Exercises in Computer-Aided Problem Solving

1. Introduction to this course

- Instructors and course information
- Purpose of this course
- Important remarks
- Assignment submission
- Schedule
- Overview of Octave/MATLAB
- Installing GNU Octave to your PC

Instructors and course information

Hitomi Anzai

Assistant professor of Institute of Fluid Science

Email: hitomi.anzai.b5@tohoku.ac.jp

Mickael Laine

Assistant professor of Graduate School of Engineering

Email: laine.mickael@tohoku.ac.jp

Yutaro Kohata

Teaching Assistant

Email: yutaro.kohata.t4@dc.tohoku.ac.jp

Course information

Google Classroom class code: kvssylh

Link for streaming session using Meet:

https://meet.google.com/lookup/el2bghbaoy

Material from previous year

www.vision.is.tohoku.ac.jp/us/course/computer-aided-problem-solving/

Purpose of this course

- Students will learn how a computer can be used to solve mathematical problems.
- Although the course will use Octave for this purpose, its focus is more on mastering mathematical skills rather than learning how to use it.
- Starting with the basic usage of Octave (or MATLAB) and how to write a program on it, students will learn how they can solve various mathematical problems by writing and executing simple programs.
- The course will cover not only mathematics that students have already learned, such as calculus, differential equation, linear algebra, etc., but also those that they have not learned, such as numerical computation, signal processing, statistics, machine learning, etc.
- The goal of this course is to have students master skills of solving the specific problems considered in this course using Octave (or MATLAB) and further obtain a concept of how they can utilize a computer to deal with novel problems.

Important remarks

- *All students* are required to use their own computers and must access the class via Google Classroom.
- Exercise problem(s) will be assigned to students for each lecture.
    - The lecture material and videos will cover the necessary topics to solve the exercises.
    - There will is no assignment for the 1st lecture, but you can submit a report as a test.
- Students are *required to submit all exercise problems* given on each class day in a week.
    - E.g., Exercises on a Monday must be submitted until the next Monday, etc.
    - Submission is done via Google Classroom (see Assignment Submission material).
- Grading will be based only on reports.
- If you have trouble, contact us either via e-mail or Google Classroom.
    - Do NOT put your questions to lecturers on your assignment file.
    - When sending e-mail, please send it to all instructors.

Assignment Submission

- **The report file** must be a PDF format and contain scripts, results (command output, plots, etc.) and an explanation.
  - Report filenames must be CAPS_02_B9TBXXXX.pdf, where “02” is the number of the lecture and “B9TBXXXX” is the student number.
- **The Script file** must also be submitted.
  - Script filenames must be CAPS_02_B9TBXXXX_ScriptName.m, where “ScriptName” can be any name.
- Submit your reports and script files via Google Classroom.
- **The deadline** is one week after the lecture (can be seen in Google Classroom).
  - You may send a revised revision after the deadline.

- There is no final examination. To get credit in this class, submit all reports for lectures from 2 to 13 before deadline.
- Showing only a script and its explanation to solve the exercise will only get an average grade. Detailed explanations of your solutions and additional work will get additional points.
- Copying from other people or past reports will NOT be evaluated.

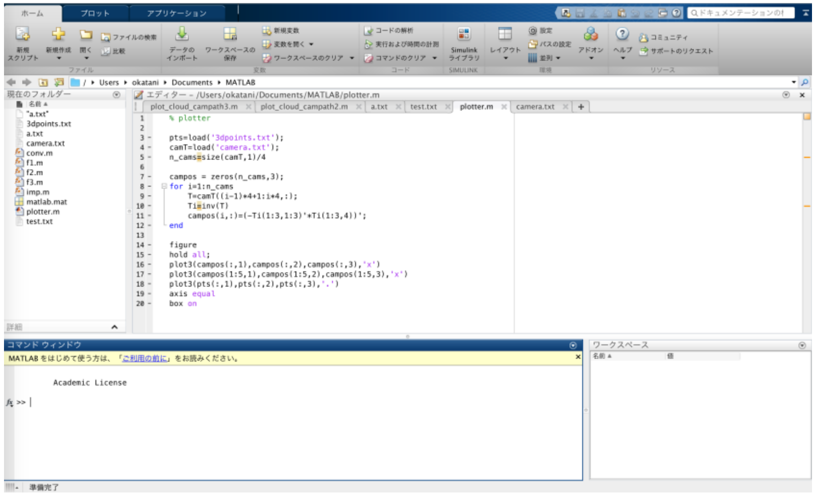
Schedule

April	12	(Mon)	1. Introduction and installation of Octave
April	16	(Fri)	2. Fundamentals of Octave/MATLAB
April	19	(Mon)	3. Matrices and linear algebra I
April	23	(Fri)	4. Roots of algebraic and transcendental equations
April	26	(Mon)	5. Least-square method and line fitting
April	30	(Fri)	6. Numerical integration and ordinary differential equations
May	3	(Mon)	Holiday
May	7	(Fri)	7. Signal processing
May	10	(Mon)	8. Probability theory: basics
May	14	(Fri)	9. Statistics I
May	17	(Mon)	10. Matrices and linear algebra II
May	21	(Fri)	11. Statistics II
May	24	(Mon)	12. Machine learning I
May	28	(Fri)	13. Machine learning II
May	31	(Mon)	14. (backup for schedule change)
June	4	(Fri)	15. (backup for schedule change)
June	7	(Mon)	16. (backup for schedule change)

MATLAB / Octave

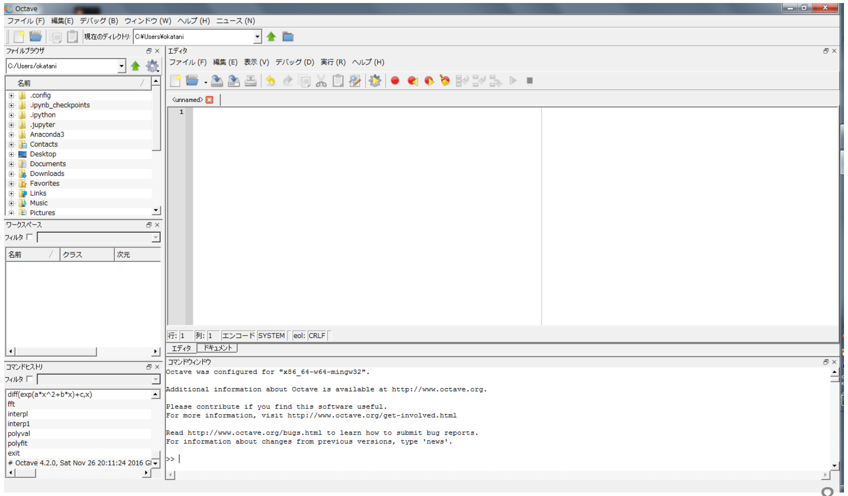
MATLAB

- A numerical computing environment and programming language developed and sold by MathWorks
- *De facto* standard in many scientific/engineering fields the world over
- A wide variety of extensions, called *toolboxes*, are available for use in a diverse field of applications



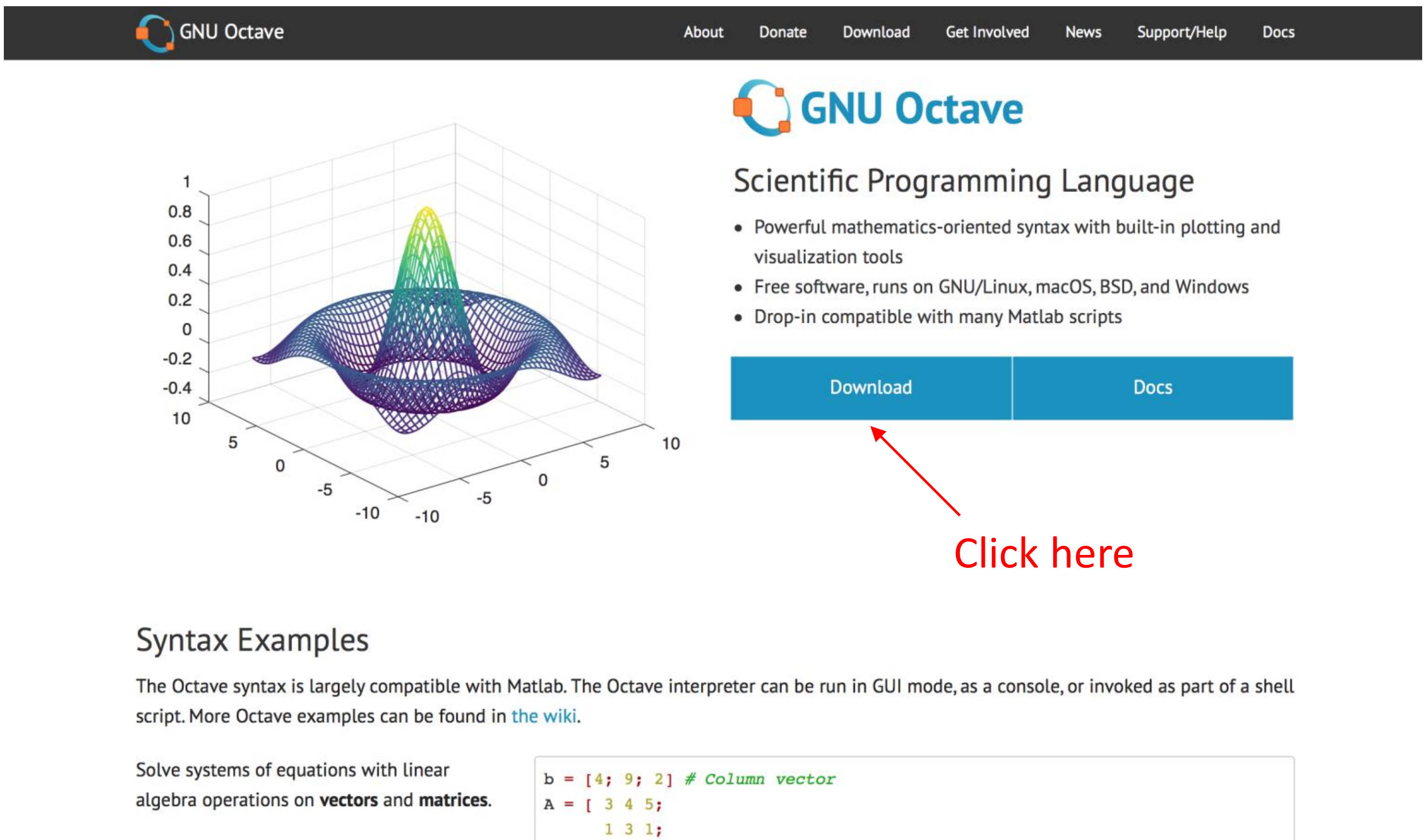
GNU Octave

- A numerical computing environment and programming language developed by volunteers and can be used for free
- Compatible to MATLAB to a certain degree
- A variety of extensions called *packages*, the counterpart of the toolboxes, is available but has only limited compatibility



