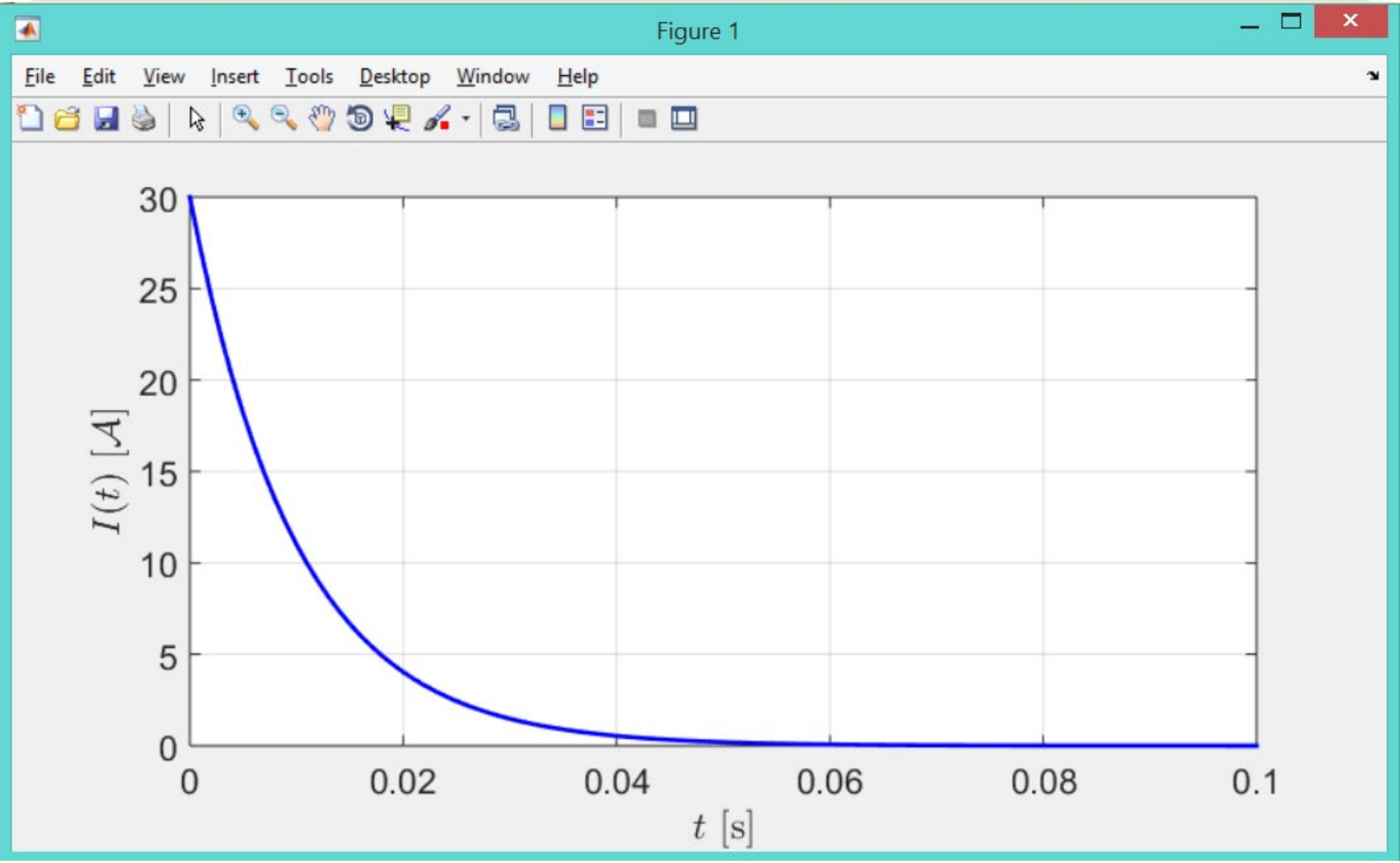
Analytical exact solution for particular circuit:

$$I(t) = 30 \times e^{-100.3302\,t}$$

MATLAB script, plotting analytical solution (for simulated electrical process)

```
%%
% Designed by Prof P.M.Trivailo (C)2019
% Series Resistance-inductance Electrical Circuit
clc; clear; close('all');
t=[0:0.01:1]*0.1;
k=-100*log(11/30);
I=30*exp(-k*t);
figure; plot(t,I,'b','LineWidth',2);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$I(t)$ [$\cal{A}$]','Interpreter','LaTeX');
set(gcf,'Position',[214   334   834   428]);
set(gca,'FontSize',16);
```

Output of the MATLAB script



!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH THE SAME EXAMPLE

!!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH THE SAME EXAMPLE

!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH THE SAME EXAMPLE

SOLUTION: Numerical (!!!) Method

1. Input of the data

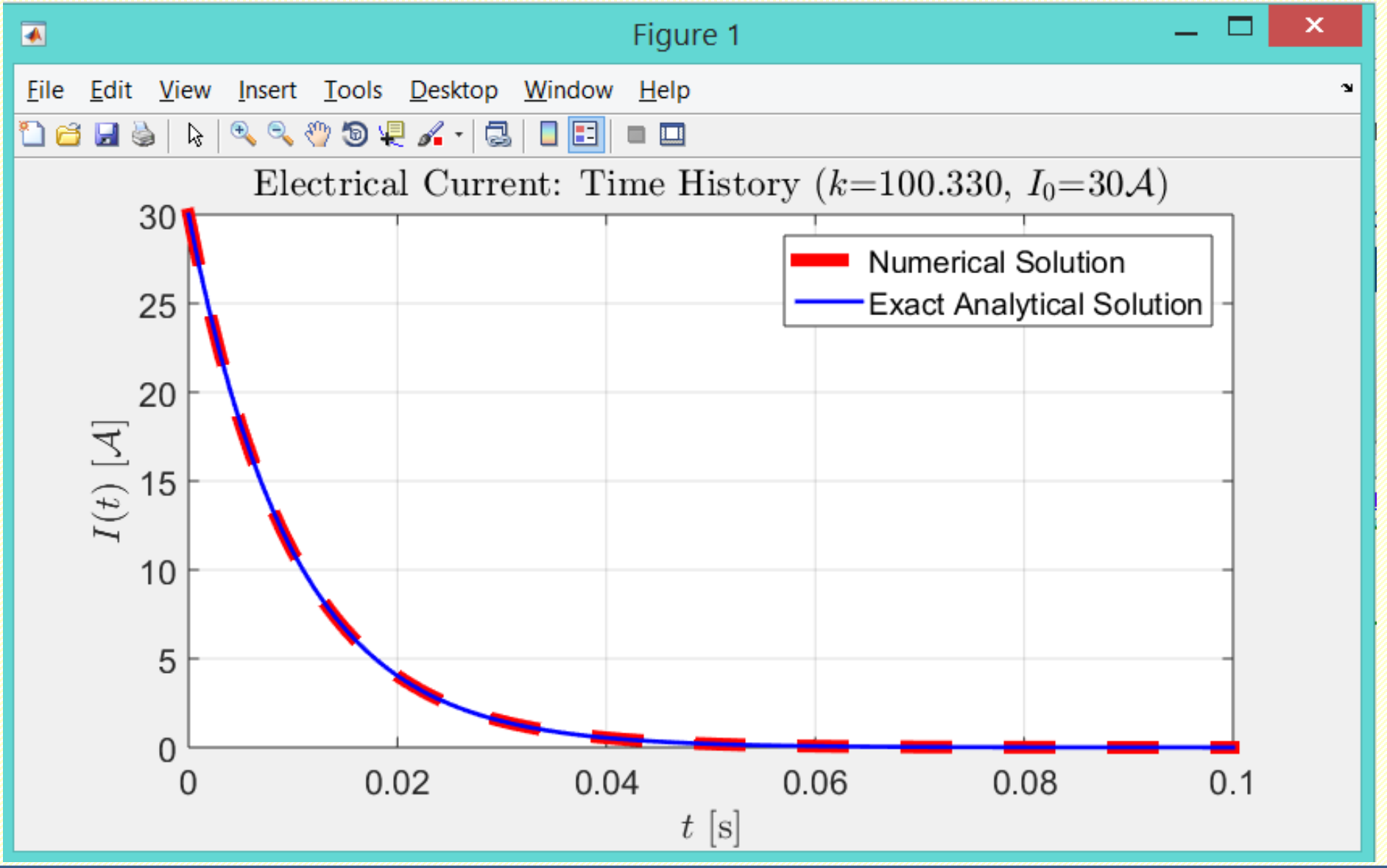
2. $$\frac{dI}{dt} = -k\,I$$

```
%% RESISTANCE-INDUCTANCE ELECTRIC
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
% Useful Ref: https://www.electronics-tutorials.ws/accircuits/ac-ind
% Fundamental Law:  dI/dt = -kI;          % here k-positive
%-------------------------------------%
I0 = 30;                                % initial current
k = -100*log(11/30);                    % here k-positive, because log(11/30)<0
tmax = 0.1;                             % s
t = [0:0.01:1]*tmax;                    % regular time array
I_xdot_anonymous = @(t, I) -k*I;  % anonymous FUNCTION IS USED
[tt,TT] = ode45(I_xdot_anonymous,t,I0);  % calling ODE
plot(tt,TT,'LineWidth',6,'Color','r','LineStyle','--');
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX'); ylabel('$I(t)$ [$\cal A$]','Interpreter','LaTeX');
str = sprintf('Electrical Current: Time History ($$k$$=%4.3f, $$I_0$$=%g$$\\cal A$$)',k,I0)
title(str, 'Interpreter','LaTeX');
set(gca,'FontSize',16); set(gcf,'Position',[214   334   834   428]);
Iexact = 30*exp(-k*t);
hold on; plot(tt,Iexact,'LineWidth',2,'Color','b');
legend('Numerical Solution','Exact Analytical Solution');
```

4. Plotting results

3. Calling ode45 procedure !

Output of the MATLAB script



FIRST ORDER

DIFFERENTIAL EQUATIONS:

SERIES RESISTANCE-INDUCTANCE ELECTRICAL CIRCUIT EXAMPLE-2

SERIES RESISTANCE-INDUCTANCE ELECTRICAL CIRCUIT

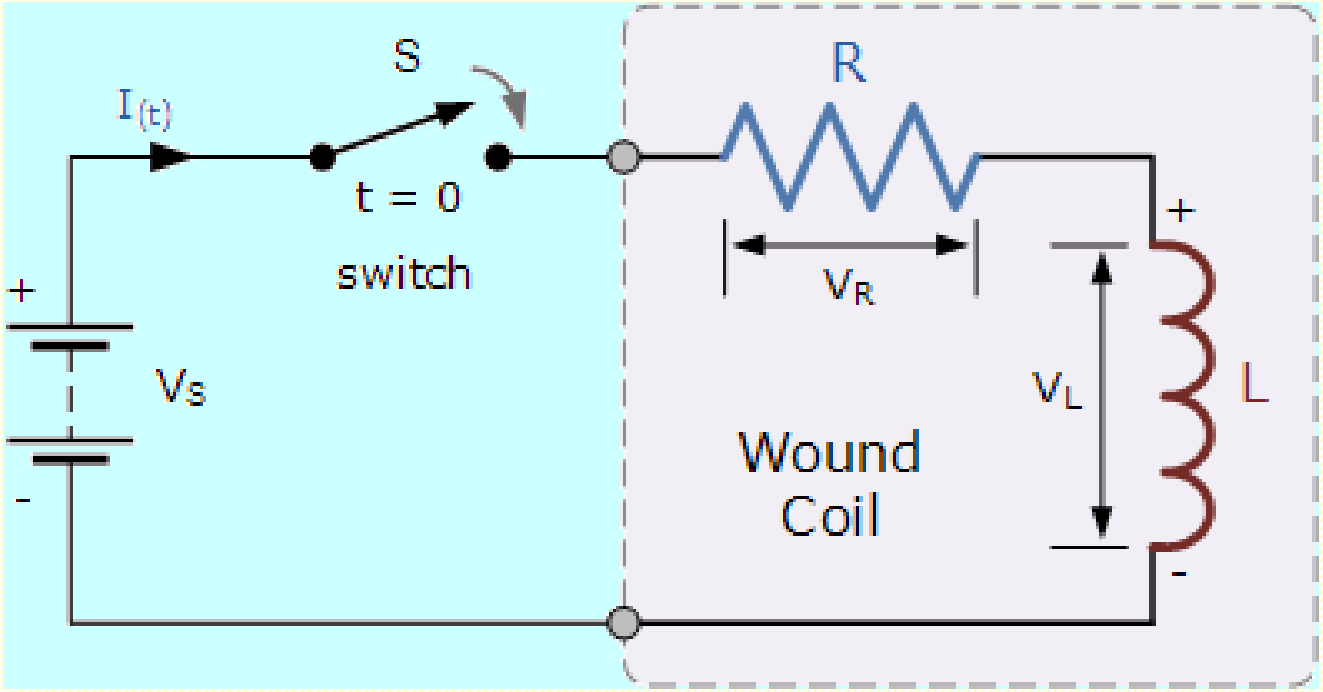


Image Courtesy :
https://www.electronics-tutorials.ws/inductor/lr-circuits.html

The *LR series circuit* is connected across a constant voltage source (the battery) and a switch.

Assume that the switch, *S* is open until it is closed at a time $t = 0$, and then remains permanently closed producing a "step response" type voltage input.

Find an expression for the current I as a function of time t.

The *LR series circuit* (shown in previous slide) is connected across a constant voltage source (the battery) and a switch.

Assume that the switch, *S* is open until it is closed at a time $t = 0$, and then remains permanently closed producing a "step response" type voltage input.

Find an expression for the current I as a function of time *t*.

SOLUTION: Modelling of the System, using Ohm's Law, Lenz's Law, Kirchhoff's Voltage Law and Fradey's Law of Induction

The current, I begins to flow through the circuit but does not rise instantly to its maximum value of Imax as determined by the ratio of V / R (**Ohm's Law**).

This limiting factor is due to the presence of the self induced emf within the inductor as a result of the growth of magnetic flux, (**Lenz's Law**).

After a time the voltage source neutralizes the effect of the self induced emf, the current flow becomes constant and the induced current and field are reduced to zero.

We can use **Kirchhoff's Voltage Law**, (KVL) to define the individual voltage drops that exist around the circuit and then hopefully use it to give us an expression for the flow of current.

Kirchhoff's Voltage Law, (KVL) gives:

$$V(t) - (V_R + V_L) = 0$$

The voltage drop across the resistor, *R* is *IR* (**Ohm's Law**):

$$V_R = IR$$

The voltage drop across the inductor, *L* is as follows (due to the **Faradey's Law of induction**):

$$V_L = L\frac{dI}{dt}$$

Resultant relationship:

$$L\frac{dI}{dt} = V - IR$$

The differential equation is separable, this allows analytical solutions:

$$\frac{dI}{dt}=\frac{V-IR}{L};\qquad \frac{dI}{V-IR}=\frac{1}{L}dt$$

$$-\frac{1}{R}\int\frac{du}{u}=\frac{1}{L}\int dt\qquad (\text{by setting } u=V-IR);\quad \Rightarrow\quad -\frac{1}{R}\ln|V-IR|=\frac{t}{L}+\text{Constant}_1$$

$$\ln|V-IR|=-\frac{R}{L}t+\text{Constant}_2;\quad (\text{the } -R \text{ gets absorbed in the Constant}_2)$$

$$V-IR=(\text{Constant}_3)\times e^{-\frac{R}{L}t};\quad (\text{the } e^{\text{Constant}_2} \text{ and the } \pm \text{ all gets absorbed in the Constant}_3)$$

$$IR=-(\text{Constant}_3)\times e^{-\frac{R}{L}t}+V;\quad I=-(\text{Constant}_4)\times e^{-\frac{R}{L}t}+\frac{V}{R};\quad \left(\frac{1}{R}\text{ gets absorbed in Constant}_4\right)$$

Initial zero balance enables us to find the value of the **Constant$_4$**:

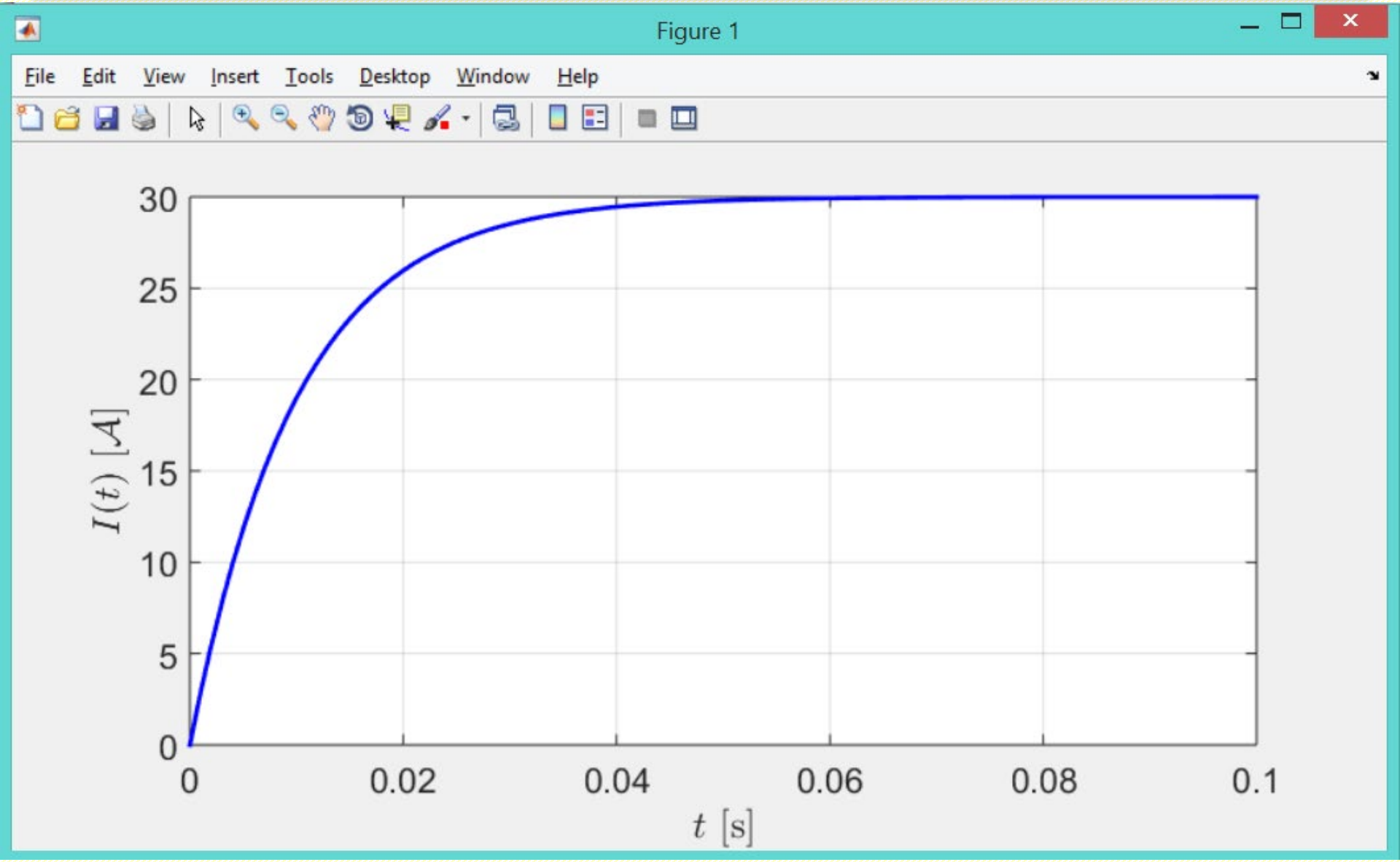
$$I\Big|_{t=0}=0;\quad \Rightarrow\quad 0=-(\text{Constant}_4)\times e^{\frac{0}{L}}+\frac{V}{R};\quad \Rightarrow\quad \text{Constant}_4=\frac{V}{R}$$

Analytical exact solution: $$I(t)=\frac{V}{R}\left(1-e^{-\frac{R}{L}t}\right)$$

MATLAB script for simulating electrical process

```
%%
% Designed by Prof P.M.Trivailo (C)2020
% Series Resistance-inductance Electric Circuit
clc; clear;
close('all');
t=[0:0.01:1]*0.1;
k=-100*log(11/30);
I=30*(1-exp(-k*t));
figure; plot(t,I,'b','LineWidth',2);
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$I(t)$ [$\cal{A}$]','Interpreter','LaTeX');
set(gcf,'Position',[214   334   834   428]);
set(gca,'FontSize',16);
```

Output of the MATLAB script



Observations:

The time required for the current flowing in the *LR* series circuit to reach its maximum steady state value is equivalent to about **5 time constants** or 5τ.

This time constant τ, is measured by $\tau = L/R$, in seconds, were *R* is the value of the resistor in Ohms and *L* is the value of the inductor in Henries. This then forms the basis of an *RL* charging circuit were 5τ can also be thought of as "5 x *L*/*R*" or the *transient time* of the circuit.

