Aalto University
School of Engineering

Application Development in Engineering

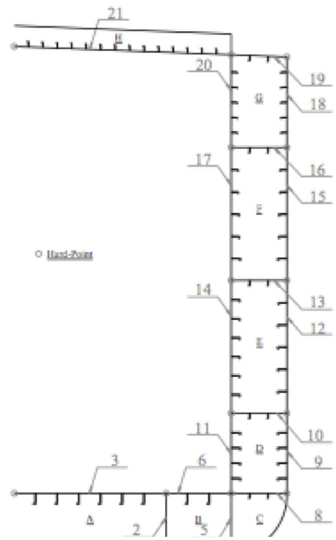
Optimization with Matlab and External Solvers

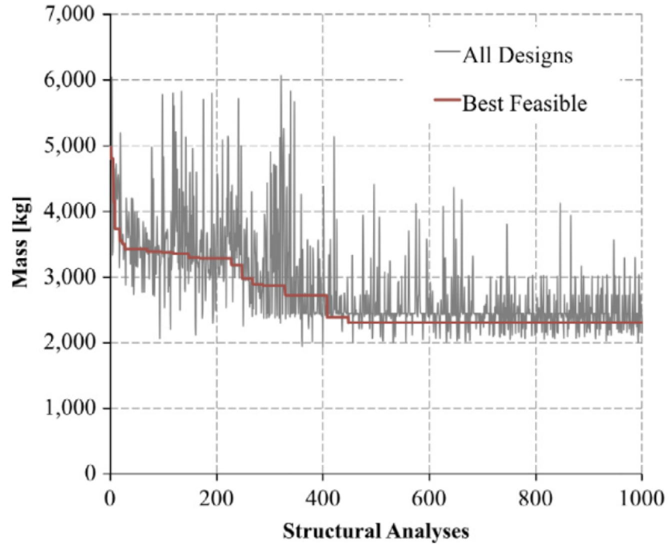
Aalto University
School of Engineering

Contens

- The aim of the lecture is to courage you to use programming, optimization and computational analyses to speed up the design processes
- Motivation
- Exercise
- Contents
  - Flow chart of optimization
  - Pre-processing
  - Analysis
  - Post-processing
- Example of process automatization in Matlab
  - opening and closing a text file
  - making vector(s) from the data of the text file
  - adding stuff to the vector
  - writing vector to text file
  - collecting results to matrix
  - writing matrix to text file
- Literature

1. Romanoff, J., “Optimization of web-core steel sandwich decks at design stage using envelope surface for stress assessment”, Eng Structures, Vol. 66, 2014, pp. 1-9.
2. User manuals: Matlab, Abaqus, Ansys, etc





Motivation

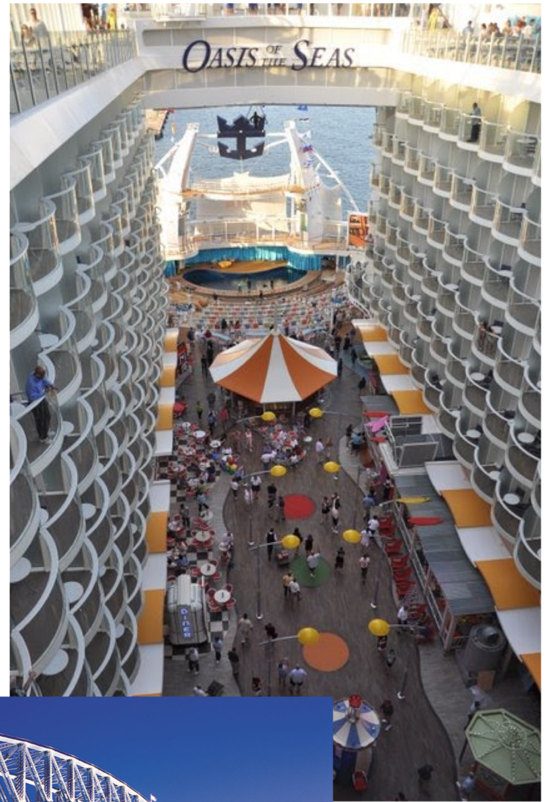
- Analysis of large complex structure/systems involves lots of work
  - Changes due to prototyping and optimization
  - Numerous analyses are needed (vibration, thermo, costs, production, flows, ultimate strength)
  - Numerous documents on analyses to be provided to authorities in form of reports
  - Etc
- Some of these tasks can be automatized ➔ requires programming
- This can be done in all stages of analyses, i.e. pre- and post processing as well as analysis itself
- Most of the solvers have their own programming language
  - Abaqus: Python, Fortran,… (Finite Elements)
  - FEMAP: API/VB (Finite Elements)
  - Etc.
- This can make the solvers sensitive to the format of files, operating system differences (Windows vs. Unix), etc

Exercise

- The idea is to create a simple script in Matlab controls execution of external solver(s) in optimization process. The process includes modifying input, executing the simulation and processing the output.
- Example input files and executables you can obtain by sending email to jani.romanoff@aalto.fi or by using your own ones
- So the script should automatically:
  1. Creates/updates an input file (*.txt, *.dat: e.g. Icore.dat Hint: do not change the length or 1st number that indicates it 19)
  2. Calls external solver (e.g. excel.exe, webcoremain.exe)
  3. Waits until analysis is completed
  4. Reads one of the output files to Matlab and processes the output (e.g. multiply by scalar)
  5. Show how you would perform looping for example in terms of optimization.
- Report
  - The written idea of the code and a flow chart, **(grade 1)**
  - The steps the application performs in commented code and example screenshots **(grades 2-4)**
  - Comment and discuss how well it works and what would be the natural way to extend **(grade 5)** in next stages of your studies

Motivation

- The structures/systems are becoming more advanced and optimized
  - Lightweight
  - Sustainable
  - Safe
- Effectiveness often requires minimization or maximization of property(ies) of the structure under given load cases and constraints
- Optimization is *mathematical method* to find optimal *solution*
  - We need optimization algorithms for search of the optimum
  - We need constraints to make the design feasible in practice
  - The key issue is to balance both constraint assessment and optimization algorithms – cost vs. accuracy
- The key question is what to optimize (dimensions, materials, shape, topology), under which conditions (loads, variable range, rules) and for what objective (mass, cost, safety, all of these)