Installing Octave to your PC (1/3)

- To install the Windows version of Octave, follow the procedures below
- Access the following URL with a Web browser and click “Download”
  - `https://www.gnu.org/software/octave/`



Installing Octave to your PC (2/3)

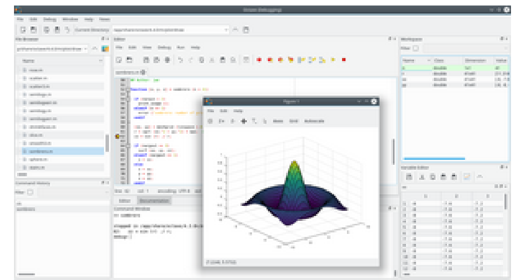
- Further select “Windows” and click the link then appeared

Install

Source | GNU/Linux | macOS | BSD | Windows

Click here

Note: All installers below bundle several **Octave Forge packages** so they don't have to be installed separately. After installation type `pkg list` to list them. Read more.



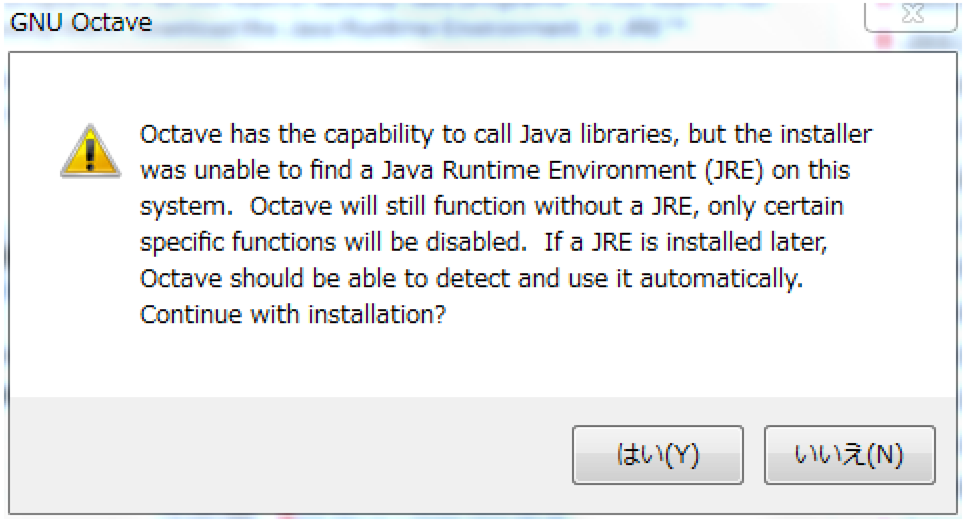
- Windows-64 (recommended)
  - octave-5.2.0_1-w64-installer.exe (~ 300 MB) [signature]
  - octave-5.2.0_1-w64.7z (~ 300 MB) [signature]
  - octave-5.2.0_1-w64.zip (~ 530 MB) [signature]
- Windows-32 (old computers)
  - octave-5.2.0_1-w32-installer.exe (~ 275 MB) [signature]
  - octave-5.2.0_1-w32.7z (~ 258 MB) [signature]
  - octave-5.2.0_1-w32.zip (~ 447 MB) [signature]

Click here if you have recent Windows (Windows 7, Windows 8, Windows 10)

Click here if you have very old Windows. If you don't know, you don't need this.

Installing Octave to your PC (3/3)

- Run the downloaded .exe file by clicking it
  - Neglect the following message about JRE(Java runtime environment) by clicking “Yes” and continuing the installation



  - You will have to wait for a few minutes until the completion

Installing Octave to your Mac (1/2)

- If you already have Homebrew, MacPorts or Fink, you should be able to Install Octave via them.
- In other cases, you can try an App Bundle

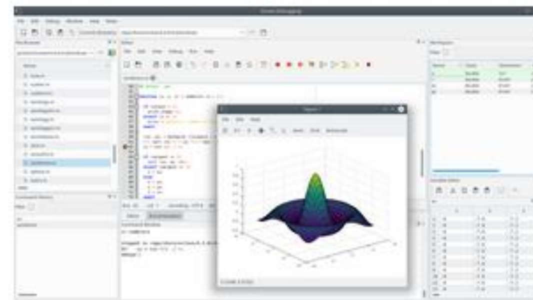
Install

Source	GNU/Linux	macOS	BSD	Windows

The Octave Wiki has instructions for installing Octave on macOS systems. Octave may also be available in third-party package managers such as Homebrew, MacPorts, or Fink.

Click here

Then, click here



Installing a macOS App Bundle [edit]

Good progress has been made on creating a reliable App bundle for Octave on macOS. Approaches using MacPorts and Homebrew have been considered. The Octave.app project provides an unofficial ready-to-use macOS app bundle installer based on Homebrew.

- macOS App Bundle of Octave 5.1.0 Beta (with GUI)
- macOS App Bundle of Octave 4.4.x (with GUI)
- macOS App Bundle of Octave 4.0.3 (with GUI) (OS X 10.9+)

Finally, click here

To compile and create the application bundle yourself, see the instructions on how to create the bundle using Homebrew. (See instructions on how to create a bundle using Macports for reference, but this approach is not currently being used.)

Installing Octave to your Mac (2/2)

Octave.app

A native Mac app distribution of GNU Octave

Home — Download — News — Contact

Octave.app Downloads

This page contains all the releases of Octave.app. If you're not sure which one to get, just grab the latest version, from the top of the page.

For betas and prereleases, including the new 5.x series see Developer Downloads.

Installation Notes

Our apps are not signed yet. This means that the first time you run them, you must right-click Octave.app and choose "Open" instead of just double-clicking it, and then choose "Run" from the dialog that pops up, when it asks you if you want to run an app from an "unkown developer".

We're working on fixing this by getting signed releases. Sorry for the inconvenience.

Be sure to follow installation notes!

Current Release

This is probably the one you want.

Click and download.

Octave 4.4.1 u1

Download: Octave-4.4.1.dmg

Release Notes: Octave 4.4.1 Release Notes

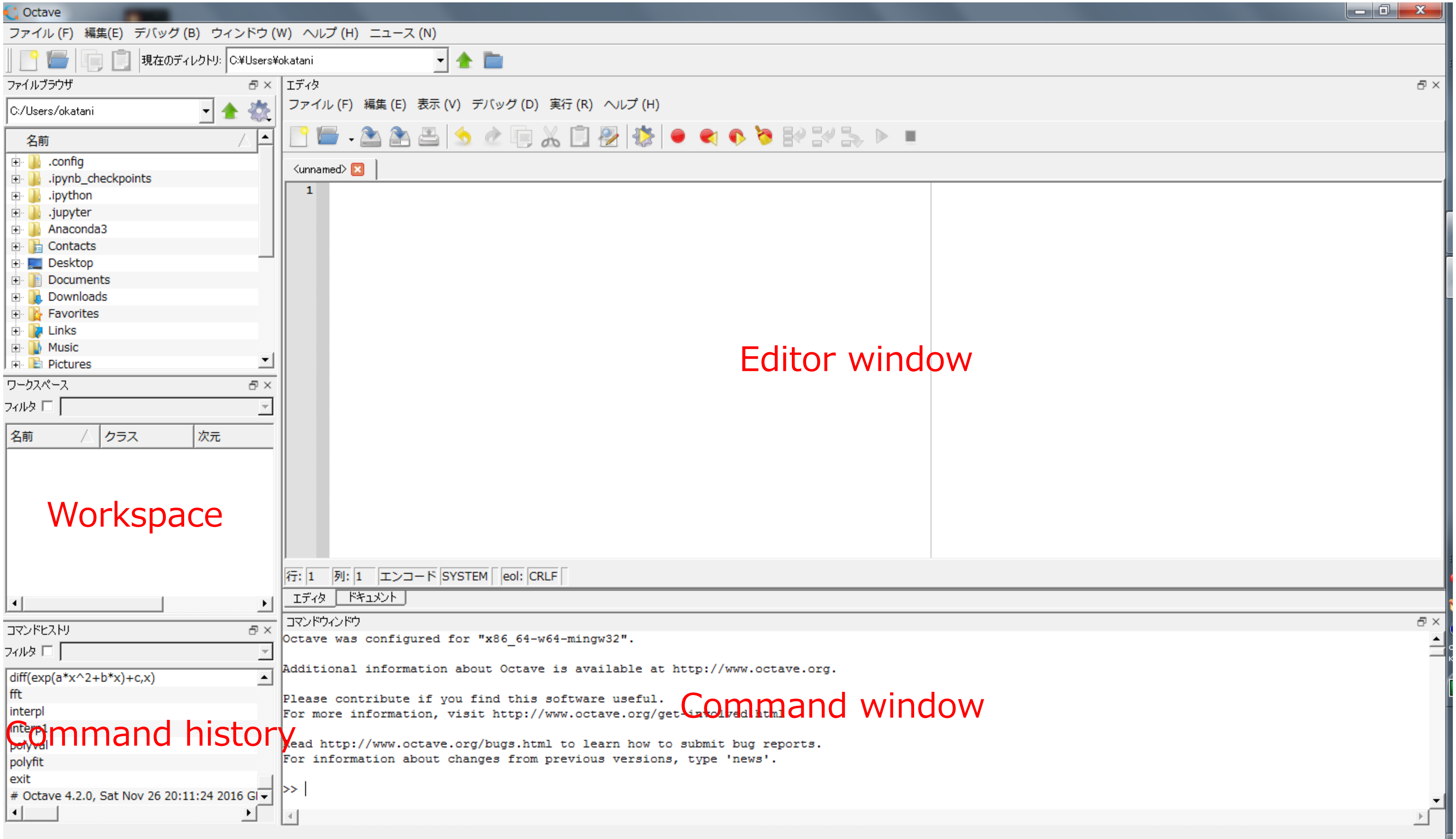
Exercises 1.1 (assignments)

- You don’t have to submit assignments for the first lecture.
- As a test, an assignment will be created under the CAPS01 topic in Google Classroom, which you can submit as practice.
  - Please follow the guidelines in “Assignment Submission” material.

