Introduction to MATLAB

Sven K. Esche
Department of Mechanical Engineering

January 12, 2010

Learning Objectives

- Using MATLAB, the student will be able to perform the following:
  - Run MATLAB
  - Manage MATLAB workspace
  - Use MATLAB Editor to create m-files
  - Input/output data from/to files
  - Define constants, variables, vectors and matrices
  - Perform simple calculations using numbers, vectors, matrices, operators and functions

Learning Objectives (cont.)

- Solve systems of linear equations
- Solve eigenvalue problems
- Generate 2D and 3D plots
- Solve systems of linear and nonlinear ODEs
- Solve unconstrained optimization problems
- Perform numerical integration
- Develop animations

MATLAB Background

- MATLAB (MATrix LABoratory) is high-performance language for technical computing
- MATLAB integrates computation, visualization, and programming in an easy-to-use environment
- Problems and solutions are expressed in familiar mathematical notation
- Basic data element is an array that does not require dimensioning
- ***demo*** MATLAB demonstration program

MATLAB Environment

- Prompt: (***>>***) indicates that MATLAB is ready to accept commands
- ***clc*** clears the Command Window
- ***%*** indicates a comment
- When a line ends with a semicolon (***;***), MATLAB performs the computation but does not display any output
- Three periods (***...***) followed by ***Return*** or ***Enter*** indicate that the statement continues on the next line
- ***format short/long/short e/long e*** formatting

MATLAB Workspace

- Command ***whos*** displays the name, size, and data type information for all variables defined in the workspace
- Command ***clear*** deletes all existing variables from the workspace

Computer Limits

- Command ***[ctype, maxsize] = computer*** displays type of computer *ctype* and max. array size *maxsize*
- Command ***intmax*** displays largest integer value
- Commands ***realmin*** / ***realmax*** display smallest / largest positive floating point number representable on the computer
- Command ***eps*** floating point accuracy

Example: Environment

- Perform the following operations:
  - Clear workspace and command window
  - Switch on echo function
  - Evaluate expression 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 - 1/8 + 1/9 - 1/10 + 1/11 - 1/12 using line continuation "..."
  - Suppress output when evaluating expression 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7;

Example: Environment (cont.)

- List name, size and data type of all variables in workspace
- Display current search path and current directory
- List all MATLAB files and all files in current directory

MATLAB Search Path

- When the name ***test*** is entered at the MATLAB prompt, the MATLAB interpreter
  - Looks for *test* as a variable
  - Checks for *test* as a built-in function
  - Looks in the current directory for a file named *test.m*
  - Searches the directories on the search path for *test.m*

MATLAB Search Path (cont.)

- Command ***path*** displays the current search path
- Command ***addpath /home/me641*** appends the directory */home/me641* to the current path
- Command ***rmpath /home/me641*** removes the directory */home/me641* from the current path
- Command ***pathtool*** starts the interactive Path Browser

MATLAB Search Path (cont.)

- Command ***cd*** (without any arguments) displays the current directory
- Command ***cd*** with a path changes the current directory
- Command ***what*** lists all MATLAB files in the current directory
- The exclamation point (***!***) is a shell escape (the rest of the input line is a command to the operating system)

M-Files

- Command ***type data.m*** displays existing m-file *data.m*
- Command ***edit*** launches the MATLAB Editor/Debugger with empty m-file
- Command ***edit data.m*** launches the MATLAB Editor/Debugger with existing m-file *data.m*
- Command ***dir*** lists all files in the current directory

Data Input / Output

- ***inputvar = input('input prompt')*** displays string 'input prompt', waits for value to be entered from keyboard and assigns it to variable *inputvar*
- ***isempty(A)*** returns 1 if array *A* is empty and 0 otherwise
- ***save('filename', 'var1', 'var2', -ascii)*** save data in var1 and var2 to ASCII file
- ***inputvar = textread('filename', '%f')*** read floating point value from file

Program Flow Control

- ***if***, ***else***, ***elseif*** execution of statements if certain expression is TRUE
- ***switch***, ***case***, ***otherwise*** execution of statements depending on value of expressions
- ***for***, ***while***, ***continue***, ***break*** loop for repeated execution of statements
- ***try***, ***catch*** error control
- ***return*** program termination

MATLAB Expressions

- Building blocks of MATLAB expressions are:
  - Constants
  - Variables
  - Operators
  - Functions

MATLAB Data Types

- Scalars
- Vectors
- Matrices
- Strings
- Special values

Numbers

- Conventional decimal notation: optional decimal point and plus or minus sign
- Scientific notation: e to denote power-of-ten factor
- Complex numbers: imaginary unit i or j
- Numbers are internally stored using long format specified by IEEE floating-point standard
- Special constants: pi, Inf, NaN

Variables

- MATLAB does not require type declarations and definition of dimensions (automatic variable creation and memory allocation)
- Variable names
  - start with letter followed by letters, numbers and underscores (no punctuation characters)
  - case sensitive
  - 31 significant characters

Operators

- Arithmetic operators:
  +, ***-***, ***.****, ***./***, ***.***, ***.^***
- Relational operators:
  <, ***<=***, ***>***, ***>=***, ***==***, ***~=***
- Logical operators:
  AND (***&***), ***OR*** (***|***), ***NOT*** (***~***)
- Special operators:
  %, ***'***, ***.***, ***:***, ***;***

Arithmetic Operators

- **+** Addition
- **-** Subtraction
- ***** Multiplication
- **/** Division
- **** Left division (described later)
- **^** Power
- **( )** Specification of evaluation order

Relational Operators

- **<** less than
- **<=** less than or equal than
- **>** greater than
- **>=** greater than or equal than
- **==** equal to
- **~=** not equal to

Logical Operators

- ***AND*** (***&***) TRUE if all inputs are TRUE
- ***OR*** (***|***) TRUE if at least one input is TRUE
- ***NOT*** (***~***) TRUE if the input is FALSE

Special Operators

- ***%*** comment
- ***'*** Complex conjugate transpose
- ***.*** elementwise execution of vector/matrix operations
- ***:*** creation of equally spaced vectors
- ***;*** output suppression

MATLAB Commands

- <command>ENTER
- <variable> = <command>ENTER
- [<variables>] = <command>ENTER

Example: Numbers

- Perform the following operations:
  - Define integer numbers 3 and -99
  - Define real numbers 0.0001, 9.6397238, 1.60210×10^{-20}, 6.02252×10^{23}
  - Define complex numbers 2+1i, -3.14159j and 3×10^{5}i
  - Define π
  - Explore expressions 1/0 and 0/0
  - Define variables my_favorite_number = 13 and x11 = 11

Mathematical Functions

- ***sin(x)***, ***cos(x)***, ***tan(x)***, ***asin(x)***, ***acos(x)***, ***atan(x)***, ***atan2(x, y)*** trigonometric functions
- ***sinh(x)***, ***cosh(x)***, ***tanh(x)***, ***asinh(x)***, ***acosh(x)***, ***atanh(x)*** hyperbolic functions
- ***exp(x)***, ***a^x***, ***log(x)***, ***log10(x)*** exponential and logarithmic functions

Mathematical Functions (cont.)

