|  |  |
| --- | --- |
|  |  |
| 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](#_Toc513559371)

[2 THEORY 3](#_Toc513559372)

[2.1 Introduction 3](#_Toc513559373)

[2.2 Input and Output Data Specification 4](#_Toc513559374)

[2.2.1 Overview 4](#_Toc513559375)

[2.2.2 Data Flow. 6](#_Toc513559376)

[2.2.3 Data Cleansing. 6](#_Toc513559377)

[2.3 Downhole card. 6](#_Toc513559378)

[2.3.1 Fourier Series. 7](#_Toc513559379)

[2.3.2 Finite Difference. 7](#_Toc513559380)

[2.3.3 Damping coefficient. 7](#_Toc513559381)

[2.4 Downhole card Analysis. 8](#_Toc513559382)

[2.4.1 Fluid Load. 8](#_Toc513559383)

[3 GUIDANCE NOTES 9](#_Toc513559384)

[3.1 Business Use 9](#_Toc513559385)

[3.2 Service Delivery 9](#_Toc513559386)

[3.2.1 Error Log 10](#_Toc513559387)

[3.3 Programming 10](#_Toc513559388)

[3.3.1 Code Layout 10](#_Toc513559389)

[3.3.2 File Description. 11](#_Toc513559390)

[4 FUTURE IMPROVEMENTS 13](#_Toc513559391)

[4.1 Technical 13](#_Toc513559392)

[4.1.1 Survey Data, Production Bore ID and Rod Depth Mapping 13](#_Toc513559393)

[4.1.2 Density of Steel 13](#_Toc513559394)

[4.2 Computational Performance 13](#_Toc513559395)

[4.3 Project Management 13](#_Toc513559396)

[5 References. 13](#_Toc513559397)

[Appendix A – PROGRAM HISTORY 14](#_Toc513559398)

[Appendix B - ERROR LOG 14](#_Toc513559399)

[B.1 Calculation Errors 14](#_Toc513559400)

[Appendix C -Typical structure folder 15](#_Toc513559401)

[Appendix D - The input/ output data 17](#_Toc513559402)

[D.1 Taper data. 17](#_Toc513559403)

[D.2 Surface card data. 17](#_Toc513559404)

[D.3 Survey data. 19](#_Toc513559405)

[Appendix E -Global Codes 20](#_Toc513559406)

[Appendix F -Detailed errors. 20](#_Toc513559407)

[F.1 Taper 23](#_Toc513559408)

[Appendix G -test cases. 23](#_Toc513559409)

[6 Plate BUCKLING. 24](#_Toc513559410)

[6.1 Failure Modes. 24](#_Toc513559411)

[6.2 Boundary and loading conditions. 24](#_Toc513559412)

[6.3 Limits. 25](#_Toc513559413)

[6.3.1 Plate Size limits: 25](#_Toc513559414)

[6.3.2 Load Limits: 25](#_Toc513559415)

[6.4 Serviceability limit states. 25](#_Toc513559416)

[6.5 Validity. 25](#_Toc513559417)

[6.6 Inputs. 25](#_Toc513559418)

[6.7 Calculation. 27](#_Toc513559419)

# 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

# THEORY

## Introduction

* Surface Dynacard is an X vs. Y plot of the surface (polished) rod load (X axis) versus surface rod position (Y-axis).
* Downhole card is calculated using methodology provided in Section 3.2.
* An overview schematic of the surface and downhole dynacards of a typical sucker rod pump is shown in Figure 3.1.

