|  |  |
| --- | --- |
|  |  |
| Plate Buckling Analysis | |
| User Manual | |
| April 25, 2018 | |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rev | Description | Date | Prep | Chk'd | Apprv'd |
| 01 | New Document | 25-April-2018 | VA |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Rev | Description | Section | Pages |
| 01 | New document | - | - |
|  |  |  |  |
|  |  |  |  |

CONTENTS

[1 INTRODUCTION 3](#_Toc512518222)

[2 Plate BUCKLING. 3](#_Toc512518223)

[2.1 Inputs. 3](#_Toc512518224)

[2.2 Calculation. 5](#_Toc512518225)

[3 THEORY 11](#_Toc512518226)

[3.1 Introduction 11](#_Toc512518227)

[3.2 Input and Output Data Specification 12](#_Toc512518228)

[3.2.1 Overview 12](#_Toc512518229)

[3.2.2 Data Flow. 14](#_Toc512518230)

[3.2.3 Data Cleansing. 14](#_Toc512518231)

[3.3 Downhole card. 14](#_Toc512518232)

[3.3.1 Fourier Series. 15](#_Toc512518233)

[3.3.2 Finite Difference. 15](#_Toc512518234)

[3.3.3 Damping coefficient. 15](#_Toc512518235)

[3.4 Downhole card Analysis. 16](#_Toc512518236)

[3.4.1 Fluid Load. 16](#_Toc512518237)

[4 GUIDANCE NOTES 17](#_Toc512518238)

[4.1 Business Use 17](#_Toc512518239)

[4.2 Service Delivery 17](#_Toc512518240)

[4.2.1 Error Log 18](#_Toc512518241)

[4.3 Programming 18](#_Toc512518242)

[4.3.1 Code Layout 18](#_Toc512518243)

[4.3.2 File Description. 19](#_Toc512518244)

[5 FUTURE IMPROVEMENTS 21](#_Toc512518245)

[5.1 Technical 21](#_Toc512518246)

[5.2 Computational Performance 21](#_Toc512518247)

[5.3 Project Management 21](#_Toc512518248)

[6 References. 21](#_Toc512518249)

# INTRODUCTION

Dynacard data program evaluates the analytics application performs the following:

* The guidance for service delivery in section XX
* The user manual for developer in section XX

The document contains the following:

* Guidance notes for business users in Sub-section 2.1
* Guidance notes for service delivery in Sub-section 2.2
* Guidance notes for developer in Sub-section 2.3

# Plate BUCKLING.

## Inputs.

The following shows how to determine the inputs for a plate buckling calculations and these inputs have been read through munch module.

* import munch
* plateGData1 = {'PlateLength': 2.69, 'PlateLength\_unit' : 'm',
  + 'PlateBreadth' : 0.70, 'PlateBreadth\_unit' : 'm',
  + 'PlateThickness' : 0.014, 'PlateThickness\_unit' : 'm',
  + 'AverageWaterDepth' : 40, 'AverageWaterDepth\_unit' : 'm',
  + 'YieldStrength' : 34 , 'YieldStrength\_unit' : 'ksi',
  + 'PoissionsRatio' : 0.30,
  + 'YoungsModulus' : 30450, 'YoungsModulus\_unit' : 'ksi'}
* plateGDataFT1 = {'PlateLength': 8.82, 'PlateLength\_unit' : 'ft',
  + 'PlateBreadth' : 2.30, 'PlateBreadth\_unit' : 'ft',
  + 'PlateThickness' : 0.046, 'PlateThickness\_unit' : 'ft',
  + 'AverageWaterDepth' : 131.23, 'AverageWaterDepth\_unit' : 'ft',
  + 'YieldStrength' : 34 , 'YieldStrength\_unit' : 'ksi',
  + 'PoissionsRatio' : 0.30,
  + 'YoungsModulus' : 30450, 'YoungsModulus\_unit' : 'ksi'}
* plateGLoading1 = {'LongtudinalStress' : 0.5, 'LongtudinalStress\_unit' : 'ksi',
  + 'TransverseStress' : 0.5, 'TransverseStress\_unit' : 'ksi',
  + 'ShearStress' : 0.7, 'ShearStress\_unit' : 'ksi'}

# How to access objects from above dictionaries (also same for JSON format files)

* constantGvalue1 = {'BucklingFactor' : 0.26,

'BCedges\_simplysupported\_long': 4,

'BC\_sideclamped\_long' : 7.00,

'Resulting material factor': 1.15,

'H4' : 101325,

'H5' : 1025 ,

'H6' : 9.81,

'H7' : 0.000145038,

'H8' : 0.001,

'BR\_transversedirection' : 1,

'Integralfactor' : 0,

'BA\_sheardirection' : 1}

The script below shows how to read above inputs through munch module.