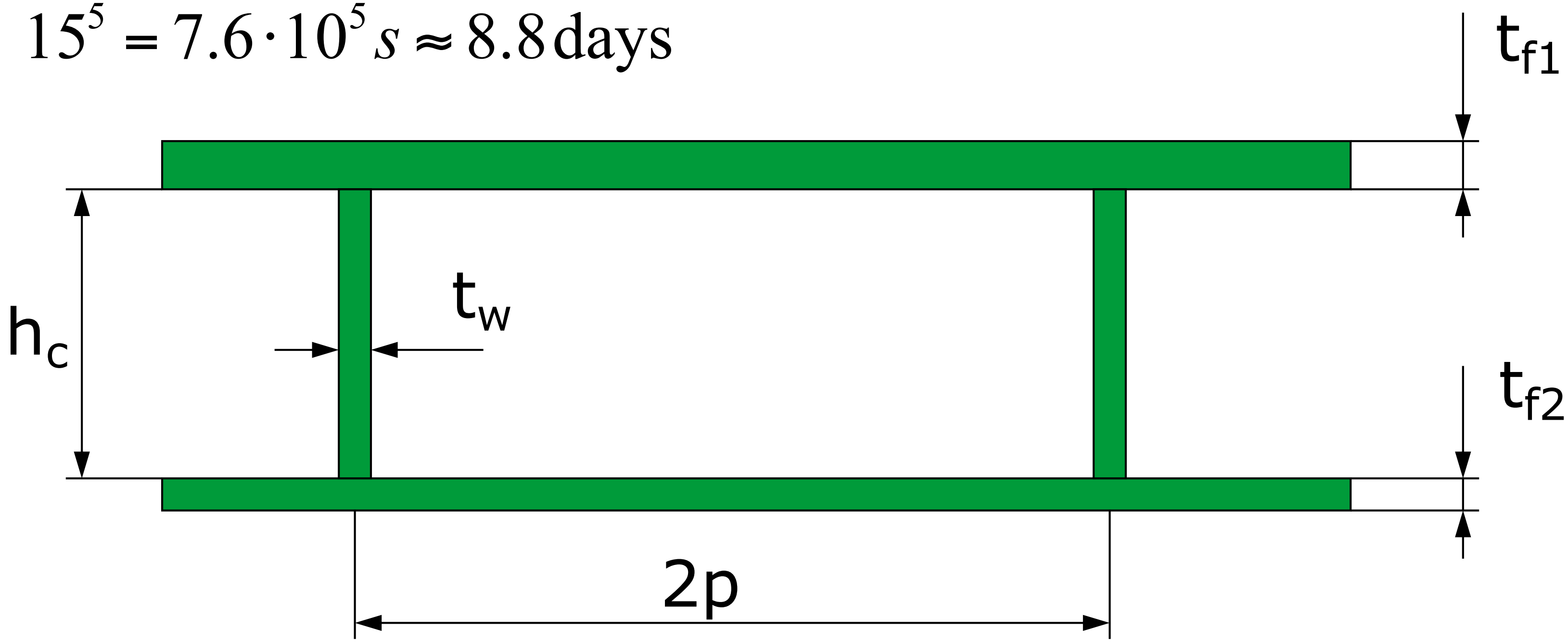
Motivation

5 variable problem

15 different design possibilities for every variable

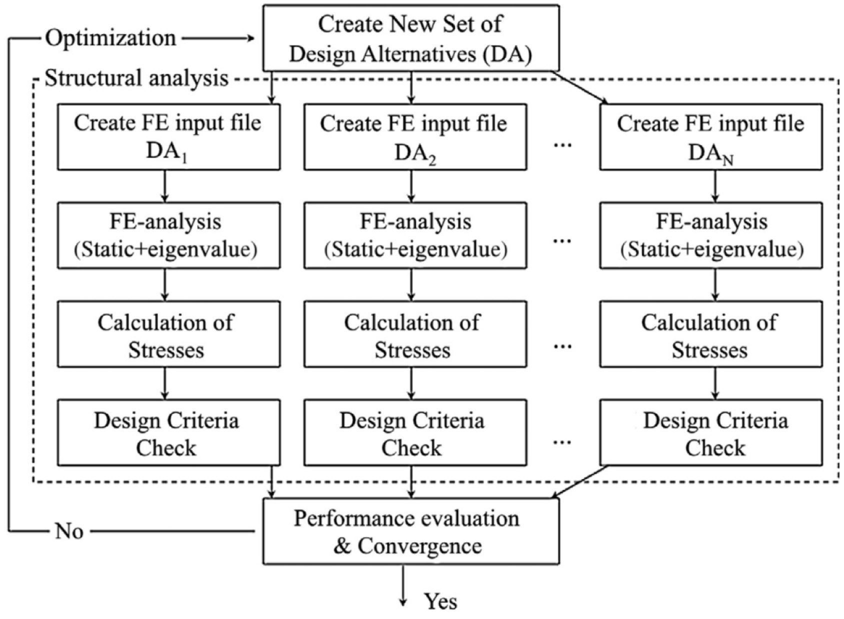
1 *s* for evaluating structural response

$$15^5 = 7.6 \cdot 10^5 s \approx 8.8 \,\text{days}$$



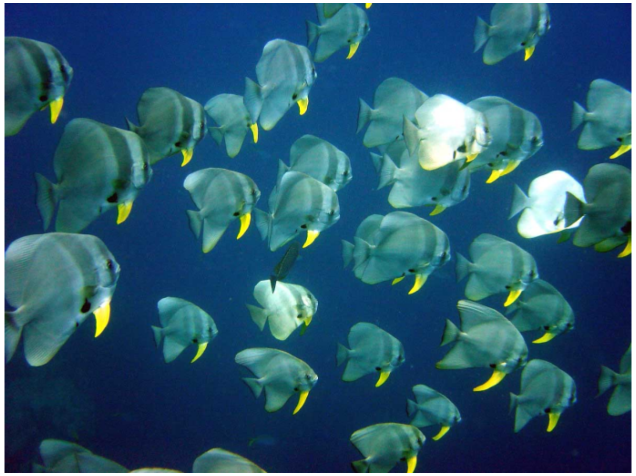
Flow of Optimization

- There are several algorithms available
  - Matlab central
  - Internet
  - Commercial codes, e.g. modeFRONTIER
- Often you need to combine several software to run different types of analyses
  - Flow solution
  - Heat transfer
  - FEA
  - Etc
- Some of these analyses take time and you need to be able to control the process
- We go through an example containing each of these parts and touch the things you need to pay attention to