- ***abs(z)***, ***angle(z)***, ***conj(z)***, ***imag(z)***, ***real(z)*** complex functions
- ***abs(x)*** absolute value
- ***sign(x)*** signum
- ***sqrt(x)*** square root

Example: Functions

- Evaluate the following functions:
  - $\sin(\pi)$, $\cos(\pi)$, $\tan(\pi)$, $\sin^{-1}(0.5)$, $\cos^{-1}(0.5)$, $\tan^{-1}(0.5)$, $\tan^{-1}(0.5/1)$
  - $\sinh(\pi)$, $\cosh(\pi)$, $\tanh(\pi)$, $\sinh^{-1}(0.5)$, $\cosh^{-1}(0.5)$, $\tanh^{-1}(0.5)$
  - $e^{2.5}$, $\pi^{2.5}$, $\ln 2.5$, $\log_{10} 2.5$
  - $|1+2i|$, $\arg(1+2i)$, $\mathrm{Re}(1+2j)$, $\mathrm{Im}(1+2j)$, $\overline{1+2i}$
  - $|-1.5|$, $\mathrm{sign}(-1.5)$, $\sqrt{-1.5}$

Simple Calculations

- Type commands directly at prompt
- Generate m-file using MATLAB Editor/Debugger (command ***edit***, select ***Save As*** from Editor/Debugger File menu, select ***Run Script*** from MATLAB Command Window File menu)
- Calculator approach: ***2+3***
- Variable approach: ***a=2; b=3; a+b***

Matrices

- Rectangular arrangement of numbers
- Special matrices:
  - 1-by-1 matrix (scalar)
  - 1-by-n or n-by-1 matrices (vectors)
- Defining a matrix in MATLAB:
  - Enter an explicit list of elements
  - Load matrices from external data files
  - Generate matrices using built-in functions
  - Create matrices with your own functions in M-files

Special Matrices

- ***zeros(m,n)*** generates m-by-n matrix of zeroes
- ***ones(m,n)*** generates m-by-n matrix of ones
- ***rand(m,n)*** generates m-by-n matrix of random numbers that are uniformly distributed in the interval (0,1)
- ***eye(n)*** generates n-by-n identity matrix

Basic Matrix Operations

- ***A'*** (complex conjugate) transpose of A
- ***fliplr(A)*** flip A left to right
- ***size(A)*** size (dimensions) of A
- ***sum(A)*** sum of elements of each column of A
- ***diag(A)*** main diagonal of A
- ***trace(A)*** sum of diagonal elements (trace) of A

Advanced Matrix Operations

- ***det(A)*** determinant of matrix A
- ***rank(A)*** rank of matrix A
- ***rref(A)*** reduced row echelon form of A
- ***cond(A)*** condition number of matrix A
- ***inv(A)*** inverse of matrix A
- ***lu(A)*** LU factorization of matrix A
- ***qr(A)*** QR factorization of matrix A
- ***orth(A)*** orthonormal basis for range of A

Matrix Subscripts

- ***A(i, j)*** element in ith row and jth column
- ***A(i, :), A(:, j)*** ith row and jth column
- Reference to element outside of range leads to error message
  "Index exceeds matrix dimensions."
- Upon storing a value in element outside of range, matrix size is automatically increased

Generation of Vectors

- ***i:j*** generates row vector containing integers from i to j with unit spacing
- ***i:j:k*** generates row vector containing integers from i to k with spacing of j
- ***a:b:c*** generates row vector containing real numbers from a to c with spacing of b
- ***linspace(a, b, n)*** generates row vector containing n real numbers from a to b

Example: Matrices

- Perform the following operations:
  - Define matrices A, B, C, D and E

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} \text{rand} & \text{rand} \\ \text{rand} & \text{rand} \\ \text{rand} & \text{rand} \end{bmatrix} \quad E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Example: Matrices (cont.)

- Determine the following

A^T

dimensions of A

$\det(A)$

condition number of A

sum of columns of A

main diagonal of A

- Refer to element A(2, 3)
- Assign value 10 to element A(2, 4)

Example: Matrices (cont.)

- Define row and column vectors a, b, c, d and e:

$$a = \begin{bmatrix} 10 & 20 & 30 \end{bmatrix}$$

$$b = \begin{bmatrix} 1/2 & 2/3 & 3/4 \end{bmatrix}^T$$

$$c = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}$$

$$d = \begin{bmatrix} 100 & 93 & 86 & \ldots & 50 \end{bmatrix}$$

$$e = \begin{bmatrix} 0 & \pi/4 & \pi/2 & 3\pi/4 & \pi \end{bmatrix}$$

Solving Systems of Equations

- Solution of system of linear equations A * **x** = **b**
- If **b** is zero vector, either det(A) is zero or **x** is zero vector
- ***x = null(A)*** solution of A * **x** = **0**
- If **b** is nonzero, det(A) must be nonzero
- ***x = A \ b*** (more accurate and faster)
- ***x = inv(A) * b*** (less accurate and slower)

Example: Singular System

- **Given**:
  - Coefficient matrix A

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

  - Right-hand-side vector b $b = \begin{bmatrix} 6 & 30 & 72 \end{bmatrix}^T$
  - Note:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 6 \\ 30 \\ 72 \end{bmatrix}$$

Example: Singular System (cont.)

- **Find**:
  - Determinant and condition number of A
  - Inverse of A
  - Check accuracy of inverse
  - Solve system of equations A x = b using inverse of A
  - Check solution accuracy
  - Solve system of equations A x = b using backslash operator
  - Check solution accuracy

Example: Singular System (cont.)

- **Solution**:
  - Determinant of A:

    $$\det(A) = 0$$

  - Condition number of A:

    $$\text{cond}(A) = 3.8131 \times 10^{16}$$

  - Inverse of A:

    $$A^{-1} = 10^{16} \begin{bmatrix} -0.4504 & 0.9007 & -0.4504 \\ 0.9007 & -1.8014 & 0.9007 \\ -0.4504 & 0.9007 & -0.4504 \end{bmatrix}$$

Example: Singular System (cont.)

- Check accuracy of inverse of A:

$$AA^{-1} = \begin{bmatrix} 2 & 0 & 2 \\ 8 & 0 & 0 \\ 16 & 0 & 8 \end{bmatrix} \neq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A^{-1}A = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 8 & 0 \\ 4 & 0 & 0 \end{bmatrix} \neq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Solve system of equations A x = b using inverse of A

$$x = A^{-1}b = 10^{17}\begin{bmatrix} -0.8106 & 1.6213 & -0.8106 \end{bmatrix}^T \neq \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$$

Example: Singular System (cont.)

- Check solution accuracy

$$Ax - b = \begin{bmatrix} -6 & -30 & -72 \end{bmatrix}^T \neq \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$$

- Solve system of equations A x = b using backslash operator

$$x = A \backslash b = 10^{17} \begin{bmatrix} -0.8106 & 1.6213 & -0.8106 \end{bmatrix}^T \neq \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$$

- Check solution accuracy

$$Ax - b = \begin{bmatrix} 26 & 98 & -72 \end{bmatrix}^T \neq \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$$

Solving Eigenvalue Problems

- Solve eigenvalue problem $A * v = \lambda * v$
- ***[V, Λ] = eig(A)***
- Matrix V containing eigenvectors $\mathbf{v}_i$ as columns
- Matrix Λ containing eigenvalues λ_i on main diagonal
- ***poly(A)*** returns row vector with coefficients of characteristic polynomial of A in order of descending powers

Example: Eigenvalue Problem

- **Given**:
  - Matrix A: $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
- **Find**:
  - Eigenvalues and eigenvectors using ***eig*** function
  - Check solution accuracy
  - Coefficients of characteristic polynomial
  - Roots of characteristic polynomial
  - Eigenvectors using ***null*** function

Example: Eigenvalue Problem (cont.)

- **Solution**:
  - Eigenvalues using ***eig*** function:

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 16.1168 & 0 & 0 \\ 0 & -1.1168 & 0 \\ 0 & 0 & -0.0000 \end{bmatrix}$$

  - Eigenvectors using ***eig*** function:

$$V = \begin{bmatrix} | & | & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ | & | & | \end{bmatrix} = \begin{bmatrix} -0.2320 & -0.7858 & 0.4082 \\ -0.5253 & -0.0868 & -0.8165 \\ -0.8187 & 0.6123 & 0.4082 \end{bmatrix}$$

Example: Eigenvalue Problem (cont.)

