Introduction Script and Matlab

w. 42. Script Introduction and basic (HG)
- Getting to know Script Language and the MATLAB GUI
- Using variables, vectors and matrices
- Using files (input and output)

w. 43. Programming in Script and Matlab (HG)
- For-loops; while-loops; if-then-else loops

w. 44. Various Matlab Applications (KHH/HG)
- Linear systems, matrix algebra
- Numerical integration and differentiation
- Differential equations

w. 45. Project Work – (KHH + HG)

w. 46. Project Work (KHH + HG)

Algorithms and Design Problem

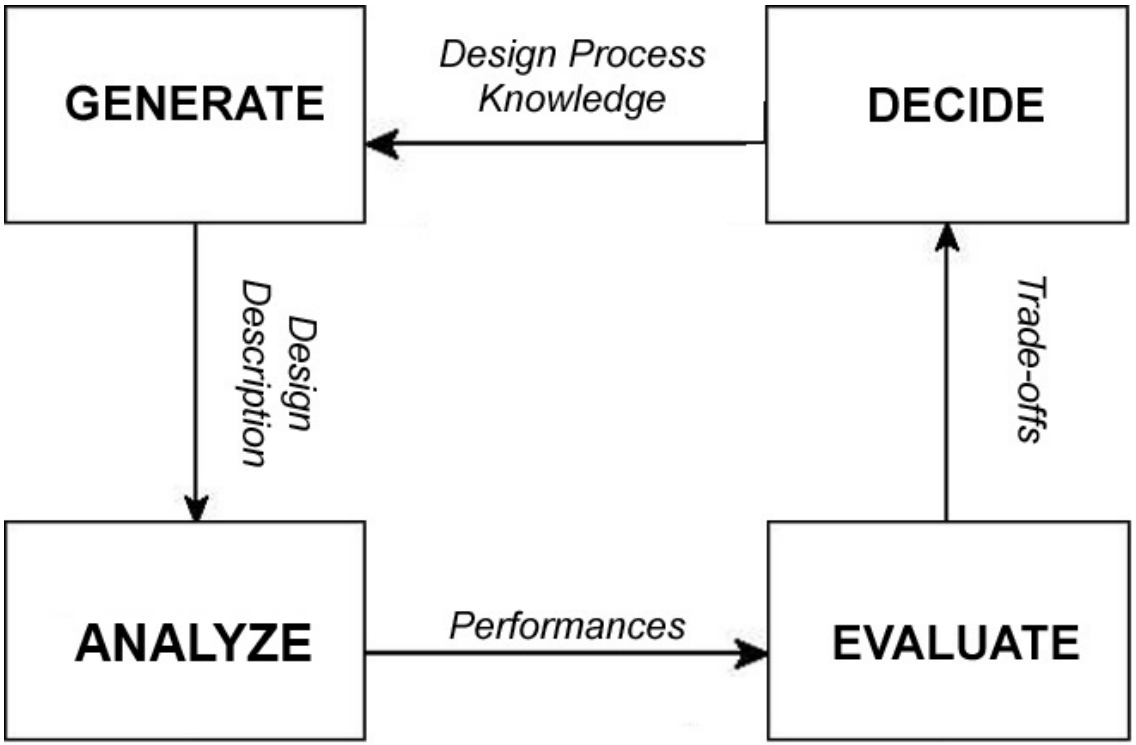
- **Design:** Mapping between form and function

Communication
Data Acquiring
GPS

Offshore Services
Drilling
Anchor Handling

Algorithms and Design Problem

- **Design:** How to Create This Mapping?