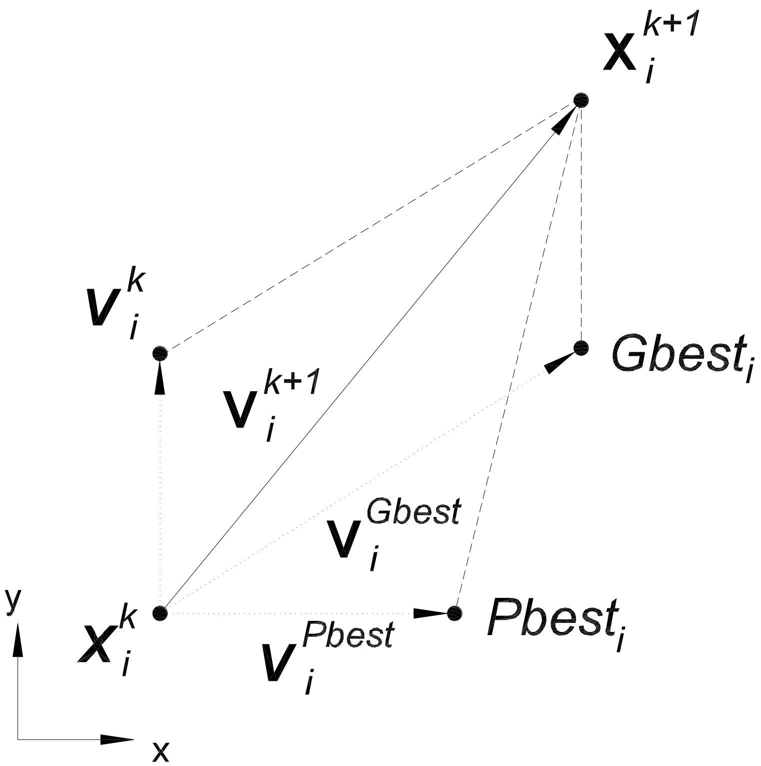


Example of Optimization Algorithm
Particle Swarm Optimization (PSO)

- Population-based optimisation technique developed by Kennedy and Eberhart (1995)
- Belongs to the group of evolutionary algorithms – similar principles as genetic algorithm
- Concept based on bird or fish swarm behavior and how knowledge is tranferred
    - Best particle in current calculation round redirects particles of next round to previous best particle
    - One-way sharing mechanism, which looks only for the best solution only
    - All particles tend to converge to the best solution



PSO



X^k: current location, X^{k+1}: new location

V^k: current velocity, V^{k+1}: new velocity

V^{Pbest}: velocity based on *Pbest*

V^{Gbest}: velocity based on *Gbest*

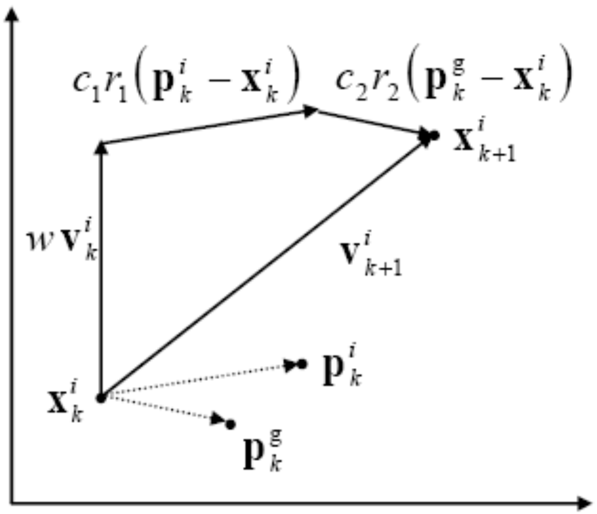
http://www.youtube.com/watch?v=IYLqvfcAzg0&feature=related

Exploration of the design space

- Speed of particle 'i' at iteration 'k' in design space - $\boldsymbol{v}_k^i$
- Particle's best location until iteration 'k' - $\boldsymbol{p}_k^i$
- Swarm's best location until iteration 'k' - $\boldsymbol{p}_k^g$
- Weight factors for the three direction components - w, c_1, c_2

$$\boldsymbol{x}_{k+1}^i = \boldsymbol{x}_k^i + \boldsymbol{v}_{k+1}^i$$

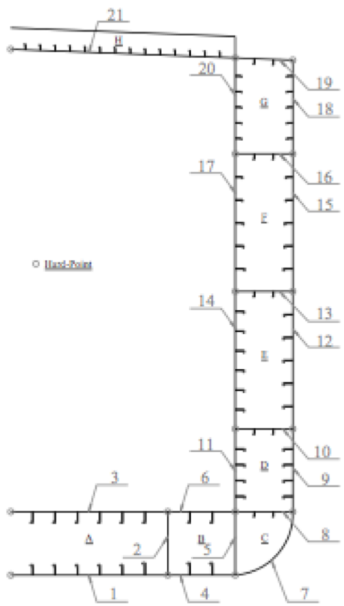
$$\boldsymbol{v}_{k+1}^i = w\boldsymbol{v}_k^i + c_1 r_1 (\boldsymbol{p}_k^i - \boldsymbol{x}_k^i) + c_2 r_2 (\boldsymbol{p}_k^g - \boldsymbol{x}_k^i)$$

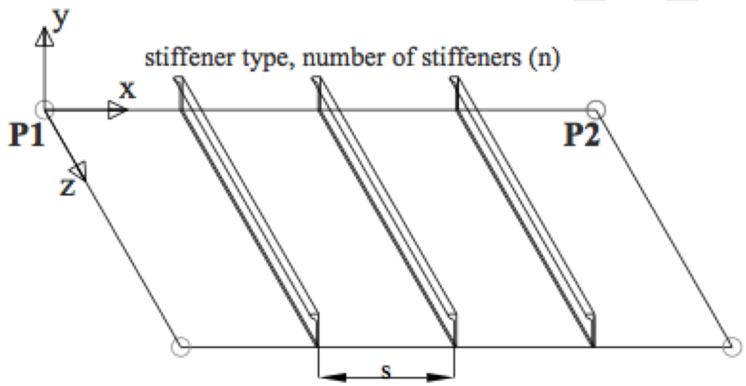


Pre-processing of parametric models

- The large complex structure can be built automatically using parametric modeling

- The logic is
  - Definition of a strake variables
  - Definition of strake lines and key points
  - Extrusion of a strake
  - Making connecting lines between strake end points
  - Defining areas based on lines
  - Meshing the areas
  - Assemble all strakes





Pre-processing of parametric models
Ansys

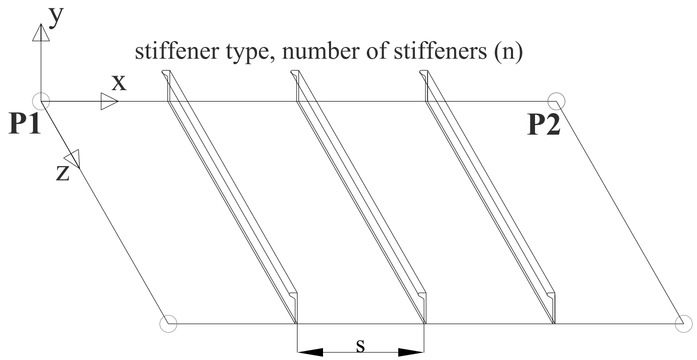


Figure 10. Strake variables

<u>Step 0: Definition of strake variables</u>

xy	x- and y-coordinates of strake hard-points (P1 and P2)
n	number of stiffeners
type	stiffener type defining the height if the stiffener, *h*
t	plate thickness
S	webframe spacing
n_str	number of strakes in the model