- Check solution accuracy:

$$\mathrm{A}\mathbf{v}_1 - \lambda_1 \mathbf{v}_1 = 10^{-14}\left[0.5329 \quad -0.1776 \quad -0.1776\right]^{\mathrm{T}}$$

$$\mathrm{A}\mathbf{v}_2 - \lambda_2 \mathbf{v}_2 = 10^{-14}\left[-0.0333 \quad -0.0958 \quad -0.2776\right]^{\mathrm{T}}$$

$$\mathrm{A}\mathbf{v}_3 - \lambda_3 \mathbf{v}_3 = 10^{-14}\left[-0.0356 \quad -0.1953 \quad 0.0532\right]^{\mathrm{T}}$$

$$\left(\mathrm{A} - \lambda_1 \mathrm{I}\right)\mathbf{v}_1 = 10^{-14}\left[0.5329 \quad -0.1776 \quad -0.0888\right]^{\mathrm{T}}$$

$$\left(\mathrm{A} - \lambda_2 \mathrm{I}\right)\mathbf{v}_2 = 10^{-14}\left[-0.0666 \quad -0.0888 \quad -0.2665\right]^{\mathrm{T}}$$

$$\left(\mathrm{A} - \lambda_3 \mathrm{I}\right)\mathbf{v}_3 = 10^{-14}\left[-0.0222 \quad -0.1776 \quad 0.0444\right]^{\mathrm{T}}$$

Example: Eigenvalue Problem (cont.)

- Coefficients of characteristic polynomial p(λ): $p(\lambda) = 1.0000\lambda^3 - 15.0000\lambda^2 - 18.0000\lambda - 0.0000$
- Roots λ_i of characteristic polynomial p(λ):

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 16.1168 & 0 & 0 \\ 0 & -1.1168 & 0 \\ 0 & 0 & -0.0000 \end{bmatrix}$$

- Eigenvectors using ***null*** function:

$$V = \begin{bmatrix} | & | & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ | & | & | \end{bmatrix} = \begin{bmatrix} 0.2320 & -0.7858 & 0.4082 \\ 0.5253 & -0.0868 & -0.8165 \\ 0.8187 & 0.6123 & 0.4082 \end{bmatrix}$$

2-D Plotting Functions

- ***plot(x, y)*** plot 2-D lines (graphs)
- ***feather(u, v)*** plot vectors with components (u, v) emanating from equally spaced points along a horizontal axis
- ***quiver(x, y, u, v)*** plot vectors as arrows with components (u, v) at points (x, y)
- ***contour(x, y, z)*** plot contour lines of surface z = f(x, y) = const.

3-D Plotting Functions

- ***plot3(x, y, z)*** plot 3-D lines (graphs)
- ***surf(x, y, z)*** plot shaded surface z = f(x, y)
- ***mesh(x, y, z)*** plot wireframe mesh of surface z = f(x, y)
- ***quiver3(x, y, z, u, v, w)*** plot vectors with components (u, v, w) at points (x, y, z)

Plot Options

- ***title('text')*** put title *text* above plot
- ***xlabel('text'), ylabel('text'), zlabel('text')*** put label *text* on x-, y- or z-axis
- ***legend('text')*** put legend on graphs
- ***grid*** put grid on plot
- ***axis('equal')*** ensure uniform scaling of x-axis and y-axis
- ***axis([xmin xmax ymin ymax])*** specify plotting range for x-axis and y-axis

Plot Options (cont.)

- ***linespec*** set line style, width, color, marker type, marker size
- ***'-', '- -', ':', '-:'*** solid, dashed, dotted, dash-dot line
- ***linewidth n*** set line width of graphs to n dots
- ***'r', 'g', 'b', 'c', 'm', 'y', 'k', 'w'*** red, green, blue, cyan, magenta, yellow, black, white line color

Plot Options (cont.)

- ***'+', 'o', '*', '.', 'x'*** plus, circle, asterisk, dot, cross marker
- ***hold on/off*** add to/overwrite current graph
- ***zoom on/off*** zoom in/out interactively
- ***view(az, el)*** set azimuth and elevation viewing angle of plot
- ***figure*** create another figure window

Exporting Graphics Files

- Generate graphic using ***plot*** command
- Select ***Export*** from ***Figure Window File*** menu
- Select file name and file type *(***.tif**, ***.jpeg**, etc.)* from ***Export Window***

Example: Plotting a Graph

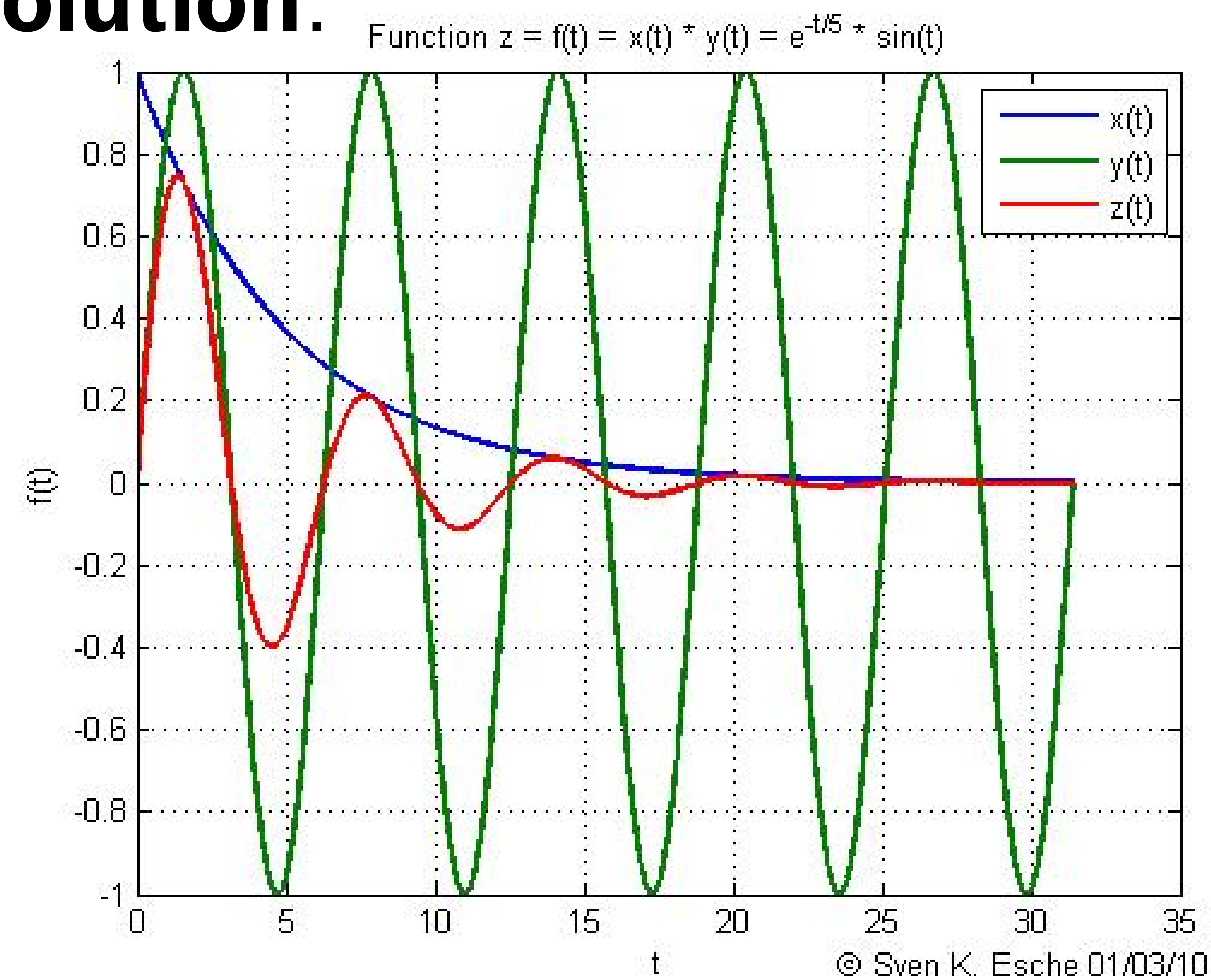
- **Given**:
  - Function z(t):

$$z = f(t) = x(t)y(t) \qquad x(t) = e^{-\frac{1}{5}t} \qquad y(t) = \sin t$$

- **Find**:
  - Plot graphs of x(t), y(t) and z(t) in interval [0, 10π]
  - Add title, axis labels, legend and grid to figure
  - Export figure as jpg file

Example: Plotting a Graph (cont.)