Algorithms and Design Problem

- In the rest of your career as designers, you will “generate, analyze, evaluate and decide” many …

Algorithms and Design Problem

- In the rest of your career as designers, you will “generate, analyze, evaluate and decide” many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many many

Algorithms and Design Problem

many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many

Algorithms and Design Problem

many many many many many many many
many many many many many many many
many many many many many many many
many many many many many many many
many many many many many many many
many many many many many many many
many many many many many many many
many many many many many many many
many many many many many many many
many many many many many many many

many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many

Algorithms and Design Problem

many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many many many
many many

Algorithms and Design Problem

many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many
many many many many many many many many

Algorithms and Design Problem

many many many many many many
many many many many many many
many many many many many many
many many many many many many
many many many many many many
many many many many many many
many many many many many many
many many many many many many
many many many many many many
many many many many many times

Algorithms and Design Problem

- In the rest of your career as designers, you will “generate, analyse, evaluate and decide” many times.
- Each design is a new problem to be solved
- You cannot create a program to solve all possible designs, but you can create a program that helps you
- Some tools are available commercially, others are free (open source)

Algorithms and Design Problem

- Matlab is a tool to create tools!
- Other similar high level/script languages
  - Mathematica (commercial)
  - Octave (Free) – based on Matlab
  - Python (Free)
  - Javascript (Free)
- Why Matlab? widely used in engineering, with a large set of “easy to use” modules, making programming easier
- But all programs here here should work there - SYNTAX

Pseudo-code and Syntax

- Pseudo-code: average age of the class:
  - Collect data from all students
  - Sum all the ages
  - Divide by the number of students
- Which “Syntax” is it?

- Are you ok with this syntax?
  - Samle inn data fra alle studenter
  - Sum alle aldre
  - Dele påååå antall studenter

Pseudo-code and Syntax

- Are you ok with this syntax?
  - Coletar a idade de todos os estudantes
  - Somar todos os valores
  - Dividir pelo numero de estudantes

- Another language, another Syntax

Algorithm, Program and Code

- **Algorithm**: A set of instructions or procedures for solving a problem
- **Program**: The set of instructions within a computer which enables it to perform the various tasks required
- **Code**: Any collection of computer instructions (possibly with comments) written using some human-readable computer language, usually as text.

Human Code x Matlab Code

- Collect data from all students

```
all_nam = {'aa','bb','cc'}
all_age = [21, 23, 27]
```

- Sum all the ages

```
sum_age = sum(all_age)
```

- Divide by the number of students

```
num_stu = length(all_age)
average = sum_age/num_stu
```

Control Flow

- Data is already inserted
- We can manipulate it according to our wish to “generate”, “analyse”, “evaluate”, “decide”
- Controlling the data – Control Flow

control flow (or alternatively, flow of control) refers to the **order** in which the individual **statements**, instructions or function calls of an imperative or a declarative program are **executed** or evaluated.

Loops

- For: allows the code to be repeatedly executed

```
for j = 1:4
    j
end
```

- “for i from 1 to 4”, do “show j”, then “end”

For

- Example:
  - For every age, find the number of days alive.
  - Pseudo-code: get every age, multiply by number days in an year (365).

```
for i = 1:num_stu
   all_age(i)*365
end
```

  - Raise your hand and explain the code, please!

For

- Saving our number of days in another vector days(…, …., ….)

```
for i = 1:num_stu
   days(i) = all_age(i)*365;
end
days
```

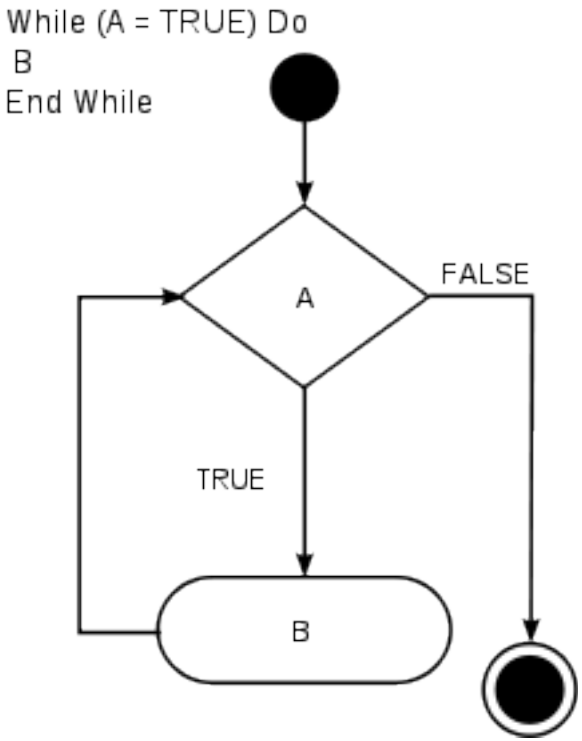
  - Raise your hand and explain the code, please!
  - How to calculate the average of days lived?