* constantGvalue = munch.munchify(constantGvalue1)
* plateGData = munch.munchify(plateGData1)
* plateGDataFT = munch.munchify(plateGDataFT1)
* plateGLoading = munch.munchify(plateGLoading1)
* l\_G = plateGDataFT["PlateLength"]
* s\_G = plateGDataFT["PlateBreadth"]
* t\_G = plateGDataFT["PlateThickness"]
* d\_G = plateGDataFT["AverageWaterDepth"]
* f\_G = plateGDataFT["YieldStrength"]
* p\_G = plateGDataFT["PoissionsRatio"]
* E\_G = plateGDataFT["YoungsModulus"]
* L\_G = plateGData["PlateLength"]
* S\_G = plateGData["PlateBreadth"]
* T\_G = plateGData["PlateThickness"]
* D\_G = plateGData["AverageWaterDepth"]
* σG\_xx = plateGLoading["LongtudinalStress"]
* σG\_yy = plateGLoading["TransverseStress"]
* τ\_G = plateGLoading["ShearStress"]
* k4\_G = constantGvalue["BucklingFactor"]
* c\_xx = constantGvalue["BCedges\_simplysupported\_long"]
* cxx = constantGvalue["BC\_sideclamped\_long"]
* ϒ\_M = constantGvalue["Resulting material factor"]
* x7 = constantGvalue["H4"]
* x8 = constantGvalue["H5"]
* x9 = constantGvalue["H6"]
* x10 = constantGvalue["H7"]
* x11 = constantGvalue["H8"]
* C\_τ = constantGvalue["BR\_transversedirection"]
* ci\_1 = constantGvalue["Integralfactor"]
* C\_τe2 = constantGvalue["BA\_sheardirection"]

# THEORY

Please add theory

# GUIDANCE NOTES

Guidance notes is provided for the following users:

* Business use
* Service Delivery
* Programming Development

The level of details included in each of the following subsections varies and is dependent on the audience.

## Business Use

TBA

For further details on the theory, see Section 0.

## Service Delivery

* The high level program flowchart is shown in Figure 2.1.
* To deploy the application in production:
  + Utilize the script commands to run the program.
  + An example bat file is also included.
* The log file help troubleshoot service delivery problems.
* The log file contains the following information:
  + Fourier series or finite difference methodology utilized for the calculation
  + Run time for each program call
* To troubleshoot in more detail, refer details in section 2.3.

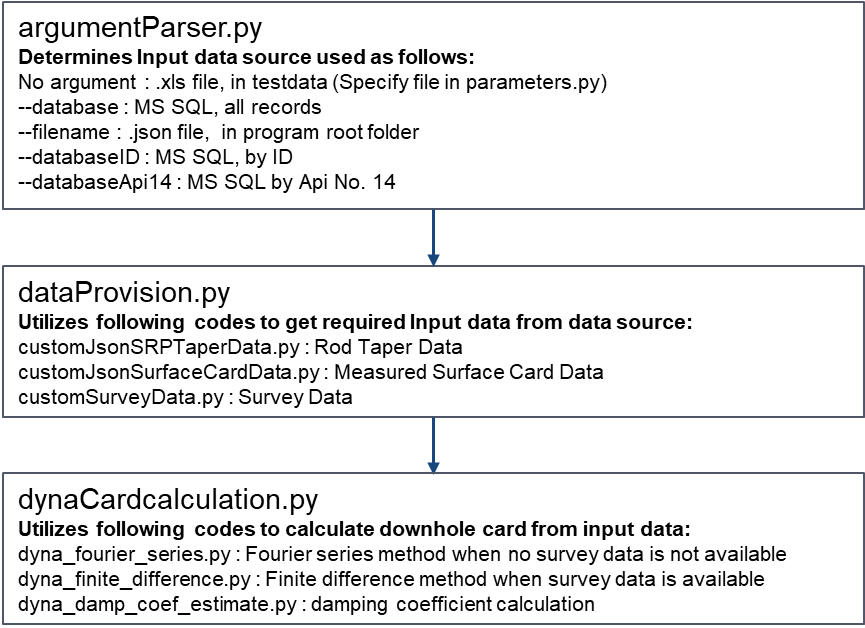


Figure 3‑1.DynaCard FlowChart

### Error Log

* Taper Data
* When the rod data has sections less than the length/Number of finite difference divisions then the program may not run
* When modulus of elasticity is zero for a any rod section, the program will not run.