2. Fundamentals of Octave (&MATLAB)

- Octave GUI(Graphical User Interface)
- Command Window
- Scripts
- Variables
- Matrices
- Arithmetic operations & special values
- Mathematical functions
- Input/output with files
- Loops
- Conditional branch & flow control
- Plotting grpahs

Octave GUI


Octave
ファイル (F) 編集(E) デバッグ (B) ウィンドウ (W) ヘルプ (H) ニュース (N)
現在のディレクトリ: C:¥Users¥okatani
ファイルブラウザ
C:/Users/okatani
名前
.config
.ipynb_checkpoints
.ipython
.jupyter
Anaconda3
Contacts
Desktop
Documents
Downloads
Favorites
Links
Music
Pictures
ワークスペース
フィルタ
名前
クラス
次元
Workspace
コマンドヒストリ
フィルタ
diff(exp(a*x^2+b*x)+c,x)
fft
interpl
polyfit
exit
Command history
エディタ
ファイル (F) 編集 (E) 表示 (V) デバッグ (D) 実行 (R) ヘルプ (H)
<unnamed>
Editor window
行: 1 列: 1 エンコード SYSTEM eol: CRLF
エディタ
ドキュメント
コマンドウィンドウ
Octave was configured for "x86_64-w64-mingw32".
Additional information about Octave is available at http://www.octave.org.
Please contribute if you find this software useful.
Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.
>>
Command window

Using Command Window

- Example: Type “1+2” and press the Enter key after the prompt “>>”

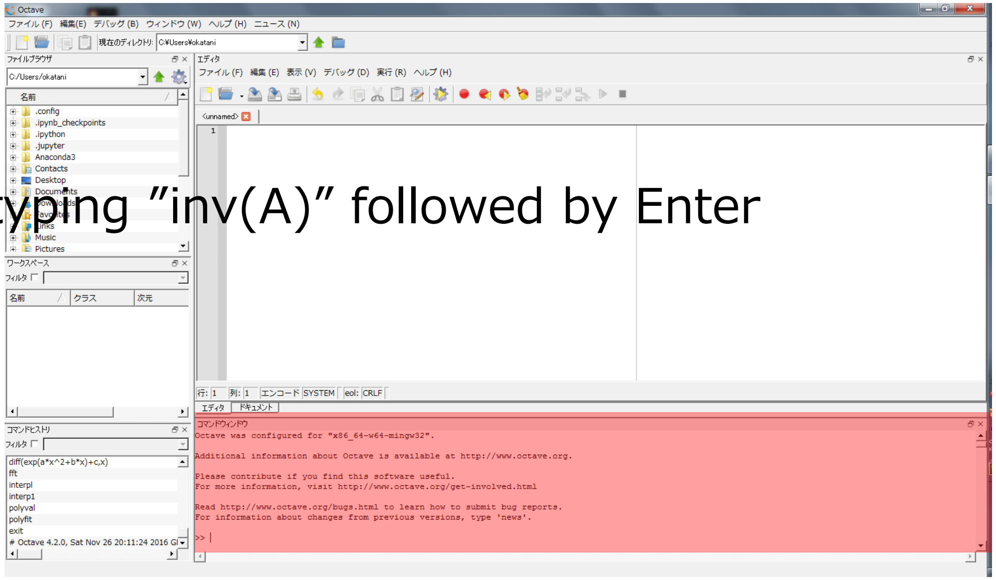
```
>> 1+2
ans =  3
>>
```

- You can create a 2x2 matrix A by typing as follows:

```
>> A=[1,2;3,4]
A =
   1   2
   3   4
```

- You can calculates its inverse by typing ”inv(A)” followed by Enter

```
>> inv(A)
ans =
  -2.00000   1.00000
   1.50000  -0.50000
```

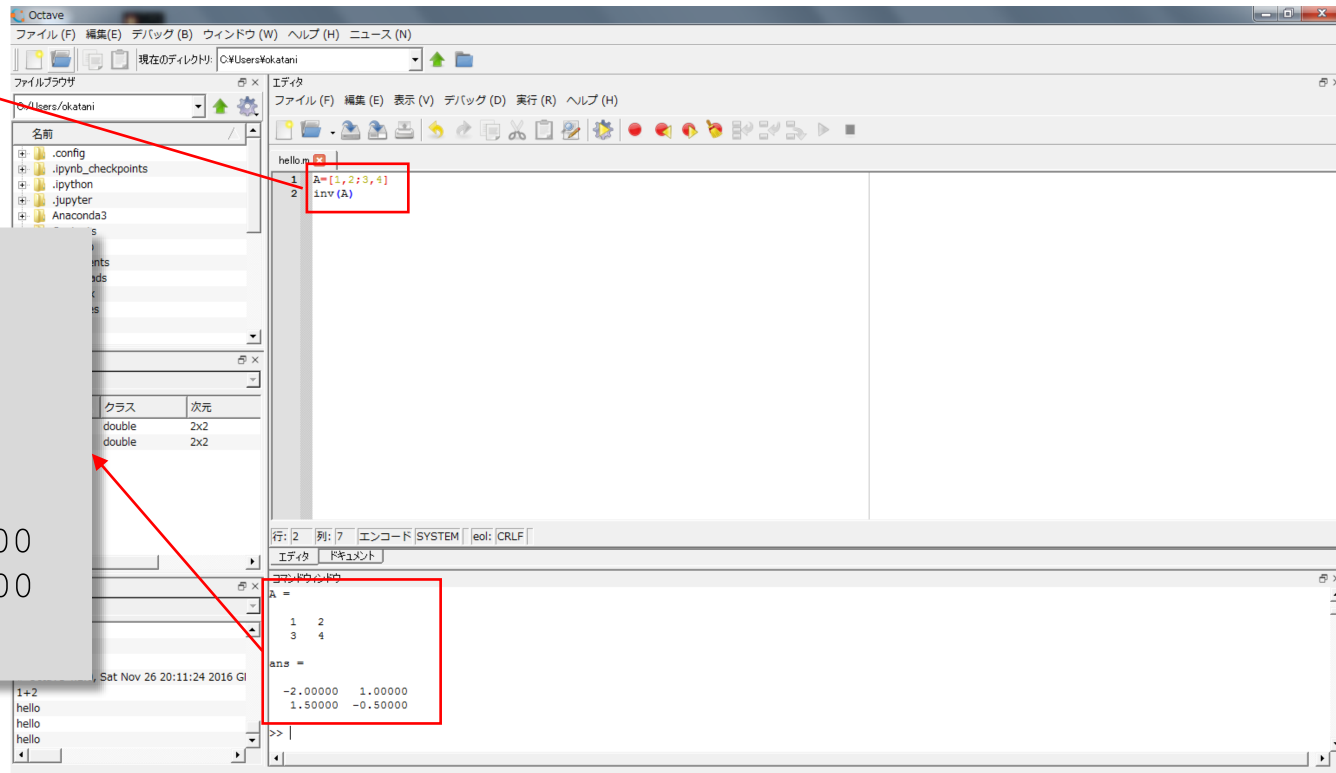


Writing a script file

- Type as follows in the Editor window, select “Save File”-“File” in the Editor window menu, type “hello”, and click “Save”
  - The script should be saved as “hello.m”
- Type “hello” followed by Enter to run the contents
  - Same as choosing “Save File and Run”-”Run” in the menu

```
A=[1,2;3,4]
inv(A)
```

```
>> A=[1,2;3,4]
A =
   1   2
   3   4
>> inv(A)
ans =
  -2.00000   1.00000
   1.50000  -0.50000
```



Using variables

- You can create and use a variable like `A` in the earlier example
  - The name of a variable should be different from existing files and variables
  - There is no limitation in the length of variable names; it must be less than 19 characters in MATLAB, though

```
>> the_1st_variable=[1;2];
>> the_1st_variable
the_1st_variable =

   1
   2
```

- Numeric characters and '_' (underscore) can be used for variable names
- Result won't be displayed by typing ';'(semicolon) at the end

- All the variables you created so far will be displayed in Workspace
- You can remove a variable with the data by typing clear

```
>> clear A
```

Using matrices

- The most fundamental data representation in Octave/Matlab
- A matrix of any size can be created by using ‘,’ to separate elements and ‘;’ to separates rows;

2x3 matrix

```
>> A=[1,2,3;2,3,4]
A =
    1   2   3
    2   3   4
```

3x2 matrix

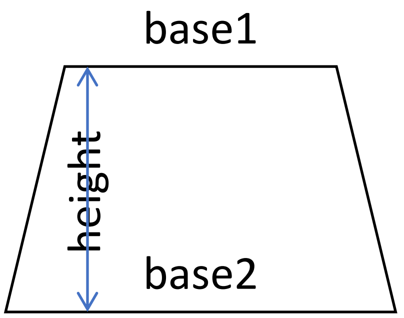
```
>> B=[1,2;2,3;3,4]
B =
    1   2
    2   3
    3   4
```

- You can get the size of a matrix using a built-in function `size`

```
>> size(A)
ans =
    2   3
>> size(B)
ans =
    3   2
```

Arithmetic operation and special values

- Basic operators : `+, -, *, /`




```
>> base1=3.0;base2=5.0;height=3.0;
>> area=(base1+base2)*height/2
area =  12
```

- Exponentiation : ^

```
>> 2^40
ans =    1.0995e+12
```

- π

```
>> pi
ans =  3.1416
```

- Imaginary unit : `i` or `j`

```
>> i
ans =  0 + 1i
>> j
ans =  0 + 1i
>> exp(-pi*i)
ans = -1.0000e+00 - 1.2246e-16i
```

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

(Euler's formula)