Pre-processing of parametric models

Ansys

Step 1: Strake line and keypoints, see Figure 11

```
a=1                                        ! Initial counter for keypoint and line numbering
nr1=0                                      ! Counting variable
count=0                                    ! Counting variable
nr=-1                                      ! Counting variable
*do,i,1,n_str,1                            ! Do loop from 1 to n_str
count=count+1                              ! Increases counter count by one
nr1=nr1+1                                  ! Counting variable is increased by one per loop
k,,xy(1,nr1),xy(3,nr1)                     ! Creates P1 in Figure 11
k,,xy(2,nr1),xy(4,nr1)                     ! Ccreates P2 in Figure 11
WPLANE,,xy(1,nr1),xy(3,nr1),,xy(2,nr1),xy(4,nr1)       ! Creates workplane in P1
CSYS,4                                     ! Places local coordinate system in P1, see Figure 11
lstr,a,a+1                                 ! Creates strake line between P1 and P2
num=n(count)                               ! Reads the correct stiffener number
htype=h(ht(nr1))                           ! Reads the stiffener height from predefined table
ldiv,a,,,num+1                             ! Divides strake line into num+1 lines
lstr,a+2,a+3                               ! Creates line between P3 and P4
ldiv,a+num+1,,,num+1                       ! Divides line between P3 and P4
u=a+3                                      ! Creates counter u
e=a+num+3                                  ! Creates counter e
*do,j,1,num                                ! Do loop to create stiffener lines, see Figure 11
u=u+1                                      ! Increases counter u by one
e=e+1                                      ! Increases counter e by one
lstr,u,e                                   ! Creates line between new points
*enddo                                     ! Closes the do loop
*endif                                     ! Closes the if loop
```

Strake do loop continues…

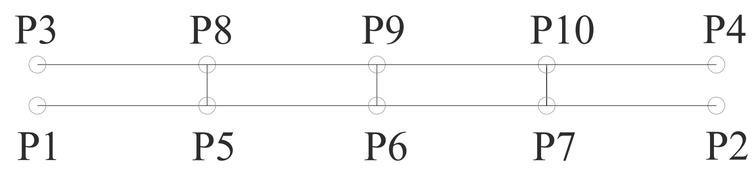


Figure 11. Keypoints and lines

Pre-processing of parametric models

Ansys

Step 2: Extend of the strake in z-direction, see Figure 12

```
kgen,2,a,a+1,1,,,S,,0                   ! Generates P11 and P12 in z-direction
*if,num,gt,0,then                       ! Initiates if loop
kgen,2,a+4,a+3+num,1,,,S,,0             ! Copies P5 to P7 in z-direction
kgen,2,a+4+num,a+3+2*num,1,,,S,,0       ! Copies P8 to P10 in z-direction
*endif                                  ! Closes the if loop
a=a+100                                 ! Sets counter a for next strake
numstr,kp,a                             ! Sets keypoint number to counter a
CSYS,0                                  ! Places coordinate system in origin
*enddo                                  ! Closes the global do loop
```

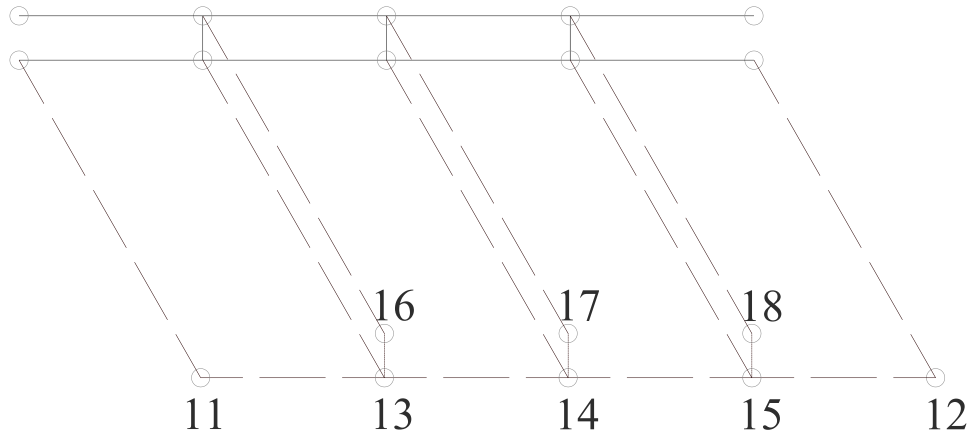


Figure 12. Extend of the strake in z-direction

Pre-processing of parametric models
Ansys

Step 3: Adding lines in z-direction, see Figure 13

```
numstr,line,0              ! sets starting points for lines
a=1                        ! Initial counter for keypoint and line numbering
count=0                    ! Counting variable
*do,i,1,n_str,1            ! Do loop from 1 to n_str
count=count+1              ! Increases counter count by one
num=n(count)               ! Reads the correct stiffener number
*if,num,gt,0,then          ! If number of stiffeners is >0, than 12 to 19 are generated
f=2*(num+2)+a              ! Creates counter f
*else                      ! Otherwise lines 12 to 19 (see Figure 13)
f=a+2                      ! Increases counter f
*endif                     ! Closes the if loop
lstr,a,f                   ! Creates line 12
lstr,a+1,f+1               ! Creates line 13
*if,num,gt,0,then          ! If number of stiffeners is >0 lines are created
f=f+1                      ! Increases counter f by one
e=a+3                      ! Creates counter e
*do,g,1,2*num              ! Start do loop
f=f+1                      ! Increases counter f by one
e=e+1                      ! Increases counter e by one
lstr,e,f                   ! Creates lines
*enddo                     ! Closes the do loop
*endif                     ! Closes the if loop
a=a+100                    ! Sets counter a for next strake
numstr,line,a              ! Sets line number to counter a
*enddo                     ! Closes the global do loop
```