In two cases, considered above, $\tau = L/R = 1/k = 0.0100$ s, so transient time is approximately equal to 0.05 s

Courtesy: Texas Instruments, https://www.electronics-tutorials.ws/inductor/lr-circuits.html

!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH THE SAME EXAMPLE

SOLUTION: Numerical (!!!) Method

1. Input of the data

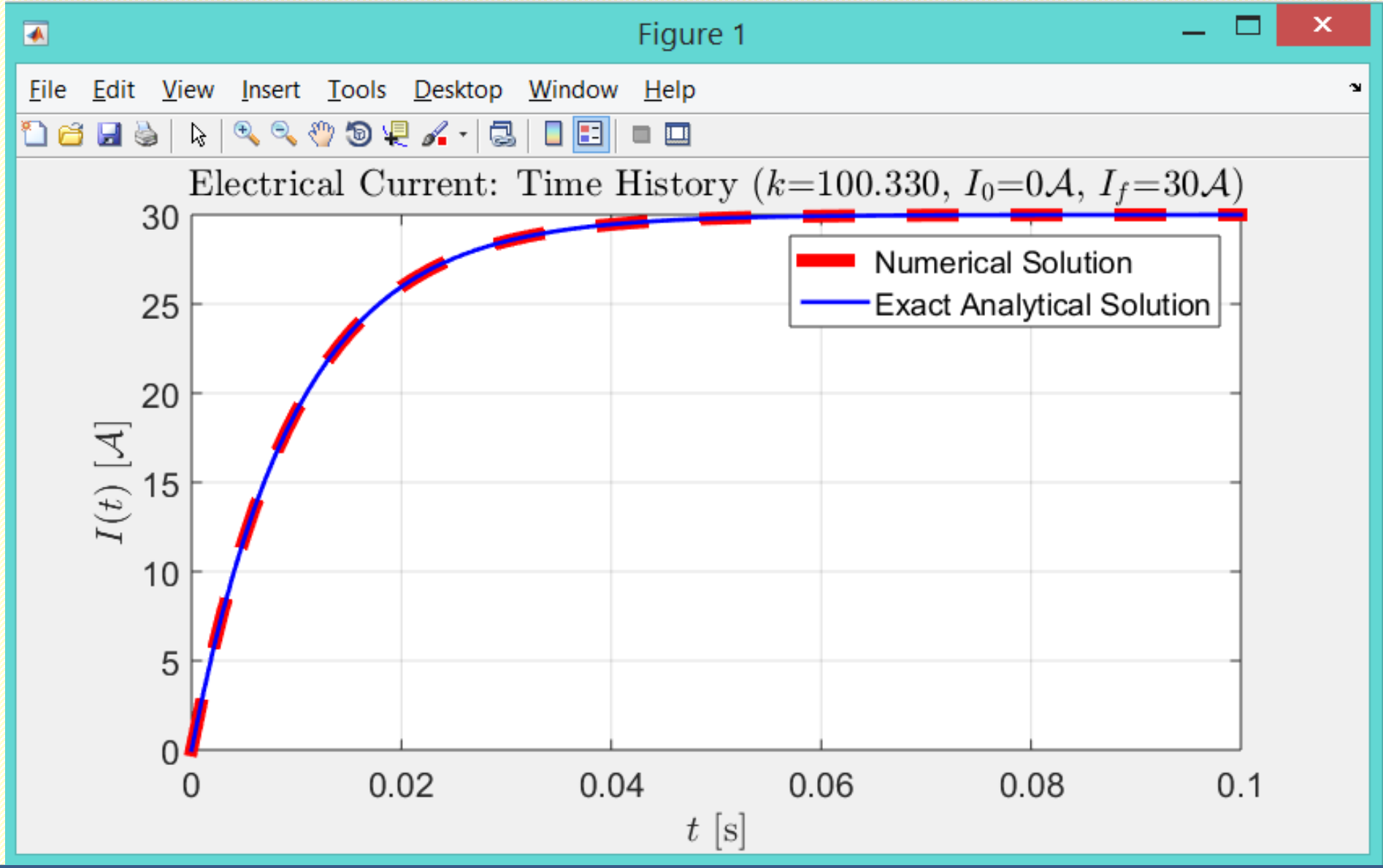
2. $$\frac{dI}{dt} = -k(I - I_f)$$

```
%% RESISTANCE-INDUCTANCE E
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
% Useful Ref: https://www.electronics-tutorials.ws/accircuits
% Fundamental Law:  dI/dt = -kI;          % here k-positive
%-------------------------------------%
I0 = 0; If=30;                          % initial and final currents
k = -100*log(11/30);                    % here k-positive, because log(11/30)<0
tmax = 0.1;                             % s
t = [0:0.01:1]*tmax;                    % regular time array
I_xdot_anonymous2 = @(t, I) -k*(I-If);  % anonymous FUNCTION IS USED
[tt2,TT2] = ode45(I_xdot_anonymous2,t,I0);% call ODE
plot(tt2,T2T,'LineWidth',6,'Color','r','LineStyle','--');
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX'); ylabel('$I(t)$ [$\cal A$]','Interpreter','LaTeX');
str = sprintf('El.Current: Time History ($$k$$=%4.3f, $$I_0$$=%g$$\\cal A$$, $$I_f$$=%g$$\\cal A$$)',k,I0,If)
title(str, 'Interpreter','LaTeX');
set(gca,'FontSize',16); set(gcf,'Position',[214   334   834   428]);
Iexact2 = 30*(1-exp(-k*t));
hold on; plot(tt2,Iexact2,'LineWidth',2,'Color','b');
legend('Numerical Solution','Exact Analytical Solution');
```

4. Plotting results

3. Calling ode45 procedure !

Output of the MATLAB script



NEWTON's LAW OF COOLING (and HEATING):

MODELLING

Newton's Law of Cooling (and Heating):

The rate at which the temperature of an object changes is proportional to the difference between the temperature of the object and the temperature of the surroundings:

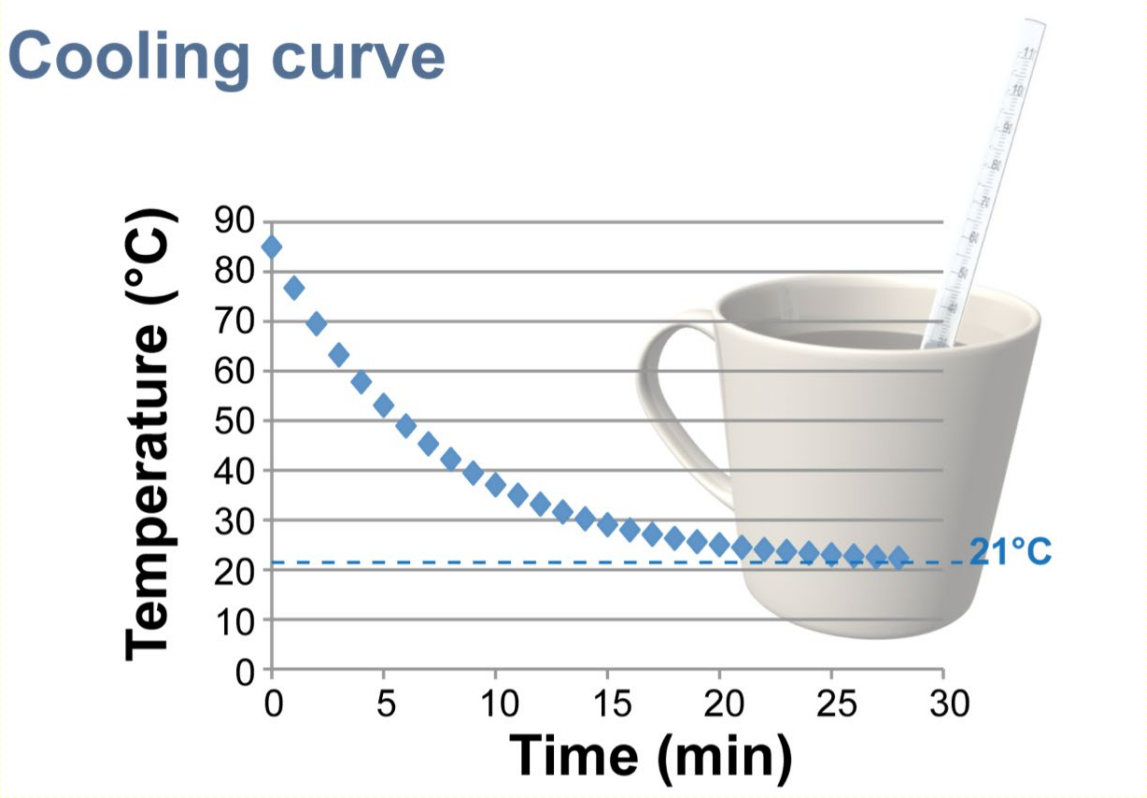
$$\frac{dT}{dt} = k(T - T_s)$$

where: t time; k - the constant of proportionality;
$T(t)$ - temperature of the object at time t;
$T_s(t)$ - temperature of the surrounding area at time t.

Courtesy: Texas Instruments, https://education.ti.com/-/media/97CF63CF732A4BB79FB2632514AD5159

Newton's Law of Cooling (and Heating): Example for Everyone to Understand

The rate of cooling depends on the temperature difference between the two objects.



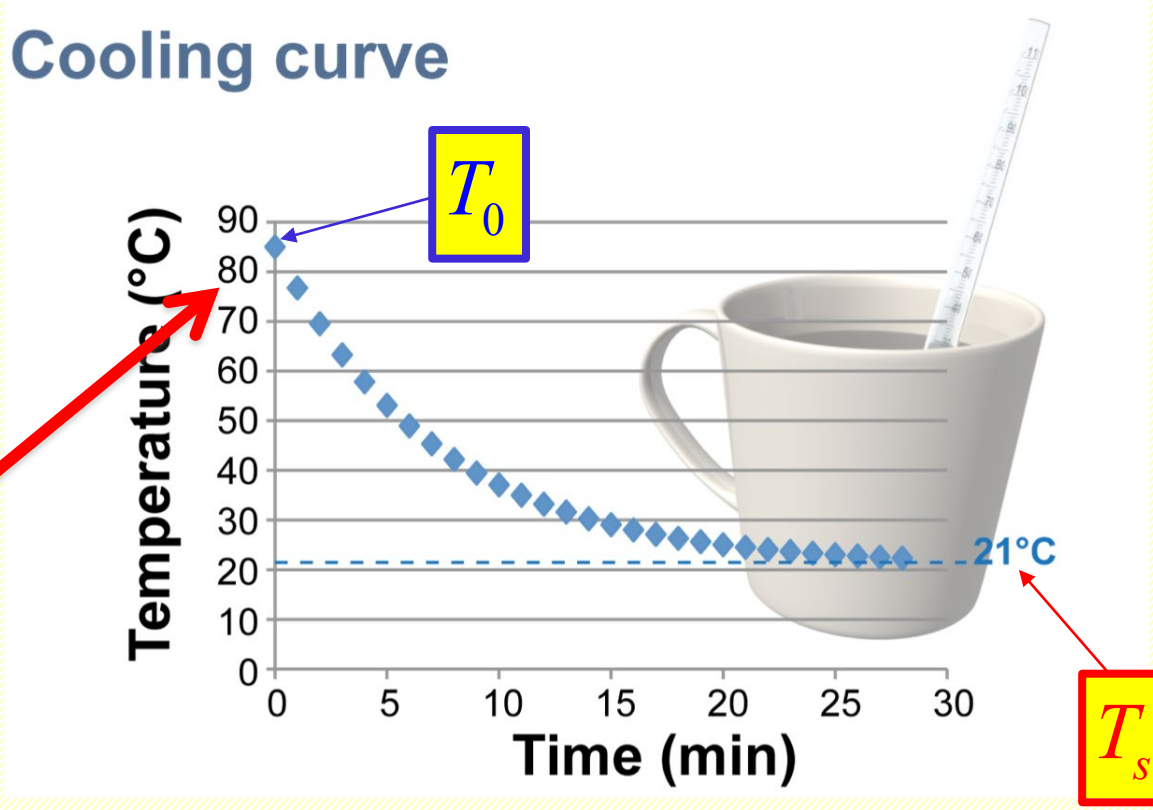
Courtesy: https://education.pasco.com/ebooks/static/FloridaPhysicsReview/BookInd-638.html

Newton's Law of Cooling (and Heating): Example for Everyone to Understand

Large temperature differences

A large temperature difference means rapid cooling.

When the coffee is much hotter than the air, its temperature drops **8°C in one minute**.



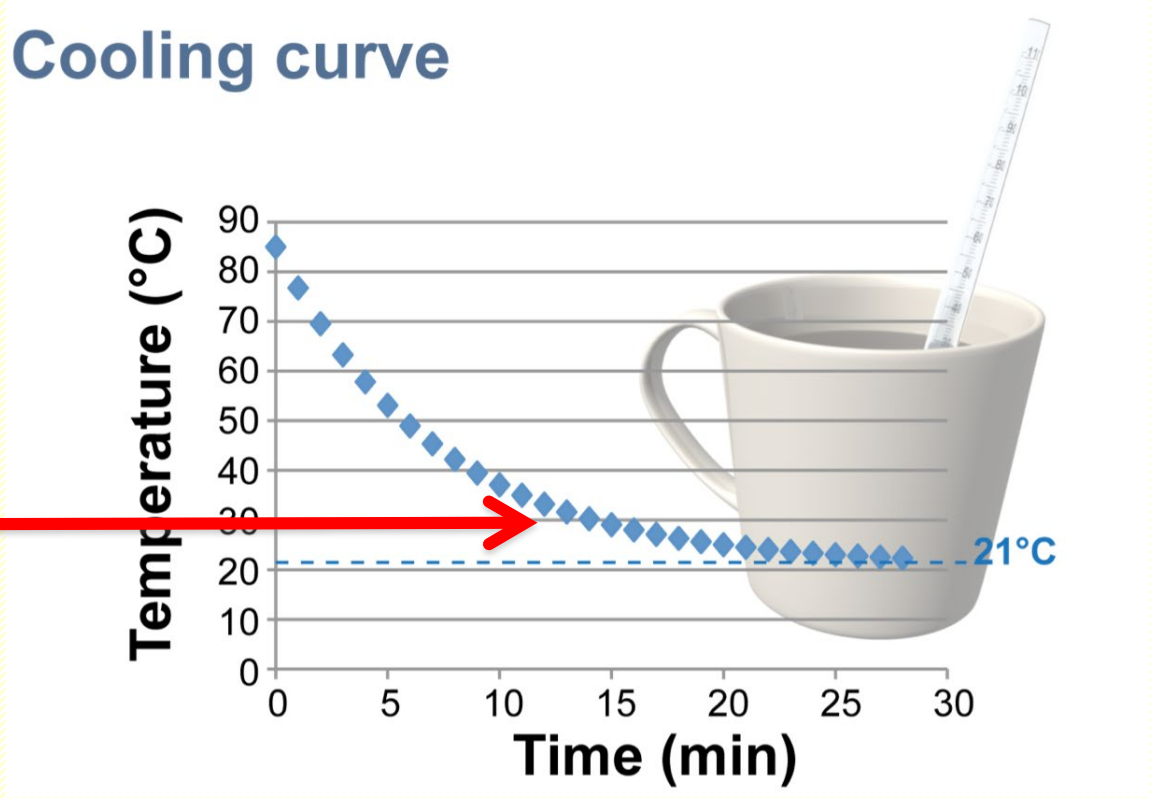
Courtesy: https://education.pasco.com/ebooks/static/FloridaPhysicsReview/BookInd-638.html

Newton's Law of Cooling (and Heating): Example for Everyone to Understand

Small temperature differences

A small temperature difference means slower cooling.

When the coffee has cooled to 30°C, the temperature changes by only **1.2°C per minute**.



Courtesy: https://education.pasco.com/ebooks/static/FloridaPhysicsReview/BookInd-638.html

Newton's Law of Cooling (and Heating):

$$\frac{dT}{dt} = k(T - T_s) \qquad \Rightarrow \qquad \frac{dT}{T - T_s} = k\,dt$$

$$\ln\left|T - T_s\right| = kt + \text{Constant}_1$$

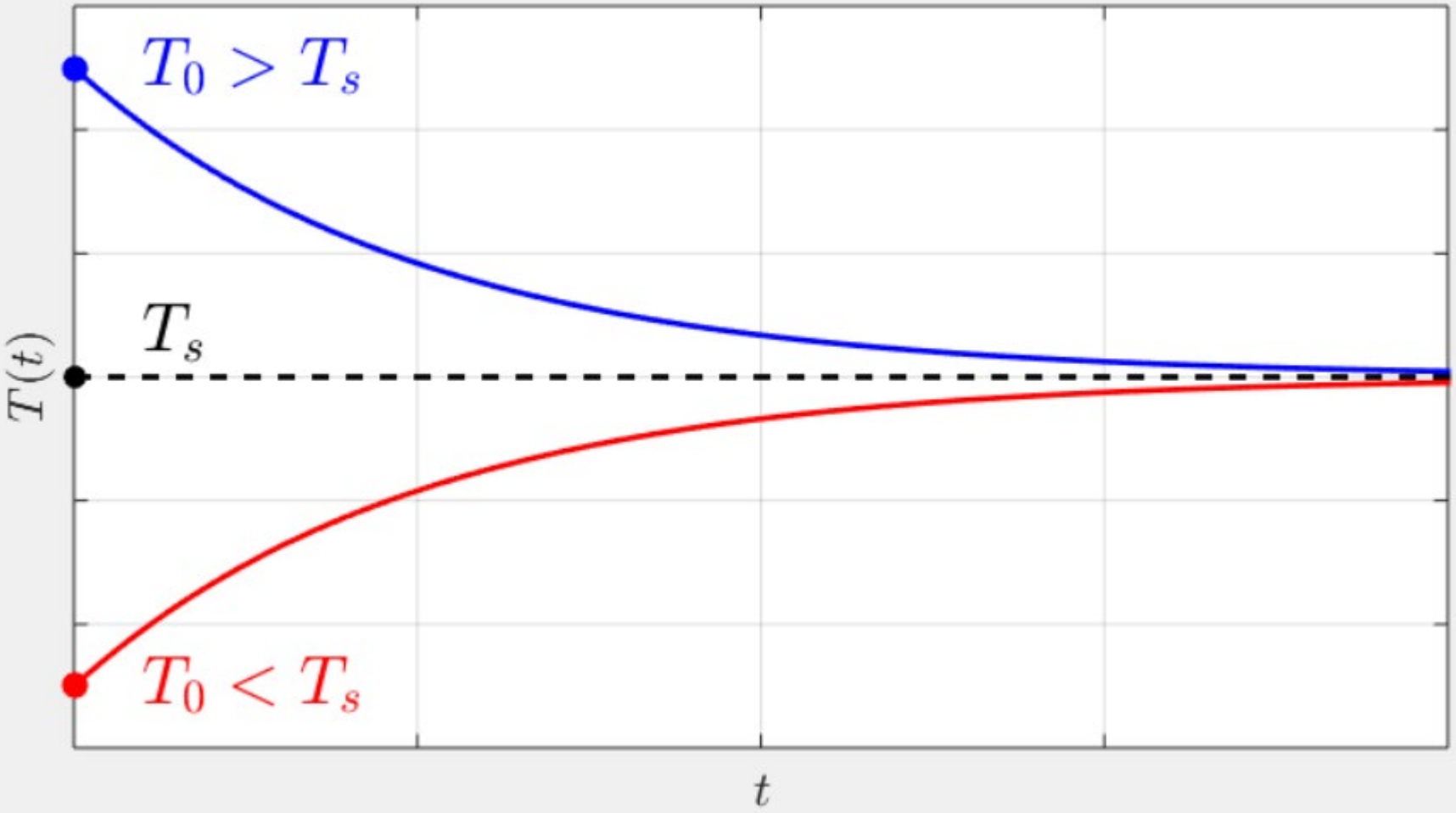
$$T - T_s = (\text{Constant}_2) \times e^{kt} \quad (\text{the } e^{\text{Constant}_1} \text{ and the } \pm \text{ all gets absorbed in the Constant}_2)$$

$$T = T_s + (\text{Constant}_2) \times e^{kt}$$

$$\text{For } t = 0: \quad T_0 = T_s + (\text{Constant}_2) \times e^{0} \quad \Rightarrow \quad (\text{Constant}_2) = T_0 - T_s$$

$$\boxed{T(t) = T_s + (T_0 - T_s) \times e^{kt}}$$

The figure below shows the general shape of $T(t)$ when $T_0 > T_s$ (i.e., in a **cooling** scenario) and when $T_0 < T_s$ (i.e., in a **heating** scenario):



LIMITATIONS:

The **Newton's Law of cooling** is applicable under the following **limitations**:

(1) This Law applies to cooling by convection and radiation and not by radiation alone. To achieve this condition the hot body is cooled in a uniform flow of air so than the radiation losses become small as compared to that due to forced convection.

(2) This Law is applicable in still air only for a temperature difference of about 20K or 30K but it is true for all temperature difference in conditions of forced convection of the air.

[*Reference: Gupta S.K. & Kumar A., Krishna's Engineering Physics. Volume II (Thermal Physics). Krishna Prakash Media, 2001, ISBN 81 87224 20 7. - p.2.56*]

NEWTON's LAW OF COOLING (and HEATING):

EXAMPLE

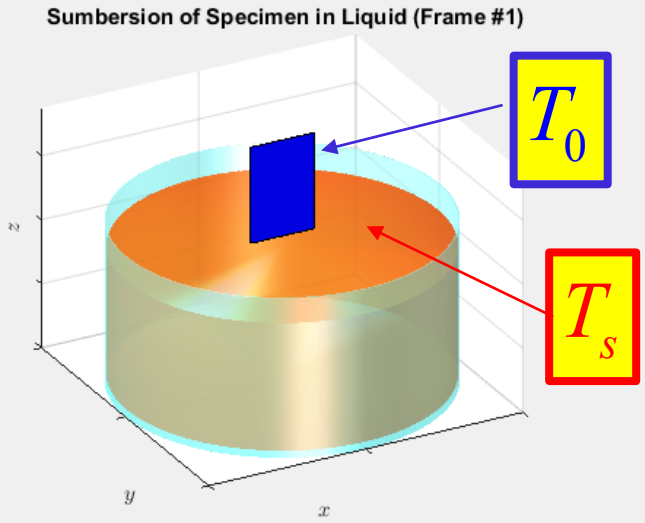
A small metal bar whose temperature is $30^{o}\,C$ is dropped into a container of $75^{o}\,C$ water.

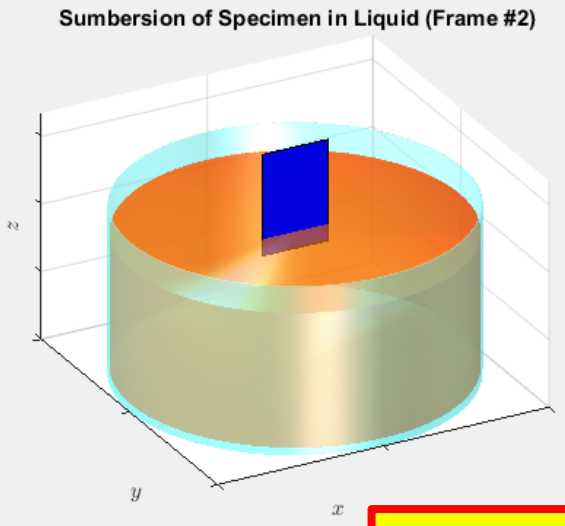
After 1 second the temperature of the bar has increased by $1^{o}\,C$ (i.e. up to $31^{o}\,C$).

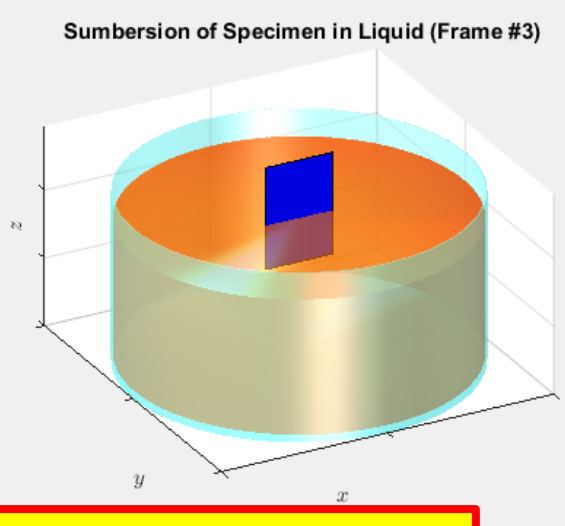
(a) How long will it take for the temperature of the bar to reach $70^{o}\,C$?

(b) 74◦C?