Loops

- while: allows the code to be repeatedly executed based on a condition

```
j = 0;
while j < 4
    j
    j = j+1;
end
```

“j equals 0. While j is less than 4, do “show j”. When j >= 4, “end”



While

- Example:
  - Same, but now for every age, find the number of days alive with while loop.
  - Pseudo-code: while I have a non calculated number, get this number, multiply by number days in an year (365). End when I calculated everything

```
i = 1;
while i <= num_stu
   all_age(i)*365
   i = i+1;
end
```

Raise your hand and explain the code, please!

Conditional

- if, else, elseif:
  - Conditional perform different computations or actions depending on whether a programmer-specified condition is true or false.

  - IF condition, then do it
  - ELSE IF other condition, do it
  - ELSE none of the conditions above, do it

Conditional

Example: create a a code that, given an age, it checks if you can buy alcohol in Norway:

- age < 18 – None
- 18 < age < 20 – Alcohol below 22%
- age > 20 – All alcohol allowed

if, else, elseif:

- age < 18 – None
- 18 < age < 20 – Alcohol below 22%
- age > 20 – All alcohol allowed

```
age = 20;
if age < 18
    disp('None is allowed');
elseif age >= 18 & age < 20
    disp('Alcohol below 22%');
else
    disp('All Allowed');
end
```

Conditional

Example: create a a code that checks if you can buy alcohol in Norway, the type of alcohol, if you can enter in a night club, and if you can teach your friend to drive:

- age < 18 – None
- 18 < age < 20 – Alcohol below 22%, no clubbing nor teach
- 20 < age < 21 Alcohol above 22%, but no clubbing nor teaching
- 21 < age < 25 – Alcohol above 22% and clubbing, but no teaching
- age > 25 – All allowed

if, else, elseif:

- age < 18 – None
- 18 < age < 20 – Alcohol below 22%, no clubbing nor teach
- 20 < age < 21 Alcohol above 22%, but no clubbing nor teaching
- 21 < age < 25 – Alcohol above 22% and clubbing, but no teaching
- age > 25 – All allowed

```
age = 20;
if age < 18
    disp('None is allowed');
elseif age >= 18 & age < 20
    disp('Alcohol below 22%,  no clubbing nor teaching');
elseif age >= 20 & age < 21
    disp('Alcohol above 22%,  no clubbing nor teaching');
elseif age >= 21 & age < 25
    disp('Alcohol above 22% and clubbing, no teaching');
else
    disp('All Allowed');
end
```

Codes

- Most of the codes are a combination of control flows of your data
- We can combine for and if, for instance!
- Pseudocode: for each student, check their allowance according to alcohol, clubbing and teaching how to drive

Code

```
all_age =[20, 21, 17, 26, 20];
for i = 1:length(all_age)
        if all_age(i) < 18
                all_age(i)
                disp('None is allowed');
        elseif all_age(i) > 18 & all_age(i) < 20
                all_age(i)
                disp('Alcohol below 22%,  no clubbing nor teaching');
        elseif all_age(i) >= 20 & all_age(i) < 21
                all_age(i)
                disp('Alcohol above 22%,  no clubbing nor teaching');
        elseif all_age(i) >= 21 & all_age(i) < 25
                all_age(i)
                disp('Alcohol above 22% and clubbing, no teaching');
        else
                all_age(i)
                disp('All Allowed');
        end
end
```