- **Solution**:

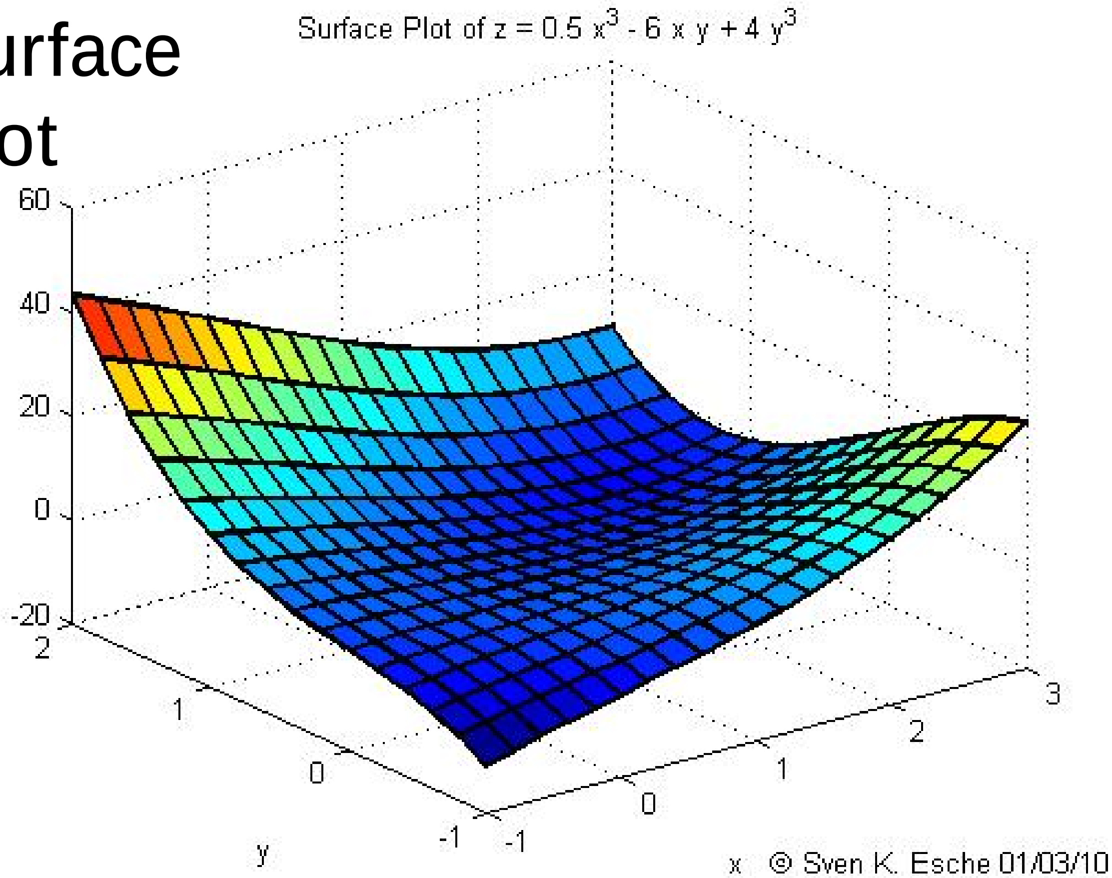


Example: Surface Plot

- **Given**:
  - Function z: $z = f(x, y) = 0.5x^3 - 6xy + 4y^3$
- **Find**:
  - Surface plot
  - Mesh plot
  - Contour plot

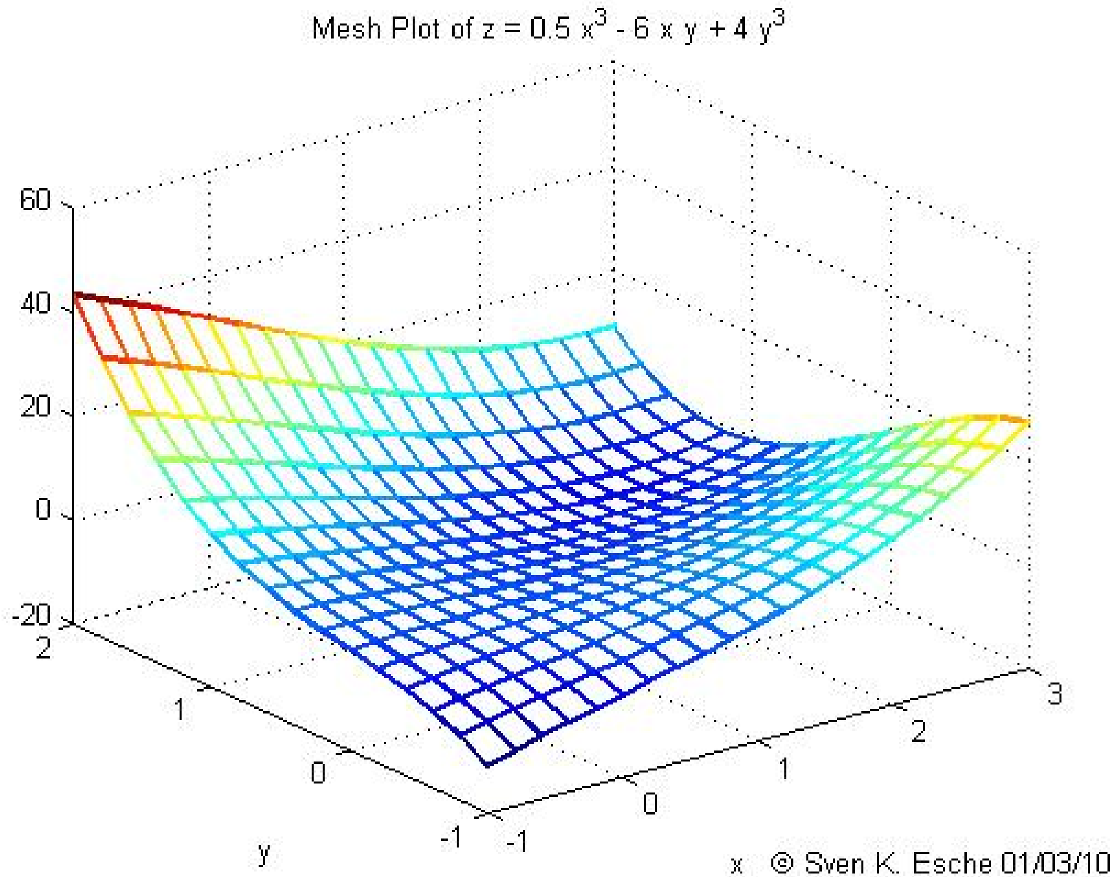
Example: Surface Plot (cont.)