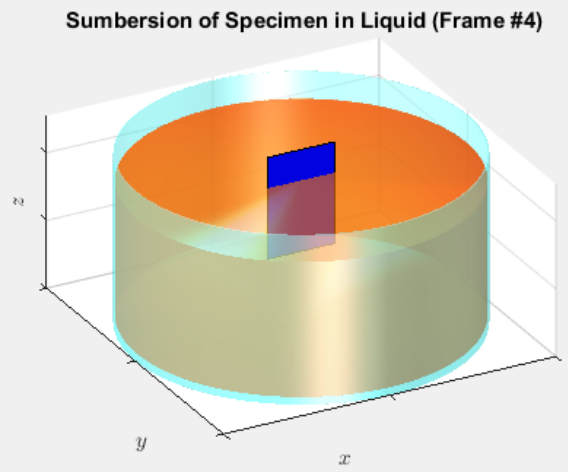
ILLUSTRATION of PROCESS by PMT with MATLAB's GRAPHICS

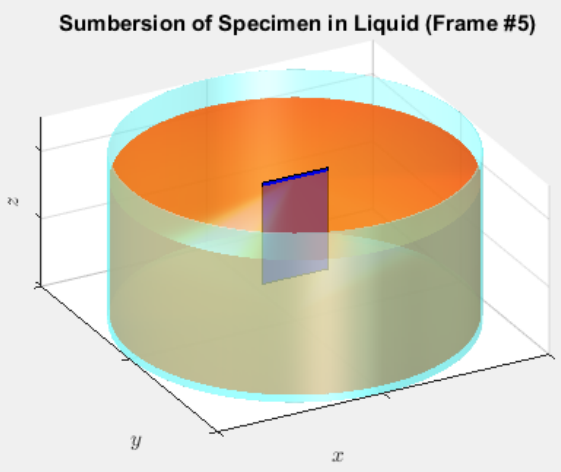


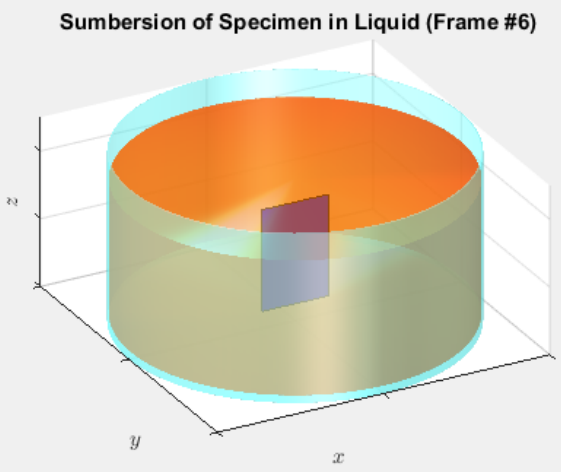




$$T(t) = T_s + (T_0 - T_s) \times e^{kt}$$







SOLUTION: (1) Determine value of *k* first

$$T(t) = T_s + (T_0 - T_s)\,e^{kt_1}$$

$$31 = 75 + (30 - 75)\,e^{k\times 1}$$

$$(75 - 31) = (75 - 30)\,e^{k}$$

$$e^{k} = \left(\frac{75-31}{75-30}\right) = \left(\frac{44}{45}\right)$$

$$k = \ln\left(\frac{44}{45}\right)$$

$$k = -0.0225$$

SOLUTION: (2) Simulating reach of 70°C

$$T(t) = T_s + (T_0 - T_s)\, e^{kt_{70}}$$

$$70 = 75 + (30 - 75)\, e^{kt_{70}}$$

$$(75 - 70) = (75 - 30)\, e^{kt_{70}}$$

$$e^{kt_{70}} = \left(\frac{75-70}{75-30}\right) = \frac{5}{45} = \frac{1}{9} \quad \Rightarrow \quad kt_{70} = \ln\left(\frac{1}{9}\right)$$

$$t_{70} = \frac{1}{k}\ln\left(\frac{1}{9}\right) = 97.7724 \text{ (sec)}$$

SOLUTION: (3) Simulating reach of 74°C

$$T(t) = T_s + (T_0 - T_s)\, e^{kt_{74}}$$

$$74 = 75 + (30 - 75)\, e^{kt_{74}}$$

$$(75 - 74) = (75 - 30)\, e^{kt_{74}}$$

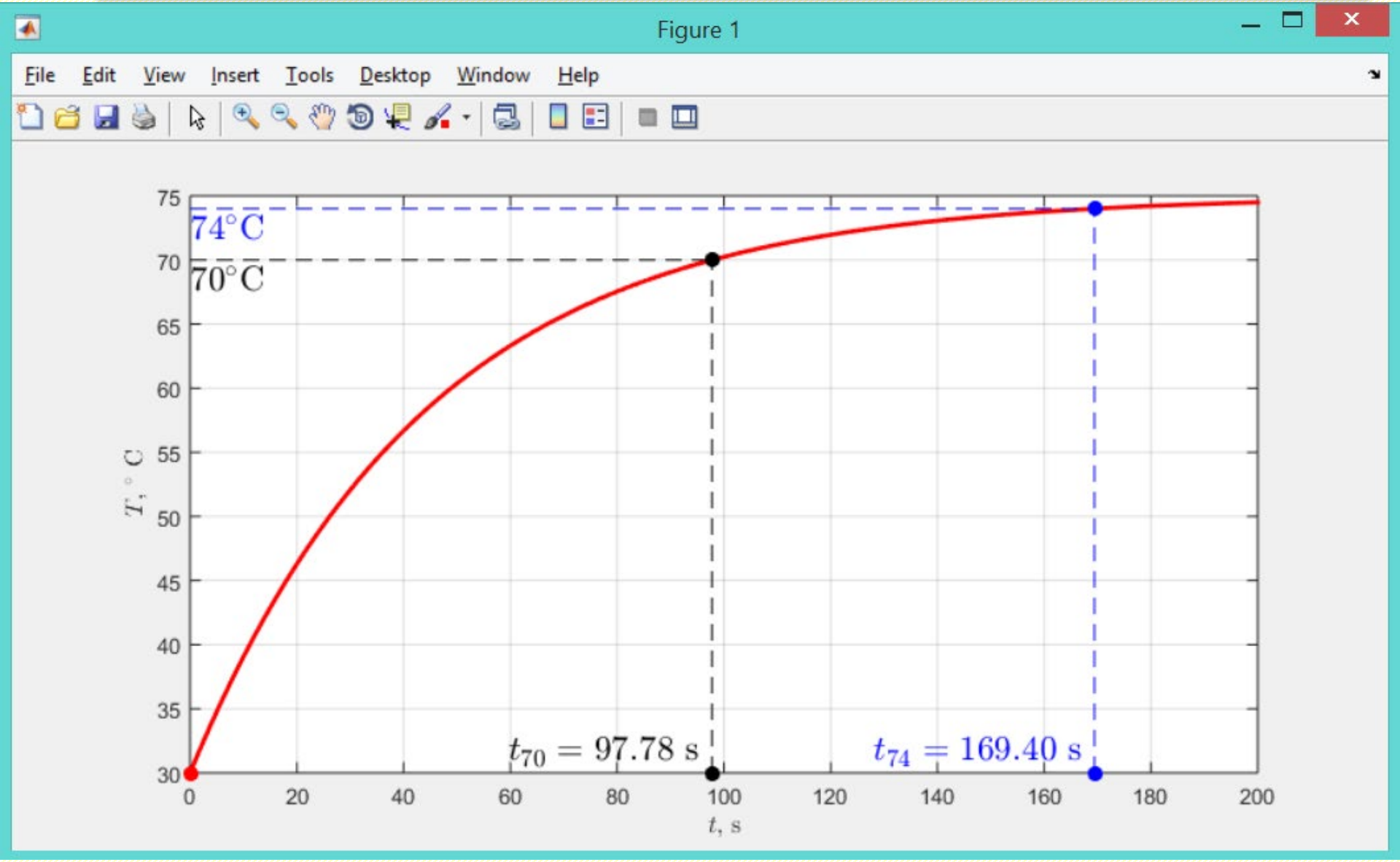
$$e^{kt_{74}} = \left(\frac{75-74}{75-30}\right) = \frac{1}{45} \quad \Rightarrow \quad kt_{74} = \ln\left(\frac{1}{45}\right)$$

$$t_{74} = \frac{1}{k}\ln\left(\frac{1}{45}\right) = 169.3894 \text{ (sec)}$$

Advanced MATLAB script for simulating process for the time range

```
%%
% Designed by Prof P.M.Trivailo (C)2020
tt=[0:0.01:1]*200;
k=log(44/45); T0=30; Ts=75;
syms T(t)
ode = diff(T,t) == k*(T-Ts)       % dT/dt= k(T-Ts)
cond = T(0) == T0; ySol(t) = dsolve(ode,cond)
TT=eval(subs(ySol,{t},{tt}));
figure; plot(tt,TT,'LineWidth',2,'Color','r'); hold on;
line('XData',tt(1),'YData',TT(1),'Marker','.','MarkerSize',24,'Color','r');
t70=interp1(TT,tt,70);
line('XData',t70,'YData',70,'Marker','.','MarkerSize',24,'Color','k');
line('XData',t70,'YData',T0,'Marker','.','MarkerSize',24,'Color','k');
line('XData',[1 1]*t70,'YData',[T0 70],'LineStyle','--','Color','k');
line('XData',[0 1]*t70,'YData',[1 1]*70,'LineStyle','--','Color','k');
text('String',sprintf('$t_{70}=%6.2f$ s ',t70),'Interpreter','LaTeX','Color','k','FontSize',16,...
    'Position',[t70 T0],'VerticalAlignment','bottom','HorizontalAlignment','right');
text('String','$70^\circ$C','Interpreter','LaTeX','Color','k','FontSize',16,...
    'Position',[0 70],'VerticalAlignment','top');
t74=interp1(TT,tt,74);
line('XData',t74,'YData',74,'Marker','.','MarkerSize',24,'Color','b');
line('XData',t74,'YData',T0,'Marker','.','MarkerSize',24,'Color','b');
line('XData',[1 1]*t74,'YData',[T0 74],'LineStyle','--','Color','b');
line('XData',[0 1]*t74,'YData',[1 1]*74,'LineStyle','--','Color','b');
text('String',sprintf('$t_{74}=%6.2f$ s ',t74),'Interpreter','LaTeX','Color','b','FontSize',16,...
    'Position',[t74 T0],'VerticalAlignment','bottom','HorizontalAlignment','right');
text('String','$74^\circ$C','Interpreter','LaTeX','Color','b','FontSize',16,...
    'Position',[0 74],'VerticalAlignment','top');
grid on; xlabel('$t$, s','Interpreter','LaTeX'); ylabel('$T$, $^\circ$ C','Interpreter','LaTeX');
set(gcf,'Position',[214   334   834   428]);
```

RESULTS: Simulation, using MATLAB



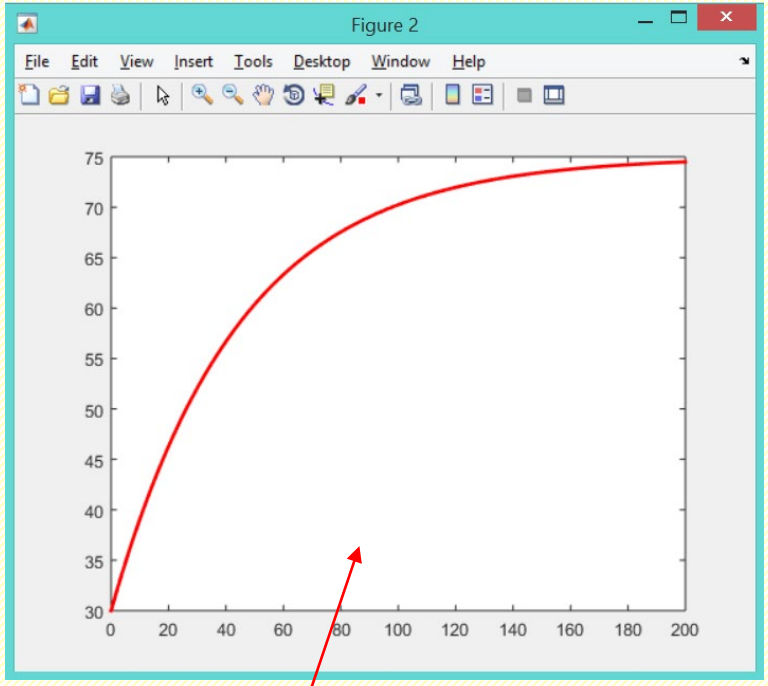
Comments of MATLAB script

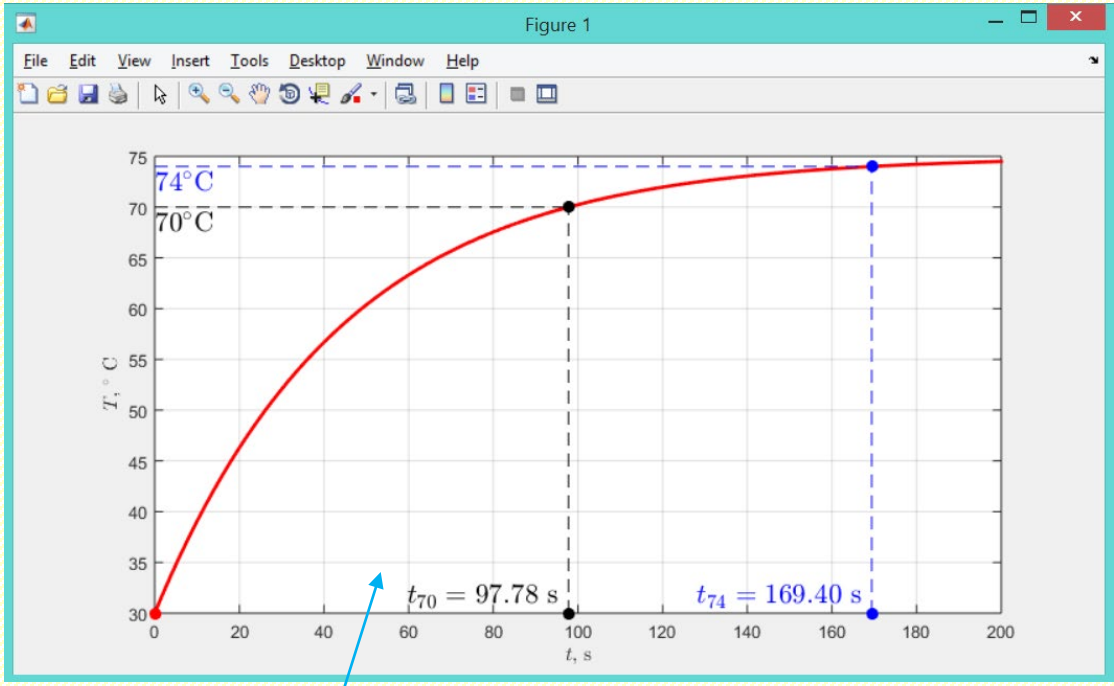
```
%%
% Designed by Prof P.M.Trivailo (C)2020
tt=[0:0.01:1]*200;
k=log(44/45); T0=30; Ts=75;
syms T(t)
ode = diff(T,t) == k*(T-Ts)       % dT/dt= k(T-Ts)
cond = T(0) == T0; ySol(t) = dsolve(ode,cond)
TT=eval(subs(ySol,{t},{tt}));
figure; plot(tt,TT,'LineWidth',2,'Color','r'); hold on;
line('XData',tt(1),'YData',TT(1),'Marker','.','MarkerSize',24,'Color','r');
t70=interp1(TT,tt,70);
line('XData',t70,'YData',70,'Marker','.','MarkerSize',24,'Color','k');
line('XData',t70,'YData',T0,'Marker','.','MarkerSize',24,'Color','k');
line('XData',[1 1]*t70,'YData',[T0 70],'LineStyle','--','Color','k');
line('XData',[0 1]*t70,'YData',[1 1]*70,'LineStyle','--','Color','k');
text('String',sprintf('$t_{70}=%6.2f$ s ',t70),'Interpreter','LaTeX','Color','k','FontSize',16,...
    'Position',[t70 T0],'VerticalAlignment','bottom','HorizontalAlignment','right');
text('String','$70^\circ$C','Interpreter','LaTeX','Color','k','FontSize',16,...
    'Position',[0 70],'VerticalAlignment','top');
t74=interp1(TT,tt,74);
line('XData',t74,'YData',74,'Marker','.','MarkerSize',24,'Color','b');
line('XData',t74,'YData',T0,'Marker','.','MarkerSize',24,'Color','b');
line('XData',[1 1]*t74,'YData',[T0 74],'LineStyle','--','Color','b');
line('XData',[0 1]*t74,'YData',[1 1]*74,'LineStyle','--','Color','b');
text('String',sprintf('$t_{74}=%6.2f$ s ',t74),'Interpreter','LaTeX','Color','b','FontSize',16,...
    'Position',[t74 T0],'VerticalAlignment','bottom','HorizontalAlignment','right');
text('String','$74^\circ$C','Interpreter','LaTeX','Color','b','FontSize',16,...
    'Position',[0 74],'VerticalAlignment','top');
grid on; xlabel('$t$, s','Interpreter','LaTeX'); ylabel('$T$, $^\circ$ C','Interpreter','LaTeX');
set(gcf,'Position',[214   334   834   428]);
```

"CORE" COMMANDS

"ANNOTATIONS"

COMPARE: “Core” & “Decorated” Plots





Plot, created with “CORE” commands only

Plot, created with “CORE” and “DECORATIONS” commands

!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH THE SAME EXAMPLE

SOLUTION: Numerical (!!!) Method

```
%% COOLING & HEATING EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
% Fundamental Law:  dT/dt = k(T-Ts);
%-------------------------------------%
T0 = 30; Ts=75;                       % initial & surroundings temperature
k = log(44/45);                       %
tmax = 200; t = [0:0.01:1]*tmax;      % regular time array

T_xdot_anonymous3 = @(t, T) k*(T-Ts);  % anonymous FUNCTION IS USED
[tt3,TT3] = ode45(T_xdot_anonymous3,t,T0);% call ODE

plot(tt3,TT3,'LineWidth',6,'Color','r','LineStyle','--');
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX'); ylabel('$T(t)$ [$^\circ$]','Interpreter','LaTeX');
str = sprintf('Temperature: Time History ($$k$$=%4.3f, $$T_0$$=%g$$^\\circ C$$, $$T_s$$=%g$$^\\circ C$$)',k,T0,Ts)
title(str, 'Interpreter','LaTeX');
set(gca,'FontSize',16); set(gcf,'Position',[214   334   834   428]);
Texact = Ts +(T0-Ts)*exp(k*t);
hold on; plot(t,Texact,'LineWidth',2,'Color','b');
legend('Numerical Solution','Exact Analytical Solution','Location','SouthEast');
axis([0 tmax T0 Ts]);
```

SOLUTION: Numerical (!!!) Method

1. Input of the data

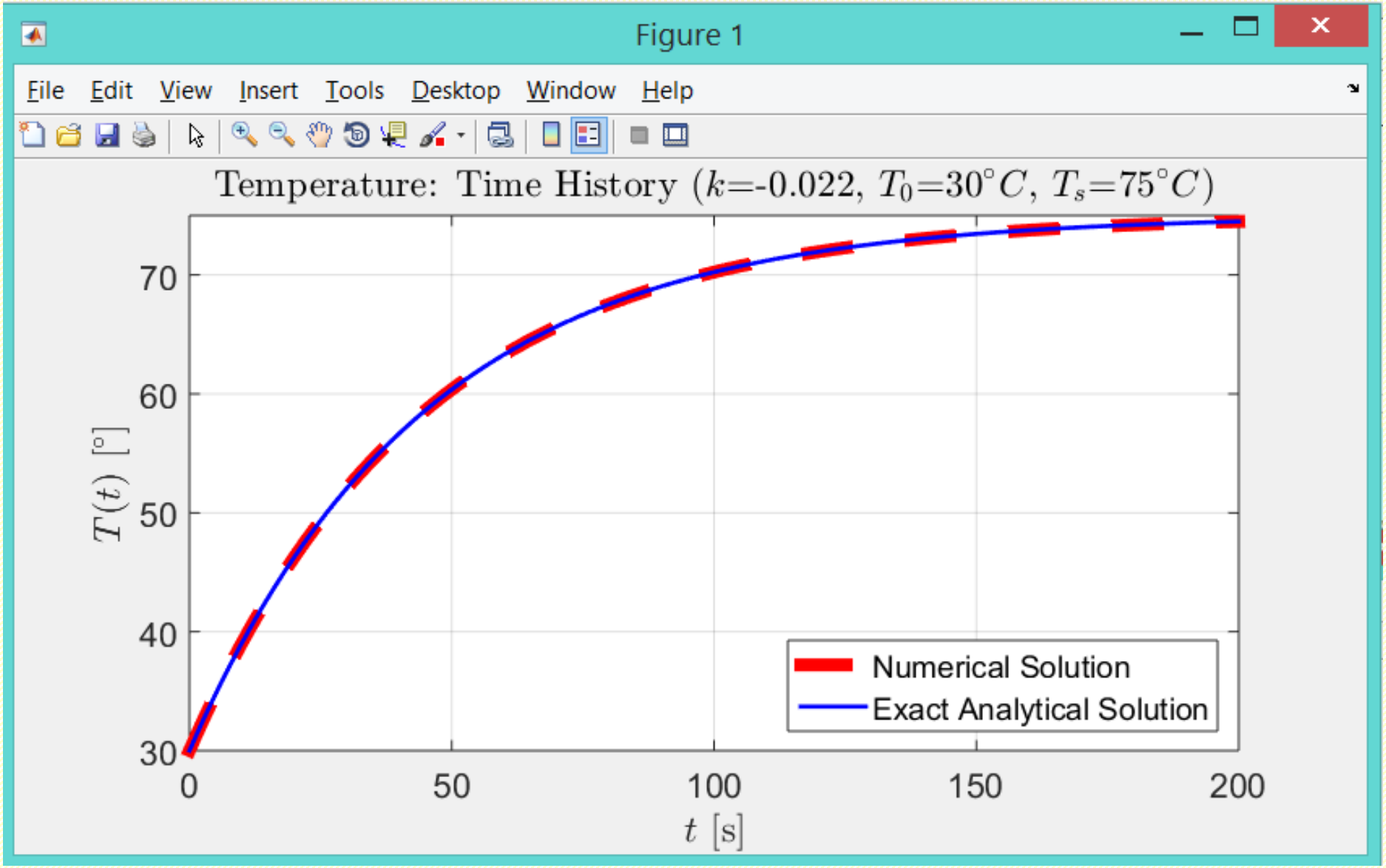
2. $$\frac{dT}{dt} = k(T - T_s)$$

```
%% COOLING & HEATING EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
% Fundamental Law:  dT/dt = k(T-Ts);
%-------------------------------------%
T0 = 30; Ts=75;                       % initial & surroundings temperature
k = log(44/45);                       %
tmax = 200; t = [0:0.01:1]*tmax;      % regular time array

T_xdot_anonymous3 = @(t, T) k*(T-Ts);  % anonymous FUNCTION IS USED
[tt3,TT3] = ode45(T_xdot_anonymous3,t,T0);% call ODE

plot(tt3,TT3,'LineWidth',6,'Color','r','LineStyle','--');
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX'); ylabel('$T(t)$ [$^\circ$]','Interpreter','LaTeX');
str = sprintf('Temperature: Time History ($$k$$=%4.3f, $$T_0$$=%g$$^\\circ C$$, $$T_s$$=%g$$^\\circ C$$)',k,T0,Ts)
title(str, 'Interpreter','LaTeX');
set(gca,'FontSize',16); set(gcf,'Position',[214   334   834   428]);
Texact = Ts +(T0-Ts)*exp(k*t);
hold on; plot(t,Texact,'LineWidth',2,'Color','b');
legend('Numerical Solution','Exact Analytical Solution','Location
axis([0 tmax T0 Ts]);
```

3. Calling ode45 proc.!

4. Plotting results

Output of the MATLAB script



HOME WORK:

PRACTICE EXAMPLE

You are given a very hot sample of metal, and wish to know its temperature.

You have a thermometer, but it only measures up to 200 ^{o}C and the metal is hotter than that!

You leave the metal in a room kept at 20 ^{o}C.

After six minutes it has cooled sufficiently that you can measure its temperature; it is 80 ^{o}C.

After another two minutes it is 50 ^{o}C.

What was the initial temperature of the metal?

FIRST ORDER DIFFERENTIAL EQUATIONS:

ANALYTICAL SOLUTION

A 1st-order linear differential equation is an equation of the form

$$y' + P(x)\, y = Q(x)$$

where *P* and *Q* are functions of *x*.

An Eq. that is written in this form is said to be in standard form.

To solve it, we identify *P(x)* and *Q(x),* then calculate *u(x)*

$$u(x) = \mathrm{e}^{\int P(x)\, dx}$$

which is called an integrating factor.

The general solution of the 1st-order differential equation is:

$$y(x) = \frac{1}{u(x)} \int Q(x)\, u(x)\; dx$$

FIRST ORDER DIFFERENTIAL EQUATIONS:

BANK TRUST EXAMPLE

A bank account that earns **8%** interest compounded continuously has an initial balance of zero. Money is deposited into the account at a constant rate of **$1,000** per year (about $2.74/day).

What is the balance in the account after 20 years?

Courtesy: http://www-personal.umich.edu/~wheeles/116Fall2012/diffmodextracredit.pdf

SOLUTION: Modelling

Let *y* represent the balance after *t* years.