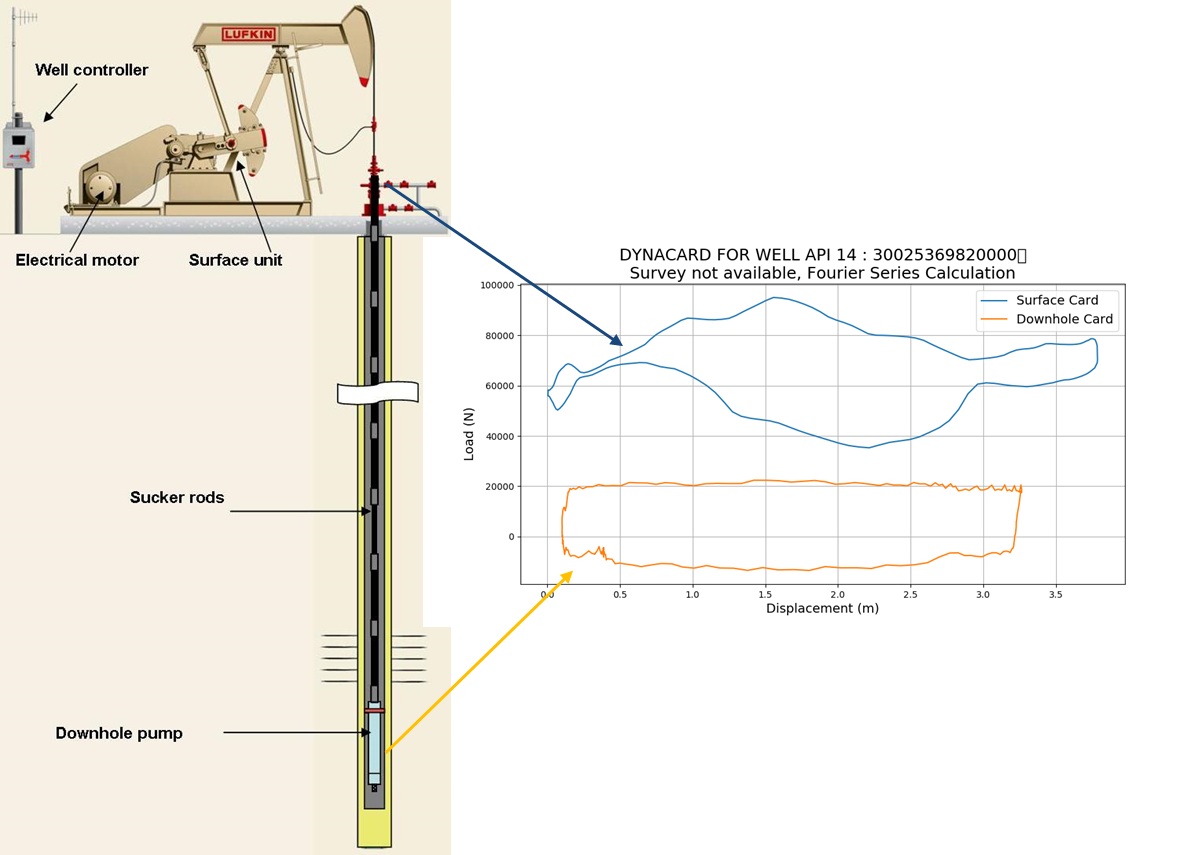


Figure 2‑1 –Surface and Downhole DynaCards,[1]

## Input and Output Data Specification

### Overview

The input data required for the program is given below:

* The taper data required to determine the rod properties all along the length of the well. This data along with units (where applicable) is given in Table 3.1.
* Is the order of the tapers important for the program? If so, we may need to rearrange data or define order before provisioning to the program
* The surface card data required along with units is given in Table 3.2
* The well survey data description defining the 3D wellbore path is given in Table 3.3
* The sample input data for the application is given in Appendix C.

The output data from the program is given below:

* The calculated downhole card data labels along with units is given in Table 3.4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Unit** | **Data Type** | **Data Range** | **Comments** |
| Api No 14 | n/a | Float | 14 numerical characters | Unique identifier |
| Rod OD | Inch | Int | Typ. 0.75 to 2 | Non-zero |
| Number Of Joints | # | Double | Typ. 1 to 200 | Non-zero |
| Joint Length | Ft | Double | Typ. 1 to 25 | Non-zero |
| Modulus Of Elasticity | Psi | Double | Typ. 8.0E+6 to 3.1E+7 | Non-zero |
| Strokes Per Minute | rpm | Double | Typ. 3 to 10 | Non-zero |

Table 2‑1.Taper Data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Unit** | **Data Type** | **Data Range** | **Comments** |
| Surface Card Displacement | inch | Double | Typ. -400 to 400 | Each point corresponds to unique load |
| Surface Card Load | lb | Double | Typ. -50,000 to +50,000 | Each point corresponds to unique load |

Table 2‑2.Surface Card Data Array

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Unit** | **Data Type** | **Data Range** | **Comments** |
| Measured Depth | ft | Double | Typ. 0 to 10,000 |  |
| Inclination | deg | Double | Typ. 0 to 15 |  |
| Azimuth | deg | Double | Typ. -360 to 360 |  |

Table 2‑3.Survey data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Unit** | **Data Type** | **Data Range** | **Comments** |
| Downhole Card Displacement | inch | Double | -400 to 400 | Each point corresponds to unique load |
| Downhole Card Load | lb | Double | -50,000 to +50,000 | Each point corresponds to unique load |