## Programming

### Code Layout

* The standard folder structure described in Appendix B is utilized for the program.
* The high level program flowchart and the service delivery instructions are given in Section 2.2.
* The main program file contents are highlighted in Figure 2.1.

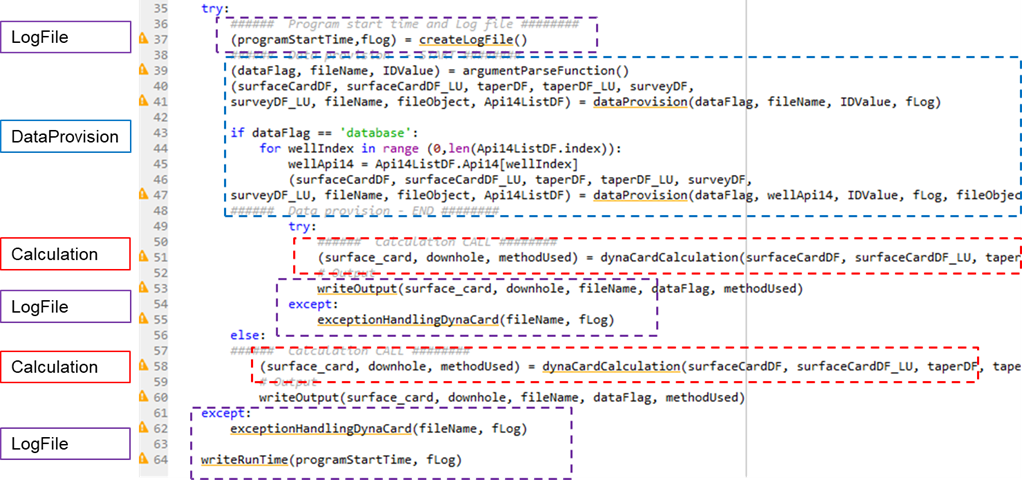


Figure 3‑2.Main Program File Contents

### File Description.

|  |  |
| --- | --- |
| **Folder/File** | **Description** |
| **Folder: DynaCard** |  |
| dynaCard.py | Main program |
| 3Run\_Env.bat | Code is used for batch run in scheduler |
| **Folder: DynaCard\lib\Calculations** | |
| dynaCardCalculation.py | Calculation file to choose between Fourier series or finite difference |
| dyna\_fourier\_series.py | Fourier series calculation method |
| dyno\_finite\_difference.py | Finite difference calculation method |
| dyno\_damp\_coef\_estimate.py | Damping coefficient estimation code |
| **Folder: DynaCard\lib\dataProvision** | |
| dataProvision.py | The main code to provision required data in a customized manner to the program. This function calls customized data query |
| parameters.py | Parameters or variables with configuration data, SQL que |
| argumentParseFunction.py | To parse the input arguments. |
| customJsonSRPTapperData.py | To get the sucker rod pump taper rod data from data source |
| customJsonSurfaceCardData.py | To get the surface card data for a given well using API No. 14 from data source |
| customSurveyData.py | To get the survey data for a given well using API No. 14 from data source |
| exceptionHandlingDynaCard.py | Exception handling |
| writeOutput.py | Write output |
| 1Create\_env.bat | Create python environment (1 time) |
| 2InstallPackages\_Env.bat | Install dependencies required in python environment |
| 4Delete\_Env.bat | Deletes the python environment |
| cx\_Oracle-6.1-cp35m-win\_amd64.whl | The public module file in python 3.5 has sql query bug. This cx\_Oracle whl file resolves this bug. |
| **Folder : dynaCard\logs** |  |
| 20171227\_12h06m.log | Contains run log files  Each program call will have a unique file with unique name.  Naming format is: YYYYMMDD\_HHhMMm, where,  YYYY: year; MM: month; DD: day; HH: hour; MM: min |
| **Folder : dynaCard\testData** |  |
| GRANTHM\_PO.xlsx | Test or development data |
| GRANTHM\_PO.xlsx\_FD.png | Test or development data |

Table 3‑1.File Description

# FUTURE IMPROVEMENTS

The improvements provided are notes for future reference to help improve the program for future updates. These may or may not be incorporated.