The balance increases in two ways:

with *interest* and with additional *deposits*:

$$\frac{dy}{dt} = \underbrace{0.08\ y}_{\text{Interest}} + \underbrace{1000}_{\text{Deposit}}$$

In standard form, this linear differential equation is:

$$y' - 0.08\ y = 1000$$

The differential equation is separable, this allows analytical solutions:

$$\frac{dy}{dt} = 0.08\,y + 1000; \qquad \frac{dy}{0.08\,y + 1000} = dt$$

$$\frac{1}{0.08}\int \frac{du}{u} = \int dt \qquad (\text{by setting } u = 0.08\,y + 1000)$$

$$\frac{100}{8}\ln\left|0.08\,y + 1000\right| = t + \text{Constant}_1$$

$$\ln\left|0.08\,y + 1000\right| = \frac{2}{25}t + \text{Constant}_2; \quad \left(\text{the } \frac{8}{100} = \frac{2}{25} \text{ gets absorbed in the Constant}_2\right)$$

$$0.08\,y + 1000 = (\text{Constant}_3) \times e^{\frac{2}{25}t}; \quad (\text{the } e^{\text{Constant}_2} \text{ and the } \pm \text{ all gets absorbed in the Constant}_3)$$

$$0.08\,y = (\text{Constant}_3) \times e^{\frac{2}{25}t} - 1000; \quad y = (\text{Constant}_4) \times e^{\frac{2}{25}t} - 12\,500; \quad \left(\frac{1}{0.08} \text{ gets absorbed in Constant}_4\right)$$

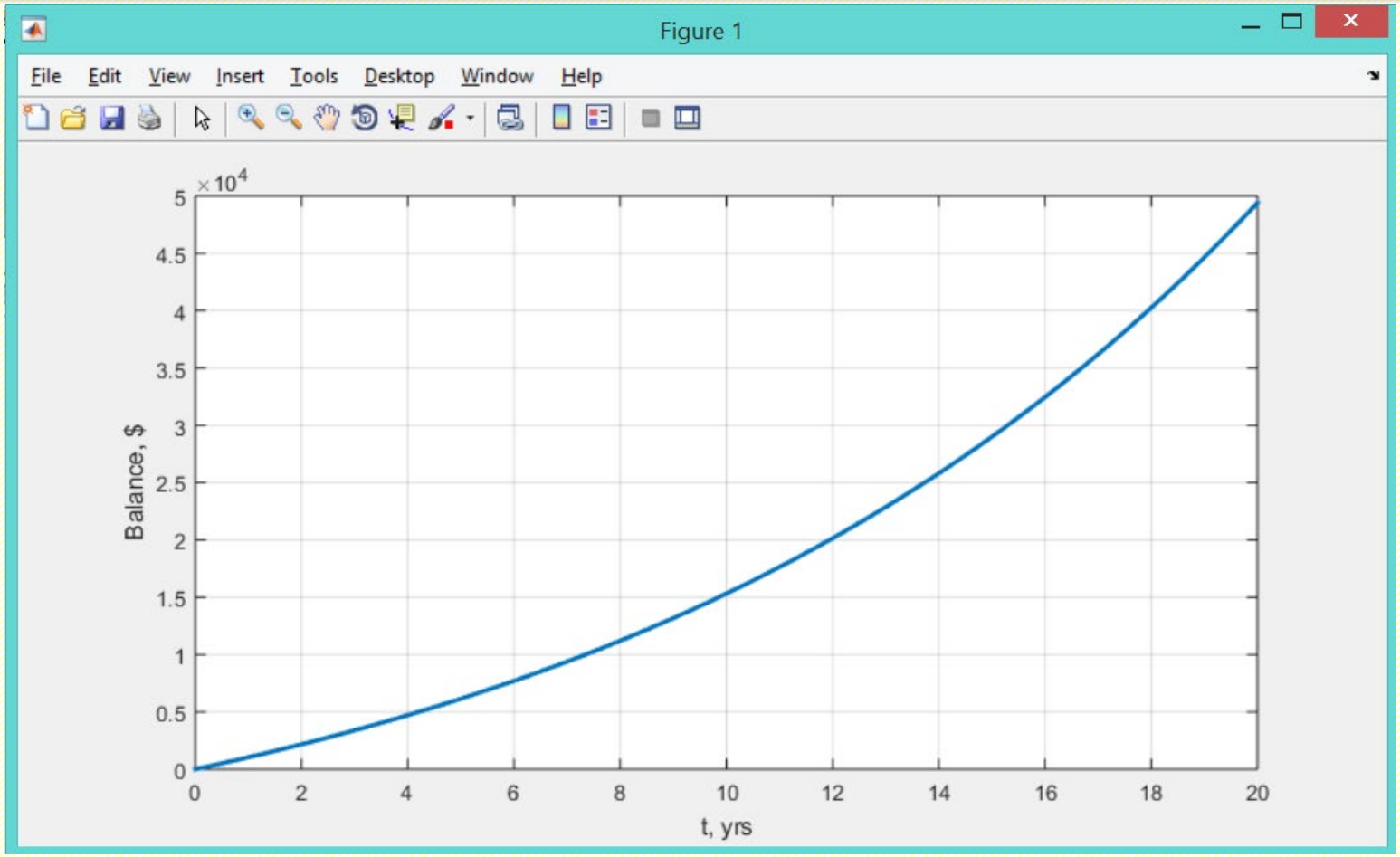
Initial zero balance enables us to find the value of the **Constant$_4$**:

$$y\big|_{t=0} = 0; \quad \Rightarrow \quad 0 = (\text{Constant}_4) \times e^{\frac{0}{25}} - 12\,500; \quad \Rightarrow \quad \text{Constant}_4 = 12\,500$$

Analytical exact solution: $y(t) = 12\,500 \times e^{\frac{2}{25}t} - 12\,500$

SOLUTION: Simulation, using MATLAB

```
t=[0:0.01:1]*20; y=12500*exp(2*t/25)-12500;
plot(t,y,'LineWidth',2);
grid on; xlabel('t, yrs'); ylabel('Balance, $');
```



To determine the balance of the account towards the end of simulation time (t=20 years), and print results in a nice format, we can use the following MATLAB command (here example of entering this command from Command Window is shown):

```
>> sprintf('y(end) = $ %8.2f',y(end))

ans =

y(end) = $ 49412.91
```

Answer: $y(20) = \$\ 49,412.91$

!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH THE SAME EXAMPLE

SOLUTION: Numerical (!!!) Method

```
%% BANK TRUST EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
% Fundamental Law:  dy/dt = ky + D;
%-------------------------------------%
close('all'); clc; clear;
D = 1000; y0=0;                         % deposit & initial bank balance
k = 0.08;                               % interest rate
tmax = 20; t = [0:0.01:1]*tmax;         % regular time array
y_xdot_anonymous4 = @(t, y) k*y + D;    % anonymous FUNCTION IS USED
[tt4,TT4] = ode45(y_xdot_anonymous4,t,y0); % call ODE
plot(tt4,TT4,'LineWidth',6,'Color','r','LineStyle','--');
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX'); ylabel('Balance $y(t)$, \$','Interpreter','LaTeX');
str = sprintf('Bank Balance: Time History ($$k$$=%4.2f\\%%, $$y_0$$=%g\\$,
$$D$$=%g\\$)',k*100,y0,D)
title(str, 'Interpreter','LaTeX');
set(gca,'FontSize',16); set(gcf,'Position',[214   334   834   428]);
yexact = 12500*exp(2*t/25)-12500;
hold on; plot(t,yexact,'LineWidth',2,'Color','b');
legend('Numerical Solution','Exact Analytical Solution','Location','SouthEast');
```

SOLUTION: Numerical (!!!) Method

1. Input of the data

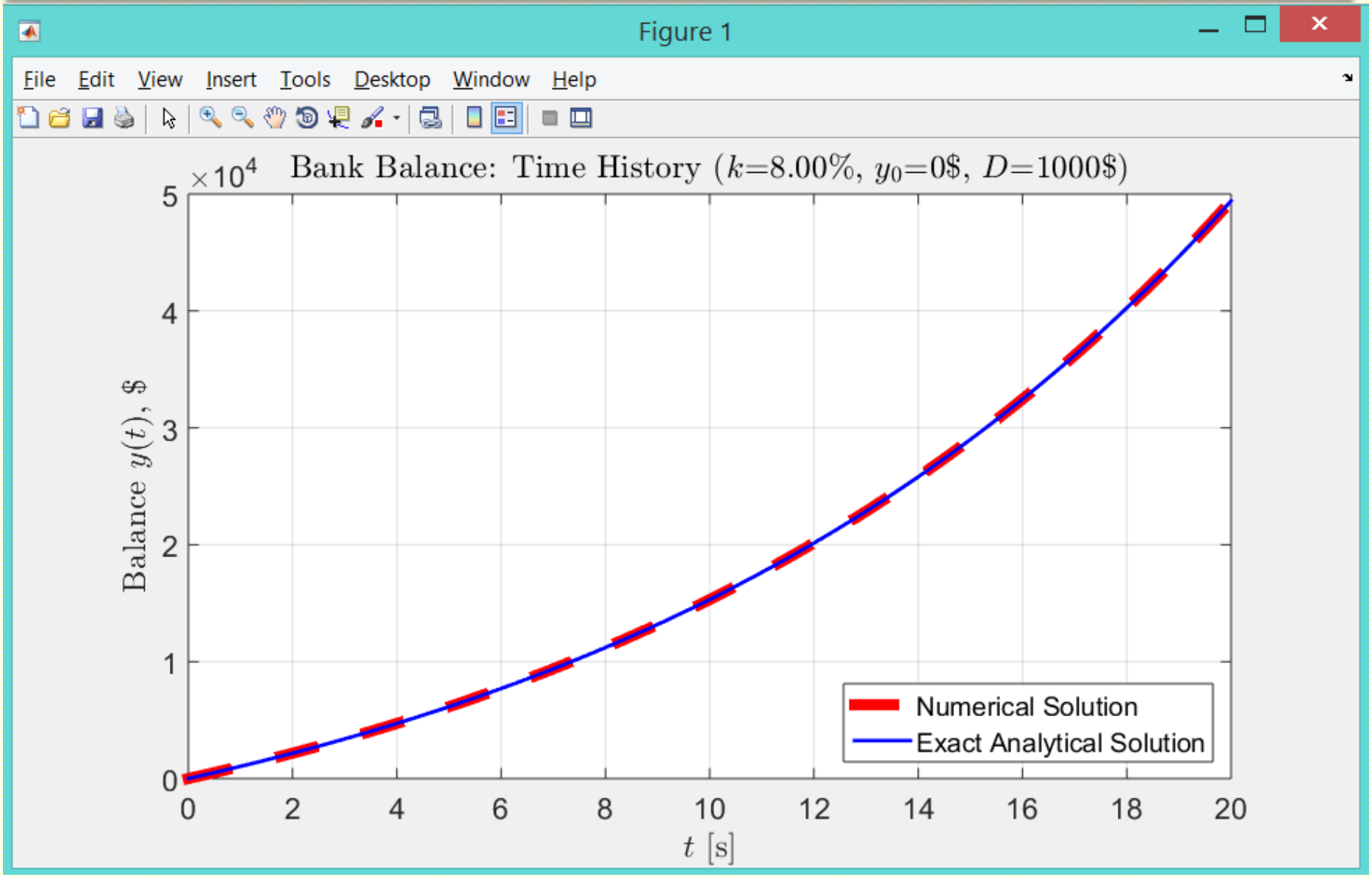
$$\frac{dy}{dt} = k\,y + D$$ 2.

```
%% BANK TRUST EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
% Fundamental Law:  dy/dt = ky + D
%-------------------------------------%
close('all'); clc; clear;
D = 1000; y0=0;                    % deposit & initial bank balance
k = 0.08;                          % interest rate
tmax = 20; t = [0:0.01:1]*tmax;    % regular time array
y_xdot_anonymous4 = @(t, y) k*y + D;     % anonymous FUNCTION IS USED
[tt4,TT4] = ode45(y_xdot_anonymous4,t,y0); % call ODE
plot(tt4,TT4,'LineWidth',6,'Color','r','LineStyle','--');
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX'); ylabel('Balance $y(t)$, \$','Interpreter','LaTeX');
str = sprintf('Bank Balance: Time History ($$k$$=%4.2f\\%%, $$y_0$$=%g\\$,
$$D$$=%g\\$)',k*100,y0,D)
title(str, 'Interpreter','LaTeX');
set(gca,'FontSize',16); set(gcf,'Position',[214   334   834   428]);
yexact = 12500*exp(2*t/25)-12500;
hold on; plot(t,yexact,'LineWidth',2,'Color','b');
legend('Numerical Solution','Exact Analytical Solution','Location
```

3. Calling ode45 procedure !

4. Plotting results

Output of the MATLAB script



FIRST ORDER DIFFERENTIAL EQUATIONS:

BANK TRUST EXAMPLE

*Changes to the previous task on this slide are shown in **red font:***

A bank account that earns **10%** interest compounded continuously has an initial balance of zero. Money is deposited into the account at a constant rate of **$1,000** per year (about $2.74/day).

What is the balance in the account after 20 years?

When will account have exactly **$40,000**?

Courtesy: http://www-personal.umich.edu/~wheeles/116Fall2012/diffmodextracredit.pdf

SOLUTION: Modelling

Let *y* represent the balance after *t* years.

The balance increases in two ways:

with *interest* and with additional *deposits*:

$$\frac{dy}{dt} = \underbrace{0.1\,y}_{\text{Interest}} + \underbrace{1000}_{\text{Deposit}}$$

In standard form, this linear differential equation is:

$$y' - 0.1\,y = 1000$$

The differential equation is separable allowing analytical solutions:

$$\frac{dy}{dt} = 0.1\,y + 1000; \qquad \frac{dy}{0.1\,y + 1000} = dt$$

$$10\int \frac{du}{u} = \int dt \qquad (\text{by setting } u = 0.1\,y + 1000)$$

$$10\,\ln\left|0.1\,y + 1000\right| = t + \text{Constant}_1$$

$$\ln\left|0.1\,y + 1000\right| = \frac{t}{10} + \text{Constant}_2; \quad \left(\text{the } \frac{1}{10} \text{ gets absorbed in the Constant}_2\right)$$

$$0.1\,y + 1000 = (\text{Constant}_3) \times e^{\frac{t}{10}}; \quad \left(\text{the } e^{\text{Constant}_2} \text{ and the } \pm \text{ all gets absorbed in the Constant}_3\right)$$

$$0.1\,y = (\text{Constant}_3) \times e^{\frac{t}{10}} - 1000; \quad y = (\text{Constant}_4) \times e^{\frac{t}{10}} - 10\,000; \quad \left(\frac{1}{0.1} \text{ gets absorbed in Constant}_4\right)$$

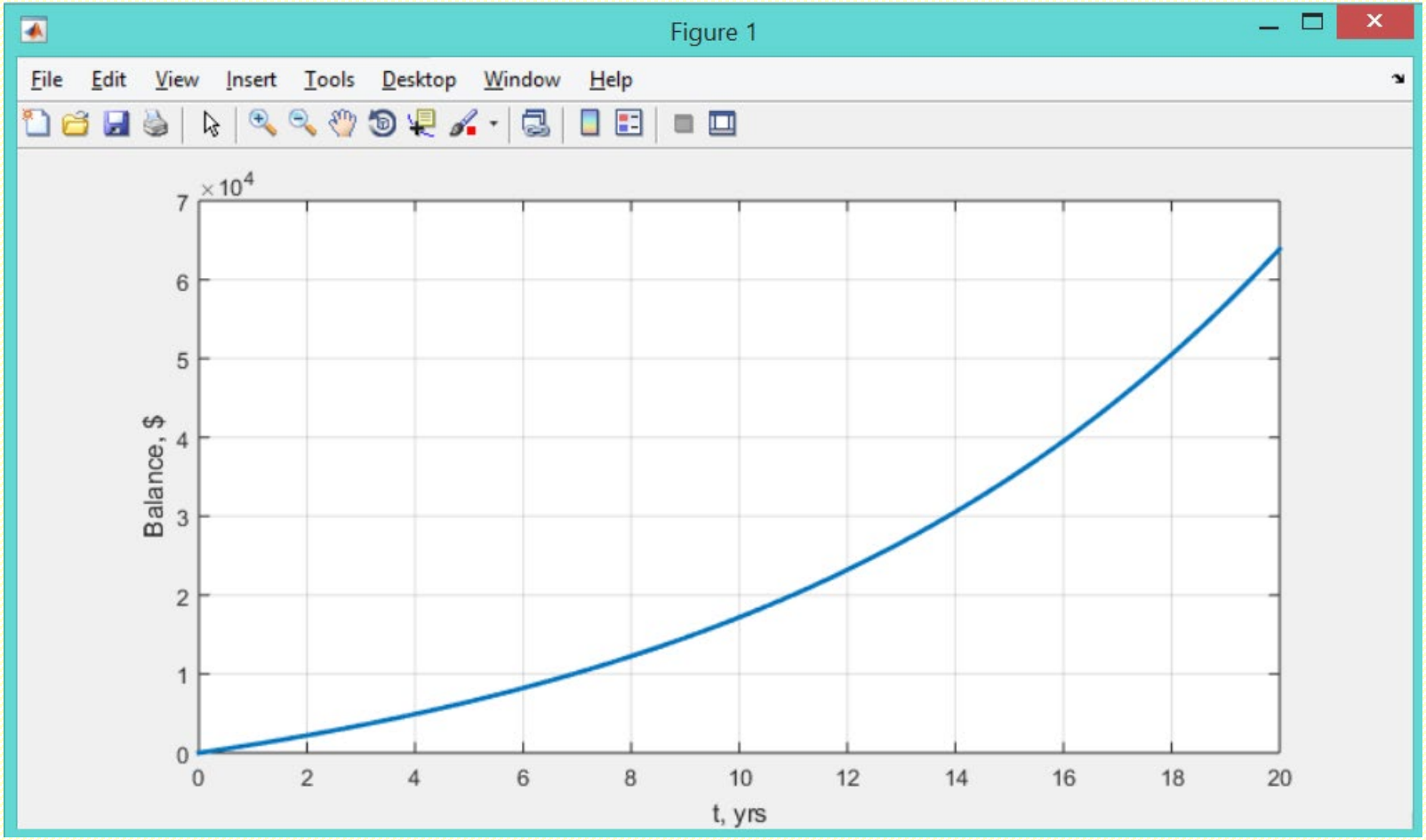
Initial zero balance enables us to find the value of the **Constant$_4$**:

$$y\big|_{t=0} = 0; \quad \Rightarrow \quad 0 = (\text{Constant}_4) \times e^{\frac{0}{10}} - 10\,000; \quad \Rightarrow \quad \text{Constant}_4 = 10\,000$$

Analytical exact solution: $y(t) = 10\,000 \times e^{\frac{t}{10}} - 10\,000$

SOLUTION: Simulation, using MATLAB

```
t=[0:0.01:1]*20; y=10000*exp(t/10)-10000;
plot(t,y,'LineWidth',2);
grid on; xlabel('t, yrs'); ylabel('Balance, $');
```



To determine the balance of the account towards the end of simulation time (t=20 years), and print results in a nice format, we can use the following MATLAB command (here example of entering this command from Command Window is shown):

```
>> sprintf('y(end) = $ %8.2f', y(end))

ans =

y(end) = $ 63890.56
```

Answer: $y(20) = \$\ 63,890.56$

!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH THE SAME EXAMPLE

SOLUTION: Numerical (!!!) Method

```
%% BANK TRUST EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
% Fundamental Law:  dy/dt = ky + D;
%-------------------------------------%
close('all'); clc; clear;
D = 1000; y0=0;                         % deposit & initial bank balance
k = 0.08;                               % interest rate
tmax = 20; t = [0:0.01:1]*tmax;         % regular time array
y_xdot_anonymous4 = @(t, y) k*y + D;    % anonymous FUNCTION IS USED
[tt4,TT4] = ode45(y_xdot_anonymous4,t,y0); % call ODE
plot(tt4,TT4,'LineWidth',6,'Color','r','LineStyle','--');
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX'); ylabel('Balance $y(t)$, \$','Interpreter','LaTeX');
str = sprintf('Bank Balance: Time History ($$k$$=%4.2f\\%%, $$y_0$$=%g\\$,
$$D$$=%g\\$)',k*100,y0,D)
title(str, 'Interpreter','LaTeX');
set(gca,'FontSize',16); set(gcf,'Position',[214   334   834   428]);
yexact = 12500*exp(2*t/25)-12500;
hold on; plot(t,yexact,'LineWidth',2,'Color','b');
legend('Numerical Solution','Exact Analytical Solution','Location','SouthEast');
```

SOLUTION: Numerical (!!!) Method

1. Input of the data

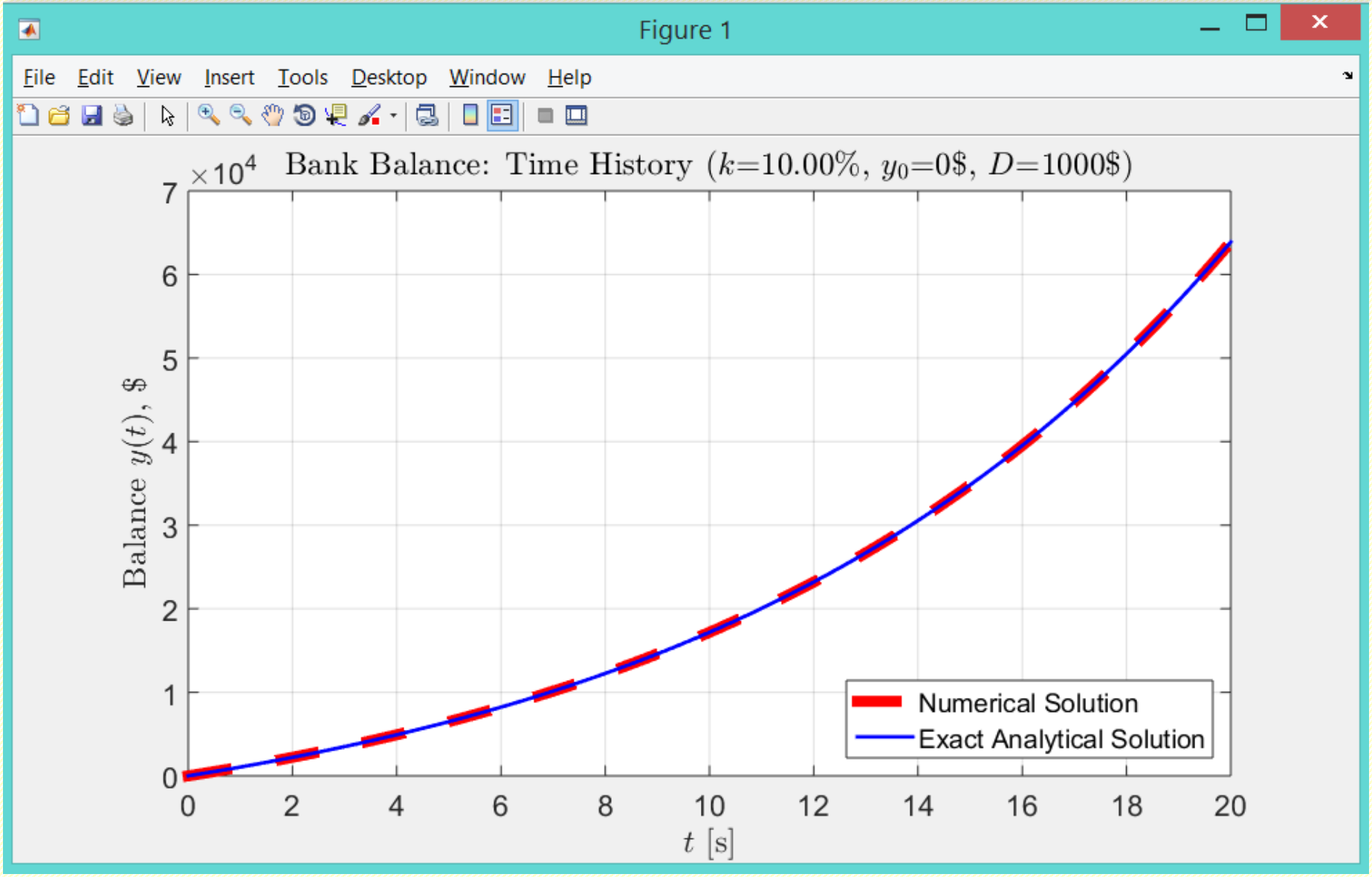
2. $$\frac{dy}{dt} = k\,y + D$$

```
%% BANK TRUST EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
% Fundamental Law:  dy/dt = ky + D
%-------------------------------------%
close('all'); clc; clear;
D = 1000; y0=0;                    % deposit & initial bank balance
k = 0.1;                           % interest rate
tmax = 20; t = [0:0.01:1]*tmax;    % regular time array
y_xdot_anonymous4 = @(t, y) k*y + D; % anonymous FUNCTION IS USED
[tt4,TT4] = ode45(y_xdot_anonymous4,t,y0); % call ODE
plot(tt4,TT4,'LineWidth',6,'Color','r','LineStyle','--');
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX'); ylabel('Balance $y(t)$, \$','Interpreter','LaTeX');
str = sprintf('Bank Balance: Time History ($$k$$=%4.2f\\%%, $$y_0$$=%g\\$,
$$D$$=%g\\$)',k*100,y0,D)
title(str, 'Interpreter','LaTeX');
set(gca,'FontSize',16); set(gcf,'Position',[214   334   834   428]);
yexact = 10000*exp(2*t/25)-10000;
hold on; plot(t,yexact,'LineWidth',2,'Color','b');
legend('Numerical Solution','Exact Analytical Solution','Location
```

3. Calling ode45 procedure !

4. Plotting results

Output of the MATLAB script



Interesting Observation

Increase of the interest rate in the simulated two examples

by 25%

(i.e. from 8% to 10%: [(10-8)/8*100% = 25%])
leads to increase of the bank balance

by 29.30%

[(63890.56-49412.91)/49412.91*100% = 29.2993%].