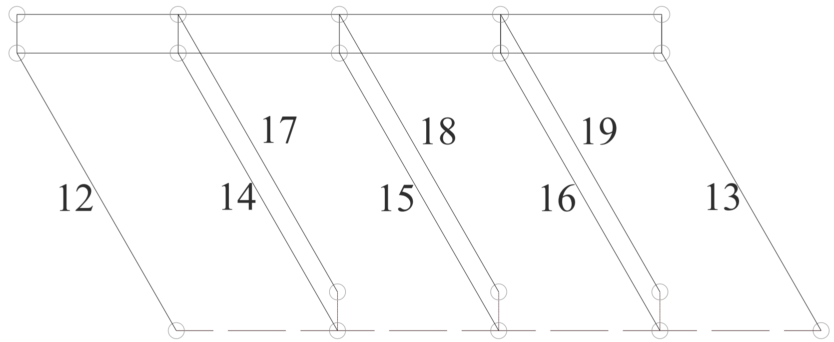


Figure 13. Lines in z-direction

Pre-processing of parametric models
Ansys

Step 4: Defining areas, see Figure 14

```
numstr,line,0                          ! Sets starting points for lines
a=1                                    ! Initial counter for keypoint and line numbering
count=0                                ! Counting variable
*do,i,1,n_str,1                        ! Do loop from 1 to n_str
count=count+1                          ! Increases counter count by one
num=n(count)                           ! Reads the correct stiffener number
*if,num,gt,0,then                      ! If number of stiffeners is >0 areas are created
e=a-1                                  ! Creates counter e
f=a+3*num+1                            ! Creates counter f
k=a+3*num+3                            ! Creates counter k
c=a+3*num+3                            ! Creates counter c
b=a+2*num+1                            ! Creates counter b
count2=0                               ! Creates counter count2
*do,i,1,2*num+1                        ! Starts do loop
count2=count2+1                        ! Increases counter count2 by one
e=e+1                                  ! Increases counter e by one
*if,count2,eq,1,then                   ! Starts if loop
f=f+1                                  ! Increases counter f by one
adrag,e,,,,,,f                         ! Creates area A1, see Figure 14
*endif                                 ! Closes if loop
*if,count2,gt,1,and,count2,le,num+1,then          ! Starts if loop
k=k+1                                  ! Increases counter k by one
adrag,e,,,,,,k                         ! Creates area A2 to A4, see Figure 14
*endif                                 ! Closes if loop
*if,count2,gt,num+1,then               ! Starts if loop
c=c+1                                  ! Increases counter c by one
b=b+1                                  ! Increases counter b by one
adrag,b,,,,,,c                         ! Creates areas A5 to A7, see Figure 14
*endif                                 ! Closes if loop
*enddo                                 ! Closes do loop
*else                                  ! Starts else loop in the case of zero stiffeners
adrag,a,,,,,,a+1                       ! Creates areas A1 to A4, see Figure 14
*endif                                 ! Closes else loop
a=a+100                                ! Sets counter a for next strake
numstr,area,a                          ! Sets area number to counter a
*enddo                                 ! Closes global do loop
```

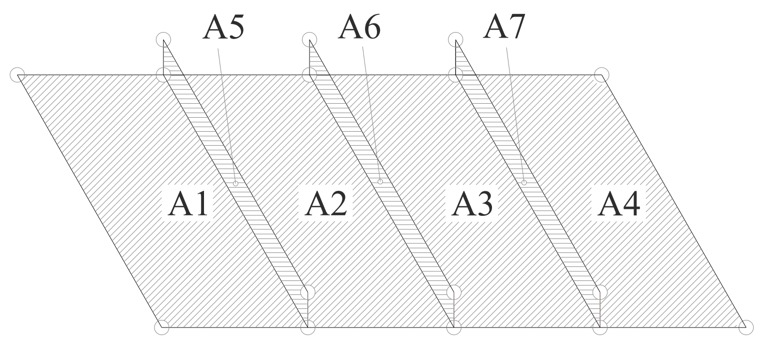


Figure 14. Areas

Pre-processing of parametric models
Ansys

Step 5: Meshing, see Figure 15

```
lsize,all,meshsize          ! Applies predefined meshsize to all lines
type,1                      ! Shell element type for plates
mat,m1                      ! Material for strake plate according to initial table
real,1                      ! Real constant defining the plate thickness
amesh,A1,A2,A3,A4           ! Meshes the plate areas of the strake
mat,m2                      ! Material for stiffeners
real,2                      ! Real constant defining the stiffener thickness
amesh,A5,A6,A7              ! Meshes the stiffener areas of the strake
type,2                      ! Beam element type for stiffeners
real,3                      ! Real constant set for beam cross-section
latt,m2,3,2,,KB,            ! Creates orientation of the unmeshed lines
lmesh,17,19                 ! Meshes lines 17 to 19 (see Figure 13)
```

The iterative nature of step 5 can be achieved by adopting the procedures presented in the previous steps.

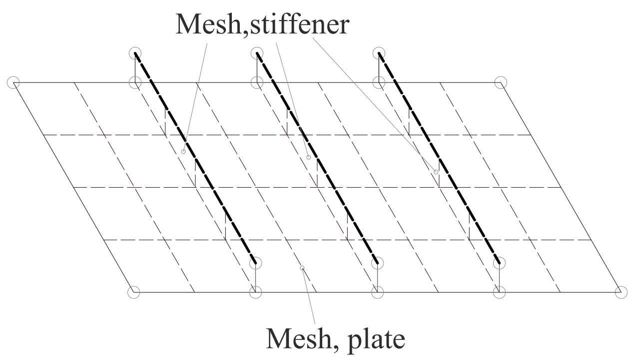


Figure 15. Sketch of meshed strake

Pre-processing of parametric models
Ansys

Step 6: Building the full FE model

The final steps of this modelling procedure include the definition of the transverse members, such as web-frames. To generate those, the line segments surrounding one section of a web-frame are identified and used to obtain the areas to be meshed. The iterative nature presented above can easily be adopted for this process. Finally the single web-frame-spacing model can be copied according to build the full three-dimensional model, see Figure 16.