- **Solution**:
  - Surface plot



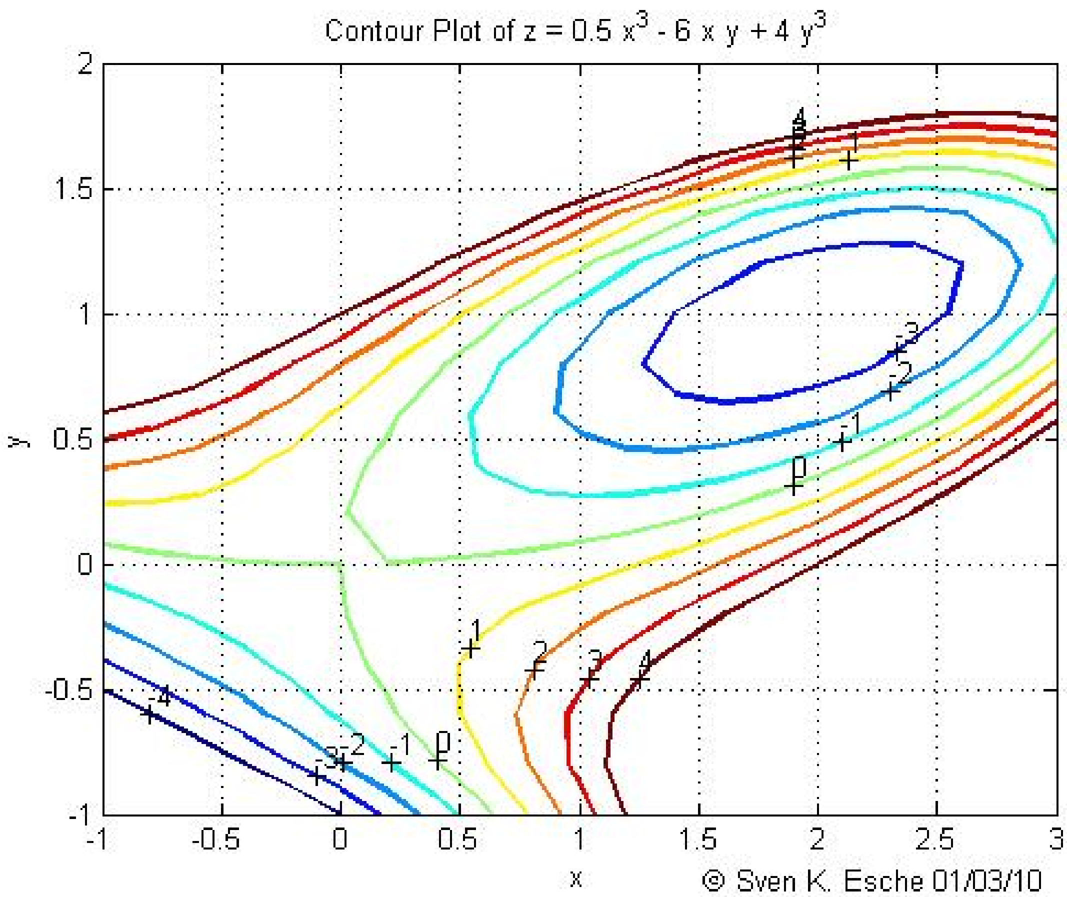
Example: Surface Plot (cont.)

- Mesh plot



Example: Surface Plot (cont.)

- Contour plot



Solving Linear ODEs

- ***ode45*** Numerical solution of ordinary differential equation *y' = f(t, y)*
- ***[t, y] = ode45('filename', t, y0)***
- ***t*** row vector with discrete time values
- ***y*** matrix with solutions for y and y' in first and second columns
- ***filename*** name of m-file containing *y' = f(t, y)*
- ***y0*** row vector with initial conditions y(0) and y'(0) in first and second columns

Example: Van der Pol's ODE

- **Given**:
  - Van der Pol's differential equation:
    $d^2x/dt^2 - m\,(1 - x^2)\,dx/dt + x = 0$
  - Initial conditions: $x(0) = 1$, $dx/dt(0) = 0$
- **Find**:
  - Numerical solution using **ode45** function
  - Time plot and phase plane plot

Example: Van der Pol's ODE (cont.)

- **Solution**:
  - Convert 2nd order ODE into system of two 1st order ODEs

$$\frac{d^2x}{dt^2} - m\left(1 - x^2\right)\frac{dx}{dt} + x = 0 \quad \dot{x} = \frac{dx}{dt} \quad \ddot{x} = \frac{d^2x}{dt^2} \quad \ddot{x} - m\left(1 - x^2\right)\dot{x} + x = 0$$

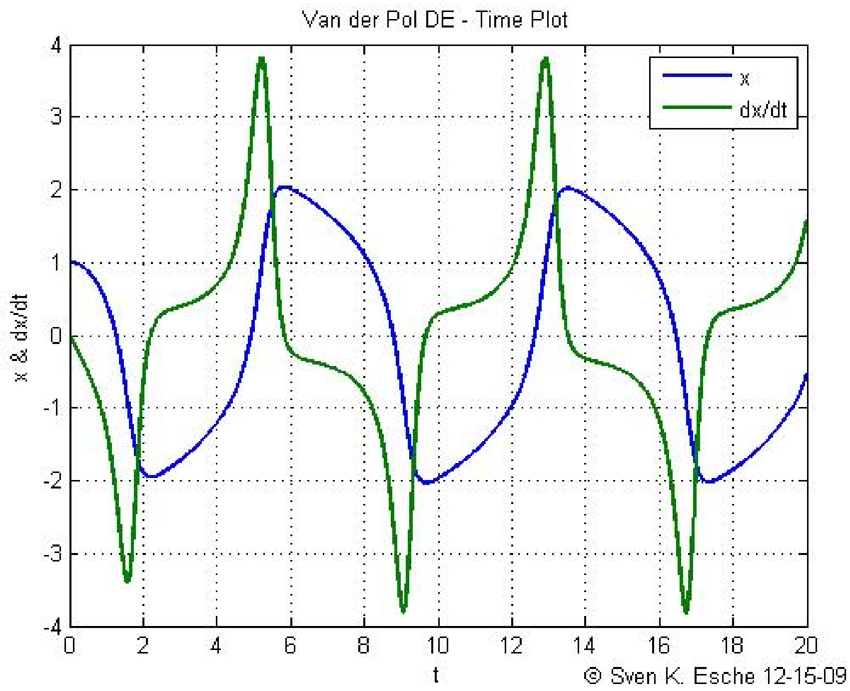
$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \qquad \dot{\mathbf{y}} = \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} y_2 \\ m\left(1 - y_1^{\,2}\right)y_2 - y_1 \end{bmatrix}$$

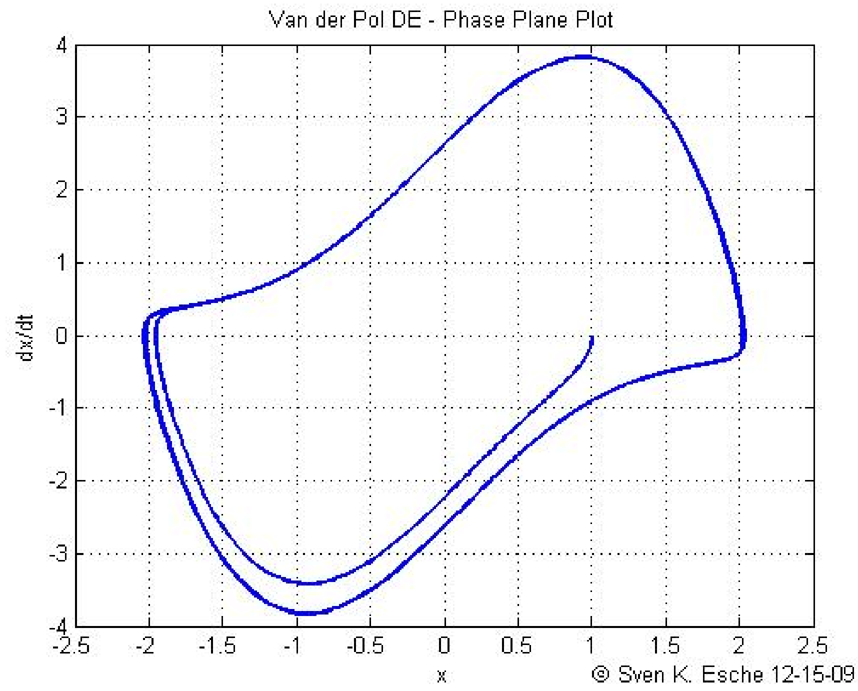
  - Define time vector **t** and initial condition vector $\mathbf{y}_0$

$$\mathbf{y}_0 = \left[x_o, \quad \dot{x}_o\right]^T = \left[1, \quad 0\right]^T$$