So, linear “scaling” of results of the first simulation can not be applied to get correct results for second simulation case.

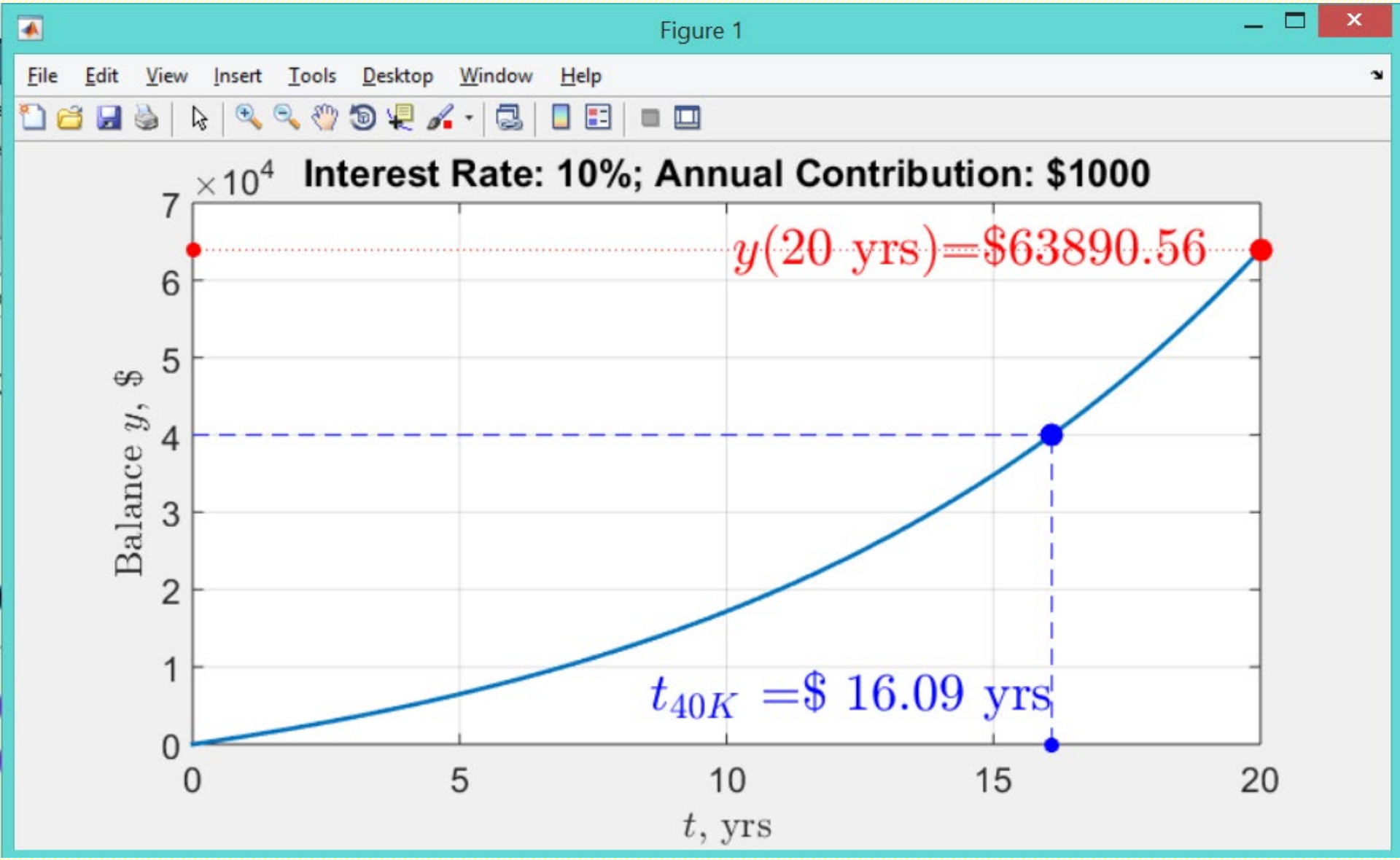
More advanced MATLAB script for the same task

```
%%
% Designed by Prof P.M.Trivailo (C)2020

tmax=20; r=10;
t=[0:0.01:1]*tmax; y=10000*exp(t/r)-10000;
plot(t,y,'LineWidth',2);
line('XData',t(end),'YData',y(end),'Marker','.','MarkerSize',36,'Color','r');
line('XData',0,'YData',y(end),'Marker','.','MarkerSize',24,'Color','r');
% line('XData',t(end),'YData',0,'Marker','.','MarkerSize',24,'Color','r');
line('XData',[0 t(end)],'YData',[1 1]*y(end),'LineStyle',':','Color','r');

str=sprintf('$$y$$(%g yrs)=\\$%8.2f',tmax,y(end));
text('String',str,'Interpreter','LaTeX',...
    'Position',[t(end)*0.95,y(end)],'FontSize',24,'Color','r',...
    'HorizontalAlignment','right',...
    'VerticalAlignment','middle');
grid on; xlabel('$t$, yrs','Interpreter','LaTeX');
ylabel('Balance $$y$$, \$','Interpreter','LaTeX');
set(gca,'FontSize',16);
str=sprintf('Interest Rate: %g%%; Annual Contribution: $1000',r)
title(str)
set(gcf,'Position',[214   334   834   428]);
```

To determine time for the balance to reach $40,000, interp1 MATLAB command can be used

```
% Continuation of the script
hold on;
yy=40000;
tt=interp1(y,t,yy);
line('XData',tt,'YData',yy,'Marker','.','MarkerSize',36,'Color','b');
line('XData',tt,'YData',0,'Marker','.','MarkerSize',24,'Color','b');
% line('XData',0,'YData',yy,'Marker','.','MarkerSize',24,'Color','b');
line('XData',[0 tt],'YData',[1 1]*yy,'LineStyle','--','Color','b');
line('XData',[1 1]*tt,'YData',[0 yy],'LineStyle','--','Color','b');
str2=sprintf('$$t_{%gK}$$ =\\$%8.2f yrs',yy/1000,tt);
text('String',str2,'Interpreter','LaTeX',...
    'Position',[tt,yy*0.05],'FontSize',24,'Color','b',...
    'HorizontalAlignment','right',...
    'VerticalAlignment','bottom');
```

Output of the advanced MATLAB script



MATLAB:

SYMBOLIC SOLUTION

More advanced MATLAB script for the same task (symbolic solution)

```
%% Designed by Prof P.M.Trivailo
% Ref: https://au.mathworks.com/help/symbolic/solve-a-single-differential-equation.html
% We are solving this 1st order ODE:  dy/dt= r y + D    (*)
% where: r-interest rate; D-annual contribution, y0-initial balance
%---------------------------
% Enter specific example data:
r=0.1; D=1000; y0=0; N=20;
%---------------------------
% Create symbolic function y(t):
syms y(t)
%---------------------------
% Define the Eq (*) using == and represent differentiation using "diff"
ode = diff(y,t) == r*y + D      % dy/dt= r y + D
% ode(t) =
% diff(y(t), t) == y(t)/10 + 1000
%---------------------------
%Solve the equation  (*) using "dsolve".
ySol(t) = dsolve(ode)
% ySol(t) =
% C3*exp(t/10) - 10000
% The constant C3 appears because no initial condition was specified.
```

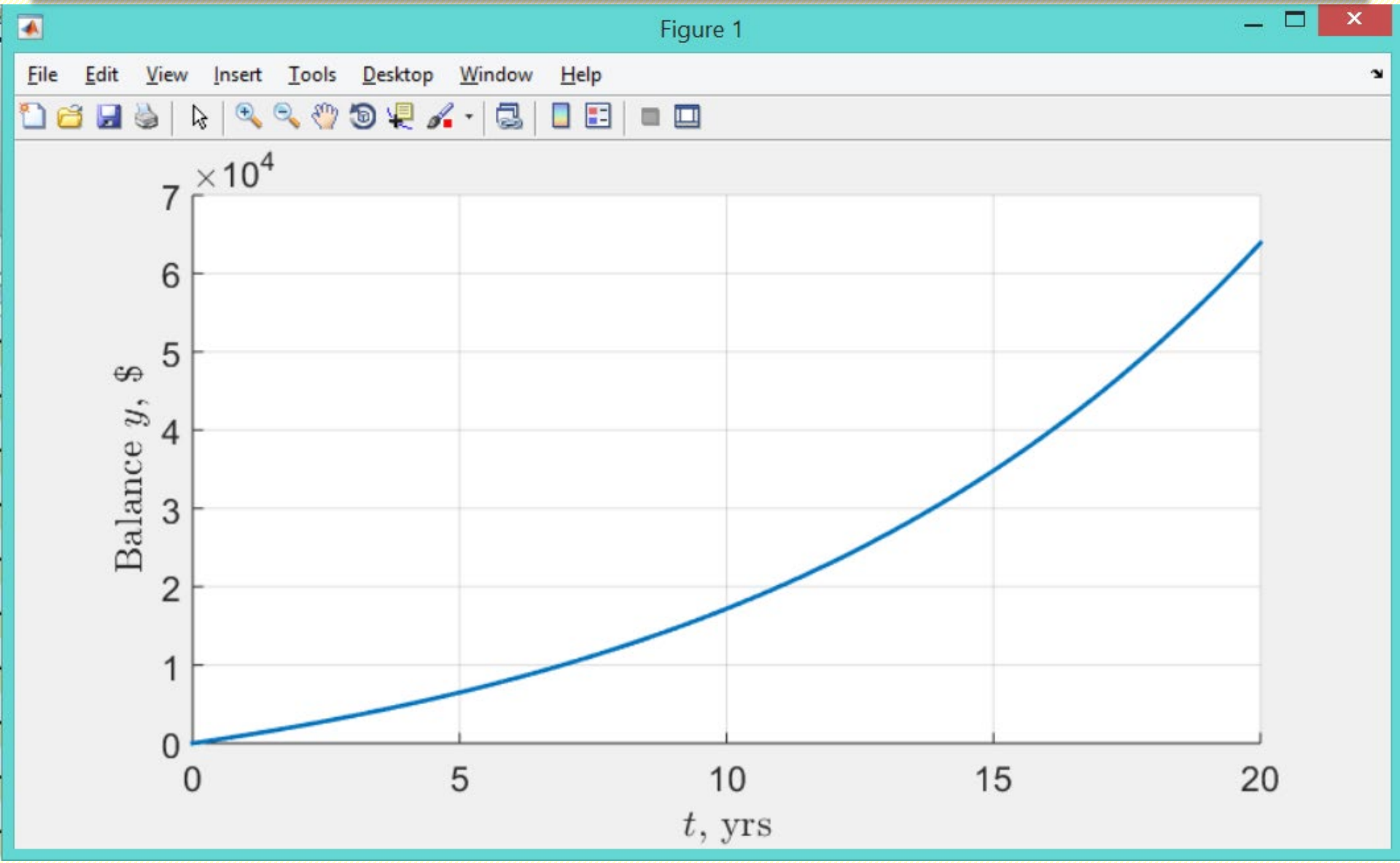
More advanced MATLAB script for the same task (symbolic solution)

```
%--------------------------
% Continuation of the script
%--------------------------
% Solve the equation with the initial condition y(0) == y0.
% The dsolve function finds a value of C3 that satisfies the condition.
cond = y(0) == y0;
ySol(t) = dsolve(ode,cond)
% ySol(t) =
% 10000*exp(t/10) - 10000

%--------------------------
% Determine balance after 20 years
eval(ySol(N))
% ans =
%   6.3891e+04

%--------------------------
% Better to use formatted print:
sprintf('Balance y(%i yrs) = $%8.2f',N,eval(ySol(N)))
% ans =
% Balance y(20 yrs) = $63890.56
```

More advanced MATLAB script for the same task (symbolic solution)

```
%--------------------------
% Continuation of the script
%--------------------------
% Better to use formatted print:
sprintf('Balance y(%i yrs) = $%8.2f',N,eval(ySol(N)))
% ans =
% Balance y(20 yrs) = $63890.56
hold on
tt=[0:0.01:1]*N;
plot(tt,ySol(tt),'LineWidth',2);
grid on; xlabel('$t$, yrs','Interpreter','LaTeX');
ylabel('Balance $$y$$, \$','Interpreter','LaTeX');
set(gca,'FontSize',16);
```

Output of the MATLAB script



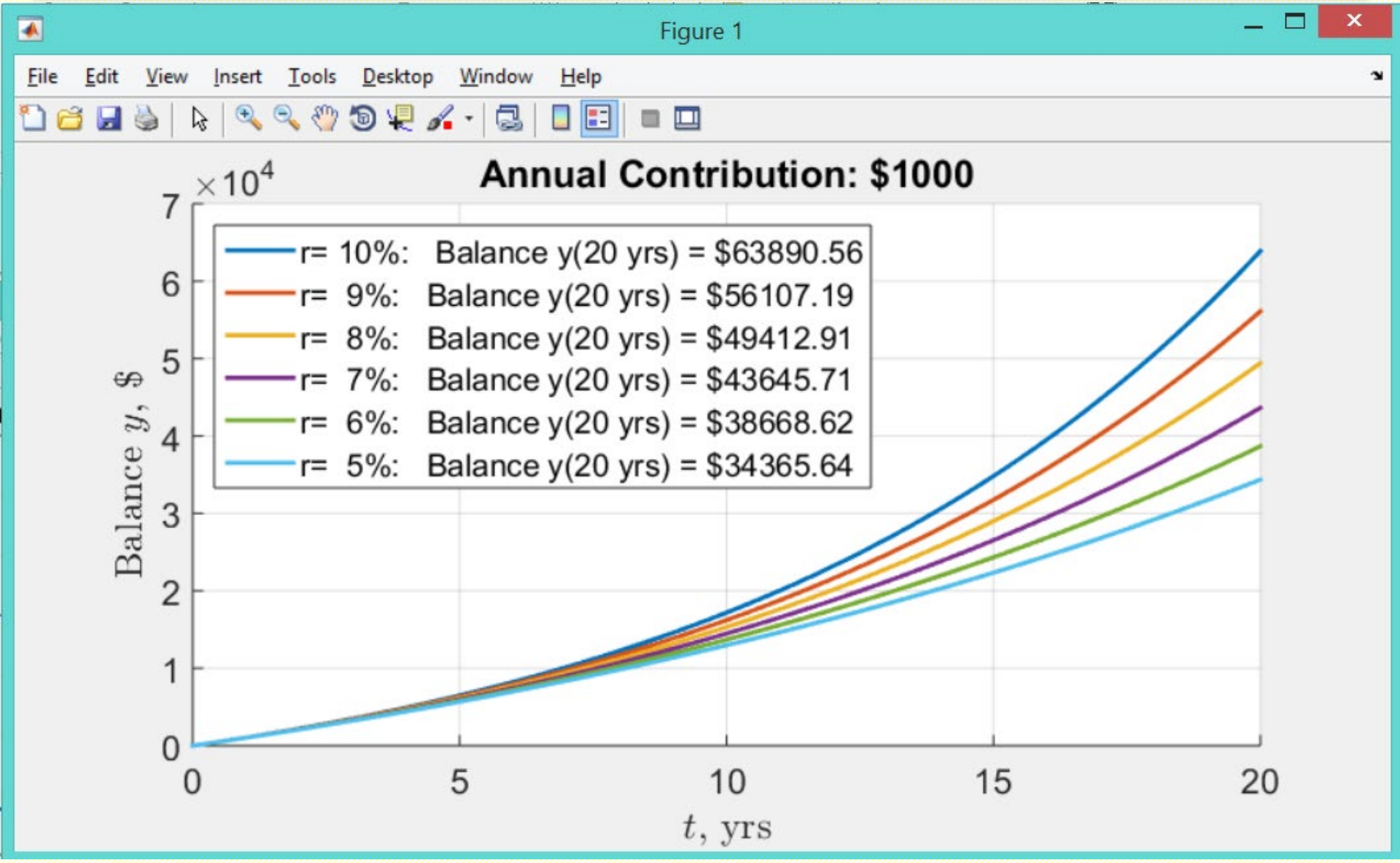
Even More advanced MATLAB script for more general task (symbolic solution)

```
%% Designed by Prof P.M.Trivailo © 2020
% We are solving this 1st order ordinary differential equation:
% dy/dt= r y + D
%--------------------------
clear; % use this command wisely: it may destroy your previous sim. data
clc; close('all');
D=1000; % annual regular annual payments
y0=0; % initial balance of the account
N=20; % years
rr=[10:-1:5]; % RANGE of the interest rates in % is simulated!!!
tt=[0:0.01:1]*N;
str=[];
siz=length(rr);
%--------------------------
% Scripts continues on the next slide …
```

Even More advanced MATLAB script for more general task (symbolic solution)

```
syms y(t)
for i=1:siz,
    r=rr(i)/100;
    ode = diff(y,t) == r*y + D      % dy/dt= r y + D
    cond = y(0) == y0;
    ySol(t) = dsolve(ode,cond)
    ySolN=eval(ySol(N));
    strC=sprintf('r=%3i%%:   Balance y(%i yrs) = $%8.2f',rr(i),N,ySolN);
    if i==siz,
        str=[str,'''',strC,''''];
    else
        str=[str,'''',strC, ''''', ','];
    end
    hold on
    plot(tt,ySol(tt),'LineWidth',2);
end
grid on; xlabel('$t$, yrs','Interpreter','LaTeX');
ylabel('Balance $$y$$, \$','Interpreter','LaTeX');
str3=sprintf('Annual Contribution: $%g',D);  title(str3)
set(gca,'FontSize',16); set(gcf,'Position',[214   334   834   428]);
eval(['lg=legend(',str,')']); set(lg,'Location','NorthWest');
```

Output of the MATLAB script



CASE STUDY:
POND POLLUTION
THEORY

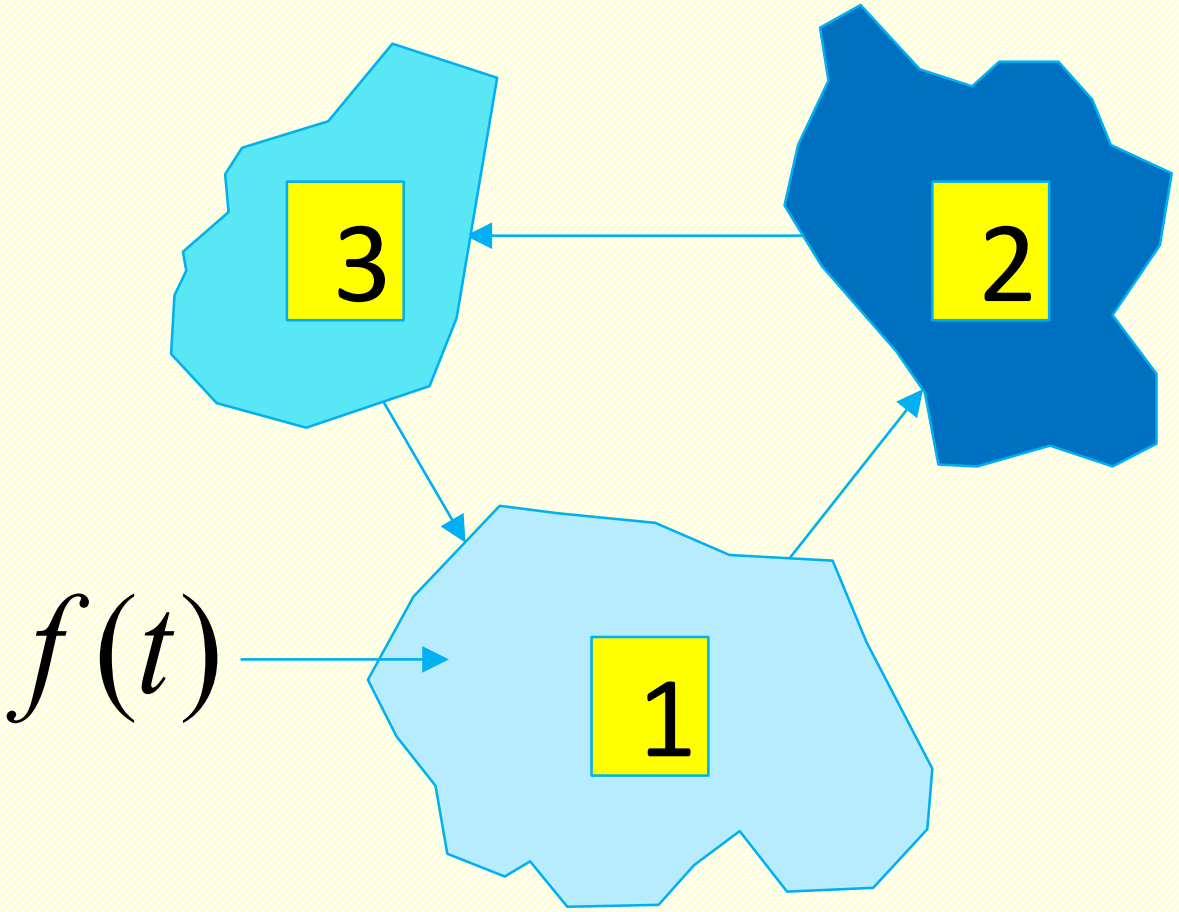
SYSTEM OF

FIRST ORDER

DIFFERENTIAL EQUATIONS:

POND POLLUTION EXAMPLE

Consider three ponds connected by streams.
The first pond has a pollution source, which spreads via the connecting streams to the other ponds.
The plan is to determine the amount of pollutant in each pond.



Three ponds 1, 2, 3 of volumes V_1, V_2, V_3 connected by streams. The pollution source *f(t)* is in pond 1.

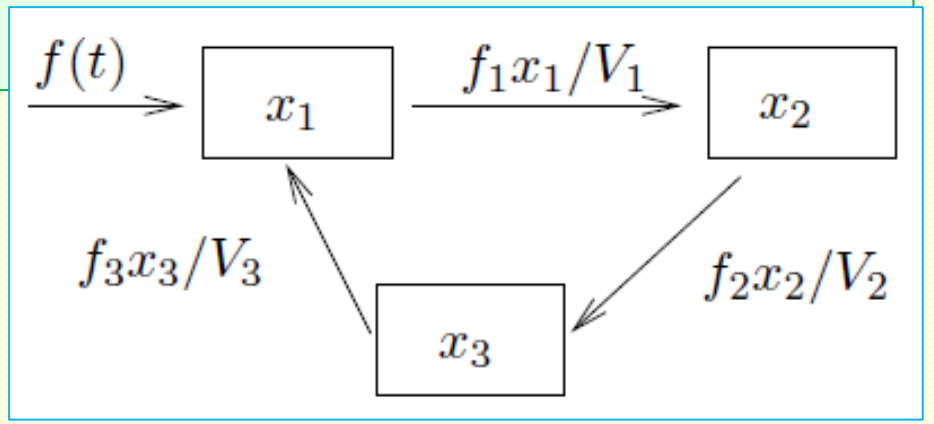
Symbol *f(t)* is the pollutant flow rate into pond 1 (kg/h).

- Symbols f_1, f_2, f_3 denote the pollutant flow rates out of ponds 1, 2, 3, respectively (m^3/h).
- It is assumed that the pollutant is well-mixed in each pond.
- The three ponds have volumes V_1, V_2, V_3 (m^3), which remain constant.
- Symbols $x_1(t), x_2(t), x_3(t)$ denote the amount (kg) of pollutant in ponds 1, 2, 3, respectively.

The pollutant flux is the flow rate times the pollutant concentration, e.g., pond 1 is emptied with flux f_1 times $x_1(t)/V_1$.

A compartment analysis is summarized in the following diagram.