## Technical

## Computational Performance

The potential improvements for the program are as follows:

* Currently running 1 card for a well at each time. Spark/cluster parallel processing can be considered in future

## Project Management

* None

# References.

[1].Modified based on rod pump training introduction material, September, 2017.

[2].SPE, “An Improved Finite Difference Calculation of Downhole Dynamometer Cards for Sucker-Rod Pumps“, February 1992.

- Code

The following first line indicates how to take parametric inputs from the input line.

* from DataProvision.parameters\_Col\_All import \*
* from math import sqrt

# How to access objects from above dictionaries (also same for JSON format files)

* σG\_xx,σG\_yy,τ\_G
* x1=s\_G/l\_G
* x2=l\_G/s\_G
* c=(2-x1)
* x3=t\_G/s\_G
* x4=s\_G/t\_G
* x5=l\_G/t\_G

# FEA Analysis Stress (No Reduction Factor is used in Spreadsheet)

* σ\_e1=sqrt(σG\_xx\*\*2+σG\_yy\*\*2-(σG\_yy\*σG\_xx)+(3\*τ\_G\*\*2)) # Vonmises Stress (σe)

# Characteristic Material Resistance, σk

* σ\_kx=f\_I
* σ\_ky=f\_I
* τ\_k=f\_I/sqrt(3)
* σ\_e=f\_I

# Edges Simply supported - Uniform Loading

* c\_yy=(1+x1\*\*2)\*\*2
* c\_τ=(5.34+4\*x1\*\*2)

# Elastic Buckling Resistance for each stress direction

* x6=3.14159\*\*2\*E\_I/12/(1-p\_I\*\*2) # PI()^2\*G38/12/(1-G37^2)
* σExx\_Simp=x6\*c\_xx\*x3\*\*2
* σEyy\_Simp=x6\*c\_yy\*x3\*\*2
* τE\_simp=x6\*c\_τ\*x3\*\*2

# Reduced Slenders ratio # σG\_xx,σG\_yy,τ\_G

* λx\_simp=round(sqrt(σ\_kx/σExx\_Simp),2)
* λy\_simp=sqrt(σ\_ky/σEyy\_Simp)
* λτ\_simp=sqrt(τ\_k/τE\_simp)
* λe\_simp=sqrt(f\_I/σ\_e1\*((σG\_xx/σExx\_Simp)\*\*c+(σG\_yy/σEyy\_Simp)\*\*c+(τ\_G/τE\_simp)\*\*c)\*\*(1/c))

# Characteristic Buckling Resistance for serviceability

* σscrx\_simp=σ\_kx/sqrt(1+λx\_simp\*\*4)
* σscry\_simp=σ\_ky/sqrt(1+λy\_simp\*\*4)
* σscrz\_simp=τ\_k/sqrt(1+λτ\_simp\*\*4)
* σescr\_simp=f\_I/sqrt(1+λe\_simp\*\*4)

# Usage factor for serviceability check, Simply Supported.

* ηsx\_simp=σG\_xx/σscrx\_simp
* FALSE=σG\_yy/σscry\_simp
* ηsz\_simp=τ\_G/σscrz\_simp
* ηse\_simp=σ\_e1/σescr\_simp