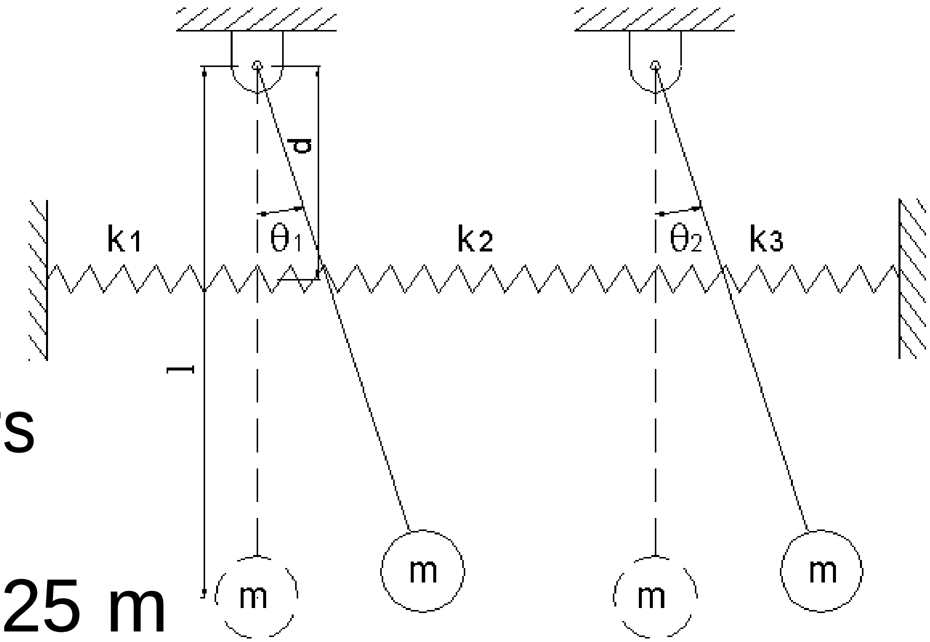
Example: Van der Pol's ODE (cont.)

- **Results**:





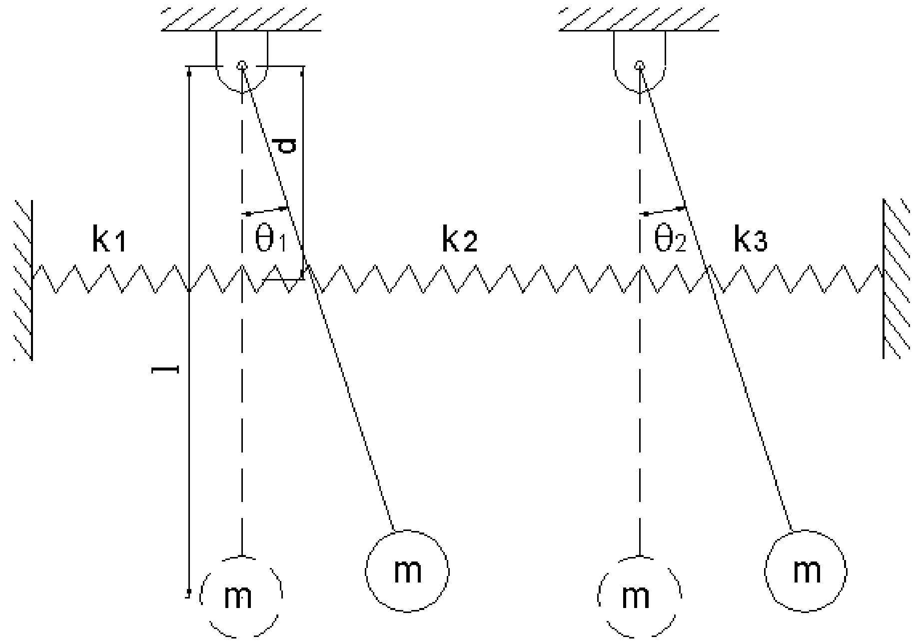
Example: Double Pendulum



- **Given**:
  - Spring-loaded double pendulum
  - System parameters
    $m_1 = m_2 = 0.1$ kg
    $d = 0.05$ m, $l = 0.25$ m
    $k_1 = k_3 = 200$ N/m, $k_2 = 20$ N/m
  - Initial conditions
    $\theta_1(0) = 0.1$ rad, $\theta_2(0) = 0.0$ rad
    $\omega_1 = \omega_2 = 0.0$ rad/s

Example: Double Pendulum (cont.)

- **Find**:
  - Numerical solution using **ode45** function
  - Time plot

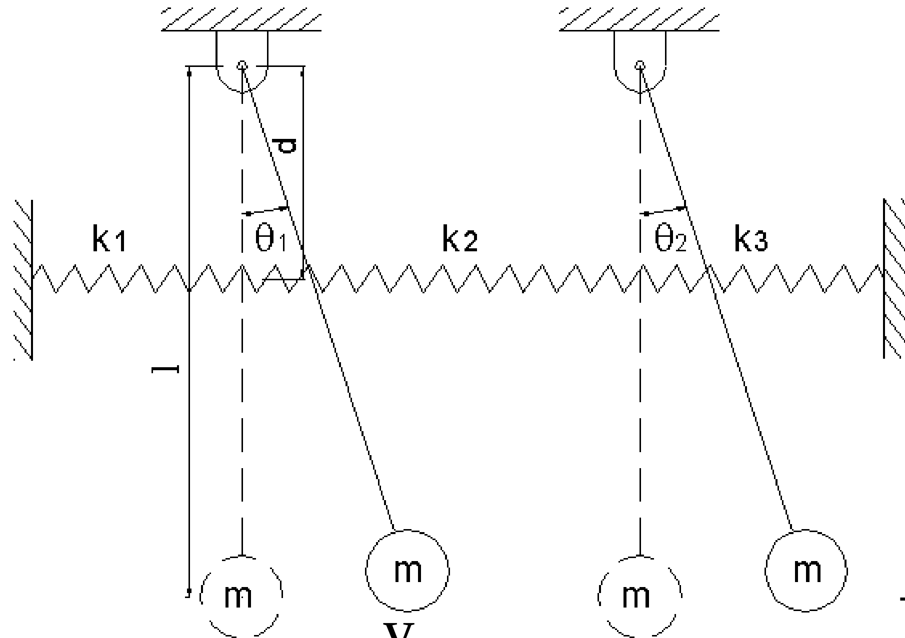


- **Solution:**
  - Governing equations (for small angles):

$$m_1 l^2 \ddot{\theta}_1 + \left[\left(k_1 + k_2\right) d^2 + m_1 g l\right] \theta_1 - k_2 d^2 \theta_2 = 0$$

$$m_2 l^2 \ddot{\theta}_2 + \left[\left(k_2 + k_3\right) d^2 + m_2 g l\right] \theta_2 - k_2 d^2 \theta_1 = 0$$

Example: Double Pendulum (cont.)