Mathematical functions

- Trigonometric functions
  - sin, sinh, asin, cos, cosh, acos, tan, tanh, atan, atan2
- Exponential, log functions, etc.
  - exp, log, log10, sqrt
- Various operations on matrix elements
  - sum, max, min, sort, mod
- Absolute value and complex numbers
  - abs, conj, imag, real

```
>> sin(pi/2)
ans =  1
>> sin(pi)
ans =    1.2246e-16
>> log(e)
ans =  1
```

```
>> A
A =
   1   2   3
   2   3   4
>> sum(A)
ans =
   3   5   7
>> sum(sum(A))
ans =  15
```

```
>> a=2.0-3.0j
a =  2 - 3i
>> imag(a)
ans = -3
>> real(a)
ans =  2
>> abs(-a)
ans =  3.6056
>> conj(a)
ans =  2 + 3i
```

Input and output with files

- You can write the value of a variable into a specified file:

```
>> save('A.txt', 'A')
```

- Then read the written value from the file:

```
>> load('A.txt')
>> A
A =
    1   2   3
    2   3   4
```

```
>> B=load('A.txt')
>> B.A
ans =
    1   2   3
    2   3   4
```

- You can also save/load the whole contents of Workspace into/from a specified file

```
>> save('workspace1')
```

```
>> load('workspace1')
```

Loops

- Repeat a series of commands with `for index=start:step:end … end`

Script

```
# loop1.m
for i=1:10
  x = 2^i;
  printf('%d: %f¥n', i, x)
endfor
```

Result

```
>> loop1
1: 2.000000
2: 4.000000
3: 8.000000
4: 16.000000
5: 32.000000
6: 64.000000
7: 128.000000
8: 256.000000
9: 512.000000
10: 1024.000000
```

```
# loop2.m
# calculate position of a vehicle
# with a constant acceleration
a = 1.0; # acceleration
for t=0.0:0.5:3 # time
  y=.5*a*t^2; # position
  printf('%f: %f¥n', t, y)
endfor
```

```
>> loop2
0.000000: 0.000000
0.500000: 0.125000
1.000000: 0.500000
1.500000: 1.125000
2.000000: 2.000000
2.500000: 3.125000
3.000000: 4.500000
```

Conditional branch & flow control

- if-elseif-else-end structure

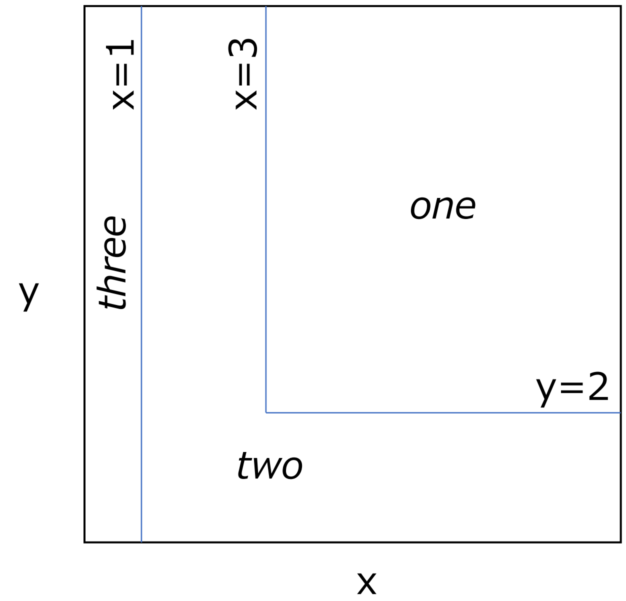
Script

```
#ifelse1.m
if x > 3.0 && y > 2.0
  disp('one')
elseif x > 1.0
  disp(’two')
else
  disp(‘three')
endif
```

Logical AND
(if both are true)

```
#ifelse2.m
if x < 3.0 || y < 2.0
  if x < 1.0
    disp(’three')
  else
    disp(‘two’)
  end
else
  disp(‘one')
endif
```

Logical OR
(if either is true)



Results

```
>> x=4;y=5;
>> ifelse1
one
>> x=2;y=5;
>> ifelse2
two
>> x=y=0;
>> ifelse1
three
```

Comparison Operators

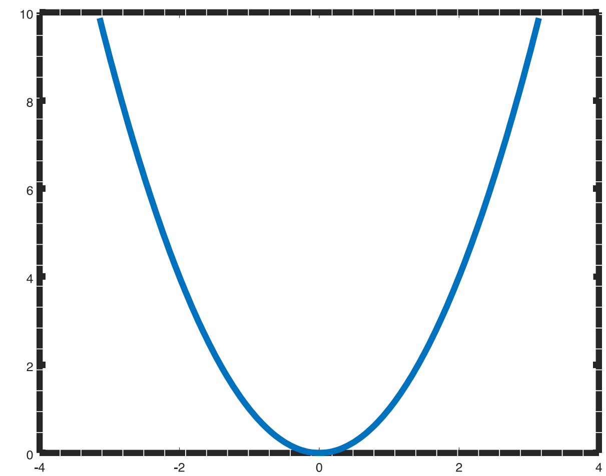
```
x <= y   less than
         or equal
x == y   equal
x >= y   greater than
         or equal
x != y   not equal
```

Plotting a graph

- `plot(x,y)`, where `x` is a vector of length m storing x coordinates and `y` is a vector of the same length storing y coordinates

```
>> x=-pi:pi/100:pi;
>> y=x.^2;
>> plot(x,y)
```

'.^' expresses squaring each element

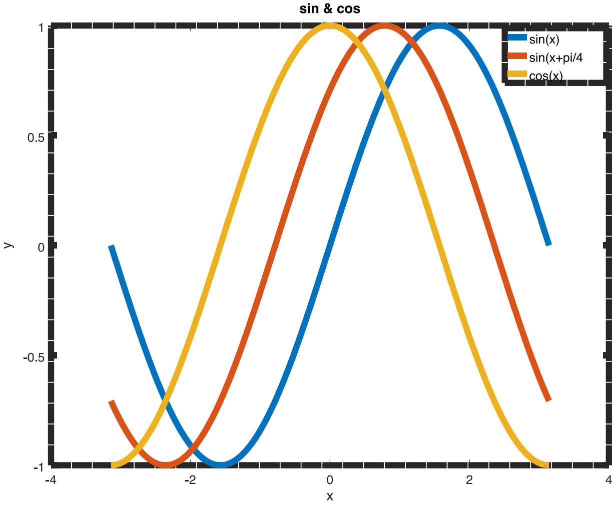


- To plot different curves in a single graph

```
>> plot(x,sin(x),x,sin(x+.25*pi),x,cos(x))
```

- To set axis labels, titles, and legends

```
>> xlabel('x'), ylabel('y'), title('sin & cos')
>> legend('sin(x)','sin(x+pi/4)','cos(x)')
```



- To change font sizes (before calling `plot`)

```
>> set(0,"defaultaxesfontsize",20)
>> set(0,"defaulttextfontsize",20)
```

Exercises 2.1 (assignments)

- Find all numbers of 3 digits such that the sum of the cubes of its digits equals the number itself; an example is 153, because $1^3+5^3+3^3 = 153$
- Revise the script below to find these numbers

```
for i = 100:999
  i1 = mod(i, 10);
  i2 = mod(floor(i/10), 10);
  i3 = floor(i/100);
  disp([i3 i2 i1])
endfor
```

Hint: This script scans every three-digit number and gets its three digits

- Write a script that finds the same numbers in a different way by filling in the blanks below:

```
for i3 = 1:9
  for i2 = 0:9
    for i1 = 0:9



    endfor
  endfor
endfor
```

3. Matrices and linear algebra I

- Accessing elements
- Basic operations
- Norms
- Inverse matrix
- Linear equation

Accessing elements

- As you have learned, ’ ; ’ indicates the end of a row; matrices of any size can be created in this way

```
>> A=[1,2,3;2,3,4]
A =
    1    2    3
    2    3    4
```

```
>> B=[1,2;2,3;3,4]
B =
    1    2
    2    3
    3    4
```

- Specify row and column indices to access an element
- A whole row or a whole column can be represented using ‘:’

```
>> A(2,3)
ans =  4
>> A(1,2)
ans =  2
```

```
>> B(3,:)
ans =
   3   4
```

```
>> B(:,1)
ans =
   1
   2
   3
```

Quick creation of several matrices by functions

- Identity matrix: eye (m)
- Matrix of all 1's: ones(m,n)
- Matrix of all 0's: zeros(m,n)

Remark: You can also use ones(m) and zeros(m) to produce square matrices.

- Matrix of random numbers: rand, randn
  - rand generates random numbers uniformly distributed in the range [0,1]
  - randn generates random numbers from the normal distribution with zero mean and variance one

```
>> eye(3)
ans =
Diagonal Matrix
   1   0   0
   0   1   0
   0   0   1
>> ones(3,2)
...
>> zeros(2,10)
...
```

```
>> rand(3,2)
ans =
   0.562728   0.057675
   0.697043   0.442021
   0.839662   0.310947
>> randn(3,2)
ans =
   1.12010  -0.96770
  -1.36156  -0.45994
   0.38406   2.33878
```

Arithmetic operations on matrices (1/2)

- Addition(+), subtraction(-), transpose(‘)

```
>> A+B`
ans =
    2    4    6
    4    6    8
```

```
>> A'+B
ans =
    2    4
    4    6
    6    8
```

```
>> A+B
error: operator +: nonconformant
arguments (op1 is 2x3, op2 is
3x2)
```

- Mutiplication

```
>> C=A*B
ans =
   14   20
   20   29
```

```
>> D=B*A
ans =
    5    8   11
    8   13   18
   11   18   25
```

- Determinant

```
>> det(C)
ans =  6.0000
>> det(C')
ans =  6.0000
```

```
>> det(D)
ans =    1.7764e-15
```

Arithmetic operations on matrices (2/2)

- Element-wise product (.*) and division (./)

```
>> A.*B'
ans =
    1     4     9
    4     9    16
```

```
>> A./B'
ans =
    1     1     1
    1     1     1
```

- Power of a square matrix (^)

```
>> (A*A')^2
ans =
   596    860
   860   1241
```

```
>> A*A'
ans =
    14   20
    20   29
```

- Element-wise power (.^)

```
>> (A*A').^2
ans =
   196    400
   400    841
```