The diagram plus compartment analysis gives the following differential equations:



$$\frac{dx_1(t)}{dt} = \frac{f_3}{V_3}x_3(t) - \frac{f_1}{V_1}x_1(t) + f(t)$$

$$\frac{dx_2(t)}{dt} = \frac{f_1}{V_1}x_1(t) - \frac{f_2}{V_2}x_2(t)$$

$$\frac{dx_3(t)}{dt} = \frac{f_2}{V_2}x_2(t) - \frac{f_3}{V_3}x_3(t)$$

For a specific numerical example, take $f_i/V_i = 0.03$, $1 \le i \le 3$, and let $f(t) = 3$ kg/h for the first 48 hours, thereafter $f(t) = 0$.

We expect due to uniform mixing that after a long time there will be 3*48 = 144 kg of pollutant uniformly deposited, which is 48 kg per pond.

Initially,
$x_1(0) = x_2(0) = x_3(0) = 0$,
if the ponds were pristine.

The specialized problem for the first 48 hours is solved numerically, using "ode45" MATLAB procedure.

$$\frac{dx_1(t)}{dt} = 0.03\,x_3(t) - 0.03\,x_1(t) + 3$$

$$\frac{dx_2(t)}{dt} = 0.03\,x_1(t) - 0.03\,x_2(t)$$

$$\frac{dx_3(t)}{dt} = 0.03\,x_2(t) - 0.03\,x_3(t)$$

MATLAB:

SYMBOLIC SOLUTION

IT INVOLVES VERY COMPLEX EXPRESSIONS!
HOWEVER, DO NOT PANICK, PLEASE: THIS TECHNIQUE IS RATHER OPTIONAL IN THIS COURSE, AND IS TAKEN AS A CONTRACT TECHNIQUE FOR THE "ODE" COMPULSORY TECHNIQUE, DEMONSTRATED LATER

Advanced MATLAB script for the task (symbolic solution)

```
%% POLLUTED PONDS EXAMPLE: SYMBOLIC SOLUTION
% Designed by Prof P.M.Trivailo (C) 2020
%--------------------------------------
clear; close all; clc;
% Define matrices and the matrix equation.
syms x1(t) x2(t) x3(t)
r=[-1  0  1;...
    1 -1  0;...
    0  1 -1]*0.03;
D=[3; 0; 0];
X = [x1; x2; x3];

odes = diff(X) == r*X + D
% odes(t) =
%
%  diff(x1(t), t) == (3*x3(t))/100 - (3*x1(t))/100 + 3
%      diff(x2(t), t) == (3*x1(t))/100 - (3*x2(t))/100
%      diff(x3(t), t) == (3*x2(t))/100 - (3*x3(t))/100
```

Advanced MATLAB script (symbolic solution) - Continued

```
% Solve the matrix equation using "dsolve". Simplify the solution by
using the simplify function.
[x1Sol(t), x2Sol(t), x3Sol(t)] = dsolve(odes);
x1Sol(t) = simplify(x1Sol(t))
x2Sol(t) = simplify(x2Sol(t))
x3Sol(t) = simplify(x3Sol(t))

% x1Sol(t) =
% C23 + t - (C24*exp(-(9*t)/200)*cos((3*3^(1/2)*t)/200))/2 + (C25*exp(-
(9*t)/200)*sin((3*3^(1/2)*t)/200))/2 + (3^(1/2)*C25*exp(-
(9*t)/200)*cos((3*3^(1/2)*t)/200))/2 + (3^(1/2)*C24*exp(-
(9*t)/200)*sin((3*3^(1/2)*t)/200))/2 + 100/3
%
% x2Sol(t) =
% C23 + t - (C24*exp(-(9*t)/200)*cos((3*3^(1/2)*t)/200))/2 + (C25*exp(-
(9*t)/200)*sin((3*3^(1/2)*t)/200))/2 - (3^(1/2)*C25*exp(-
(9*t)/200)*cos((3*3^(1/2)*t)/200))/2 - (3^(1/2)*C24*exp(-
(9*t)/200)*sin((3*3^(1/2)*t)/200))/2
%
% x3Sol(t) =
% C23 + t + C24*exp(-(9*t)/200)*cos((3*3^(1/2)*t)/200) - C25*exp(-
(9*t)/200)*sin((3*3^(1/2)*t)/200) - 100/3
```

Note: symbolic solutions are given by very long expressions, which can not fit one line. Therefore, when I "cut-and-paste" them from MATLAB, being a one-line each, and inserting in the script as a commented one-line strings, they are taking a few lines on this PPT slide.

Advanced MATLAB script (symbolic solution) - Continued

```
% The constants C23, C24 and C25 appear because no conditions are specified.
% Solve the system with the initial conditions x1(0)=0, x2(0)=0 and x3(0)=0.
% When specifying equations in matrix form, you must specify initial conditions in matrix form.
% "dsolve" finds values for the constants that satisfy these conditions.

C = X(0) == [0; 0; 0];
[x1Sol(t), x2Sol(t), x3Sol(t)] = dsolve(odes,C)

% x1Sol(t) =
% t - (100*cos((3*3^(1/2)*t)/200))/(3*exp(t)^(9/200)) +
(100*3^(1/2)*sin((3*3^(1/2)*t)/200))/(9*exp(t)^(9/200)) + 100/3
%
% x2Sol(t) =
% t - (200*3^(1/2)*sin((3*3^(1/2)*t)/200))/(9*exp(t)^(9/200))
%
% x3Sol(t) =
% t + (100*cos((3*3^(1/2)*t)/200))/(3*exp(t)^(9/200)) +
(100*3^(1/2)*sin((3*3^(1/2)*t)/200))/(9*exp(t)^(9/200)) - 100/3
```

Initial conditions are specified here as matrix.

Note: All solutions in symbolic form are found by MATLAB!!! Exciting !!!

Advanced MATLAB script: Commands only, without annotations

```
%% POLLUTED PONDS EXAMPLE: SYMBOLIC SOLUTION
% Designed by Prof P.M.Trivailo (C) 2020
%--------------------------------------
% Define matrices and the matrix equation.
syms x1(t) x2(t) x3(t)
r=[-1  0  1;  1 -1  0;  0  1 -1]*0.03;
D=[3; 0; 0]; X = [x1; x2; x3];
odes = diff(X) == r*X + D

% Solve the matrix equation using "dsolve".
[x1Sol(t), x2Sol(t), x3Sol(t)] = dsolve(odes);

% Solve the system with zero initial conditions.
C = X(0) == [0; 0; 0];
[x1Sol(t), x2Sol(t), x3Sol(t)] = dsolve(odes,C)
```

Symbolic solutions, derived by MATLAB

$$x_1(t) = t - \frac{100\cos\left(\frac{3\sqrt{3}\,t}{200}\right)}{3\left(e^t\right)^{\frac{9}{200}}} + \frac{100\sqrt{3}\sin\left(\frac{3\sqrt{3}\,t}{200}\right)}{9\left(e^t\right)^{\frac{9}{200}}} + \frac{100}{3}$$

$$x_2(t) = t - \frac{200\sqrt{3}\sin\left(\frac{3\sqrt{3}\,t}{200}\right)}{9\left(e^t\right)^{\frac{9}{200}}}$$

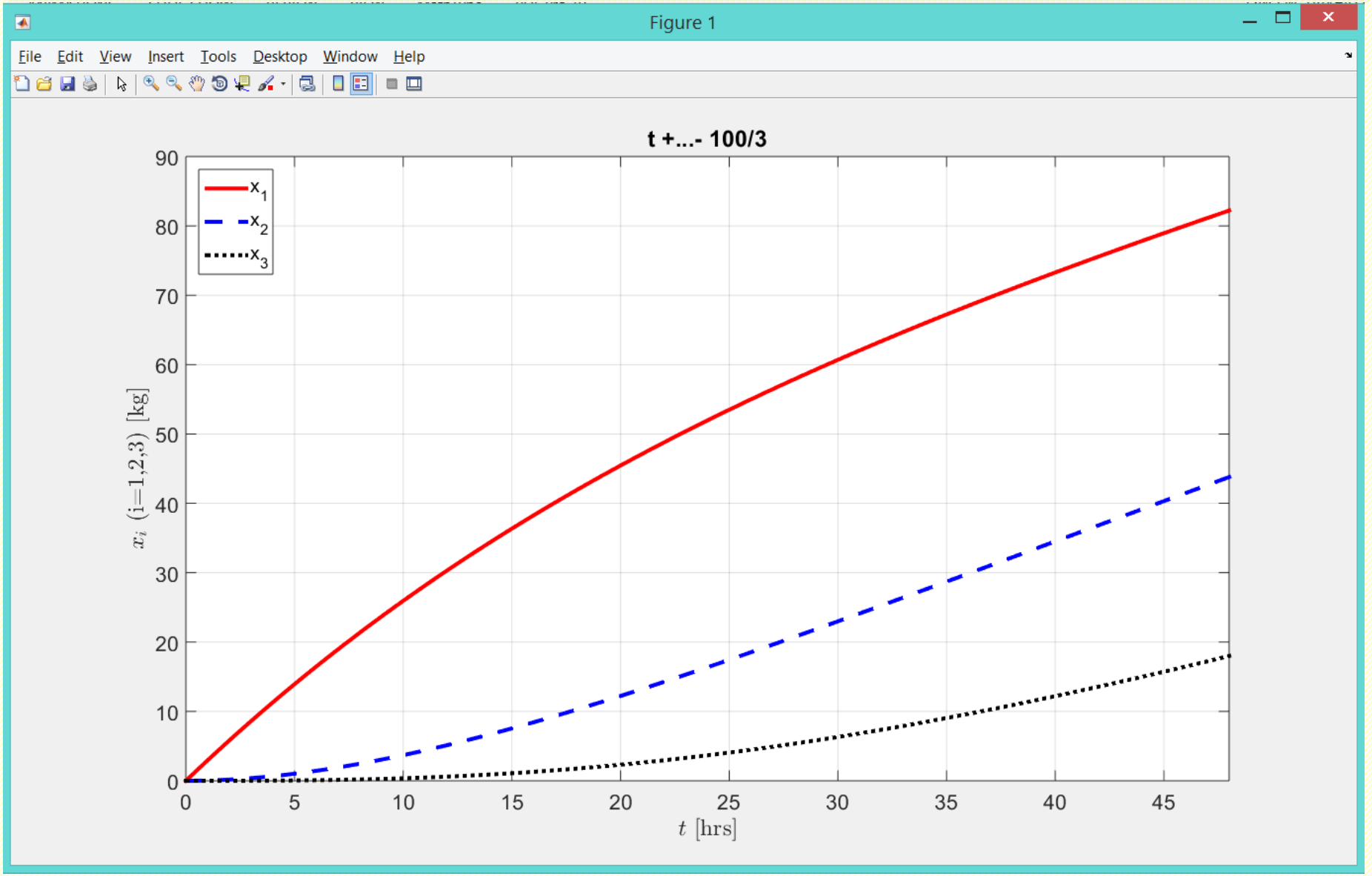
QUITE IMPRESSIVE! IS NOT IT?

$$x_3(t) = t + \frac{100\cos\left(\frac{3\sqrt{3}\,t}{200}\right)}{3\left(e^t\right)^{\frac{9}{200}}} + \frac{100\sqrt{3}\sin\left(\frac{3\sqrt{3}\,t}{200}\right)}{9\left(e^t\right)^{\frac{9}{200}}} - \frac{100}{3}$$

Advanced MATLAB script (symbolic solution) - Continued

```
% Continuation of the script. It is split into parts for the PPT presentation only.
%-------------------------------------------------------------------
% Visualize the solution using fplot. Before R2016a, use ezplot instead.


%fplot(x1Sol)
h1=ezplot(x1Sol,[0 48 0 90]); set(h1,'LineStyle','-','Color','r');
hold on
%fplot(x2Sol)
h2=ezplot(x2Sol,[0 48 0 90]); set(h2,'LineStyle','--','Color','b');
%fplot(x3Sol)
h3=ezplot(x3Sol,[0 48 0 90]); set(h3,'LineStyle',':','Color','k');
set([h1, h2, h3],'LineWidth',3);
grid on
legend('x_1','x_2','x_3','Location','NorthWest');
xlabel('$t$ [hrs]','Interpreter','LaTeX');
ylabel('$x_i$ (i=1,2,3) [kg]','Interpreter','LaTeX');
set(gca,'FontSize',16);  set(gcf,'Position',[488 -90 1361 852]);
```

Output of the MATLAB script



!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH THE SAME EXAMPLE

MATLAB:

NUMERICAL SOLUTION

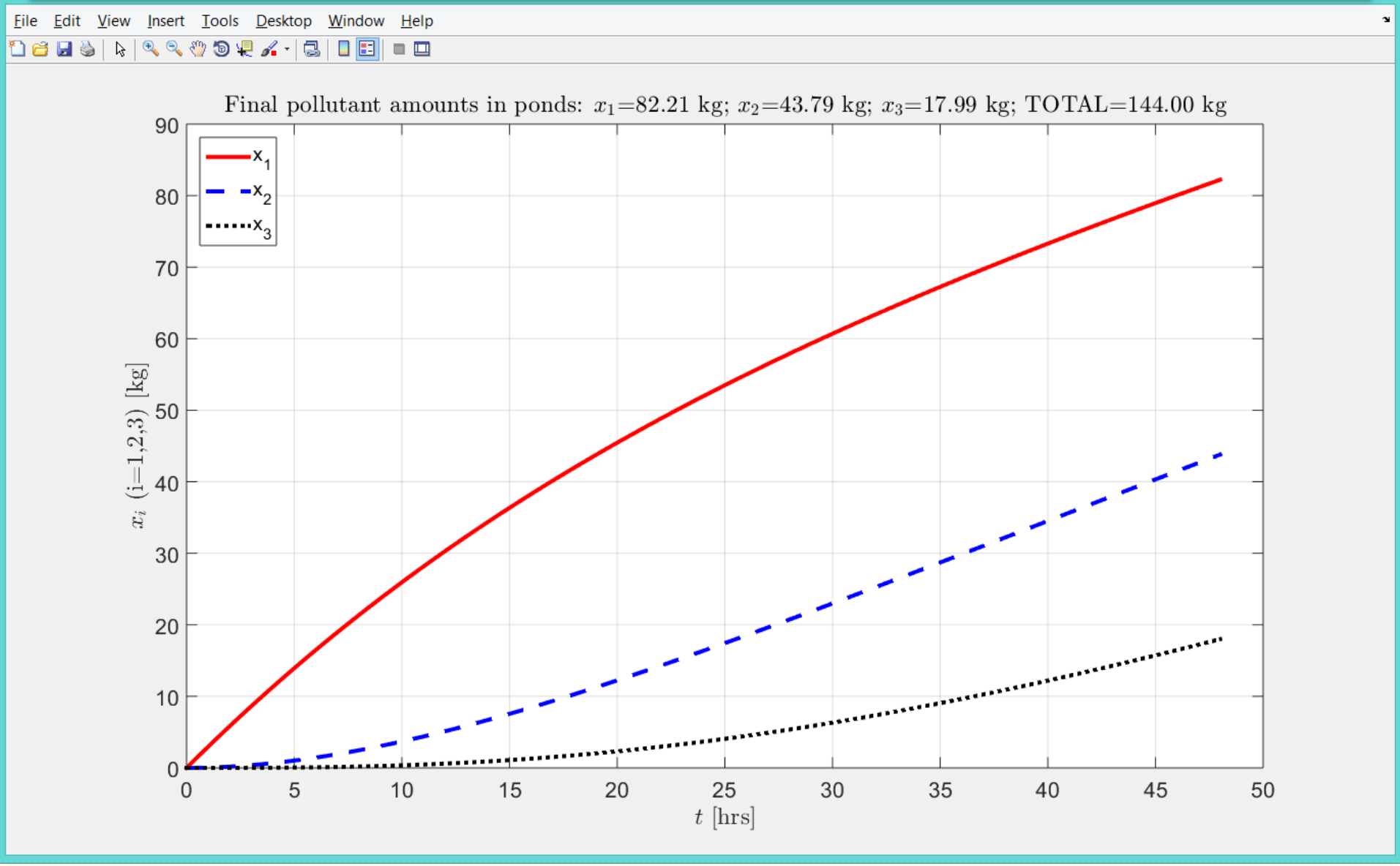
MATLAB script for numerical solution of the problem

```
%% POLLUTED PONDS EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
%--------------------------------------
clear; close all; clc;
r=[-1  0  1;...
    1 -1  0;...
    0  1 -1]*0.03;
D=[3; 0; 0];
tmax=24*2; t=[0:0.01:1]*tmax; % hrs
mass_xdot_anonymous = @(t, x) (r*x+D);
[tt,zz]=ode45(mass_xdot_anonymous,t,[0;0;0]);
plot(tt,zz(:,1),'r-',tt,zz(:,2),'b--',tt,zz(:,3),'k:','LineWidth',3);
legend('x_1','x_2','x_3','Location','NorthWest');
xlabel('$t$ [hrs]','Interpreter','LaTeX');
ylabel('$x_i$ (i=1,2,3) [kg]','Interpreter','LaTeX');
grid on
str1='Final pollutant amounts in ponds: ';
str=sprintf('%s$$x_1$$=%5.2f kg; $$x_2$$=%5.2f kg; $$x_3$$=%5.2f kg; TOTAL=%6.2f kg',str1,zz(end,:),sum(zz(end,:)));
title(str,'Interpreter','LaTeX')
set(gca,'FontSize',16);  set(gcf,'Position',[488 -90 1361 852]);
```

CORE or "HEART" OF THE SCRIPT !!!

"DECORATIONS" commands

Output of the MATLAB script



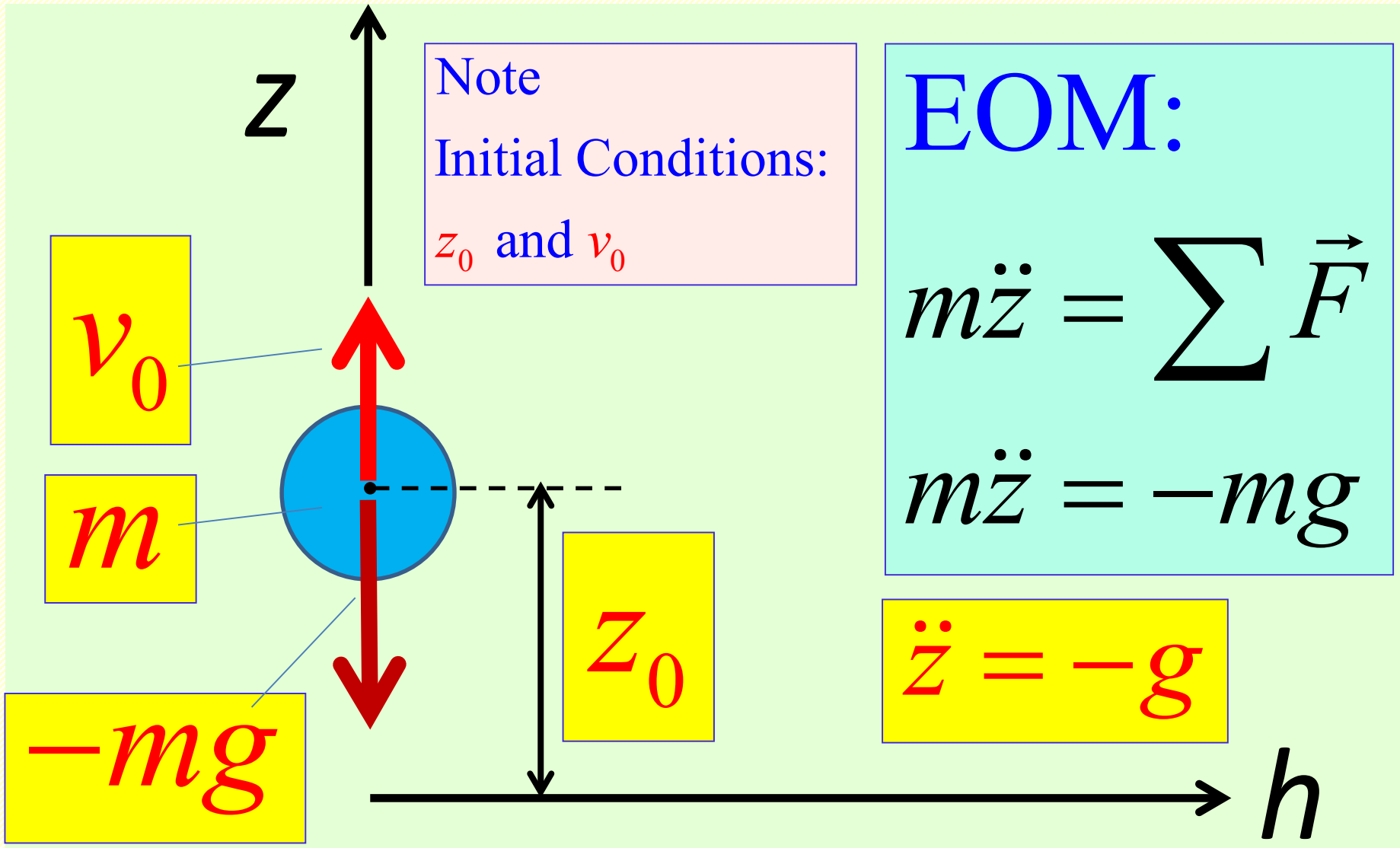
CASE STUDY: FALLING MASS

THEORY

SOLVING RESPONSES:

MATLAB and ODExxx

Modelling a Falling Mass



Note

Initial Conditions:

z_0 and v_0

EOM:

$$m\ddot{z} = \sum \vec{F}$$

$$m\ddot{z} = -mg$$

$$\ddot{z} = -g$$

SOLUTION: Analytical Method

Integration of this differential equation allows analytical solution:

$$\frac{d^2z}{dt^2} = -g; \quad \Rightarrow \quad \frac{dz}{dt} = -gt + C_1; \quad \Rightarrow \quad z = -\frac{gt^2}{2} + C_1 t + C_2$$

Application of Initial Conditions enables us to find the values of constants C_1 and C_2 :

$$z\big|_{t=0} = z_0; \quad \Rightarrow \quad z_0 = C_2$$

$$\left.\frac{dz}{dt}\right|_{t=0} = v_0; \quad \Rightarrow \quad v_0 = C_1$$

Analytical exact solution: $z(t) = z_0 + v_0 t - \frac{gt^2}{2}$

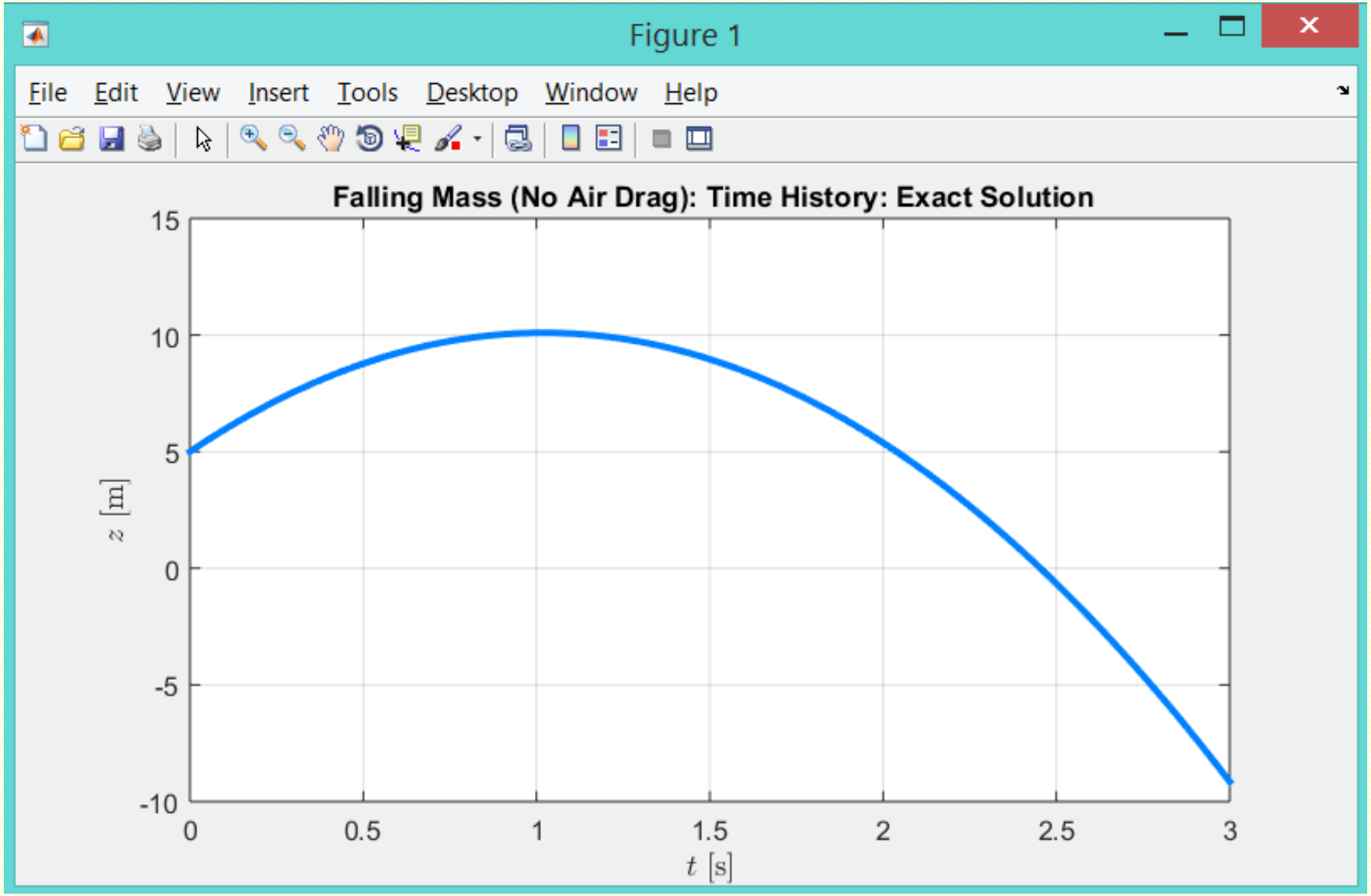
SOLUTION: Analytical Method

MATLAB SCRIPT (plotting results)

```
%% FALLING MASS (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2019
%-----------------------------------
        %                ^ z
% Initial Conditions:    |
v0=10;   % m/s           |
z0=5;    % m             |
tmax=3;  % s             +--                > t
t=[0:0.01:1]*tmax;
zz=z0+v0*t-g*t.^2/2;  % Note: pseudo-square operation !!!
plot(tt,zz,'LineWidth',3,'Color',[0 0.5 1]);
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$z$ [m]','Interpreter','LaTeX');
title('\bf Falling Mass (No Air Drag): Time History: Exact Solution');
set(gca, 'FontSize',12);
```