Code – adding name

```
all_age =[20, 21, 17, 26, 20];
nam_all = {'Robert', 'Liza', 'Pål', 'Mario', 'Madalena'};
for i = 1:length(all_age)
        if all_age(i) < 18
                text_to_display = [char(nam_all(i)), ' - None' ];
                disp(text_to_display);
        elseif all_age(i) > 18 & all_age(i) < 20
                text_to_display = [char(nam_all(i)), ' -
Alcohol below 22%,  no clubbing nor teaching' ];
                disp(text_to_display);
        elseif all_age(i) >= 20 & all_age(i) < 21
                text_to_display = [char(nam_all(i)), ' -
Alcohol above 22%,  no clubbing nor teaching' ];
                disp(text_to_display);
        elseif all_age(i) >= 21 & all_age(i) < 25
                text_to_display = [char(nam_all(i)), ' -
Alcohol above 22% and clubbing, no teaching' ];
                disp(text_to_display);
        else
                text_to_display = [char(nam_all(i)), ' - All
allowed' ];
                disp(text_to_display);
        end
end
```

Code – adding name

```
all_age =[20, 21, 17, 26, 20];
nam_all = {'Robert', 'Liza', 'Pål', 'Mario', 'Madalena'};
for i = 1:length(all_age)
        if all_age(i) < 18
                text_to_display = [char(nam_all(i)), ' - None' ];
                disp(text_to_display);
        elseif all_age(i) > 18 & all_age(i) < 20
                text_to_display = [char(nam_all(i)), ' -
Alcohol below 22%,  no clubbing nor teaching' ];
                disp(text_to_display);
        elseif all_age(i) >= 20 & all_age(i) < 21
                text_to_display = [char(nam_all(i)), ' -
Alcohol above 22%,  no clubbing nor teaching' ];
                disp(text_to_display);
        elseif all_age(i) >= 21 & all_age(i) < 25
                text_to_display = [char(nam_all(i)), ' -
Alcohol above 22% and clubbing, no teaching' ];
                disp(text_to_display);
        else
                text_to_display = [char(nam_all(i)), ' - All
allowed' ];
                disp(text_to_display);
        end
end
```

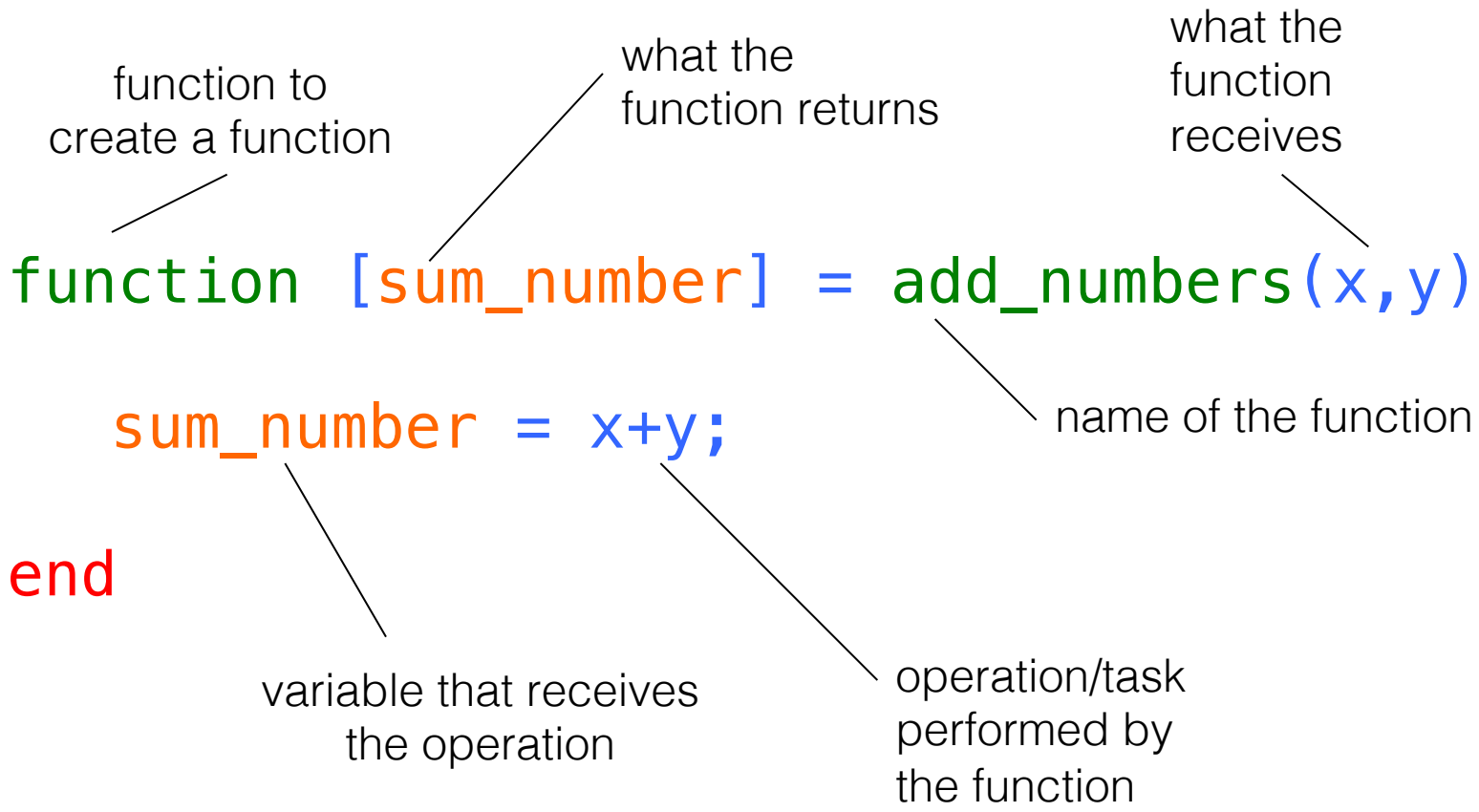
this is getting longer...