Norm of vectors and matrices

- Norm of a vector : norm(x,p)

$$\|\mathbf{x}\|_p = \sqrt[p]{\sum_{i=1}^{m} x_i^p} \qquad \mathbf{x} = [x_1, x_2, \cdots, x_m]^\top$$

```
>> x=[1,3,2];
>> norm(x)
ans =     3.7417
>> norm(x,2)
ans =     3.7417
>> norm(x,1)
ans =      6
>> norm(x,inf)
ans =      3
```

$$\|\mathbf{x}\|_2 = \|\mathbf{x}\| = \sqrt{\sum_{i=1}^{m} x_i^2}$$

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{m} |x_i|$$

$$\|\mathbf{x}\|_\infty = \max(x_1, \cdots, x_m)$$

- Norm of a matrix : norm(X,p)
    - E.g., Frobenius norm*

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}^2} = \sqrt{\mathrm{trace}(\mathbf{X}^\top \mathbf{X})}$$

```
>> X=randn(3,4);
>> norm(X,'fro')
ans =
    3,4349
```

```
>> sqrt(trace(X*X'))
ans =
    3.4349
```

*https://en.wikipedia.org/wiki/Matrix_norm

Inverse matrices

- The inverse A^{-1} of a square matrix A can be calculated by inv

```
>> A=randn(3,3)
A =
   0.087948   1.279500   0.060176
  -1.494407  -0.188317  -0.918068
  -1.063032   1.306333   0.734150
>> B=inv(A)
B =
   0.4055585  -0.3289932  -0.4446546
   0.7923708   0.0491297  -0.0035107
  -0.8226907  -0.5637950   0.7245167
>> B*A
ans =
   1.00000   0.00000  -0.00000
  -0.00000   1.00000   0.00000
   0.00000  -0.00000   1.00000
>> A*B
ans =
   1.00000   0.00000   0.00000
   0.00000   1.00000   0.00000
  -0.00000   0.00000   1.00000
```

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

Linear equations

- Use operator '¥' (Gaussian elimination) or inversion inv

```
>> A=[2,2,1;3,-1,3;2,-1,-3]
A =
     2    2    1
     3   -1    3
     2   -1   -3
>> b=[0;3;-1]
b =
     0
     3
    -1
>> A¥b
ans =
     0.19512
    -0.51220
     0.63415
>> inv(A)*b
ans =
     0.19512
    -0.51220
     0.63415
```

Remark: In general, inverse matrices should not be used for solving linear equations, particularly very large ones, from the perspective of computational efficiency and numerical accuracy

A simultaneous equation:

$$2x + 2y + z = 0$$
$$3x - y + 3z = 3$$
$$2x - y - 3z = -1$$

Its vector-matrix notation:

$$\begin{bmatrix} 2 & 2 & 1 \\ 3 & -1 & 3 \\ 2 & -1 & -3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ -1 \end{bmatrix}$$

The solution:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.19512 \\ -0.51220 \\ 0.63415 \end{bmatrix}$$

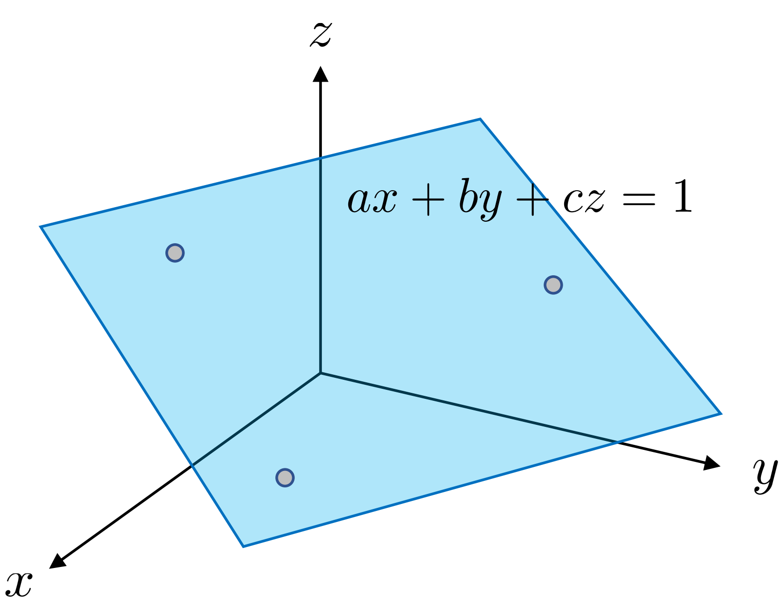
Gaussian elimination*

System of equations	Row operations	Augmented matrix
$\begin{alignedat}{4} 2x &+ y &- z &= 8 \\ -3x &- y &+ 2z &= -11 \\ -2x &+ y &+ 2z &= -3 \end{alignedat}$		$\left[\begin{array}{ccc\|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right]$
$\begin{alignedat}{4} 2x &+ y &- z &= 8 \\ &\tfrac{1}{2}y &+ \tfrac{1}{2}z &= 1 \\ &2y &+ z &= 5 \end{alignedat}$	$L_2 + \frac{3}{2}L_1 \to L_2$ $L_3 + L_1 \to L_3$	$\left[\begin{array}{ccc\|c} 2 & 1 & -1 & 8 \\ 0 & 1/2 & 1/2 & 1 \\ 0 & 2 & 1 & 5 \end{array}\right]$
$\begin{alignedat}{4} 2x &+ y &- z &= 8 \\ &\tfrac{1}{2}y &+ \tfrac{1}{2}z &= 1 \\ & &-z &= 1 \end{alignedat}$	$L_3 + -4L_2 \to L_3$	$\left[\begin{array}{ccc\|c} 2 & 1 & -1 & 8 \\ 0 & 1/2 & 1/2 & 1 \\ 0 & 0 & -1 & 1 \end{array}\right]$
The matrix is now in echelon form (also called triangular form)		
$\begin{alignedat}{4} 2x &+ y & &= 7 \\ &\tfrac{1}{2}y & &= \tfrac{3}{2} \\ & &-z &= 1 \end{alignedat}$	$L_2 + \frac{1}{2}L_3 \to L_2$ $L_1 - L_3 \to L_1$	$\left[\begin{array}{ccc\|c} 2 & 1 & 0 & 7 \\ 0 & 1/2 & 0 & 3/2 \\ 0 & 0 & -1 & 1 \end{array}\right]$
$\begin{alignedat}{4} 2x &+ y & &= 7 \\ &y & &= 3 \\ & &z &= -1 \end{alignedat}$	$2L_2 \to L_2$ $-L_3 \to L_3$	$\left[\begin{array}{ccc\|c} 2 & 1 & 0 & 7 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right]$
$\begin{alignedat}{4} x & & &= 2 \\ &y & &= 3 \\ & &z &= -1 \end{alignedat}$	$L_1 - L_2 \to L_1$ $\frac{1}{2}L_1 \to L_1$	$\left[\begin{array}{ccc\|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right]$

*https://en.wikipedia.org/wiki/Gaussian_elimination

Exercises 3.1

- Suppose we have three points in 3D space and their coordinates are $(x,y,z)=(0.2+r_{x1}, -0.1+r_{y1}, 1.0+r_{z1})$, $(3.0+r_{x2}, 0.1+r_{y2}, -1.0+r_{z2})$, and $(1.0+r_{x3}, -2.0+r_{y3}, -0.5+r_{z3})$, respectively. r is a random number between -0.1 and 0.1. Find a plane passing through these three points. Note that the equation of a plane that does not pass through the origin (0,0,0) is given by $ax + by + cz = 1$



A plane in 3D space passing through three points and not through the origin

Hint : Set up simultaneous linear equations and solve it to determine unknowns (a,b,c)

4. Roots of algebraic and transcendental equations

- Roots of algebraic (polynomial) equations
- User-defined functions
- Roots of transcendental equations
- Symbolic computation

Roots of polynomial equations: roots

- To find the roots of a 2nd order polynomial equation x^2-x-2=(x-2)(x+1)=0, type as follows:

```
>> C=[1,-1,-2];
>> roots(C)
ans =
    2
   -1
```

- Roots of a 3rd order equation x^3+1=0 are calculated as follows:

```
>> C=[1,0,0,1];
>> roots(C)
ans =
   -1.00000 + 0.00000i
    0.50000 + 0.86603i
    0.50000 - 0.86603i
```

User-defined functions

- You can define an arbitrary function by writing a script of the form:

```
function [y1,...,yN] = myfun(x1,...,xM)
y1 = ...
...
endfunction
```

  - Save the following script into, say, “myfun.m”

```
#myfun.m
function y = myfun(x)
  y = x^2+sin(x)-1;
endfunction
```

  - You can call it as a function in the following ways:

```
>> myfun(0)
ans = -1
>> myfun(1)
ans = 0.84147
```

Remark: These commands must be run in the same directory (folder) as myfun.m was saved. Or you can add the directory where myfun.m exists to Octave’s load path; type “help path” for details.

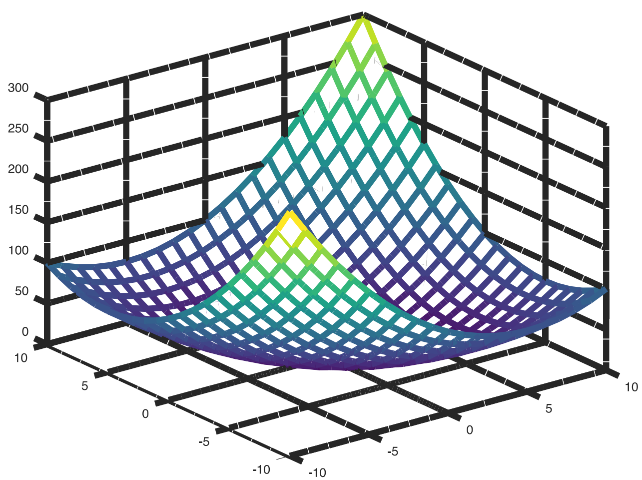
Anonymous function

- You can use *anonymous function*, which is another way of creating a user-defined function

```
>> myfun1 = @(x) (x^2+sin(x)-1);
>> myfun1(1)
ans = 0.84147
```

  - An example of functions with two (and more) variables:

```
>> myfun2 = @(x,y) (x.^2+y.^2+x.*y);
>> [X,Y] = meshgrid(-10:10);
>> mesh(X,Y,myfun2(X,Y))
```

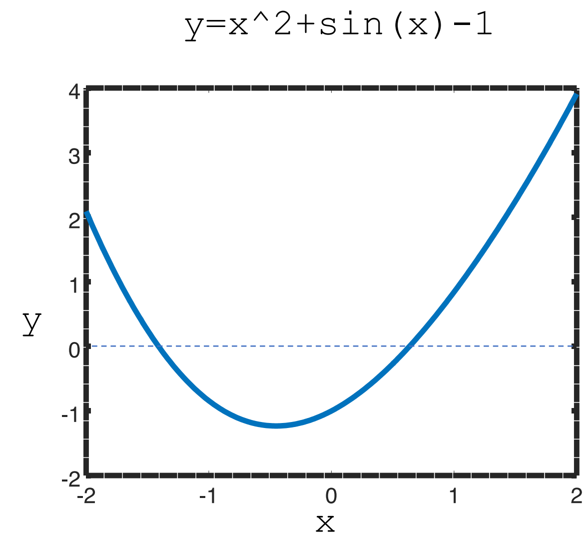


Remark: The use of x.^2 instead of x^2 above makes it possible to deal with the case when x is a matrix (or a vector or even a tensor).