- Convert ODEs

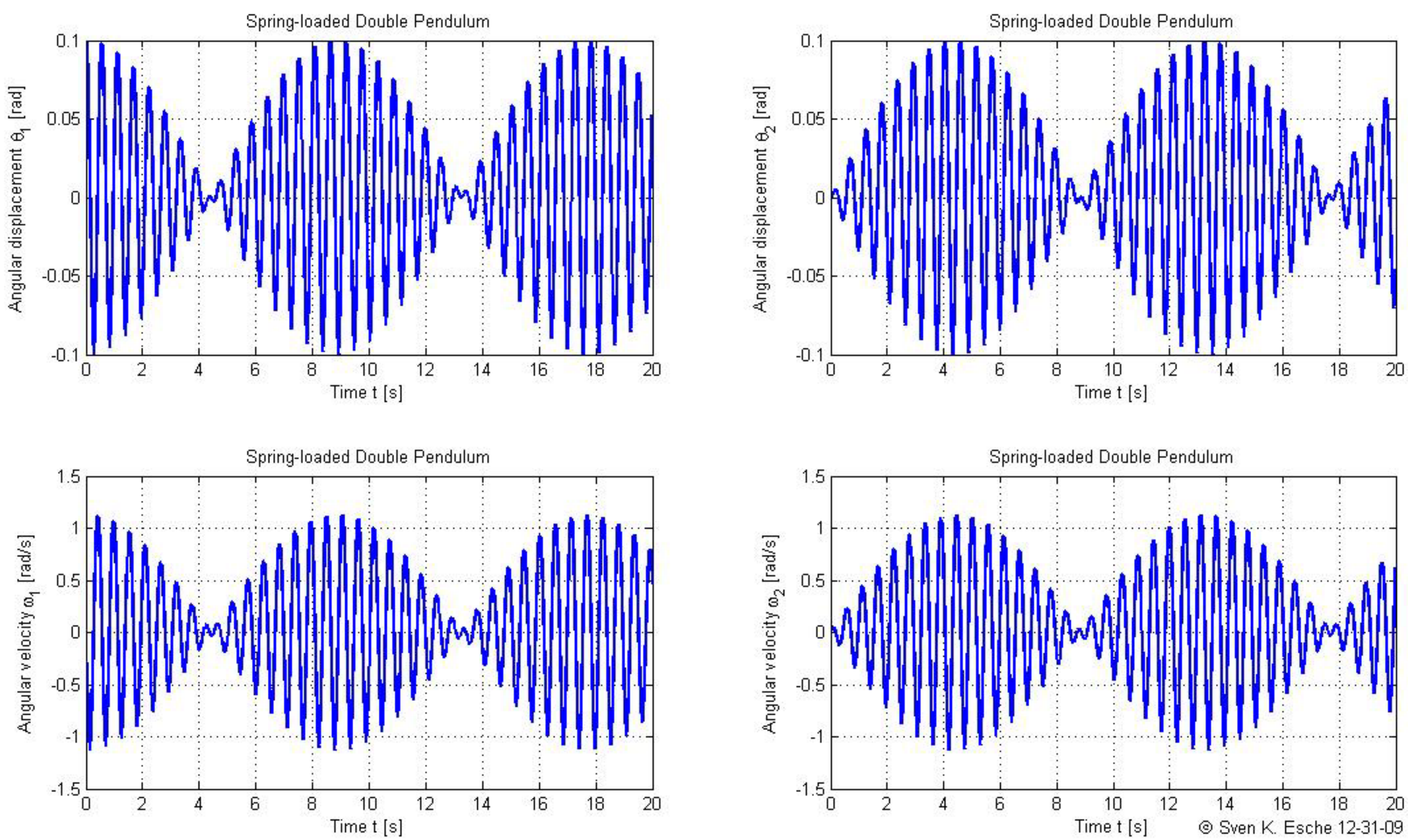
$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \omega_1 \\ \theta_2 \\ \omega_2 \end{bmatrix}$$

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \\ \dot{y}_4 \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \alpha_1 \\ \omega_2 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ \left\{k_2 d^2 y_3 - \left[\left(k_1 + k_2\right) d^2 + m_1 g l\right] y_1\right\} / m_1 l^2 \\ y_4 \\ \left\{k_2 d^2 y_1 - \left[\left(k_2 + k_3\right) d^2 + m_2 g l\right] y_3\right\} / m_2 l^2 \end{bmatrix}$$

$$\mathbf{y}_0 = \left[\theta_{10} \quad \omega_{10} \quad \theta_{20} \quad \omega_{20}\right]^T$$

Example: Double Pendulum (cont.)

- **Results**:



Unconstrained Minimization

- Define (column) residue vector **R** = f(**x**)
- Define (scalar) objective function F(**x**) = **R**(**x**)^T **R**(**x**)
- Minimize objective function using ***[x, fval] = fminsearch('filename', x0, options)***
- ***filename*** name of m-file containing F(**x**)
- ***x0*** row vector with initial guess for **x**

Unconstrained Minimization (cont.)

- Options
  - ***Display off, iter, final*** level of output display during iteration
  - ***TolX*** termination tolerance for x
  - ***TolFun*** termination tolerance for F(**x**)
  - ***MaxIter*** maximum number of iterations allowed
  - ***MaxFunEvals*** maximum number of function evaluations allowed

Example: Banana Function

- **Given**:
  - Rosenbrock's banana function (objective function:

$$f(\mathbf{x}) = f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

$$f(\mathbf{x}) = \mathbf{R}(\mathbf{x})^T \mathbf{R}(\mathbf{x}) = \begin{bmatrix} 1 - x & 10y - 10x^2 \end{bmatrix} \begin{bmatrix} 1 - x \\ 10y - 10x^2 \end{bmatrix}$$

  - Initial vector:

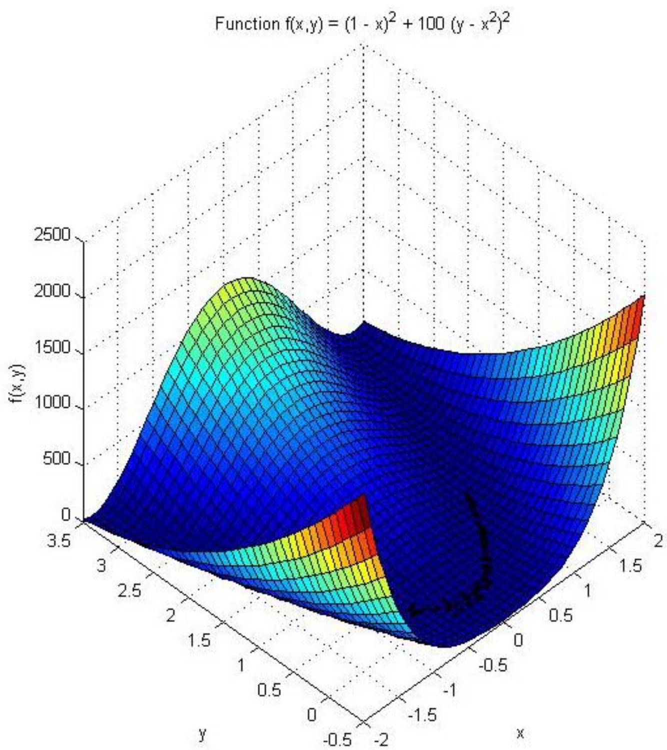
$$\mathbf{x}_0 = \begin{bmatrix} x_0 & y_0 \end{bmatrix}^T = \begin{bmatrix} -1.2 & 1 \end{bmatrix}^T$$

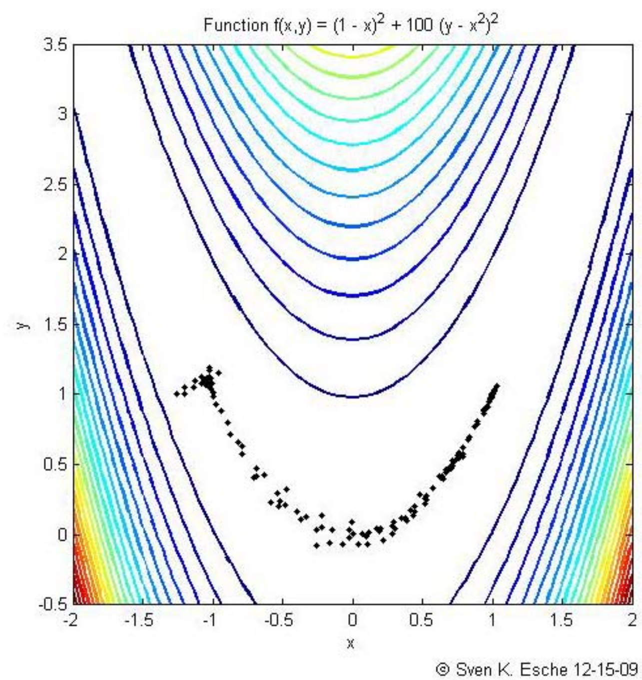
- **Find**:
  - Surface and contour plots of f(x, y)
  - Local minimum using **fminsearch** function

Example: Banana Function

- **Solution**:
  - Result (minimum): $\mathbf{x}^* = \left[x^* \quad y^*\right]^T = \left[1 \quad 1\right]^T$

$$f\left(\mathbf{x}^*\right) = 100\left[y^* - \left(x^*\right)^2\right]^2 + \left(1 - x^*\right)^2 = 100\left[1^2 - \left(1\right)^2\right]^2 + \left(1-1\right)^2 = 0$$