### Data Flow.

The high level data for the

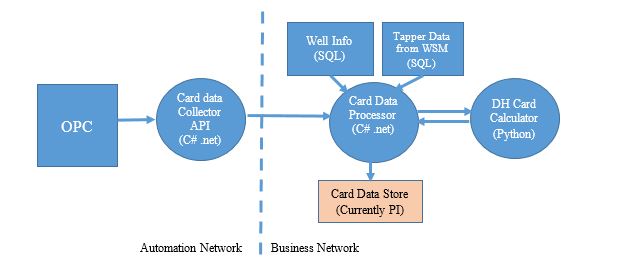


Figure 2‑2.High level Data

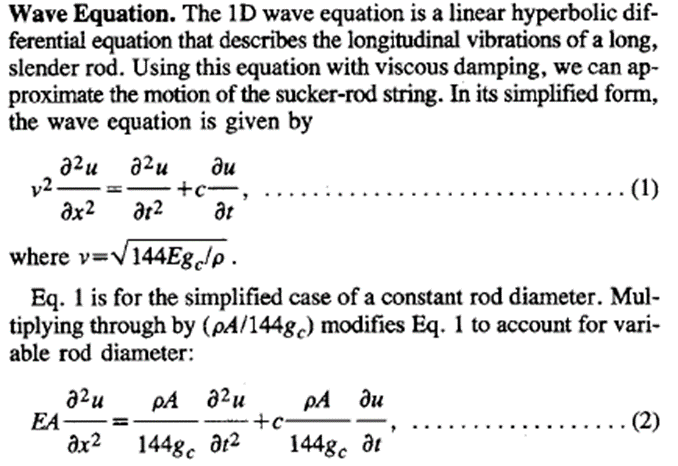
### Data Cleansing.

* Remove zero modulus of elasticity
* Remove any rod sections?!?
* TBA

## Downhole card.

The downhole card is calculated using the wave theory. The wave dynamics is given below:

* Due to the up and down motion of the pump, waves travel inline along the length of rod string
* The surface dynacard boundary condition along with the laws of reflection are used to calculate the downhole dynacard.



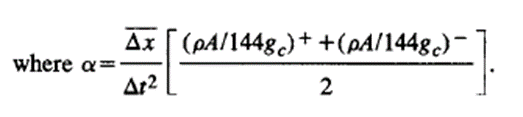


Figure 2‑3.Wave Equation

### Fourier Series.

TBA

### Finite Difference.

* The key implementation in finite difference method is the implementation of the friction along the length of the rod.
* Essentially, the downhole cards force envelop is reduced marginally due to friction between rod and wellbore.

### Damping coefficient.

TBA. Possibly move it into its own separate document as this may be used by multiple functions as required.

## Downhole card Analysis.

### Fluid Load.

* Pump Fillage
* Pump In-take Pressure (PIP)
* Pump Efficiency
* Pump Displacement
* Inferred Production