"OENG1116_falling_mass_NO_air_drag_EXACT.m" file

Output of the MATLAB script



!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

STATE-SPACE FORM
OF THE EQUATION OF MOTION

Modelling a Falling Mass: EOM

$$\ddot{z} = -g$$

Let us introduce new variables:

$$x_1 = z$$

$$x_2 = \dot{z}$$

then

$$\begin{cases} \dot{x}_1 = x_2 \end{cases}$$

Modelling a Falling Mass: EOM

$$\ddot{z} = -g$$

Also, with new variables:

$$x_1 = z$$

$$x_2 = \dot{z}$$

$$\begin{cases} \\ \dot{x}_2 = -g \end{cases}$$

EOM in the STATE-SPACE FORM

$$\ddot{z} = -g$$

Let us introduce new variables:

$$x_1 = z$$

$$x_2 = \dot{z}$$

and

then

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g \end{cases}$$

!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH FALLING MASS EXAMPLE

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (no annotations)

```
%% FALLING MASS (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
%----------------------------------------
        %                ^ z
% Initial Conditions:    |
v0=10;   % m/s           |
z0=5;    % m             |
tmax=3; % s              +--------------> h
t=[0:0.01:1]*tmax;
f_mass_xdot_anonymous = @(t, x) ([x(2); -9.81]);
[tt,zz]=ode45(f_mass_xdot_anonymous,t,[z0; v0]);
plot(tt,zz(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$z$ [m]','Interpreter','LaTeX');
title('\bf Falling Mass (No Air Drag): Time History');
set(gca, 'FontSize',12);
```

"OENG1116_falling_mass_NO_air_drag_ANONYMOUS.m" file

SOLUTION: Numerical (!!!) Method

"OENG1116_falling_mass_NO_air_drag_ANONYMOUS.m" file

```
%% FALLING MASS (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
%-----------------------------------
        %                  ^ z
% Initial Conditions:     |
v0=10;   % m/s            |
z0=5;    % m              |
tmax=3;  % s              +--------------> h
t=[0:0.01:1]*tmax;
f_mass_xdot_anonymous = @(t, x) ([x(2); -9.81]);
[tt,zz]=ode45(f_mass_xdot_anonymous,t,[z0; v0]);
plot(tt,zz(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$z$ [m]','Interpreter','LaTeX');
title('\bf Falling Mass (No Air Drag): Time
set(gca, 'FontSize',12);
```

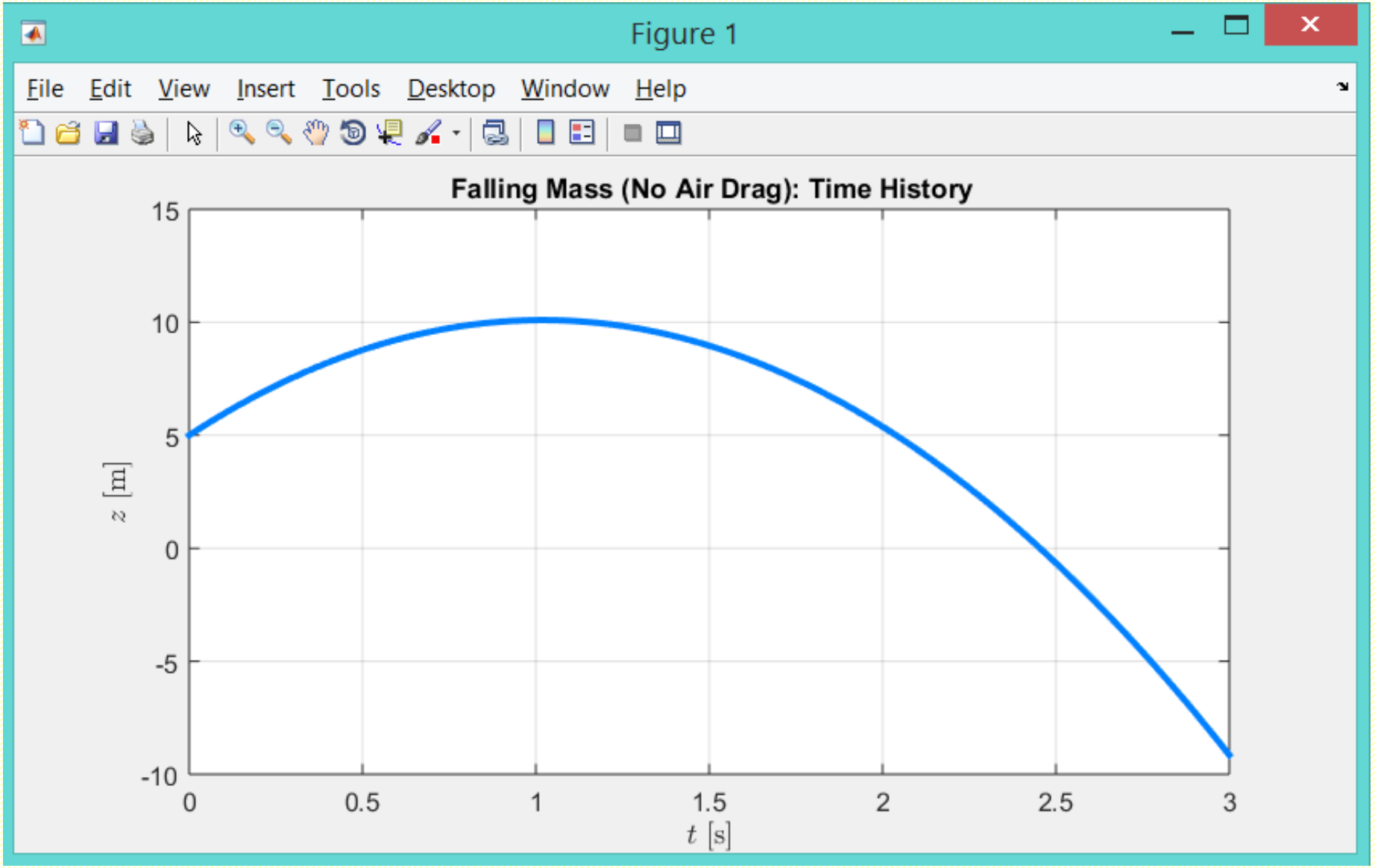
1. Input of the data

2. $$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g \end{cases}$$

3. Calling ode45 procedure !

4. Plotting results

Output of the MATLAB script



MATLAB SCRIPT: 2 files solution

OENG1116_falling_mass.m

Main Program

```
%% FALLING MASS (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C)
%-----------------------------------
        %                ^ z
% Initial Conditions:    |
v0=10;   % m/s           |
z0=5;    % m             |
tmax=3; % s              +--------------> x
t=[0:0.01:1]*tmax;
[tt,zz]=ode45('falling_mass_xdot',t,[z0; v0]);
plot(tt,zz(:,1),'LineWidth',3,'Color',[0 0 1]);
grid on; xlabel('t [s]'); ylabel('z [m]');
```

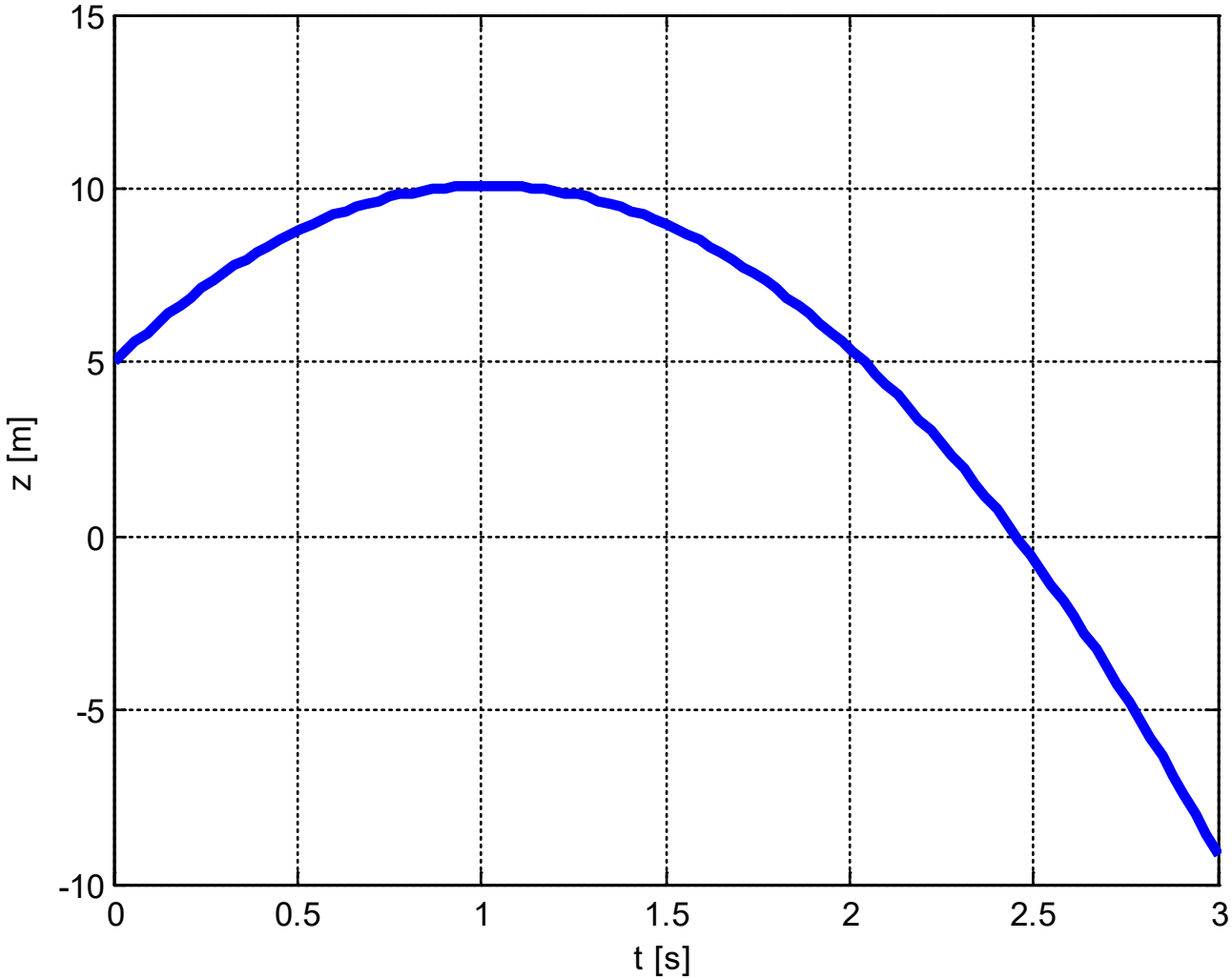
Must be the Same !!!!!!!!

falling_mass_xdot.m

'Xdot' Function

```
function [xdot]= falling_mass_xdot(t,x)

xdot(1,1)=x(2);
xdot(2,1)=-9.81;
```

RESULTS OF THE SIMULATION:



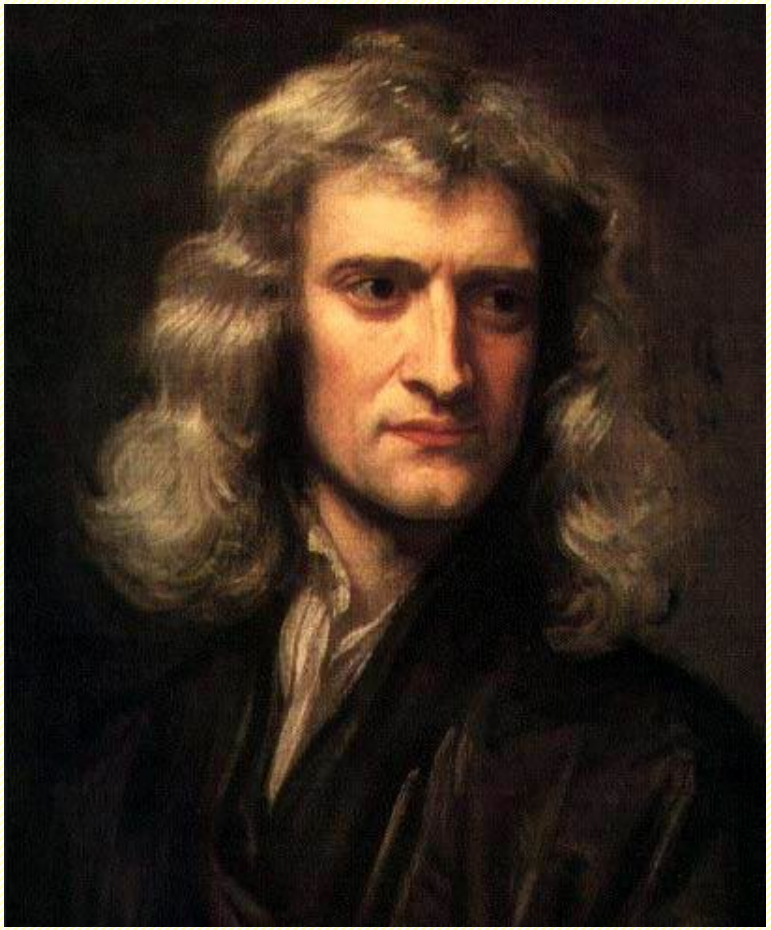
Discussion on the Script & Results

1. **NO DRAG !!!**
2. **Positive ‘z’ is UP**
3. Results (States) are in the ‘zz’ matrix.
4. Use ‘size(zz)’ to know the dimension.
5. ‘xdot’ Function **MUST** return a column vector!
6. Initial conditions are in the ‘column’ format.
7. Ask a question: WHEN the mass crosses the *z=0* level?

FUNDAMENTAL LAWS:

NEWTON's SECOND LAW EXAMPLE

ISAAC NEWTON



Sir **Isaac Newton** PRS (25 Dec. 1642 - 20 March 1727 [NS: 4 Jan. 1643 - 31 March 1727]) was an English physicist, mathematician, astronomer, natural philosopher, alchemist, and theologian, who has been ”considered by many to be the greatest and most influential scientist who ever lived.” His monograph *Philosophi Naturalis Principia Mathematica*, published in 1687, lays the foundations for most of classical mechanics. In this work, Newton described universal gravitation and the three laws of motion, which dominated the scientific view of the physical universe for the next three centuries.

Courtesy: Wikipedia https://en.wikipedia.org/wiki/Isaac_Newton

NEWTON'S LAWS of MOTION

Newton's First Law (also known as the Law of Inertia) states that an object at rest tends to stay at rest and that an object in uniform motion tends to stay in uniform motion unless acted upon by a net external force.

The meaning of this law is the existence of reference frames (called inertial frames) where objects not acted upon by forces move in uniform motion (in particular, they may be at rest).

Courtesy: Wikipedia https://en.wikipedia.org/wiki/Newton%27s_laws_of_motion

NEWTON'S LAWS of MOTION (Cont'd)

Newton's Second Law states that an applied force, ***F*** , on an object equals the rate of change of its momentum, ***p***, with time.

$$\sum \vec{\mathbf{F}} = \frac{d\vec{\mathbf{p}}}{dt} = \frac{d(m\vec{\mathbf{v}})}{dt}$$

$$\mathbf{F} = \frac{d\mathbf{p}}{dt} = \frac{d(m\mathbf{v})}{dt} = \mathbf{v}\frac{dm}{dt} + m\frac{d\mathbf{v}}{dt}$$

If applied to an object with constant mass

$$\frac{dm}{dt} = 0$$

the first term vanishes, and by substitution using the definition of acceleration, the equation can be written in the iconic form

$$\mathbf{F} = m\,\mathbf{a}$$

NEWTON'S LAWS of MOTION (Cont'd)

Newton's Third Law states that for every action there is an equal and opposite reaction. This means that any force exerted onto an object has a counterpart force that is exerted in the opposite direction back onto the first object.

MODELLING:

ONE-DOF SYSTEM EXAMPLE

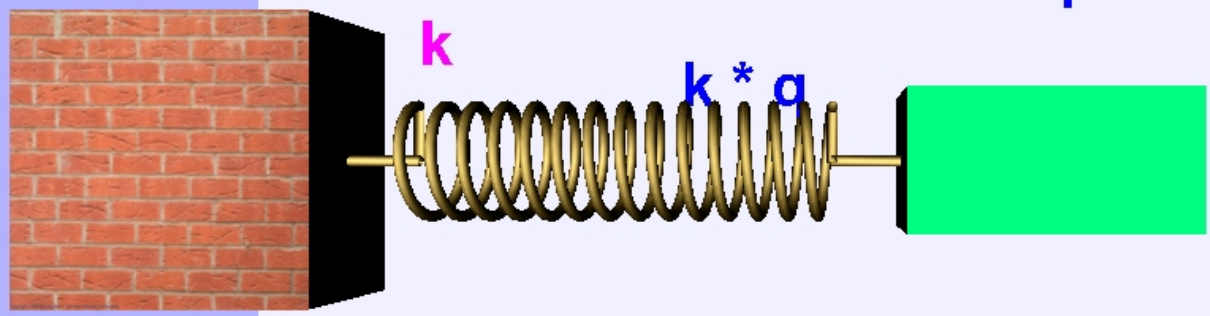
illustrated in Virtual Reality

One-DOF mass-spring system:

Derivation of EOM using Newton's Second Law & Free-Body Diagrams (FBDs).

1-DOF System: Static Equilibrium

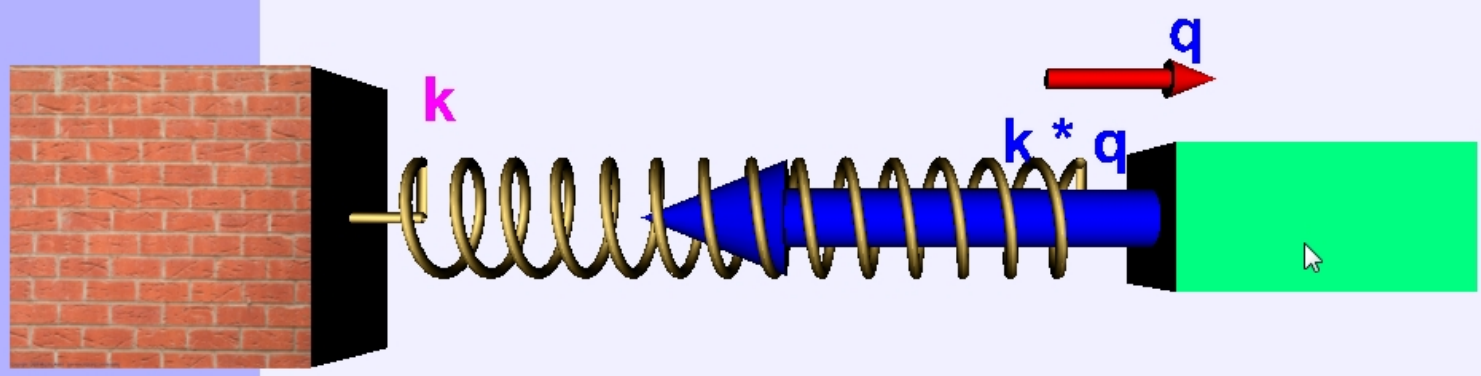
VR Model



RMIT, SAMME
Copyright P.M.Trivailo March-2013

To use sensors PLACE POINTER OVER Cortona3D

1-DOF System: Disturbed

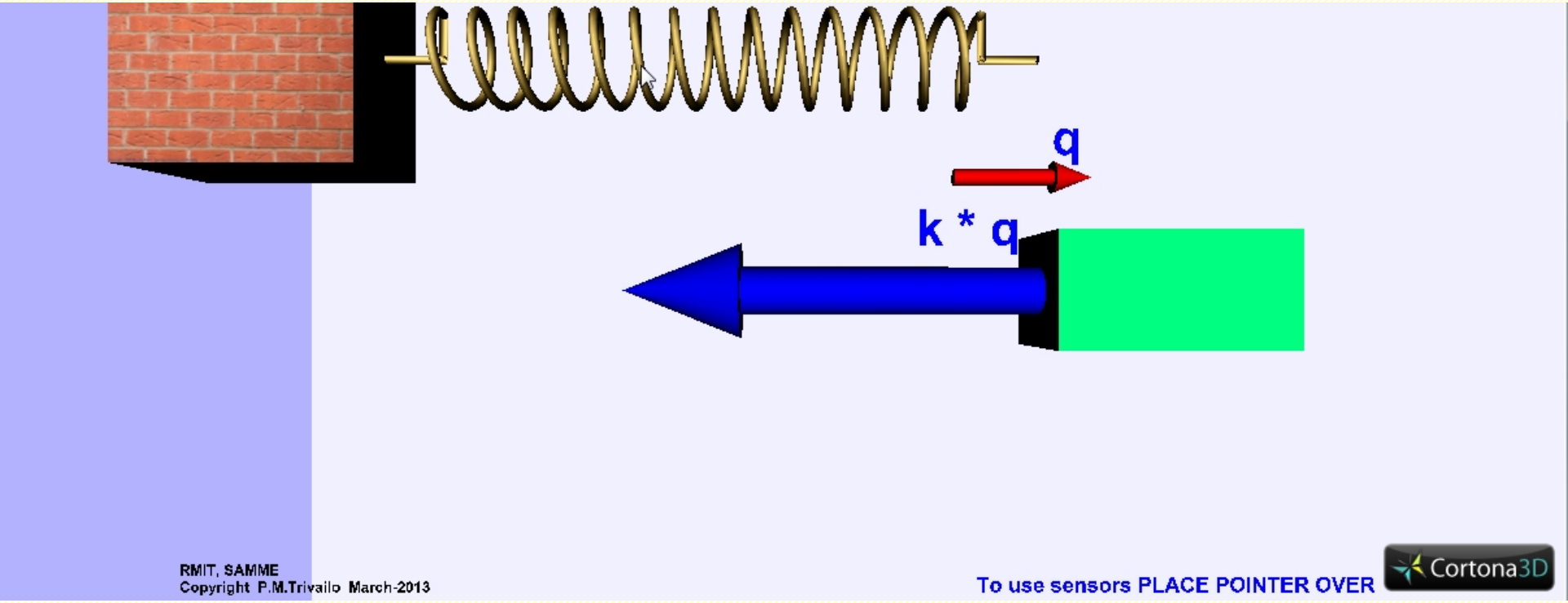


RMIT, SAMME
Copyright P.M.Trivailo March-2013

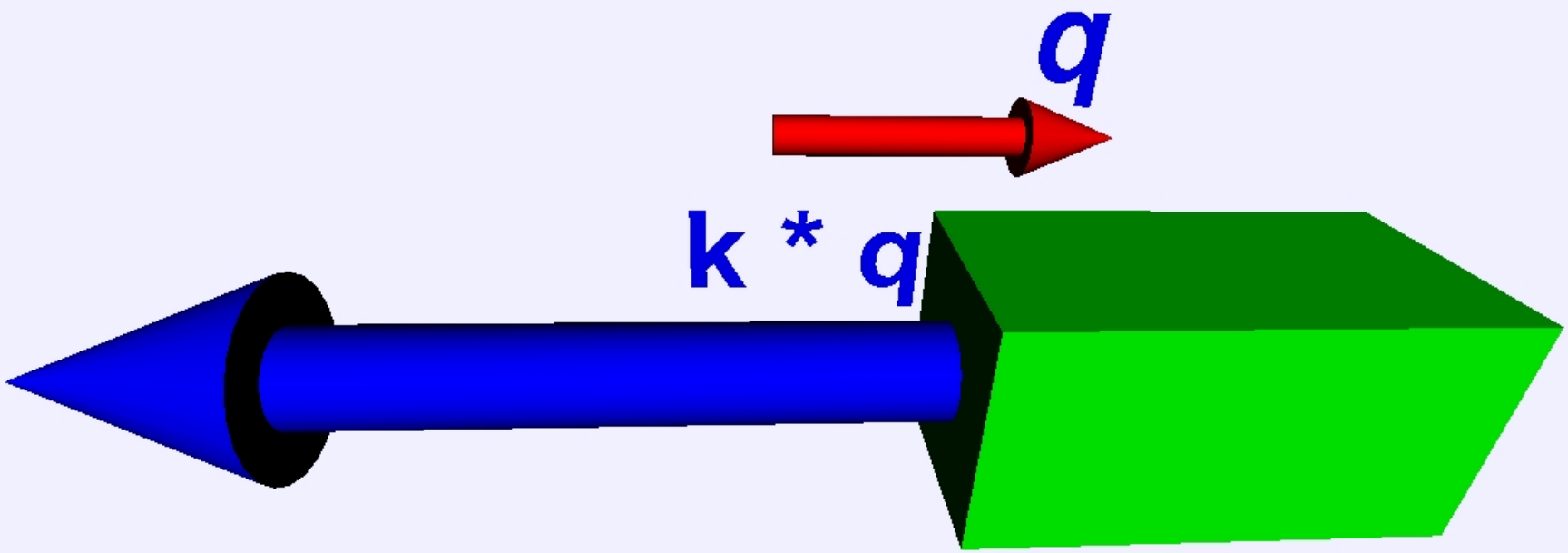
Press + Drag to CHANGE "q"