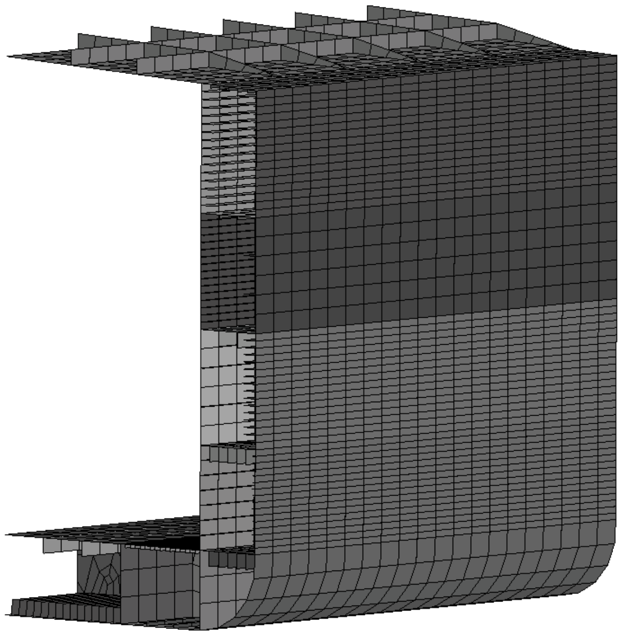


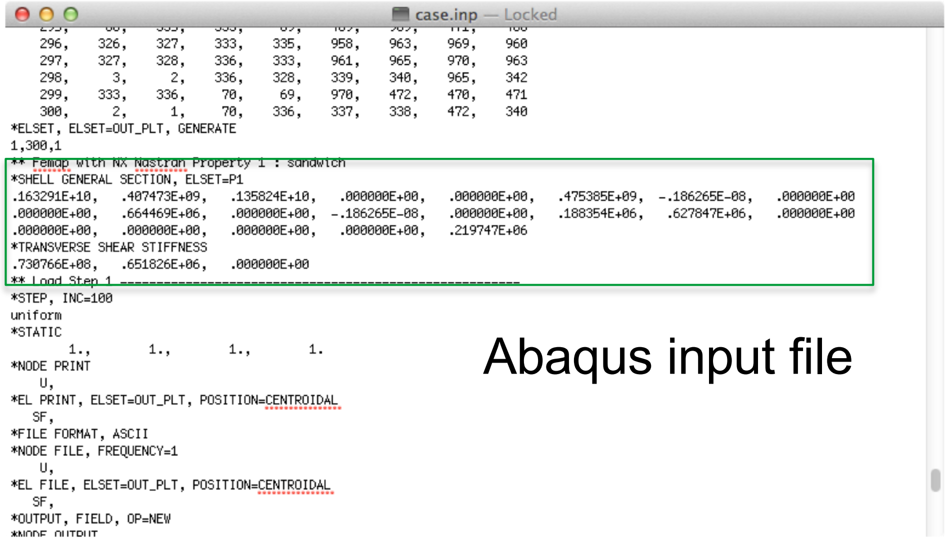
Figure 16. The full finite element model

Pre-processing by changing material definition

- Programming can be also used to change the input file properties
  - Equivalent stiffness of shells (e.g. in scantling optimization)
  - Offset beams and their properties
  - Nodal coordinates in geometrical optimization
  - Etc

- Pay attention to input file format
  - Space
  - Tab
  - Enter

- Process
  1. Create FE mesh
  2. Calculate the equivalent stiffness, e.g. in Matlab
  3. Print the result in right format to input file




```
    296,    326,    327,    333,    335,    958,    963,    969,    960
    297,    327,    328,    336,    333,    961,    965,    970,    963
    298,      3,      2,    336,    328,    339,    340,    965,    342
    299,    333,    336,     70,     69,    970,    472,    470,    471
    300,      2,      1,     70,    336,    337,    338,    472,    340
*ELSET, ELSET=OUT_PLT, GENERATE
1,300,1
** Femap with NX Nastran Property 1 : sandwich
*SHELL GENERAL SECTION, ELSET=P1
.163291E+10,  .407473E+09,  .135824E+10,  .000000E+00,  .000000E+00,  .475385E+09, -.186265E-08,  .000000E+00
.000000E+00,  .664469E+06,  .000000E+00, -.186265E-08,  .000000E+00,  .188354E+06,  .627847E+06,  .000000E+00
.000000E+00,  .000000E+00,  .000000E+00,  .000000E+00,  .219747E+06
*TRANSVERSE SHEAR STIFFNESS
.730766E+08,  .651826E+06,  .000000E+00
** Load Step 1 -------------------------------------------------------
*STEP, INC=100
uniform
*STATIC
       1.,        1.,        1.,        1.
*NODE PRINT
    U,
*EL PRINT, ELSET=OUT_PLT, POSITION=CENTROIDAL
   SF,
*FILE FORMAT, ASCII
*NODE FILE, FREQUENCY=1
    U,
*EL FILE, ELSET=OUT_PLT, POSITION=CENTROIDAL
   SF,
*OUTPUT, FIELD, OP=NEW
*NODE OUTPUT
```

Abaqus input file

Pre-processing by changing material definition
Matlab Creates Input for Abaqus

/Users/jromanoff/Documents/MATLAB/work/webcoreoptimizationtest/equivalentstiffnessplate.asv

```
function [abd,Dt,Dw,Db,Etps,Ewps,Ewrs,Ebps,Gt,Gw,Gb,kQy,massplate,Qt,Qw,Qwr,Qb]=equivalentstiffnessplate(Icore,Pseries,Mseries)

%*******************************************************************
%
%   CALCULATION OF EQUIVALENT STIFFNESS PROPERTIES FOR I-CORE
%   1. Local bending stiffness
%   2. ABD-Matrix (Reissner-Mindlin)
%   3. ABD0-Matrix (Local bending of faces excluded)
%   4. Shear stiffness matrix (1st order)
%   5. Print into file (ABAQUS format)
%
%*******************************************************************

      %---Geometry and materials

      Pdirection=Icore(1);

      tt=Icore(2);
      tw=Icore(3);
      tb=Icore(4);
      s=Icore(5);
      hc=Icore(6);

      Et=Icore(7);
      Ew=Icore(8);
      Eb=Icore(9);

      nut=Icore(10);
      nuw=Icore(11);
      nub=Icore(12);

      kthet=Icore(13);
      ktheb=Icore(14);

      rhot=Icore(15);
      rhow=Icore(16);
      rhob=Icore(17);

      %---area weight for plate
```