Example: Four-bar Linkage

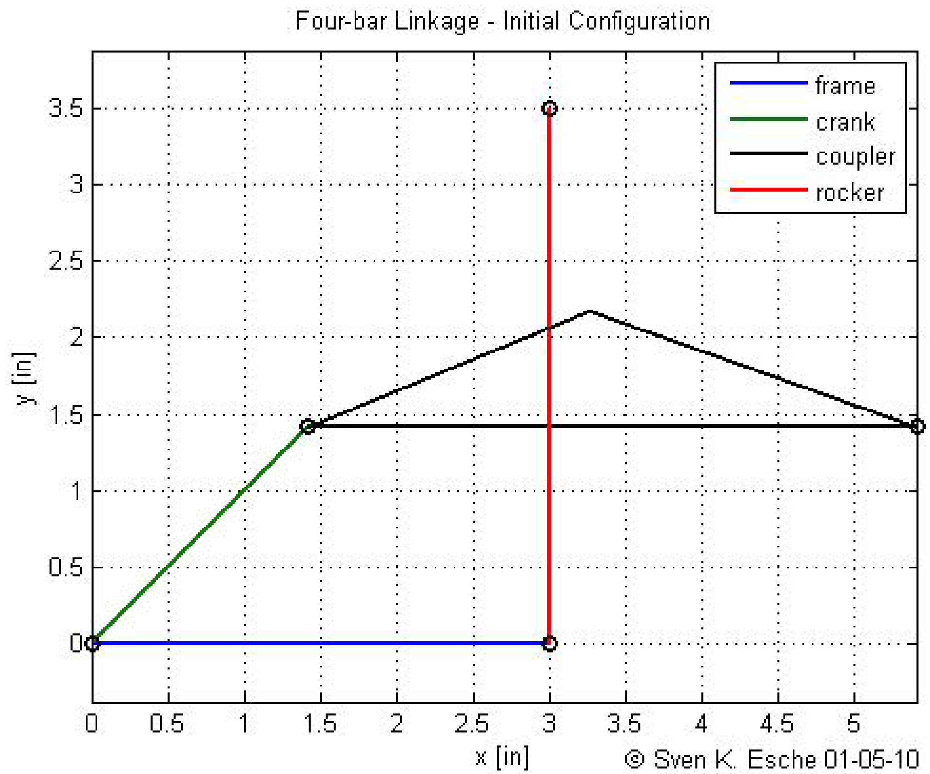
- **Given**:
  - Link lengths: l_1 = 3 in, l_2 = 2 in, l_3 = 4 in, l_4 = 3.5 in
  - Link orientations: $\theta_1 = 0°$, $\theta_2 = 45°$
- **Find**:
  - Assembled configuration
  - Animation for full crank rotation

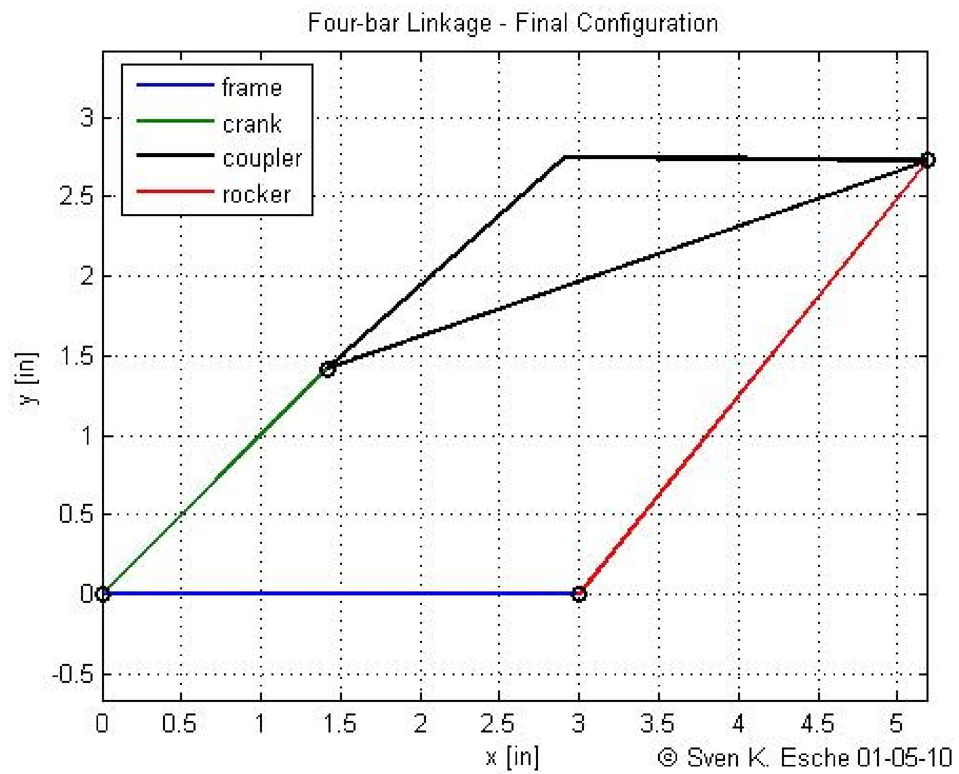
Example: Four-bar Linkage (cont.)

- **Solution**:
  - Define residue vector $\mathbf{R} = f(x_i, y_i, \theta_i)$
    - x_i: *x-coordinates of link centers of gravity*
    - y_i: *y-coordinates of link centers of gravity*
    - θ_i: *link orientations*
  - Define (scalar) objective function $F = \mathbf{R}(x_i, y_i, \theta_i)^T \mathbf{R}(x_i, y_i, \theta_i)$
  - Minimize objective function using ***fminsearch('filename', x0, options)***

Example: Four-bar Linkage (cont.)

- Initial and final configurations:





Numerical Integration Using *quad*

- ***quad('filename', a, b, tol)*** evaluate definite integral using adaptive Simpson quadrature
- ***filename*** name of m-file containing function f(x) to be integrated
- ***a, b*** finite integration limits
- ***tol*** absolute error tolerance

Example: Quadrature

- **Given**:
  - Function f(x) = sin(x)
- **Find**:
  - Euclidean norm of f(x) on interval [0, 2π]
- **Solution**:
  - Analytical solution:

$$\left\|f(x)\right\|_2 = \sqrt{\langle f(x), f(x) \rangle} = \sqrt{\int_a^b [f(x)]^2 dx} = \sqrt{\int_0^{2\pi} \sin^2(x) dx}$$

$$\left\|f(x)\right\|_2 = \sqrt{\left[\frac{x}{2} - \frac{\sin 2x}{4}\right]_0^{2\pi}} = \sqrt{\pi} = 1.7725$$

Example: Quadrature (cont.)

- Main program:
  - *n2sin2 = quad('matlab_quad_sub', 0, (2 * pi));*
  - *n2sin = sqrt(n2sin2)*
- Subroutine:
  - *function n2sin2 = matlab_quad_sub(x);*
  - *n2sin2 = sin(x) .^ 2;*
- Numerical result: $||\sin(x)||_2 = 1.7725$

Numerical Integration Using *trapz*

- ***trapz(x, y)*** evaluate definite integral using trapezoidal numerical integration
- ***x*** (uniformly or nonuniformly spaced) integration grid (row vector)
- ***y*** function values $y_i = f(x_i)$

Example: Trapezoidal Rule

- **Given**:
  - Function f(x) = sin(x)
- **Find**:
  - Definite integral of f(x) on interval [0, π]
- **Solution**:
  - Analytical solution:

$$\int_0^{\pi} \sin x dx = -\cos x\Big|_0^{\pi} = 1-(-1) = 2$$

Example: Trapezoidal Rule (cont.)

- MATLAB program:
  - *x = 0.0:pi/180.0:pi;*
  - *y = sin(x);*
  - *trapz(x, y)*
- Numerical result:

$$\int_0^{\pi} \sin x dx = 1.9999$$