# 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.

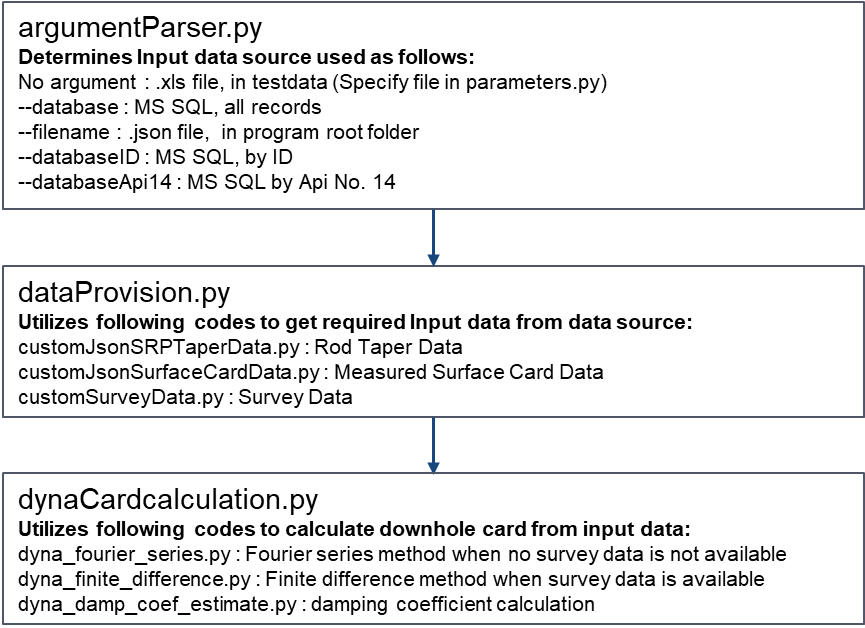


Figure 3‑1.DynaCard FlowChart

### 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.

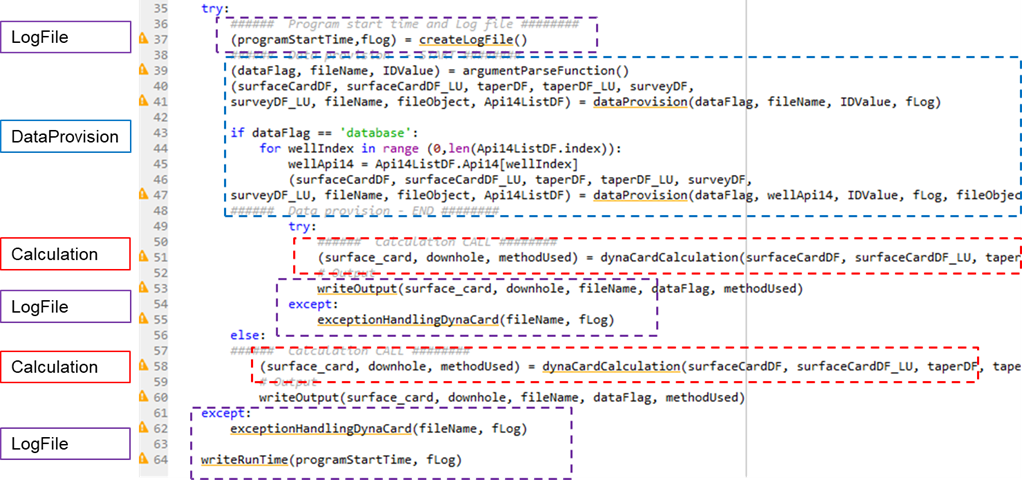


Figure 3‑2.Main Program File Contents

### 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

### Survey Data, Production Bore ID and Rod Depth Mapping

* Mapping of the survey data on to the well pump rod with well elevation /depth:
  + Instead of using "Number of Joints" and "Joint length", we will use "Start Elevation" and "End Elevation" of each section in next update. This data can be utilized to map the right survey for each elevation as required by the dynaCard program.

### Density of Steel

The finite difference/fourier series algorithms use steel density of 8500 g/m3. The damping algorithm uses steel density of 7850 kg/m3. Suggest to use consistent density of 7850 kg/m3.

## 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.

– PROGRAM HISTORY

The program history is provided in this section.

|  |  |  |
| --- | --- | --- |
| Revision Date | Features |  |
| 2017-12-31 | Modular design for input data  Ability to select data source  Unitconverter as dedicated function. |  |

Table 5‑1.Program History

- ERROR LOG

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table 5‑2.Error Log.

Calculation Errors

The typical errors encountered while running the calculation program are given in this section.

-Typical structure folder

The folder structure utilized by the program is described in Table B.1 and a typical example folder structure is shown in Figure B.1.

|  |  |
| --- | --- |
| **Folder Name** | **Description** |
| !global Codes | * Contains common shared codes shared globally * May also be shared by other programs * Codes are typically generic functions to connect to database for getting data, unit conversion, etc. |
| dynaCard | * Main program folder * Contains the main program code(s) and a batch file to run program in production. * Folder name is same as the analytics program (typically) * Contains following subfolders “Calculations” and “dataProvision” |
| calculations | * Contains code(s) utilized for the key calculations |
| data Provision | * Contains the data provision codes which provide data to the program * These can get required custom data for each program |
| logs |  |

Table 5‑3.Folder Structure Description.