plain text file Ln 1 Col 1

Pre-processing by changing material definition

Matlab Creates Input for Abaqus

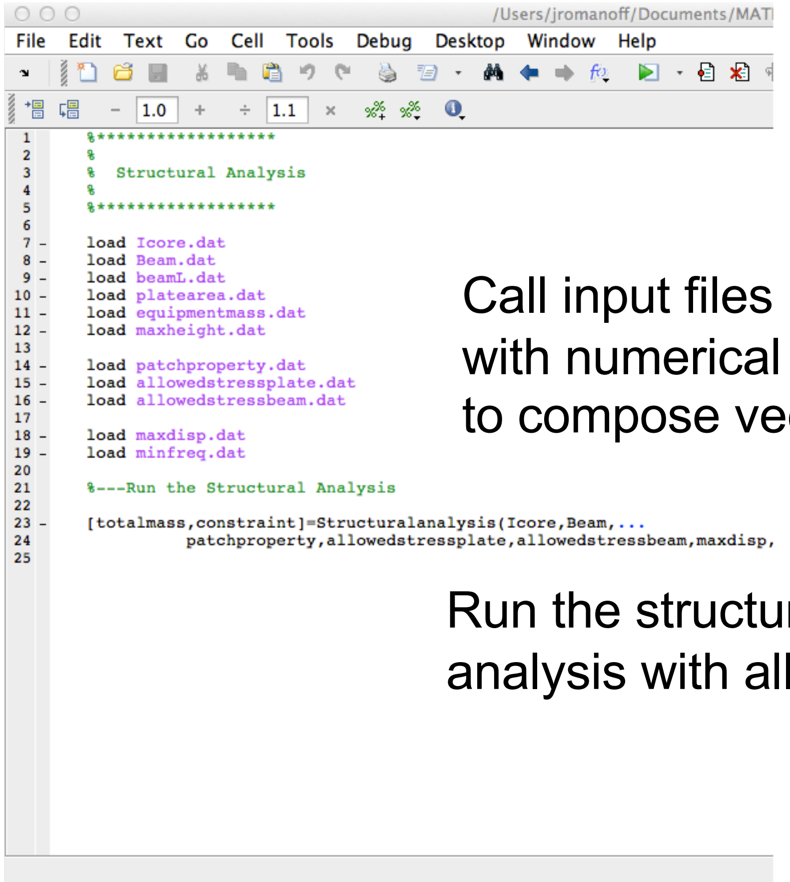
/Users/jromanoff/Documents/MATLAB/work/webcoreoptimizationtest/equivalentstiffnessplate.asv

```
        %------DQx----
        naz=(tt*s*d+1*tw/2*d^2)/(s*(tt+tb)+tw*d);
        Iy=tt*s*d^2+tw/3*d^3-naz^2*(s*(tt+tb)+tw*d);
        kter=Iy^2/(((d-naz)^2*s^3*tt/12+(d/20-naz/4)*d^4*tw+d^3*(-tb*s*naz+tw*naz^2)/3+tb*s*naz^2*d^2+tb^2/tw*s^2*naz^2*d+naz^2*tb/12*s^3)*(s*tt+s*tb+tw
        DQx=kter*((z0-z1)*Gt+(z1-z2)*Gw*tw/s+(z2-z3)*Gb);

        %------DQy----
        kQy=(1+12*(Dt(2,2)/s/kthet-Dt(2,2)/s/ktheb)+6*Dt(2,2)/Dw(2,2)*d/s)/(1+12*Dt(2,2)/Dw(2,2)*d/s+Dt(2,2)/Db(2,2));
        DQy=12*Dw(2,2)/s^2/(kQy*(Dw(2,2)/Db(2,2)+6*d/s)+12*Dw(2,2)/ktheb/s-d/s*2);

        DQtemp(2,2)=zeros;
        DQtemp(1,1)=DQx;
        DQtemp(2,2)=DQy;

        DQ=Txzyz*DQtemp*Txzyzt;

        fidOUT = fopen('stiffness.dat','a');

            fprintf(fidOUT,'%s','**FEMAP with NX NASTRAN Property: sandwich');
            fprintf(fidOUT,'%-10d\n',Pseries);
            fprintf(fidOUT,'%s','*SHELL GENERAL SECTION, ELSET=P');
            fprintf(fidOUT,'%-10d\n',Mseries);
            fprintf(fidOUT,'%-14.8e %-1s %14.8e %-1s %14.8e %-1s %14.8e %-1s %14.8e %-1s %14.8e %-1s %14.8e %-1s %14.8e %-1s\n',ABD0(1,1),',',ABD0(1,2),
            fprintf(fidOUT,'%-14.8e %-1s %14.8e %-1s %14.8e %-1s %14.8e %-1s %14.8e %-1s %14.8e %-1s %14.8e %-1s %14.8e %-1s\n',ABD0(4,3),',',ABD0(4,4),
            fprintf(fidOUT,'%-14.8e %-1s %14.8e %-1s %14.8e %-1s %14.8e %-1s %14.8e\n',ABD0(6,2),',',ABD0(6,3),',',ABD0(6,4),',',ABD0(6,5),',',ABD0(6,6)
            fprintf(fidOUT,'%s\n','*TRANSVERSE SHEAR STIFFNESS');
            fprintf(fidOUT,'%-14.8e %-1s %-14.8e %-1s %-14.8e\n',DQ(1,1),',',DQ(2,2),',',DQ(1,2));

            fprintf(fidOUT,'%s','**FEMAP with NX NASTRAN Property: massshell');
            fprintf(fidOUT,'%-10d\n',Pseries);
            fprintf(fidOUT,'%s','*SHELL SECTION, ELSET=P');
            fprintf(fidOUT,'%-3d',Pseries*2);
            fprintf(fidOUT,'%s',', MATERIAL=M');
            fprintf(fidOUT,'%-3d\n',Pseries);
            fprintf(fidOUT,'%-14.8e %-1s\n',tequivalent,',');

fclose('all');
```

Analysis

- The analysis can be controlled by programming, e.g. in optimization sequence of tasks is important
  - Create new design
  - Analyze the design with FEM for
    - Static
    - Dynamic
  - Extract the FE-results to optimization algorithm

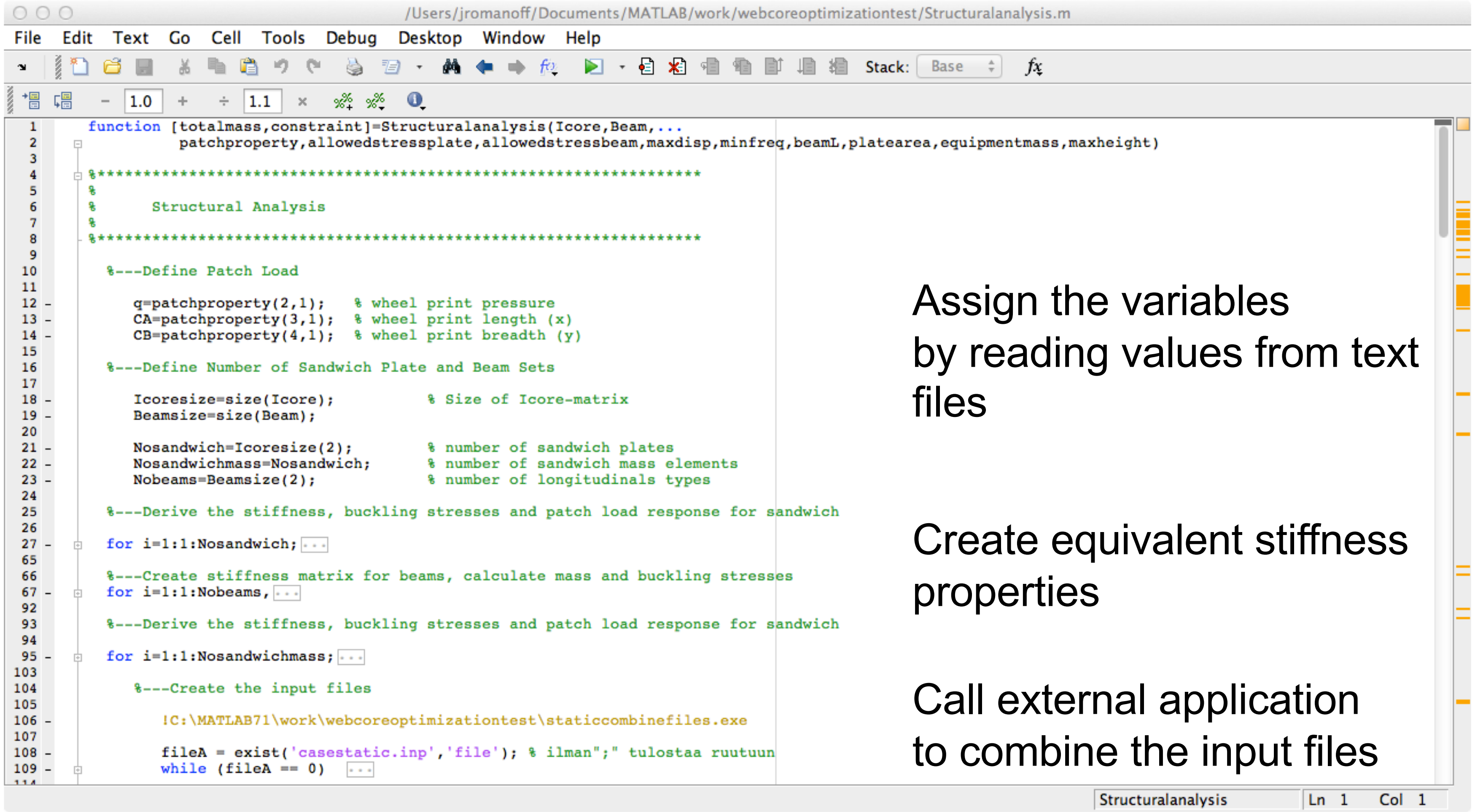
- Another example is that during the analysis user defined material model can be used. This is coded in FE solver programming language