Playing with Functions

- function: a named section of a program that performs a specific task.
- In this sense, a function is a type of procedure or routine.
- Some programming languages make a distinction between a function, which returns a value, and a procedure, which performs some operation but does not return a value.
- Matlab: Muste be in another .m file
- Examples: disp(), sum(), length(), plot()...

Functions

- function to add any two numbers:



Playing with Functions

- add_numbers: returns the sum of any two numbers.

```
add_numbers(2,3)
ans = 5

add_numbers(10,32)
ans = 42
```

Example in Class

- Create the following functions:
  - Transform years in days
  - Calculate the average of a list
  - Check if an age can drink/club/teach how to drive
  - Check if a list of ages can drink/club/teach how to drive
  - Check if a list of ages can drink/club/teach how to drive, showing the name of the person
  - Your own function

Design Problem - Crane

- Design a crane, varying square cross section and load. Check if crane collapses (σ_{max} = 250MPa)

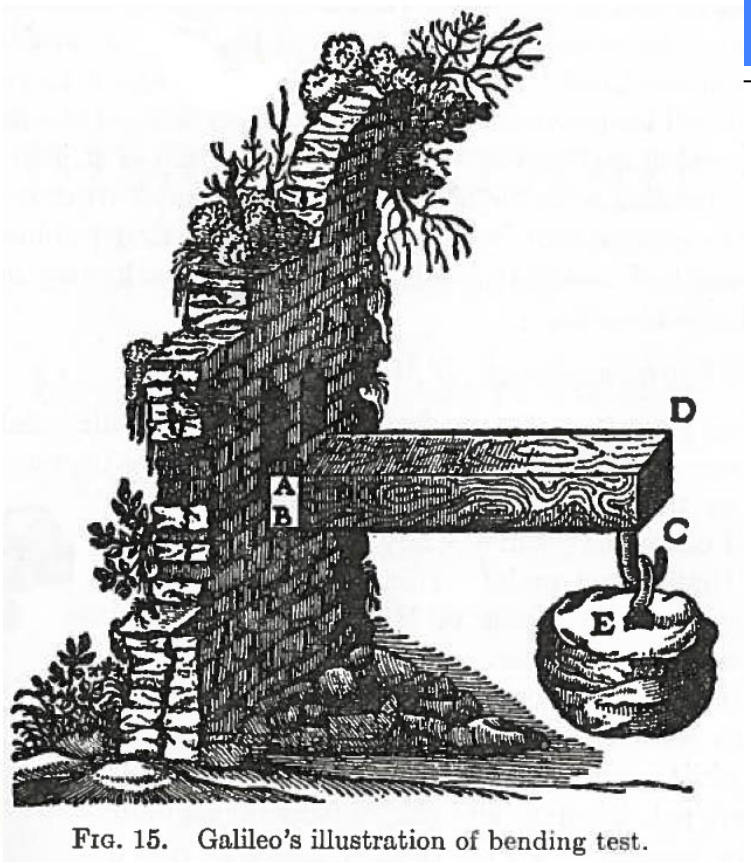
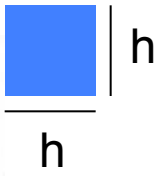
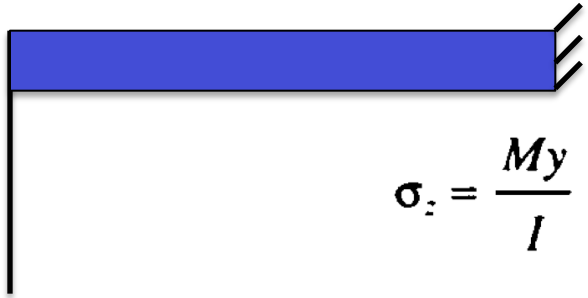