Roots of transcendental equation: fsolve

- To find roots of $x^2+\sin(x)-1=0$, type as follows:

```
>> fsolve(@(x) x^2+sin(x)-1, 1.0)
ans =  0.63673
>> fsolve(@(x) x^2+sin(x)-1, -1.0)
ans = -1.4096
```



  - fsolve tries to find a root starting from given initial value
  - It can fail to find any root; the success depends on the equation and the provided initial values

20.1 Solvers From https://www.gnu.org/software/octave/doc/

Octave can solve sets of nonlinear equations of the form

```
F (x) = 0
```

using the function *fsolve*, which is based on the MINPACK subroutine *hybrd*. This is an iterative technique so a starting point must be provided. This also has the consequence that convergence is not guaranteed even if a solution exists.

Function File: **fsolve** *(fcn, x0, options)*
Function File: *[x, fvec, info, output, fjac]* = **fsolve** *(fcn, …)*

Solve a system of nonlinear equations defined by the function *fcn*.

fcn should accept a vector (array) defining the unknown variables, and return a vector of left-hand sides of the equations. Right-hand sides are defined to be zeros. In other words, this function attempts to determine a vector *x* such that `fcn (x)` gives (approximately) all zeros.

x0 determines a starting guess. The shape of *x0* is preserved in all calls to *fcn*, but otherwise it is treated as a column vector.

Symbolic package

- Extends Octave to enable symbolic computation
  - Function `solve` in MATLAB has not been implemented as of today
- To install `symbolic` package, visit https://github.com/cbm755/octsympy and follow the instruction.
- To use this package, type the following in Command Window:

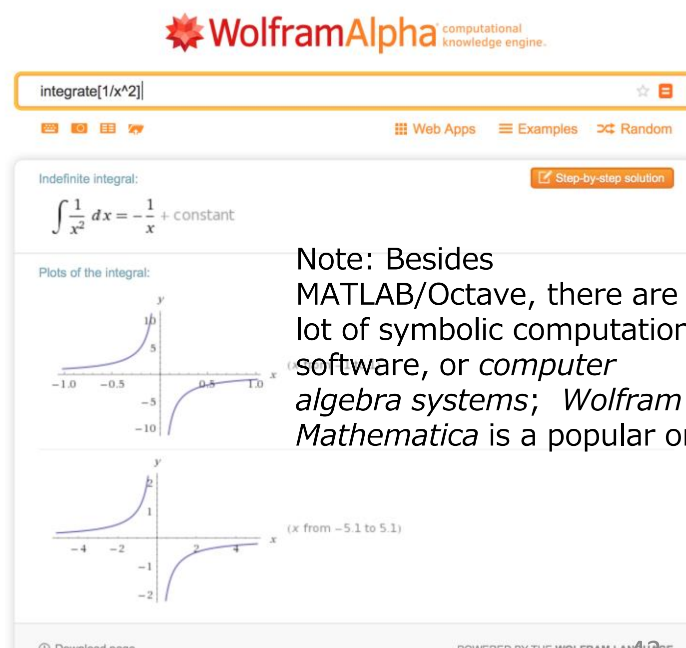
```
>> pkg load symbolic
```

- To start symbolic computation, you must first declare a symbolic variable by syms

```
>> syms x
```

- A symbolic representation of a function:

```
>> x^2+sin(x)-1
ans = (sym)
   2
  x  + sin(x) - 1
```



Note: Besides MATLAB/Octave, there are a lot of symbolic computation software, or *computer algebra systems*; *Wolfram Mathematica* is a popular one

http://www.wolframalpha.com

Symbolic package: factorization

- Factorization of a polynomial: factor

```
>> syms x
>> f=x^3+13*x^2-105*x+171;
>> factor(f)
ans = (sym)
             2
  (x - 3) *(x + 19)
```

```
>> syms x y
>> f=x^3*y-3*x^3-4*x^2*y+12*x^2-3*x*y+9*x+18*y-54;
>> factor(f)
ans = (sym)
             2
  (x - 3) *(x + 2)*(y - 3)
```

Symbolic package: differential

- Symbolic differential: diff

```
>> diff(x^2+sin(x)-1)
ans = (sym) 2*x + cos(x)
```

```
>> diff(exp(-x*sin(x)))
ans = (sym)
                                -x*sin(x)
   (-x*cos(x) - sin(x))*e
```

Remark: If some special characters such as e^x or √ are not displayed properly, try the "sympref display ascii" command to switch to ascii mode.

Symbolic package: indefinite integral

- Indefinite integral : int

```
>> int(x^2+sin(x)-1)
ans = (sym)

    3
   x
   -- - x - cos(x)
   3
```

```
>> int(sin(log(x)))
ans = (sym)

   x*sin(log(x))   x*cos(log(x))
   ------------- - -------------
         2               2
```

Exercises 4.1

- Find all the roots to the following equation

$$10 \cdot sin^2(Ax) \cdot \exp\left(-\frac{Bx}{2}\right) + 0.01(C+D)x - 0.3 = 0, (0 \leq x \leq 5)$$

  - A, B, C, D are constant value, which is identified by your student number.
  - If your student number is ‘C6TB1234’, A=1, B=2, C=3, and D=4.

e.g.) **C 6 T B <u>1</u> <u>2</u> <u>3</u> <u>4</u>**

$$\underset{A}{\overset{\shortparallel}{}}\ \underset{B}{\overset{\shortparallel}{}}\ \underset{C}{\overset{\shortparallel}{}}\ \underset{D}{\overset{\shortparallel}{}}$$

  - Hint: You must specify good initial values to use `fsolve`. To do so, plot the function y=f(x) in the interval [0,5] as follows and make guesses of possible roots.

```
>> x=0:0.01:5;
>> y=10*sin(A*x).^2.*exp(-B*x/2) + 0.01*(C+D)-0.3;
>> y0=zeros(1,length(x));
>> plot(x,y,x,y0)
```

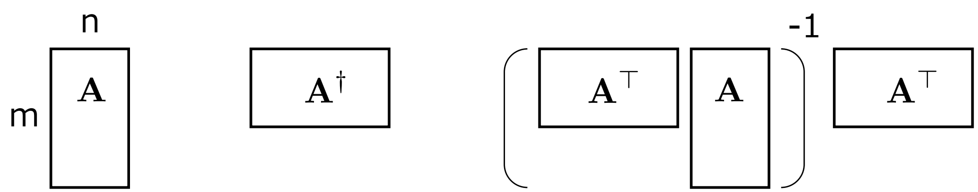
5. Least square method and line fitting

- Pseudoinverse
- Overdetermined system of linear equations
- Line fitting

Pseudoinverse (aka Moore-Penrose pseudoinverse or generalized inverse)

- Assuming that a $m \times n$ matrix A is a real matrix and $A^T A$ is invertible, the pseudoinverse $A^{\dagger}$ for matrix A is defined to be

$$\mathbf{A}^{\dagger} \equiv (\mathbf{A}^{\top}\mathbf{A})^{-1}\mathbf{A}^{\top}$$



- The following always holds:

$$\mathbf{A}^{\dagger}\mathbf{A} = \mathbf{I}$$

  - This is because:

$$\mathbf{A}^{\dagger}\mathbf{A} = ((\mathbf{A}^{\top}\mathbf{A})^{-1}\mathbf{A}^{\top})\mathbf{A} = (\mathbf{A}^{\top}\mathbf{A})^{-1}(\mathbf{A}^{\top}\mathbf{A}) = \mathbf{I}$$

- Note that if $m \neq n$, the following always holds:

$$\mathbf{A}\mathbf{A}^{\dagger} \neq \mathbf{I}$$

Calculating a pseudoinverse

- Function pinv gives the pseudoinverse of a given matrix

```
>> A=randn(5,3)
A =
   -1.000354   0.027611   0.065035
   -3.013282  -0.687265  -0.462170
   -1.345817  -0.410357   1.915242
   -0.480726   0.027323   1.544261
   -0.512782   0.230256  -0.269629
>> pinv(A)
ans =
   -0.3005504  -0.1638335   0.0394693  -0.1490451  -0.3649408
    1.1103074  -0.5201691  -0.5397881   0.7475318   1.6065569
    0.0075412  -0.1606289   0.2720726   0.2571976  -0.0259860
```

- The left multiplication to A yields an identity matrix

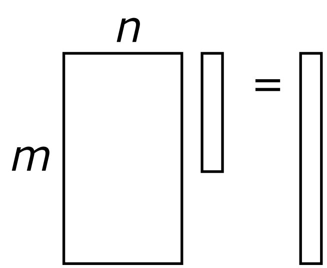
```
>> pinv(A)*A
ans =
    1.0000e+00   2.7756e-16  -1.5266e-16
   -5.5511e-16   1.0000e+00   7.2164e-16
    2.9490e-17   7.9797e-17   1.0000e+00
```

Remark: the right multiplication does not yield an identity

Overdetermined system of linear equations

- Consider a system of linear equations with a more number of equations than unknowns
  - A: m x n matrix（m>n）