Cortona3D

1-DOF System: Constrains Removed



1-DOF System: Free-Body Diagram



$$m\mathbf{a} = \sum \mathbf{F} \quad \Rightarrow \quad m\ddot{q} = -kq$$

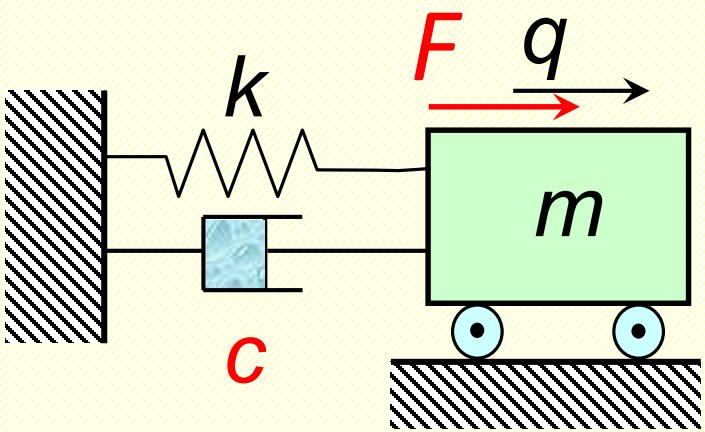
!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH MASS-SPRING EXAMPLE

1-DOF System: Free-Body Diagram



EXCITATION FORCE & DAMPING CAN ALSO BE ADDED.
RESULTANT Equation of Motion (EOM) IS:

$$m\ddot{q} + c\dot{q} + kq = F$$

LET US CONVERT IT IN THE STATE-SPACE FORM:

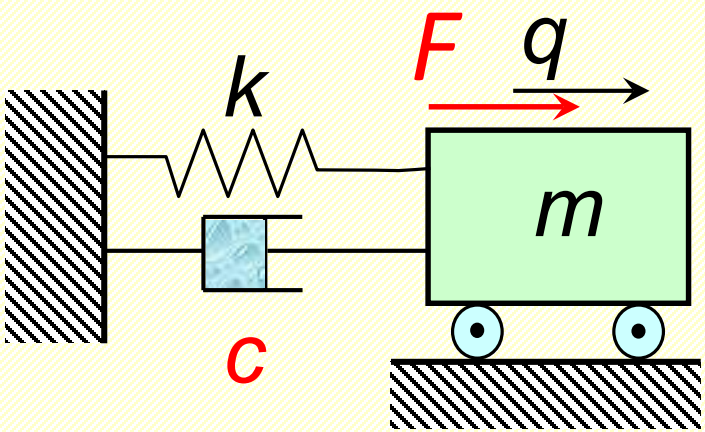
If $x_1 = q$ and $x_2 = \dot{q}$ then $\dot{x}_1 = x_2$.

Also, the EOM of the mass-damper-spring system can be written as:

$$\ddot{q} = -\frac{k}{m}q - \frac{c}{m}\dot{q} + \frac{F}{m} \quad \text{or} \quad \dot{x}_2 = -\frac{k}{m}x_1 - \frac{c}{m}x_2 + \frac{1}{m}F$$

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F$$

1-DOF System: Free-Body Diagram



EXCITATION FORCE & DAMPING CAN ALSO BE ADDED.
RESULTANT Equation of Motion (EOM) IS:

$$m\ddot{q} + c\dot{q} + kq = F$$

EOM IN THE STATE-SPACE MATRIX FORM:

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F$$

OR (vector format of the same):

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}, \quad \mathbf{x} = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} q \\ \dot{q} \end{Bmatrix}, \quad A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}$$

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (no annotations)

```
%% MASS-SPRING-DAMPER SYSTEM EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
%---------------------------------------
m=10; k=100; c=10;
% Initial Conditions:
q0=3;          % m
q_dot0=15;     % m/s
tmax=6;        % s
t=[0:0.01:1]*tmax;
A=[0 1; -k/m -c/m]; B=[0; 1/m]; F=0;
mck_xdot_anonymous = @(t, x) A*x+B*F;
[tt,xx]=ode45(mck_xdot_anonymous,t,[q0; q_dot0]);
plot(tt,xx(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$q$ [m]','Interpreter','LaTeX');
str1='\bf Free Vibration of Mass-Damper-Spring System: Time History ';
str2=sprintf('(m=%g kg; c=%g N*s/m; k=%g N/m)',m,c,k);
title([str1 str2]);
set(gca, 'FontSize',16); set(gcf,'Position',[240 -50 1200 750]);
```

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (with annotations)

```
%% MASS-SPRING-DAMPER SYSTEM EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
%-----------------------------------
m=10; k=100; c=10;
% Initial Conditions:
q0=3;           % m
q_dot0=15;      % m/s
tmax=6;         % s
t=[0:0.01:1]*tmax;
A=[0 1; -k/m -c/m]; B=[0; 1/m]; F=0;
mck_xdot_anonymous = @(t, x) A*x+B*F;
[tt,xx]=ode45(mck_xdot_anonymous,t,[q0; q_dot0]);
plot(tt,xx(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$q$ [m]','Interpreter','LaTeX');
str1='\bf Free Vibration of Mass-Damper-Spring System: T
str2=sprintf('(m=%g kg;
title([str1 str2]);
set(gca, 'FontSize',16)
```

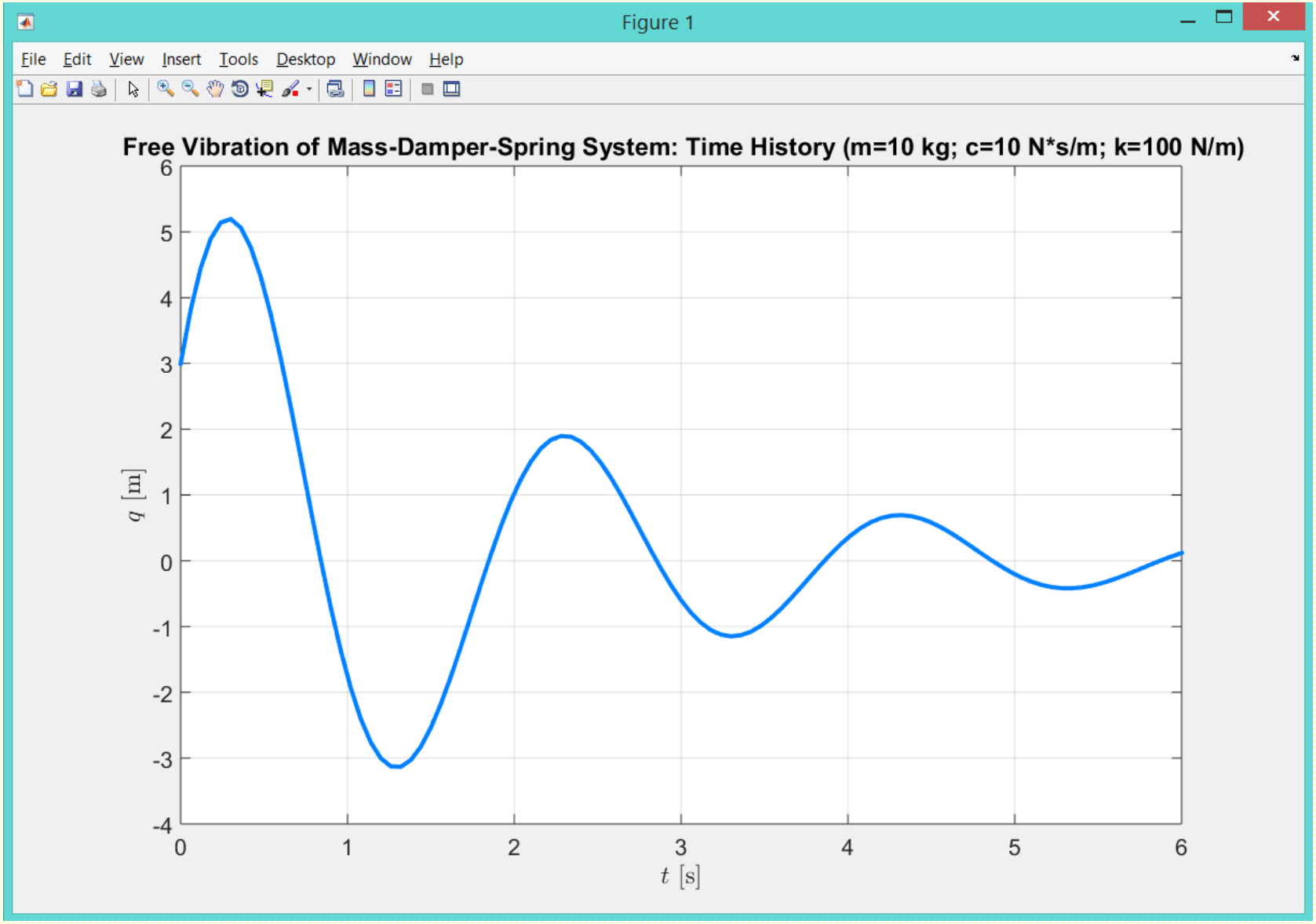
1. Input of the data

2. $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$

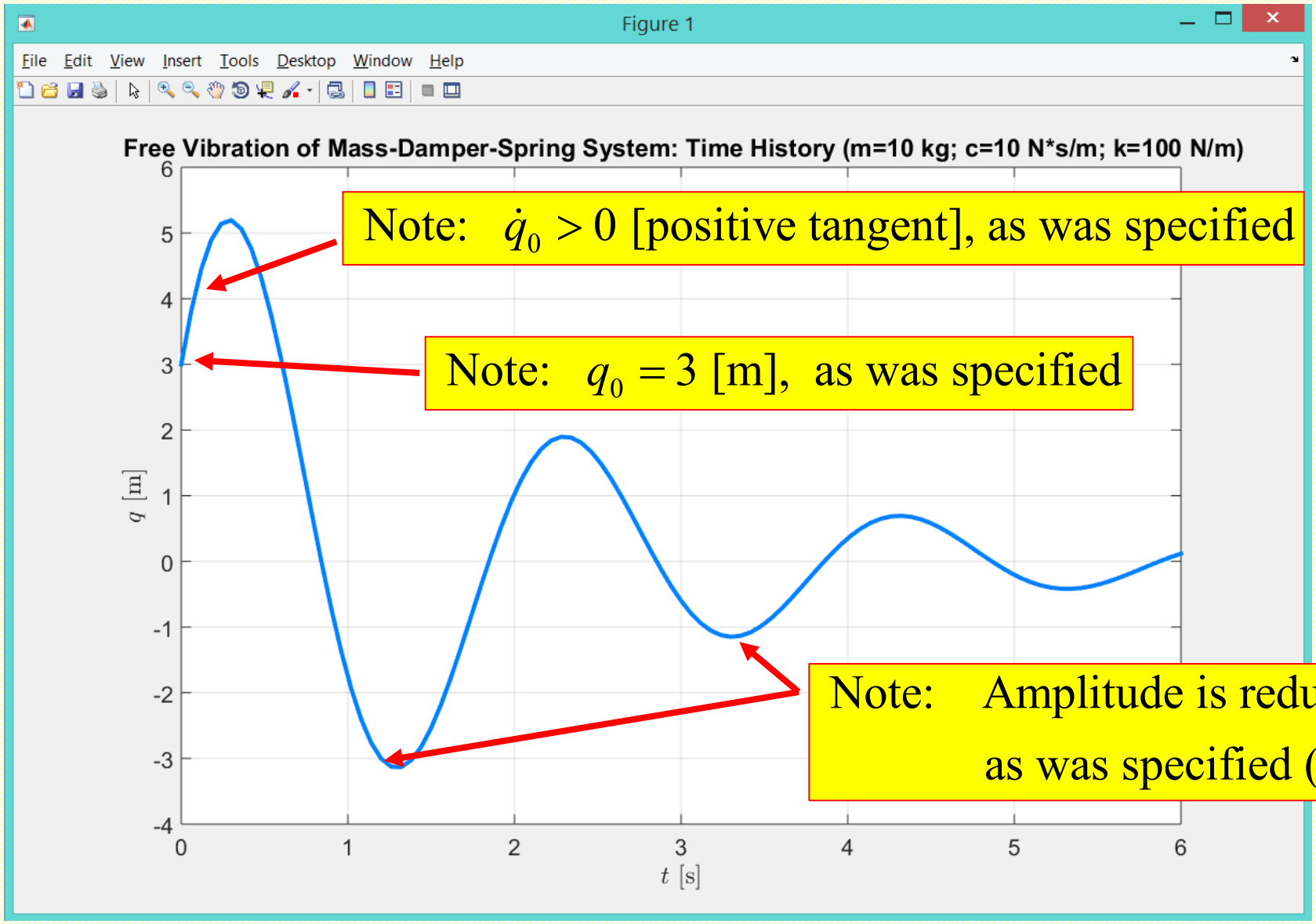
3. Calling ode45 procedure !

4. Plotting results

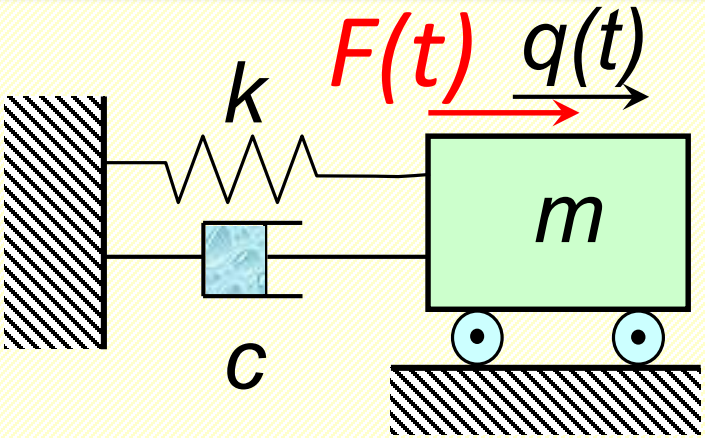
Output of the MATLAB script

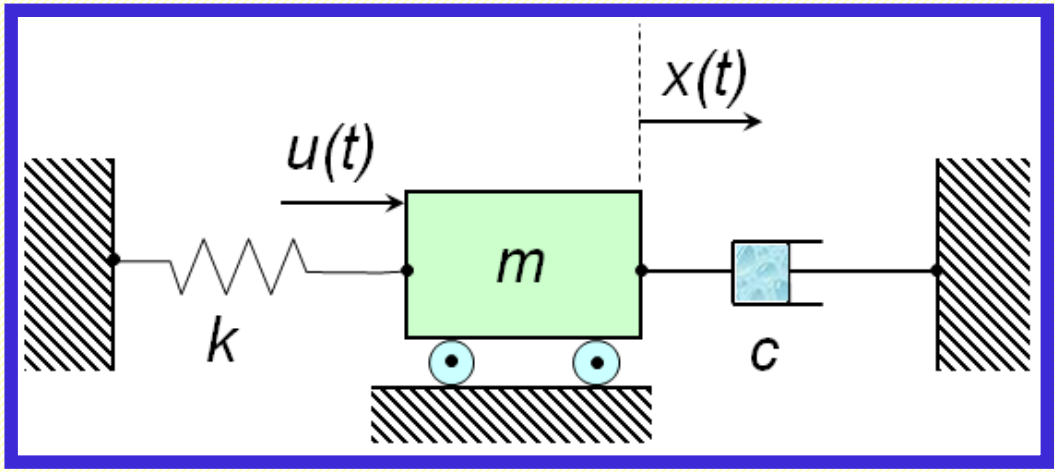


Output of the MATLAB script (annotated)



Some Comments:





1. Mass, damper, stiffness arrangements.
2. Adding springs in parallel/sequence & combination.
3. Initial Conditions – must be a vector!
4. Inspect dimensions of the results.
5. Which option to take: One-file or two-files?
6. Best choice: analytical, numerical and/or symbolic?

One-DOF mass-spring-damper system:

ADVANCED CASE: FORCED EXCITATION

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (no annotations)

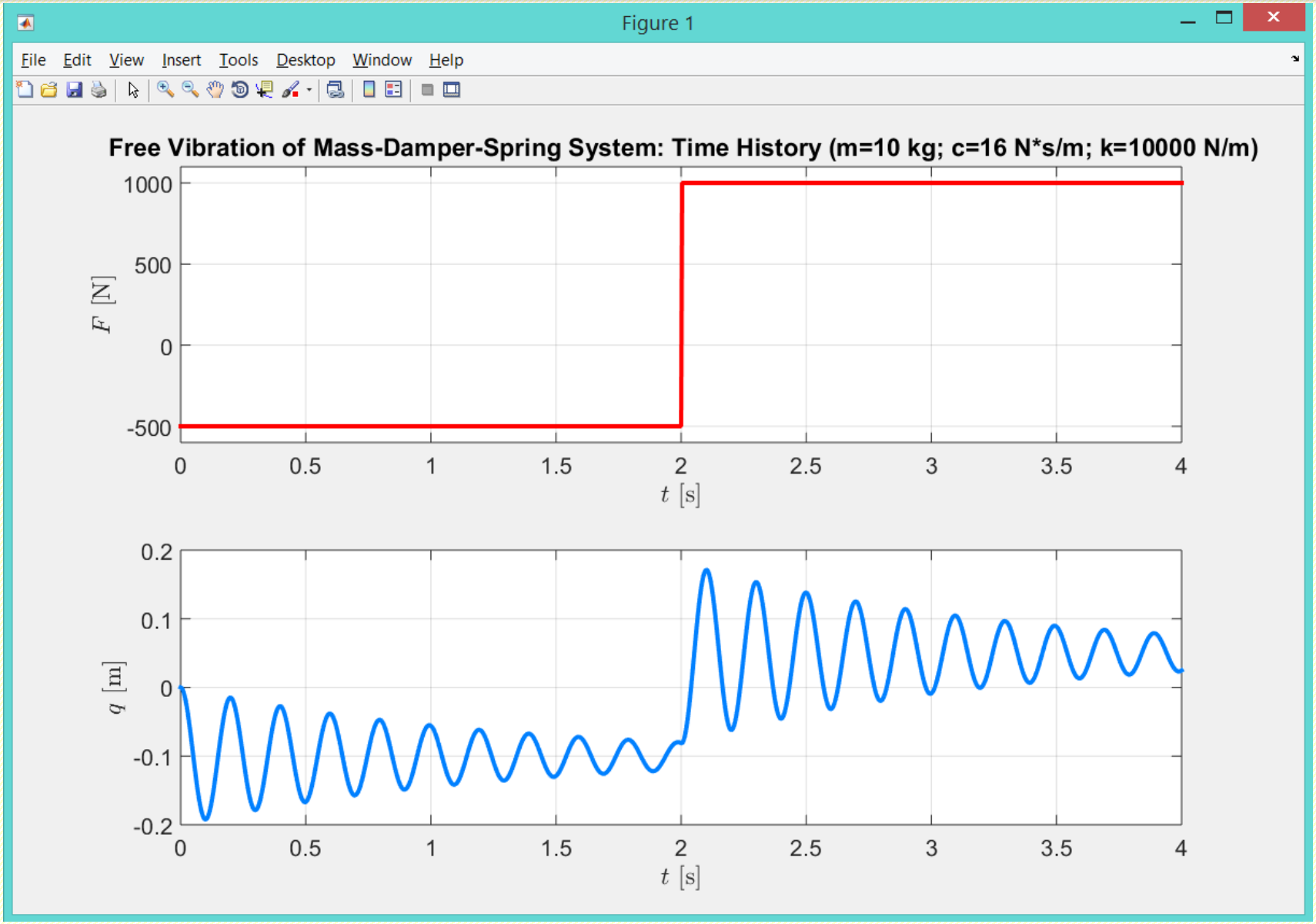
```
%% MASS-SPRING-DAMPER SYSTEM EXAMPLE: FORCED CASE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
%--------------------------------------
clear; close all; clc;
m=10; k=10000; c=16;
% Initial Conditions:
q0=0;          % m
q_dot0=0;    % m/s
tmax=4;        % s
t=[0:0.001:1]*tmax;
A=[0 1; -k/m -c/m]; B=[0; 1/m];
mck_xdot_anonymous = @(t, x) A*x+B*(-1000*(t<=2)+500*(t>2));
F_anonymous = @(t) (-500*(t<=2)+1000*(t>2));
[tt,xx]=ode45(mck_xdot_anonymous,t,[q0; q_dot0]);
%--------------------
```

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (no annotations)

```
% Continuation of the script
%--------------------
subplot(2,1,1);
plot(t,F_anonymous(t),'LineWidth',3,'Color','r');
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$F$ [N]','Interpreter','LaTeX');
axis([0 tmax -600 1100]);
set(gca, 'FontSize',16);
%--------------------
str1='\bf Free Vibration of Mass-Damper-Spring System: Time History ';
str2=sprintf('(m=%g kg; c=%g N*s/m; k=%g N/m)',m,c,k);
title([str1 str2]);
%--------------------
subplot(2,1,2);
plot(tt,xx(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$q$ [m]','Interpreter','LaTeX');
set(gca, 'FontSize',16);
set(gcf,'Position',[ 240  -50 1200  750]);
```

Output of the MATLAB script



END OF LECTURE-1b SLIDES