FIG. 15. Galileo's illustration of bending test.





$$\sigma_z = \frac{My}{I}$$

$$M_{max} = M(0) = PL$$

$\sigma = PL*6/h^3$

σ_{max} = 250
L = 3m

Design Problem - Crane

Pseudocode

Create a load P

Create a cross-section h

Create a lenght L

generate

Create a function which calculates σ, given P,L,h

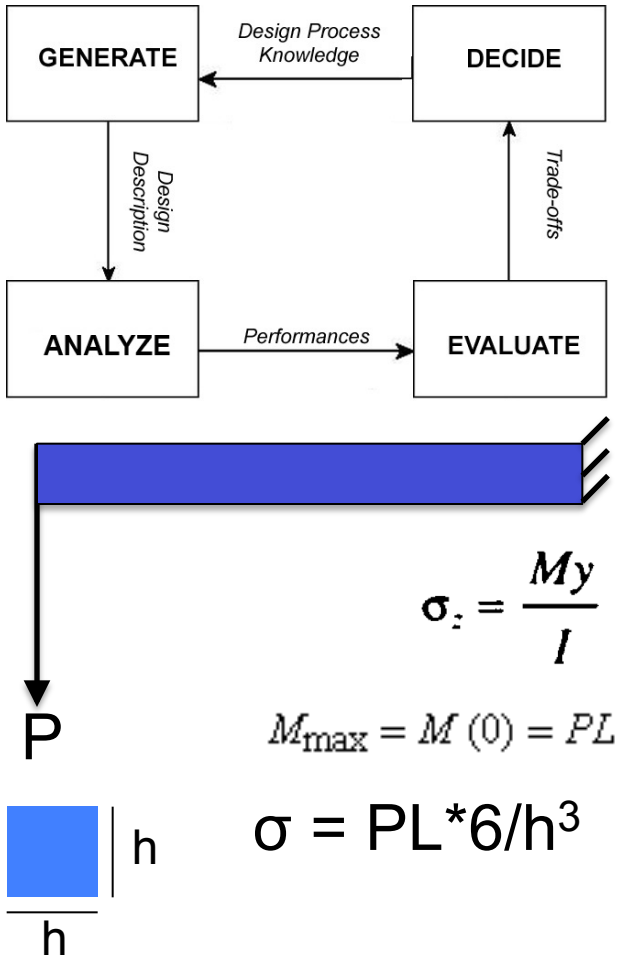
analyse

Test if tension larger than 250MPa, the crane collapses

evaluate

Is this crane ok?

decide



Design Problem - Crane

Create a load P

Create a cross-section h

Create a length L

```
load_c = 10000
section_c = 100
length_c = 3000
```

Create a function which calculates σ given P,L,h

```
function [sigma] = tension(P,L,h)
    sigma = P*L*6/(h^3);
end
```