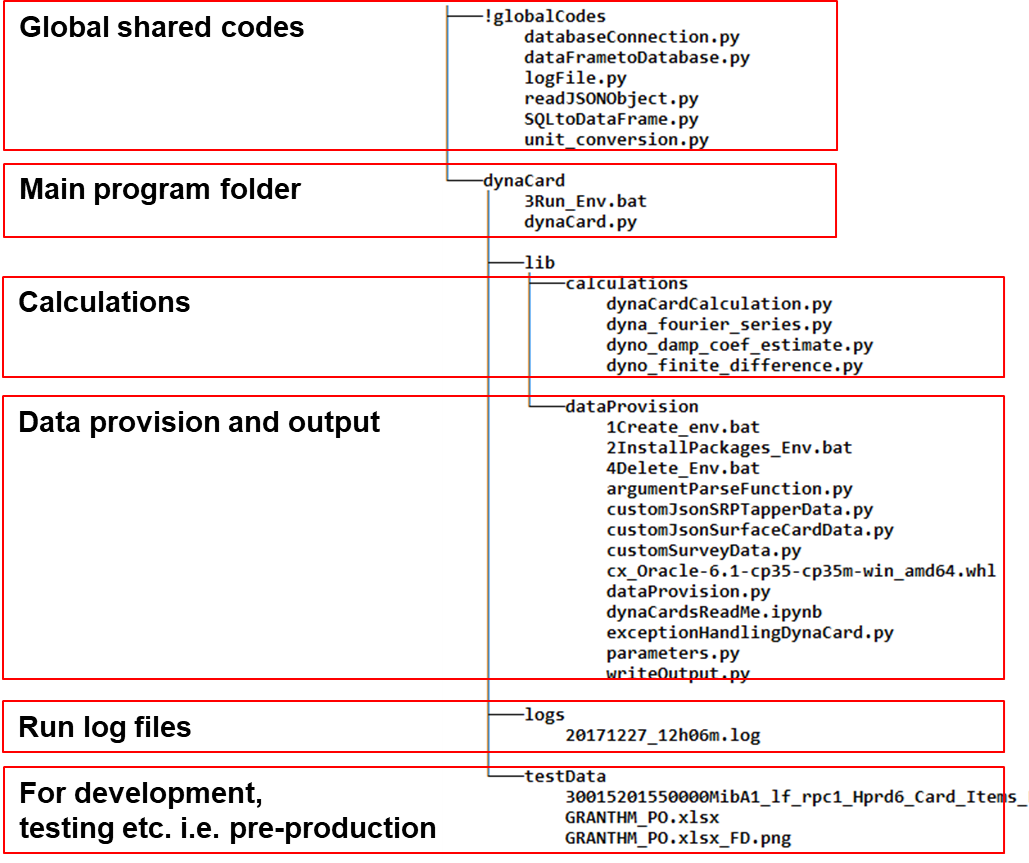


Figure 5‑1.Example Folder Structure.

- The input/ output data

The input and output data supplied to the program is given in this section.

Taper data.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Api No 14 | Rod OD (Inch) | Number Of Joints | Joint Length (ft) | Modulus Of Elasticity (Pa) | Strokes Per Minute |
| 42415039920000 | 1.0 | 1 | 4.0 | 31000000.0 | 8.5714 |
| 42415039920000 | 1.0 | 1 | 10.0 | 31000000.0 | 8.5714 |
| 42415039920000 | 1.0 | 98 | 25.0 | 31000000.0 | 8.5714 |
| 42415039920000 | 0.875 | 159 | 25.0 | 31000000.0 | 8.5714 |
| 42415039920000 | 1.0 | 6 | 25.0 | 31000000.0 | 8.5714 |
| 42415039920000 | 0.75 | 1 | 2.0 | 31000000.0 | 8.5714 |

Table 5‑4.Taper Data

Surface card data.

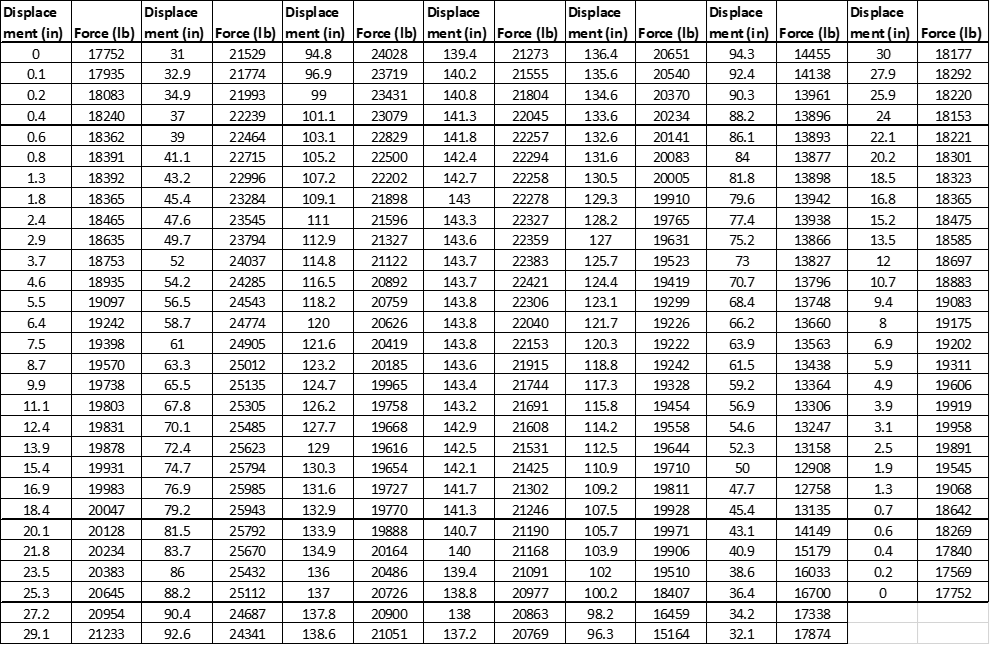


Table 5‑5.Surface Card Data

Survey data.