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$



m>n → Called overdetermined

m<n → Called underdetermined

  - In general, an overdetermined system does not have a solution
- We calculate a “solution” as follows:

$$\mathbf{x} = \mathbf{A}^{\dagger}\mathbf{b}$$

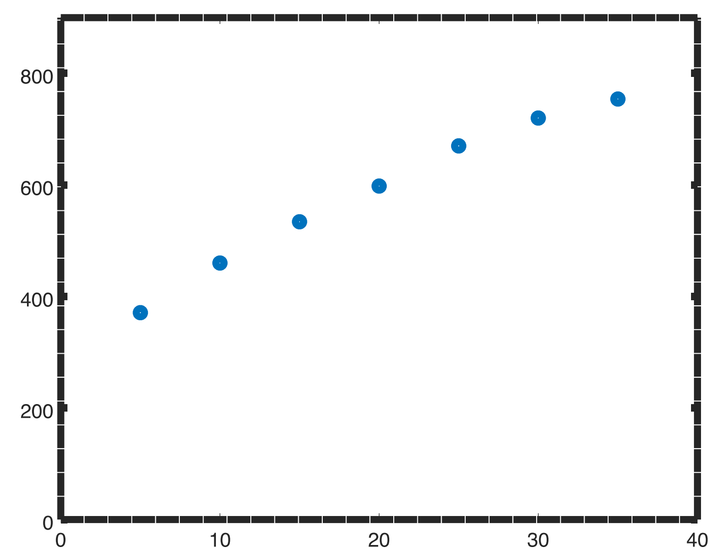
  - It can be shown that this solution x minimizes $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$
  - This solution is thus called the least square solution

Line fitting: an example

- Salary and years of service of employees in Japan

Years of service	<5	<10	<15	<20	<25	<30	<35
Salary (mil. JPY)	370.8	459.4	533.8	597.7	669.7	719.7	753.8

```
>> years=5:5:35
years =
     5    10    15    20    25    30    35
>> income=[371,460,534,598,670,720,754];
>> plot(years,income,”o”)
>> axis([0,40,0,900])
>> set(gca,”fontsize”,14)
```



Line fitting: least square method (1/2)

- Fit a line $y=ax+b$ to a set of points $\{(x_1,y_1), \cdots, (x_N,y_N)\}$ so that the difference in y axis will be small for each (x_i,y_i)

$$\varepsilon_i \equiv \|y_i - \hat{y}_i\| = \|y_i - (ax_i + b)\|$$

- To do so, find (a,b) that minimizes the sum of the differences for all the points

$$\sum_{i=1}^{N} \varepsilon_i^2 = \sum_{i=1}^{N} \|y_i - (ax_i + b)\|^2$$

The right hand side can be rewritten as:

$$\sum_i \|y_i - (ax_i + b)\|^2 = \left\| \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} ax_1 + b \\ ax_2 + b \\ \vdots \\ ax_N + b \end{bmatrix} \right\|^2 = \left\| \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_N & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2$$

$$ax_i + b = \begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

Line fitting: least square method (2/2)

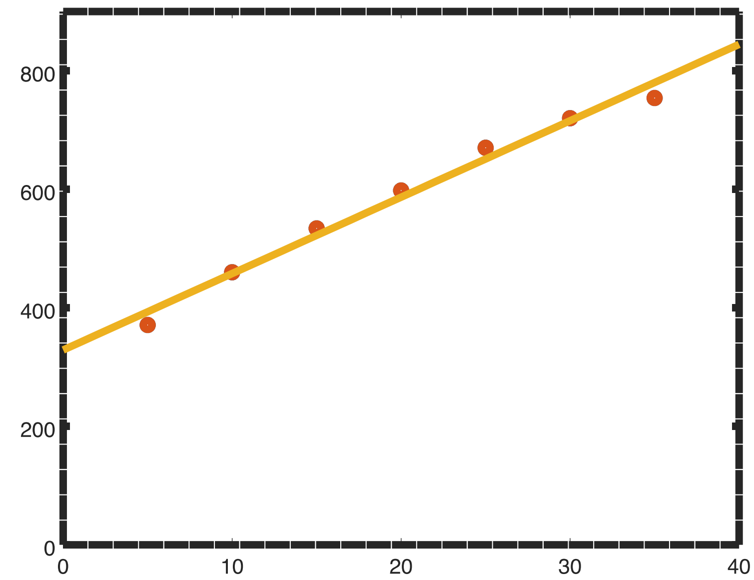
- Thus, the problem reduces to solution of a linear equation Xp=y

$$\|\mathbf{Xp}-\mathbf{y}\|^2 \to \min \qquad \mathbf{X} \equiv \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_N & 1 \end{bmatrix}, \quad \mathbf{p} \equiv \begin{bmatrix} a \\ b \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

- Its solution (i.e., least square solution) is given using pseudoinverse X† as

$$\hat{\mathbf{p}} \equiv \mathbf{X}^{\dagger}\mathbf{y}$$

```
>> X=ones(7,2);
>> X(:,1)=years';
>> y=income';
>> p=pinv(X)*y;
>> hold on
>> xx=0:1:40;
>> plot(xx,p(1)*xx+p(2))
```



Exercises 5.1

- The table to the right shows the number of Nobel laureates per capita (i.e., divided by population) and chocolate consumption per capita for different countries
- It has been discovered that there is a strong link between these two cultural traits (Nobel laureates and chocolate consumption)
    - Franz H. Messerli, Chocolate Consumption, Cognitive Function, and Nobel Laureates, the New England Journal of Medicine, 367, 1562-1564, 2012
- Fit a line to the data and plot the results
    - You can download the file (‘Nobel_vs_choco.txt‘) from Google Class CAPS05 assigment.
- Add an imaginary „CAPS Kingdom“, which has 10×(A+B) Nobel laureates per capita and consumes 0.5×(C+D) kg/y/head of chocolate, then show and plot how the fitted line changes. A, B, C and D are the last 4 digits from your student number (see Excercise 4.1).

	Nobel laureates per capita	Chocolate consumption per capita (kg/y/head)
Sweden	31.855	6.6
Switzerland	31.544	10.8
Denmark	25.255	8.6
Austria	24.332	7.9
Norway	23.368	9.8
UK	18.875	10.3
Ireland	12.706	8.8
Germany	12.668	11.4
USA	10.706	5.1
Hungary	9.038	3.5
France	8.99	7.4
Belgium	8.622	6.8
Finland	7.6	7
Australia	5.451	6
Italy	3.265	3.3
Poland	3.124	4.5
Lithuania	2.836	6.1
Greece	1.857	4.5
Portugal	1.855	4.5
Spain	1.701	3.3
Japan	1.492	2.2
Bulgaria	1.421	2.2
Brazil	0.05	2.5

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3743834/

6. Numerical integration and ordinary differential equation

- Numerical integration (definite integral)
- Double integral
- Initial value problem of ODEs

Numerical integration

- The value of a definite integral can be calculated using quad
- E.g., To calculate the following definite integral:

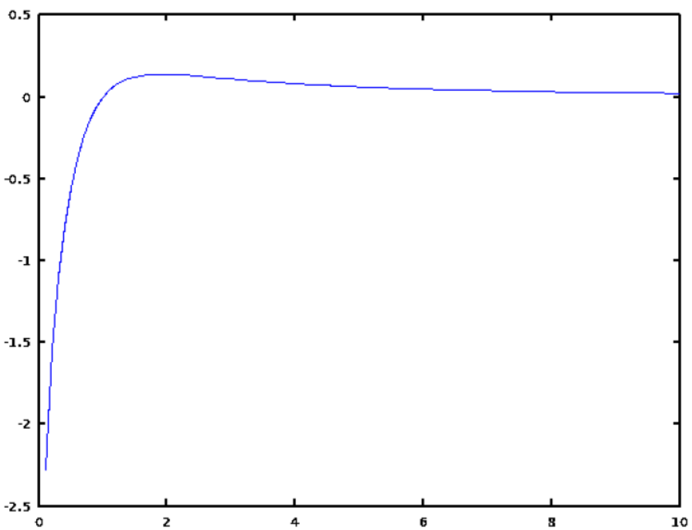
$$\int_0^{10} \frac{\log x}{1+x^2} dx$$

```
>> quad(@(x)(log(x)/(1+x^2)), 0, 10)
ans = -0.32938
```

- You can plot the original function by

```
>> x=0:0.1:10;
>> plot(x, log(x)./(1+x.^2)))
```

Remark: Recall element-wise operations of matrices/vectors have a preceding period, e.g., './' and '.^'



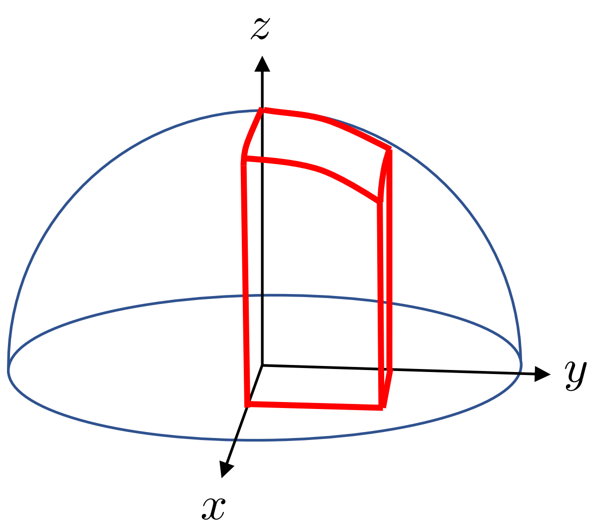
Double integral

- The value of double integral can be calculated using dblquad
- E.g., To calculate the volume of a part of the hemisphere of a unit sphere

$$x^2 + y^2 + z^2 = 1$$

$$z = \sqrt{1 - x^2 - y^2}$$

```
>> dblquad(@(x,y)(sqrt(1-x.^2-y.^2)),0,0.5,0,0.5)
ans =  0.22774
```