Test if tension larger than 250MPa, the crane collapses

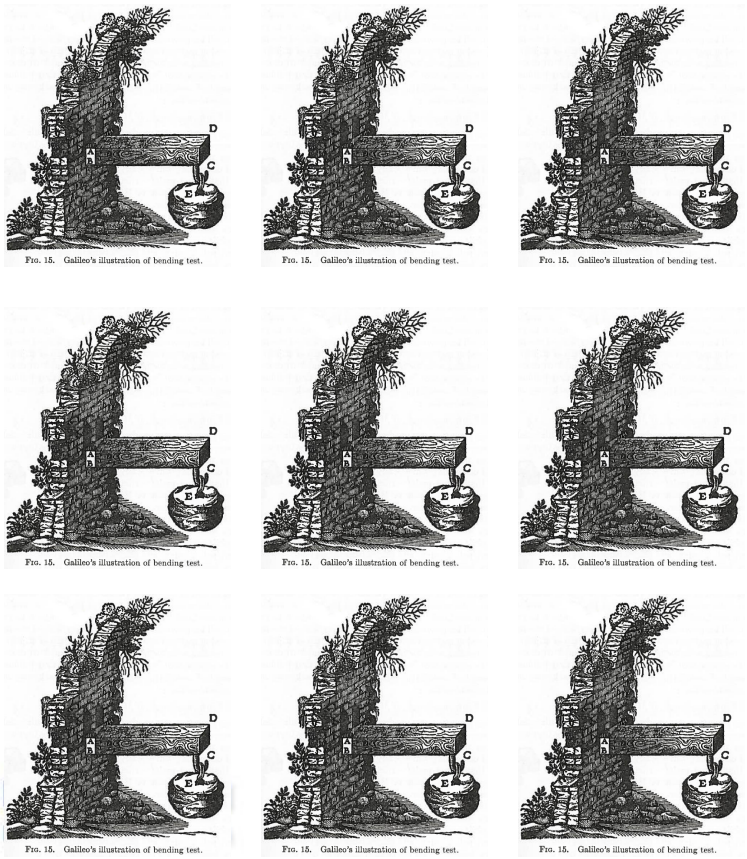
Is this crane ok?

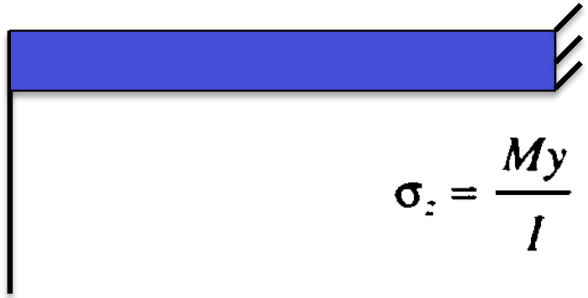
```
sigma_value =
tension(load_c,length_c,section_c)

if sigma_value > 250
    disp('Collapse')
else
    disp('OK')
end
```

1 design
1 condition
1 analysis

Design Problem - Crane

- 1 case – easier to do by hand
- Large number of cases - computer





$$\sigma_z = \frac{My}{I}$$

$$M_{max} = M(0) = PL$$

$\sigma = PL*6/h^3$

$\sigma_{max} = 250$

$L = 3m$

Design Problem - Crane

Example 2:

- Given loads from 10kgf - 1000kgf, test:
  a. h = 40, L = 3000
- plot (P, σ)

```
load_c = 100:100:1000
section_c = 40
length_c = 3000
```

```
function [sigma] = tension(P,L,h)
    sigma = P*L*6/(h^3);
end
```

```
for   i= 1:length(load_c)
    sigma_cal(i) = tension(load_c(i),
length_c , section_c );
end

sigma_cal

plot(load_c,sigma_cal)
```

1 design
10 conditions
10 analysis

Design Problem - Crane

Example 3:

- Given loads from 10kgf-1000kgf, test:

  [L] = 500:500:5000,

  h=40mm

  plot (L, σ) for every P

```
load_c = 100:100:1000
section_c = 40
length_c = 500:500:5000

 function [sigma] = tension(P,L,h)
     sigma = P*L*6/(h^3);
 end
```

```
for   i= 1:length(load_c)
    for j = 1:length(length_c)
            sigma_cal(i,j) = tension(load_c(i),length_c(j),40);
    end
end

hold on
for i=1:length(sigma_cal)
    plot(length_c, sigma_cal(i,1:length(sigma_cal)))
end
```