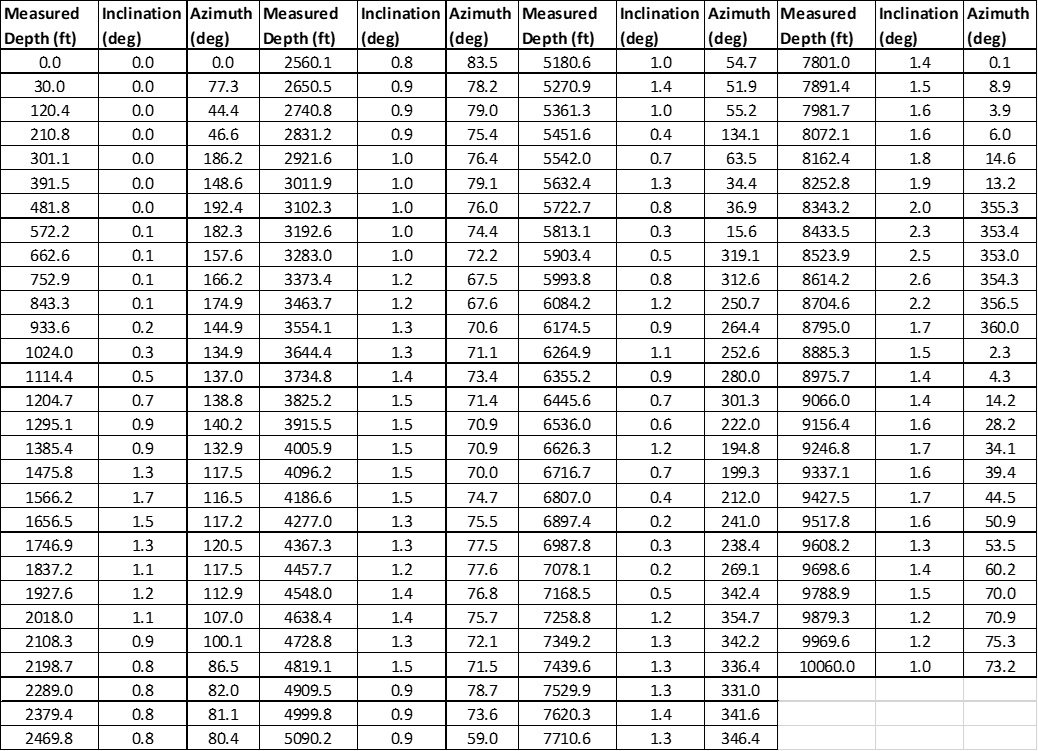


Table 5‑6.Survey Data

-Global Codes

The global codes utilized by multiple programs are described in this section. These are reusable standard code modules shared by multiple programs.

|  |  |
| --- | --- |
| **Folder/File** | **Description** |
| **Folder : !globalCodes** |  |
| databaseConnection.py | Code to connect to mssql or Oracle databases |
| dataFrametoDatabase.py | Insert a dataFrame to a Database. Easier to use when data order and data columns match between dataFrame and Database |
| logFile.py | Log file |
| readJSONObject.py | Code to read the JSONObject |
| SQLtoDataFrame.py | Get data frame from database using SQL query |
| unit\_conversion.py | Unit conversion |

Table 5‑7.Global Codes

-Detailed errors.