# Characteristic Buckling Resistance for Ultimate check.

* σucrx\_simp1=(σ\_kx/(sqrt(1+λx\_simp\*\*4)))
* σucrx\_simp2=σ\_kx/sqrt(2)/λx\_simp
* if(λx\_simp<1):
* print("The value of σucrx\_simp1 is ",σucrx\_simp1)
* else:
* print("The value of σucrx\_simp2 is",σucrx\_simp2)
* σucry\_simp1=(σ\_ky/(sqrt(1+λy\_simp\*\*4)))
* σucry\_simp2=σ\_ky/sqrt(2)/λy\_simp
* if(λy\_simp<1):
* print("The value of σucry\_simp1 is ",σucry\_simp1)
* else:
* print("The value of σucry\_simp2 is",σucry\_simp2)
* σucrz\_simp1=(τ\_k/(sqrt(1+λτ\_simp\*\*4)))
* σucrz\_simp2=τ\_k/sqrt(2)/λτ\_simp
* if(λτ\_simp<1):
* print("The value of σucrz\_simp1 is ",σucrz\_simp1)
* else:
* print("The value of σucrz\_simp2 is",σucrz\_simp2)
* σeucr\_simp1=(σ\_e/(sqrt(1+λe\_simp\*\*4)))
* σeucr\_simp2=σ\_e/sqrt(2)/λe\_simp
* if(λe\_simp<1):
* print("The value of σeucr\_simp1 is ",σeucr\_simp1)
* else:
* print("The value of σeucr\_simp2 is",σeucr\_simp2)

# Usage factor for ultimate check, , Simply Supported.

* ηux\_simp=σG\_xx/σucrx\_simp1
* ηuy\_simp=σG\_yy/σucry\_simp2
* ηuz\_simp=τ\_G/σucrz\_simp1
* ηue\_simp=σ\_e1/σeucr\_simp2

# Sides clamped - Uniform Loading

* cyy=(1+2.5\*x1\*\*2+5\*x1\*\*4)
* cτ=(9+5.6\*x1\*\*2)

# Elastic Buckling Resistance for each stress direction.

* σExx\_Simp=x6\*cxx\*x3\*\*2
* σEyy\_Simp=x6\*cyy\*x3\*\*2
* τE\_Simp=x6\*cτ\*x3\*\*2

# Reduced Slenders ratio.

* λx\_side=sqrt(σ\_kx/σExx\_Simp)
* λy\_side=sqrt(σ\_ky/σEyy\_Simp)
* λτ\_side=sqrt(τ\_k/τE\_Simp)
* λe\_side=sqrt(f\_I/σ\_e1\*((σG\_xx/σExx\_Simp)\*\*c+(σG\_yy/σEyy\_Simp)\*\*c+(τ\_G/τE\_Simp)\*\*c)\*\*(1/c))

# Characteristic Buckling Resistance for serviceability.

* σscrx\_side=σ\_kx/sqrt(1+λx\_side\*\*4)
* σscry\_side=σ\_ky/sqrt(1+λy\_side\*\*4)
* σscrz\_side=τ\_k/sqrt(1+λτ\_side\*\*4)
* σescr\_side=f\_I/sqrt(1+λe\_side\*\*4)

# Usage factor for serviceability check, Sides Clamped.

* ηsx\_side=σG\_xx/σscrx\_side
* ηsy\_side=σG\_yy/σscry\_side
* ηsz\_side=τ\_G/σscrz\_side
* ηse\_side=σ\_e1/σescr\_side

# Characteristic Buckling Resistance for Ultimate Check.

* σucrx\_side1=σ\_kx/(sqrt(1+λx\_side\*\*4))
* σucrx\_side2=σ\_kx/sqrt(2)/λx\_side
* if(λx\_side<1):
* print("The value of σucrx\_side1 is ",σucrx\_side1)
* else:
* print("The value of σucrx\_side2 is",σucrx\_side2)
* σucry\_side1=σ\_ky/(sqrt(1+λy\_side\*\*4))
* σucry\_side2=σ\_ky/sqrt(2)/λy\_side
* if(λy\_side<1):
* print("The value of σucry\_side1 is ",σucry\_side1)
* else:
* print("The value of σucry\_side2 is",σucry\_side2)
* σucrz\_side1=τ\_k/(sqrt(1+λτ\_side\*\*4))
* σucrz\_side2=τ\_k/sqrt(2)/λτ\_side
* if(λτ\_side<1):
* print("The value of σucrz\_side1 is ",σucrz\_side1)
* else:
* print("The value of σucrz\_side2 is",σucrz\_side2)
* σeucr\_side1=σ\_e/(sqrt(1+λe\_side\*\*4))
* σeucr\_side2=σ\_e/sqrt(2)/λe\_side
* if(λe\_side<1):
* print("The value of σeucr\_side1 is",σeucr\_side1)
* else:
* print("The value of σeucr\_side2 is",σeucr\_side2)