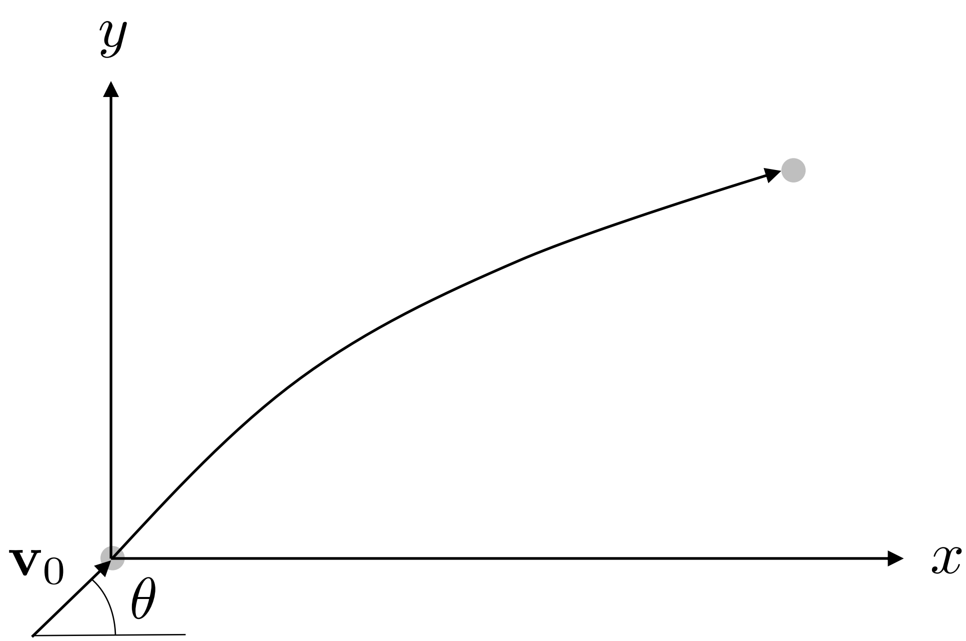
Initial value problem of ODEs

- Four steps to solve an initial value problem of an ODE
  1. Derive differential equations describing the target system
  2. If they are 2^{nd} and higher order ODEs, convert them into a system of 1^{st} order ODEs by incorporating new variables
  3. Create a function (a script file) that calculates the derivatives of the variables from their values and time
  4. Calculate how each variable changes with time using function ode45 by providing it with initial values of the variables and a time interval to consider.

Example

- Suppose that a metal ball with mass m [kg] is thrown into space with elevation angle θ [rad] and initial velocity v_0 [m/s]
- The equation of motion is represented with coordinates (x,y) as

$$\frac{d^2x}{dt^2} = 0$$ (Const. velocity) $$\frac{d^2y}{dt^2} = -g$$ (Standard acceleration due to gravity)



How to solve the example problem (1/2)

- Let (v_x,v_y) be the velocities of the ball in the *x and y* axis, respectively
- Convert the equations in the last page into the 1st order diffenretial eq. wrt. *x, y, v_x*, and v_y

$$\frac{dx}{dt} = v_x \qquad \frac{dy}{dt} = v_y \qquad \frac{dv_x}{dt} = 0 \qquad \frac{dv_y}{dt} = -g$$

- Create a function that calculates these derivatives
  - Let *p* be a 4-vector storing *x, y, v_x, v_y* at time *t*

$$\mathbf{p} = (x, y, v_x, v_y)$$

  - Write a function that calculates the derivative *d***p***/dt* from *t* and **p**

```
function dp = deriv_fun(t, p)
g = 9.81;
dp = [p(3), p(4), 0, -g];
```

$$\frac{d\mathbf{p}}{dt} = \left(\frac{dx}{dt}, \frac{dy}{dt}, \frac{dv_x}{dt}, \frac{dv_y}{dt} \right)$$

How to solve the example problem (2/2)

- Call function `ode45` with a time interval and initial values

Only in older Octave versions

User-defined func. of *d***p***/dt*

Time interval

Initial values of *x, y,* v_x, v_y at *t=0*

$$\mathbf{v}_0 = (v_0 \cos\theta, v_0 \sin\theta)$$

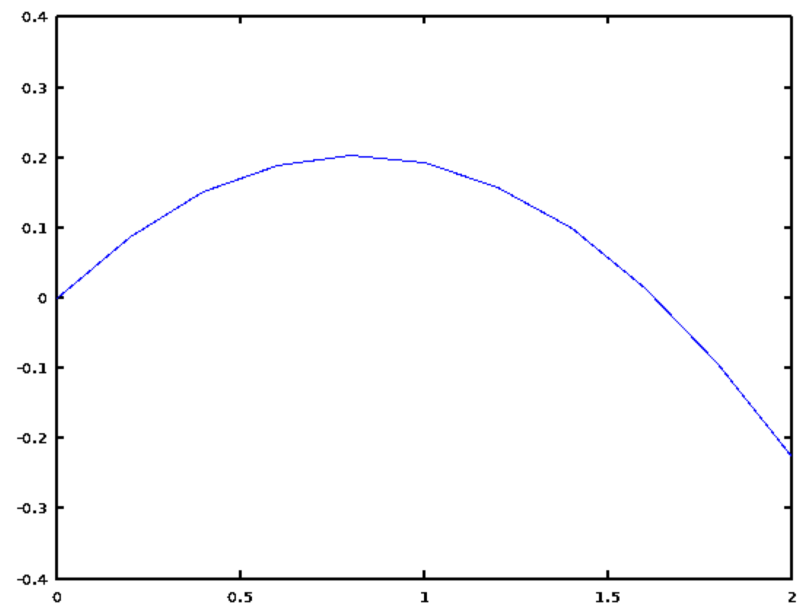
```
>> pkg load odepkg
>> [T, result] = ode45(@deriv_fun, [0,0.5], [0,0,4.0,2.0])
```

Results:

```
warning: Option "RelTol" not set, new value 0.000001 is used
warning: called from    ode45 at line 113 column 5
warning: Option "AbsTol" not set, new value 0.000001 is used
warning: Option "InitialStep" not set, new value 0.050000 is used
warning: Option "MaxStep" not set, new value 0.050000 is used
T =
   0.00000
   0.05000
   0.10000
   0.15000
   0.20000
   0.25000
   0.30000
   0.35000
   0.40000
   0.45000
   0.50000
   0.50000
result =
   0.00000   0.00000   4.00000   2.00000
   0.20000   0.08774   4.00000   1.50950
   0.40000   0.15095   4.00000   1.01900
   0.60000   0.18964   4.00000   0.52850
   0.80000   0.20380   4.00000   0.03800
   1.00000   0.19344   4.00000  -0.45250
   1.20000   0.15855   4.00000  -0.94300
   1.40000   0.09914   4.00000  -1.43350
   1.60000   0.01520   4.00000  -1.92400
   1.80000  -0.09326   4.00000  -2.41450
   2.00000  -0.22625   4.00000  -2.90500
   2.00000  -0.22625   4.00000  -2.90500
```

Plot of a trajectory of the metal ball

```
>> plot(result(:,1), result(:,2))
```

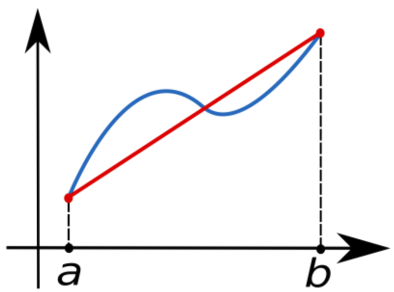


Quadrature rules and Runge-Kutta method*

- Definite integral is numerically computed by several approximation methods, e.g., the trapezoidal rule or Simpson rule

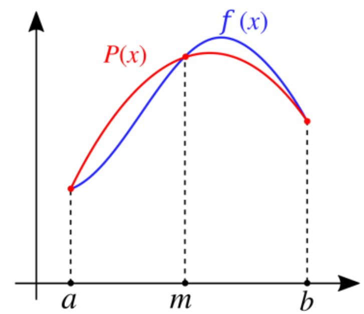
The trapezoidal rule

$$\int_a^b f(x)\,\mathrm{d}x \approx (b-a)\frac{f(a)+f(b)}{2}$$



Simpson rule

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]$$

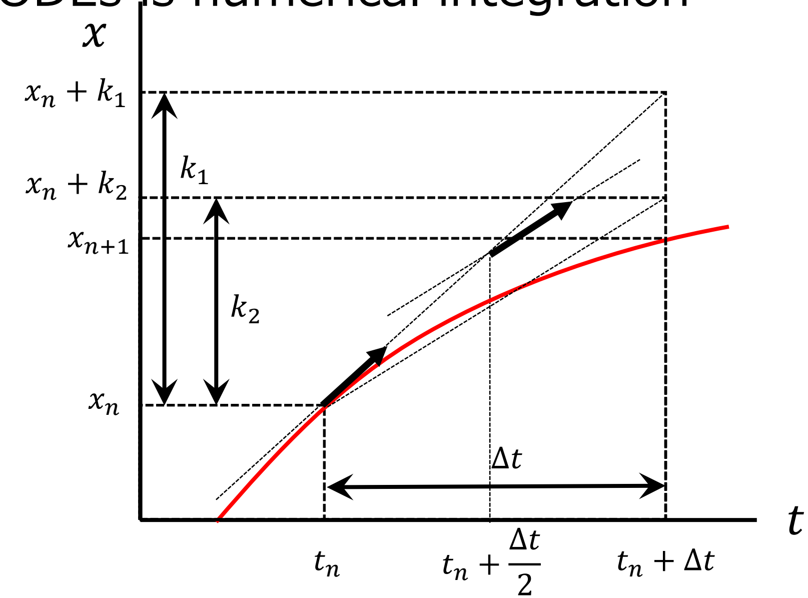


- The core of numerical solutoins to ODEs is numerical integration
  - 2nd order Runge-Kutta method

$$k_1 = \Delta t\, f(t_n, x_n)$$

$$k_2 = \Delta t\, f(t_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}k_1)$$

$$x_{n+1} = x_n + k_2(+O(\Delta t^3))$$



Exercise 6.1

Consider a mass *m, to which* a spring with spring constant *k* and a damper with damping constant *c are attached* as shown in the diagram. Assume that the mass can move only in the *x*. The equation of motion is given by

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = 0$$

When setting *c* to ((your birth month) modulo 3)+1) and *k* to ((your birth day) modulo 7)+1), plot *x*(*t*) with *m=1*, *x(0)=1* and *dx/dt(0)=0*.

E.g., If your birth month and date is 13th August, then *c*=3 and *k*=7