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error** | **Solution** |
| 1 | Traceback (most recent call last):  File "dynaCard.py", line 56, in <module>  (surface\_card, downhole, methodUsed) = dynaCardCalculation(surfaceCardDF, taperDF, surveyDF, fileName, fLog)  File "lib/calculations\dynaCardCalculation.py", line 17, in dynaCardCalculation  (surface\_card, downhole) = dynaCardFourierSeriesCalculation(surfaceCardDF, taperDF)  File "lib/calculations\dyna\_fourier\_series.py", line 375, in dynaCardFourierSeriesCalculation  (rod\_information) = rod\_data(taperDF)  File "lib/calculations\dyna\_fourier\_series.py", line 104, in \_\_init\_\_  self.c = dampingEstimator(self, viscosity = dampEstimator\_by\_mu\_default)  File "lib/calculations\dyna\_damp\_coef\_estimate.py", line 127, in dampingEstimator  sim\_params.tap\_L[i])  File "lib/calculations\dyna\_damp\_coef\_estimate.py", line 65, in estimate\_damping\_coeff  /(ID\_well/2.0 - OD\_rod/2.0)  ZeroDivisionError: float division by zero |  |
| 2 | Traceback (most recent call last):  File "dynacard.py", line 56, in <module>  (surface\_card, downhole, methodUsed) = dynaCardCalculation(surfaceCardDF, taperDF, surveyDF, fileName, fLog)  File "lib/calculations\dynaCardCalculation.py", line 24, in dynaCardCalculation  (surface\_card, downhole) = dynaCardFiniteDifferenceCalculation(surfaceCardDF, taperDF, surveyDF)  File "lib/calculations\dyna\_finite\_difference.py", line 743, in dynaCardFiniteDifferenceCalculation  removeJump = removeJumpValue)  File "lib/calculations\dyna\_finite\_difference.py", line 688, in main  coeffs.\_set\_angles()  File "lib/calculations\dyna\_finite\_difference.py", line 416, in \_set\_angles  phi\_vector = reshape(self.sur.f\_phi(x), (Nx, 1))  File "lib/calculations\dyna\_finite\_difference.py", line 160, in f\_phi  return interp1d(self.s, self.phi, kind = 'linear')(s)  File "C:\Users\AceEngineer\_Server2\AppData\Local\conda\conda\envs\dynaCard\_env\lib\site-packages\scipy\interpolate\polyint.py", line 79, in \_\_call\_\_  y = self.\_evaluate(x)  File "C:\Users\AceEngineer\_Server2\AppData\Local\conda\conda\envs\dynaCard\_env\lib\site-packages\scipy\interpolate\interpolate.py", line 634, in \_evaluate  below\_bounds, above\_bounds = self.\_check\_bounds(x\_new)  File "C:\Users\AceEngineer\_Server2\AppData\Local\conda\conda\envs\dynaCard\_env\lib\site-packages\scipy\interpolate\interpolate.py", line 666, in \_check\_bounds  raise ValueError("A value in x\_new is above the interpolation "  ValueError: A value in x\_new is above the interpolation range. |  |
| 3 | Traceback (most recent call last):  File "dynacard.py", line 56, in <module>  (surface\_card, downhole, methodUsed) = dynaCardCalculation(surfaceCardDF, taperDF, surveyDF, fileName, fLog)  File "lib/calculations\dynaCardCalculation.py", line 24, in dynaCardCalculation  (surface\_card, downhole) = dynaCardFiniteDifferenceCalculation(surfaceCardDF, taperDF, surveyDF)  File "lib/calculations\dyna\_finite\_difference.py", line 743, in dynaCardFiniteDifferenceCalculation  removeJump = removeJumpValue)  File "lib/calculations\dyna\_finite\_difference.py", line 662, in main  sur = survey\_data(taperDF, surveyDF)  File "lib/calculations\dyna\_finite\_difference.py", line 156, in \_\_init\_\_  self.D\_phi = np.gradient(self.phi)  File "C:\Users\AceEngineer\_Server2\AppData\Local\conda\conda\envs\dynaCard\_env\lib\site-packages\numpy\lib\function\_base.py", line 1642, in gradient  "Shape of array too small to calculate a numerical gradient, "  ValueError: Shape of array too small to calculate a numerical gradient, at least two elements are required. |  |
| 4 | Traceback (most recent call last):  File "dynacard.py", line 56, in <module>  (surface\_card, downhole, methodUsed) = dynaCardCalculation(surfaceCardDF, taperDF, surveyDF, fileName, fLog)  File "lib/calculations\dynaCardCalculation.py", line 17, in dynaCardCalculation  (surface\_card, downhole) = dynaCardFourierSeriesCalculation(surfaceCardDF, taperDF)  File "lib/calculations\dyna\_fourier\_series.py", line 396, in dynaCardFourierSeriesCalculation  downhole = solver(surface\_card, rod\_information, sim)  File "lib/calculations\dyna\_fourier\_series.py", line 343, in solver  alpha, beta = eigenvalues(sim, 0) # second term = 0 means down  File "lib/calculations\dyna\_fourier\_series.py", line 206, in eigenvalues  alpha[:, j] = n \* sim.w / sim.a[j] / sqrt(2) \* sqrt( 1 + sqrt(inside))  IndexError: invalid index to scalar variable. |  |
|  |  |  |

Taper

-test cases.

The program got to handle discontinuities as shown in figure below.