# Usage factor for ultimate check, Sides Clamped.

* ηux\_side=σG\_xx/σucrx\_side1
* ηuy\_side=σG\_yy/σucry\_side2
* ηuz\_side=τ\_G/σucrz\_side1
* ηue\_side=σ\_e1/σeucr\_side2

# Buckling resistance stress in longitudinal direction.

* λ\_p=0.525\*x4\*sqrt(f\_I/E\_I)
* Cx=(λ\_p-0.22)/λ\_p\*\*2
* if(λ\_p>0.673):
* print("The value for slendrness grater than equal to (0.673)",Cx)
* else:
* print("The value is",1)
* σxrd=Cx\*f\_I/ϒ\_M

# Buckling resistance stress in Transverse direction

* λ\_c=1.1\*x4\*sqrt(f\_I/E\_I)
* µ=0.21\*(λ\_c-0.2)
* k1=1 # if(l\_c<=0.2): print("the value of k",k)
* k2=1/(2\*λ\_c\*\*2)\*((1+µ+λ\_c\*\*2)-sqrt((1+µ+λ\_c\*\*2)\*\*2-4\*λ\_c\*\*2))
* k3=1/(2\*λ\_c\*\*2)+0.07
* p\_Sd\_pa=101325+1025\*D\_G\*x9
* p\_Sd\_ksi=p\_Sd\_pa\*x10\*x11
* x12= 2\*(x3\*\*2)\*f\_I #x7=2\*(t\_G/s\_G)^2\*f\_y
* #IF(0.05\*G43-0.75<0,0,0.05\*G43-0.75)
* h\_α1=0.05\*x4-0.75
* h\_α2=0.05\*x4-0.75
* if(h\_α1<0):
* print(" The value of h\_α1 is",0)
* else:
* print(" The value of h\_α is",h\_α2)
* Kp1=1
* Kp2=1-h\_α2\*((p\_Sd\_ksi/f\_I)-2\*x3\*\*2)
* if(p\_Sd\_ksi<=p\_Sd\_pa):
* print(" The value of Kp is",Kp1)
* else:
* print(" The value of Kp is",Kp2)
* σy\_R=(1.3\*t\_G/l\_G\*sqrt(E\_I/f\_I)+k4\_G\*(1-1.3\*t\_G/l\_G\*sqrt(E\_I/f\_I)))\*f\_I\*Kp1
* σy\_rd=σy\_R/ϒ\_M

# Buckling resistance stress in Shear direction.

* kl\_1=5.34+4\*(x1)\*\*2
* kl\_2=5.34\*x1\*\*2+4
* if(x1<1):
* print("The value of kl\_1 is",kl\_1)
* else:
* print("The value of kl\_2 is",kl\_2)
* λ\_w=0.795\*x4\*sqrt(f\_I/(E\_I\*kl\_1))
* if(λ\_w>1.2):
* print(0.9/λ\_w)
* if(λ\_w>0.8):
* print(1-0.625\*(λ\_w-0.8))
* else:
* print("The value of C\_τ is",C\_τ)
* τ\_rd=C\_τ/ϒ\_M\*f\_I/sqrt(3)

# Buckling resistance stress in Bi-axial with Shear direction

* ci\_2=(1-s\_G/(120\*t\_G))
* if(x4>120):
* print("The value of ci\_1",ci\_1)
* else:
* print("The value of ci\_2",ci\_2)
* k\_l=kl\_1
* λ\_w=λ\_w
* C\_τe1=(1-0.8\*(λ\_w-0.8))
* if(λ\_w>1.25):
* print(1/λ\_w\*\*2)
* if(λ\_w>0.8):
* print("The value of C\_τe1 is",C\_τe1)
* else:
* print("The value of C\_τe2 is",C\_τe2)
* τrd=C\_τe2/ϒ\_M\*f\_I/sqrt(3)
* σ\_xrd=σxrd
* σ\_yrd=σy\_rd
* τ\_rd=τrd
* x15=(σG\_xx/σ\_xrd)\*\*2+(σG\_yy/σ\_yrd)\*\*2-ci\_2\*(σG\_yy/σ\_xrd)\*(σG\_yy/σ\_yrd)+(τ\_G/τ\_rd)\*\*2

# DNV-RP-C201 Usage factor

* Longitudinal=σG\_xx/σxrd
* Transverse=σG\_yy/σy\_rd
* Shear=τ\_G/τ\_rd
* Biaxial=sqrt(x15)