Call input files with numerical info to compose vectors

Run the structural analysis with all tasks

Analysis



Assign the variables by reading values from text files

Create equivalent stiffness properties

Call external application to combine the input files

Analysis

/Users/jromanoff/Documents/MATLAB/work/webcoreoptimizationtest/Structuralanalysis.m

File Edit Text Go Cell Tools Debug Desktop Window Help

```matlab

    timestatic1=clock;
    STATICTIME=60*(timestatic1(1,5)-timestatic0(1,5))+(timestatic1(1,6)-timestatic0(1,6));
    disp(['STATIC ANALYSIS TIME=' sprintf('%4i',STATICTIME) 's'])


    timevib0=clock;

    !C:\Abaqus\6.6-1\exec\abq661.exe job=casevib

    A=0;
    B=0;

    A = exist('casevib.sta','file'); % ilman";" tulostaa ruutuun
    B = exist('casevib.lck','file');
    while (A <= B)

        A = exist('casevib.sta','file');
        B = exist('casevib.lck','file');

    end


    timevib1=clock;
    VIBTIME=60*(timevib1(1,5)-timevib0(1,5))+(timevib1(1,6)-timevib0(1,6));
    disp(['VIBRATION ANALYSIS TIME=' sprintf('%4i',VIBTIME) 's'])

    disp('END OF ABAQUS ANALYSES');

  %---Run ABAQUS script to pick up stress resultant

    !C:\Abaqus\6.6-1\exec\abq661.exe script=sectanddisp.py

    A = exist('sectanddisp.sta','file'); % ilman";" tulostaa ruutuun
    while (A == 0)
        A = exist('sectanddisp.sta','file');
    end

   %---Run ABAQUS script to pick up stress resultant

   !C:\Abaqus\6.6-1\exec\abq661.exe script=freq.py
```

Structuralanalysis Ln 1 Col 1

Run the vibration analysis in Abaqus and wait that it is Completed (identified by .sta & .lck file existence/non-existence)

Collect the results

Post-Processing

- Some tasks in post processing can be automated
  - Critical stress check
  - Critical displacement check
  - Lowest eigenfrequency check
  - Etc
  - Reporting

freq.py — Locked

```
#*****************************************************************************************
***
#
#
#      SCRIPT TO PICK SECTION FORCES IN SHELL ELEMENTS
#
#     Script does following things:
#     1. Picks the section forces Nx, Ny...
#     2. Prints them into different file
#
#*****************************************************************************************
***

outputFiledisplacement = open('freq.txt','w+')
```

Create result file for Matlab

```
# ------Read the output from case.odb

from odbAccess import *

odb = openOdb(path='2d2mesh.odb')
```

Open Abaqus result file

```
#---Picking all load and boundary conditions
a = ['Step-1'] #,'Step-2'
b = [-5, -4, -3, -2, -1]

for x in a:
        for y in b:

                laststep=odb.steps[x]
                #lastFrame=lastStep.frames[-5]
                lastFrame=lastStep.frames[y]


                # frequencydesc=lastFrame.description[0:70]
                frequency=lastFrame.frequency
```

Pick the value of interest

```
                print '***********'

                print frequency

                outputFiledisplacement.write('%s\n'  % (frequency))


#------CLOSE ALL OUTPUT FILES

outputFiledisplacement.close()
```

Close result file for Matlab

Example of process automatization in Matlab

1. Opening a text file "*load textfile.txt*"
2. Making vector(s) from the data of the text file "*a=textfile(1,:)"*
3. Adding stuff to the vector and making matrix
    1. Find length of vector = *L*
    2. Create a new vector with input and output, e.g. *b=…*
    3. Add numbers to vector at location L+1, L+2,…
    4. Collect vector to matrix (i,:), column i with undefined length :
4. Writing matrix to text file (*fopen, fprintf*)

The file type etc depends on the simulation tool, i.e. external solver

Conclusions

- Often we need to run analyses several times during optimization using various external solvers
- This requires automatization, file handling, timing of processes etc
- Matlab is good environment for of controlling such processes:
  – Contains many open source optimization algorithms
  – Can handle external solvers in batch-mode
  – Has good visualization options
  – Can create executable with GUI

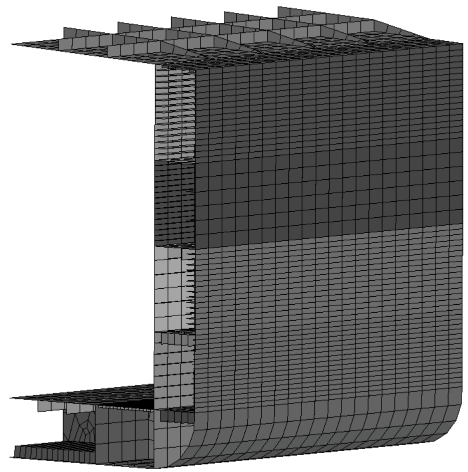


Figure 16. The full finite element model