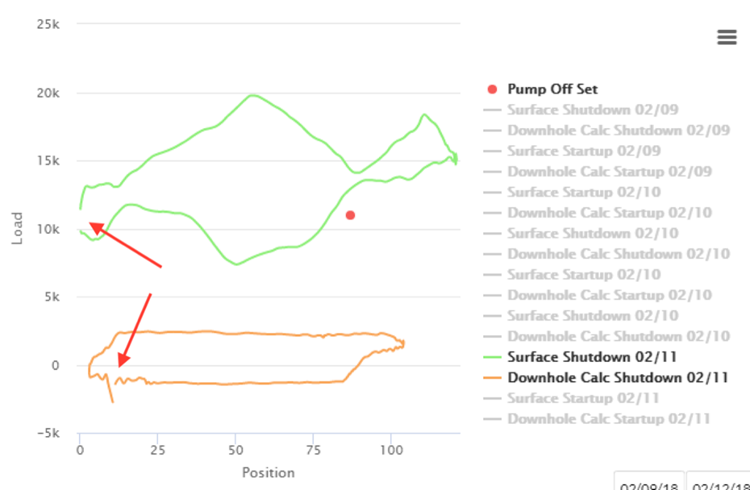


Figure 5‑2.Test Cases.

# Plate BUCKLING.

Buckling of flat plates may be experienced when the plate is excessively stressed in compression along opposite edges, or in shear uniformly distributed around all edges of the plate or a combination of both. This necessitates establishment of values for the critical buckling stress in compression (a) and in shear (G).

## Failure Modes.

This recommended practice addresses failure modes for unstiffened and stiffened plates, which are not covered by the cross sectional check of members. Such failure modes are:

* Yielding of plates in bending due to lateral load.
* Buckling of slender plates (high span to thickness ratio) due to in-plane compressive stresses or shear stresses.

Guidance for determining resistance is given both for individual plates (unstiffend plates), stiffened plates and for girders supporting stiffended plate panels. For stiffened panels the recommendations cover panel buckling, stiffener buckling as well as local buckling of stiffener and girder flanges, webs and brackets.

## Boundary and loading conditions.

The Serviceability and Ultimate are the loading conditions that will also change some factors and hence the results.

* Loading conditions
  + Serviceability.
  + Ultimate.

The simply supported and sides clamped are the boundary condition that will also change the results. We have to capture this as well by repeating the calculation.

* Boundary conditions.
  + Simply supported.
  + And sides clamped.

## Limits.

### Plate Size limits:

The plate size limits for a given load such as the plate length or breadth becomes longer, it will fail. So we can find the plate thickness, plate size limits for a given loading on a plate for any component. There are 3 variations and this will be 3-D surface plot if we try find the result.

### Load Limits:

The load limits for a given plate size are the loads (longitudinal, transverse, shear) that can be varied to find the limiting loads on a given plate. There are 3 variations and this will be 3-D surface.

## Serviceability limit states.

Check of serviceability limit states for slender plates relatedto out of plane deflection may normally be omitted if the smallest span of the plate is less than 120 times the plate

thickness.

## Validity.

This Recommended Practice is best suited to rectangularplates and stiffened panels with stiffener length being larger than the stiffener spacing ( l > s ). It may also be used for girders being orthogonal to the stiffeners and with the girder having significant larger cross-sectional dimensions than the stiffeners.

## 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"]

## Calculation.

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

Buckling strength analyses shall be based on the characteristic buckling strength for the most unfavourable buckling mode. The characteristic buckling strength shall be based on the

lower 5th percentile of test results.

# 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.

Buckling checks of unstiffened plates in compression shall be made according to the effective width method. The reduction in plate resistance for in-plane compressive forces is expressed by a reduced (effective) width of the plate which is multiplied by the design yield strength to obtain the design resistance.

The design buckling resistance of an unstiffened plate under longitudinal compression force may be calculated as.

* 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.

In case of linear varying transverse stress the capacity check can be done by use of the design stress value at a distance l1 from the most stressed end of the plate, but not less than 0.75

of maximum σy,Sd.

The design buckling resistance of a plate under transverse compression force may be found from:

* 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.

A plate subjected to biaxially loading with shear should fulfil the following requirement where if both σx,Sd and σy,Sd is compression (positive) then ci\_2=(1-s\_G/(120\*t\_G)) for s/t <= 120

And ci\_2=0 for s/t > 120.

If either of σx,Sd and σy,Sd or both is in tension (negative), then ci = 1.0.

In order to perform cross sectional checks for members subjected to plate buckling the local buckling effects can be accounted for by checking the resistance by using the effective width.

* 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)