10 designs
10 conditions
100 analysis

Design Problem - Crane

Example 4:

- Given loads from 10kgf-1000kgf, test:

  [L] = 500:500:5000,
  h=10:10:100

```
load_c = 100:100:1000
section_c = 10:10:100
length_c = 500:500:5000

 function [sigma] = tension(P,L,h)
     sigma = P*L*6/(h^3);
 end
```

```
for   i= 1:length(load_c)
    for j = 1:length(length_c)
            for k = 1:length(section_c)
                    sigma_cal(i,j,k) =
tension(load_c(i),length_c(j),section_c(k));
            end
    end
end
```

100 designs
10 conditions
1000 analysis

Design Problem - Crane

- 1000 designs in a blink of an eye
- How to analyze 10^4? 10^5? 10^{100}?
- Large set of data: lets use computer to filter it!

Largest (Real Double) Matrix and Total Workspace Size by Platform (MATLAB 7.4, part of R2007a)

Operating system and configuration	Total Workspace Size in MB	Largest Matrix Size in MB	Number of Elements in Largest Real Double Array	Number of Elements in Largest int8 Array
32-bit Windows XP	~1700MB	~1189MB	~155e6	~1246e6
32-bit Windows Vista	~1643MB	~1428MB	~187e6	~1497e6
32-bit Windows XP, best case, with 3GB switch	~2700MB	~1536MB	~200e6	~1610e6
32-bit Linux	~2683MB	~2385MB	~312e6	2^{31}-2 (~2147e6)
MAC OS X running 32-bit MATLAB	~2919MB	~1532MB	~200e6	~1606e6
64-bit Windows XP running 32-bit MATLAB	~3155MB	~2047MB	~268e6	2^{31}-2 (~2147e6)
64-bit Linux running 32-bit MATLAB	~3558MB	~2292MB	~300e6	2^{31}-2 (~2147e6)
Solaris running 32-bit MATLAB	~3535MB	~3072MB	~402e6	2^{31}-2 (~2147e6)
64-bit Windows XP, Linux or Solaris running 64-bit MATLAB 7.4 and earlier	~<8TB	16GB (double array) / 2GB (int8 array)	2^{31}-2 (~2147e6)	2^{31}-2 (~2147e6)
64-bit Windows XP, Linux or Solaris running 64-bit MATLAB 7.5 and later	~<8TB	<8TB	2^{48}-1 (~2.8e14)	2^{48}-1 (~2.8e14)

Design Problem - Crane

Check from example 4 how many designs (L,h) are acceptable for all conditions (tension bellow 250MPa)

```
load_c = 10:1000:10000;
section_c = 10:10:100;
length_c = 500:500:5000;
num_desig_ok = 0;
for  i= 1:length(section_c)
    for j = 1:length(length_c)
           design_ok = 1;
                      for k = 1:length(load_c)
                                 sig = tension(load_c(k),length_c(i),section_c(j));
                                 sigma_cal(i,j,k) = sig;
                if sig > 250
                                                design_ok = 0;
                end
            end
           if design_ok == 1
                      text_to_display = ['L - ', num2str(length_c(i)), ', h  -  ',
    num2str(section_c(j)), ' - Design OK'];
                      disp(text_to_display)
            num_desig_ok = num_desig_ok + 1;
           end
    end
end
text_end =['Number of Designs Approved: ',num2str(num_desig_ok)]
disp(text_end)
```

100 designs
10 conditions
1000 analysis

HØGSKOLEN I ÅLESUND

Introduction to MATLAB

w. 41. MATLAB Introduction and basic (KHH)
- Getting to know MATLAB and the MATLAB GUI
- Using variables, vectors and matrices
- Plotting in MATLAB
- Using files (input and output)

w. 42. Programming in MATLAB (HG)
- For-loops; while-loops; if-then-else conditions

w. 43. Various Applications (KHH/HG)
- Linear systems, matrix algebra
- Numerical integration and differentiation
- Differential equations

w. 44. Project 1 – Mechanical problem (KHH)

w. 45. Project 2 – Fleet logistics problem (HG)