|  |  |
| --- | --- |
|  |  |
| Fatigue Curve Analysis | |
| User Manual | |
| July 6, 2018 | |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rev | Description | Date | Prep | Chk'd | Apprv'd |
| 01 | New Document | 6-July-2018 | VA |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Revision History

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

CONTENTS

[1 INTRODUCTION. 2](#_Toc519002399)

[1.1 Fundamentals. 2](#_Toc519002400)

[1.2 Basic Curves. 2](#_Toc519002401)

[1.3 Miscellaneous data. 3](#_Toc519002402)

[1.4 Theory. 3](#_Toc519002403)

[2 Python Modules. 4](#_Toc519002404)

[3 Fatigue curves. 4](#_Toc519002405)

[3.1 Fatigue basic curve. 4](#_Toc519002406)

[3.2 Linear slope data. 6](#_Toc519002407)

[3.3 Shear data calculation. 7](#_Toc519002408)

[4 Standard Reffernce. 10](#_Toc519002409)

# INTRODUCTION.

To understand the theory behind the Fatigue Curves and modelling tools (shear7) by using python application which helps oil & gas industry.

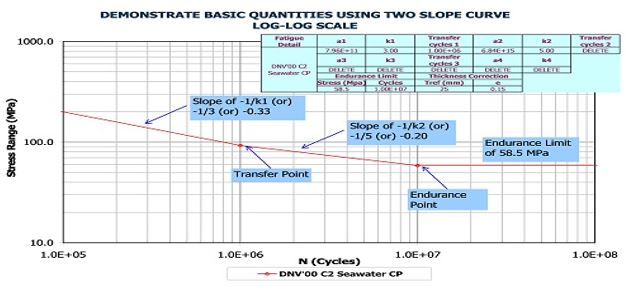
## Fundamentals.

The following are the major concepts to be involved before starting the project.

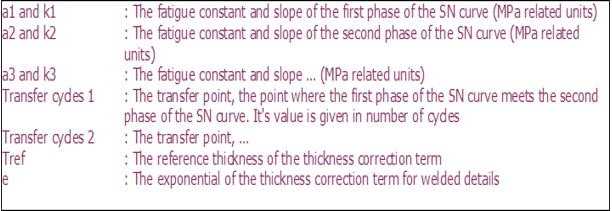
* Fatigue: the weakening of a material caused by repeatedly applied loads. It is the progressive and localized structural damage that occurs when a material is subjected to cyclic loading. It consists of cyclic and variable loadings.
* Fatigue limit: this is that load limit below which material gains infinite life.
* Endurance: The maximum value of completely reversed bending stress that a material can withstand without any failure.
* Endurance limit: The maximum value of completely reversed bending stress that a material can withstand without any failure for ‘infinite number of cycles’.
* The very basic difference between fatigue and endurance limit is that the endurance limit has cycle number mentioned with (I).
* Note: S-N curve a plot of the magnitude of an alternating stress versus the number of cycles to failure for a given material.

## Basic Curves.

Through the following image explanation of points and limits of basic curves are represented.



The slope point explanation of basic curves are represented as follows.



## Miscellaneous data.

|  |  |
| --- | --- |
| **Data Description** | |
| tref | Reference thickness equal 25 mm for welded connections other than tubular joints |
| For tubular joints | The reference thickness is 32 mm |
| For bolts | Tref = 25 mm |
| **Different slopes selection based on project requirement** | |
| Single slope | (0 Stress Cut off) |
| **Note:** Please note that highest stress range curve from multi-slope data is used for single-slope | |
| Single slope | (with Endurance limit/ stress cut-off) |
| Multi-slope | (0 Stress Cut off) |
| Multi-slope | (with Endurance limit/ stress cut-off) |
| **Note:** Please note that endurance limit is probably only used for research purposes and typically NOT for project work | |

## Theory.

The theory behind the fatigue curves is explained by the following equation.

Where,

S=Stress Range (Mpa)

a=Value corresponding to S in stress range.

The modelling tools are (Shear7 for vortex-induced vibration analysis).

# Python Modules.

The modules needed for compiling data and plotting in python are as mentioned below.

|  |  |  |
| --- | --- | --- |
| **Module** | **Software** | **Description** |
| Xlrd | Python | Reading the data using excel file. |
| Pandas | Python | Manipulating excel. |
| Matplotlib | Python | For plotting curves. |
| Xlwt | Python | Writing the data to excel file. |

# Fatigue curves.

The following steps determines the procedure followed in performing the three different plot data calculations through python.

## Fatigue basic curve.

The raw data has been prepared in the excel sheet and from this the data is read in python file. For example, reading data for slope1 is

* fatigueRawdata = pd.read\_excel(" File directory")
* fatigueConstant = fatigueRawdata["Log a1~"]

The Fatigue constant calculation using numpy and updating the column with calculated values are as follows.

* fatigueConstant1 = np.power(10,fatigueConstant)
* fatigueRawdata["Log a1"] = fatigueConstant1

Assigning the reference variable and updating the column with calculated values.

* refernceConstantCycleandStressdata = fatigueRawdata["# of Slopes"]

slope = fatigueRawdata["m1"]

transferCycle1 = np.where(refernceConstantCycleandStressdata > 1,

* + - fatigueRawdata["Transfer Cycles 1"], np.nan )

fatigueRawdata["Transfer Cycles 1"] = transferCycle1

transferStress1 = np.where(refernceConstantCycleandStressdata > 1,

* + - fatigueRawdata["Transfer Stress 1 (Mpa)"],np.nan)
* fatigueRawdata["Transfer Stress 1 (Mpa)"] = transferStress1

The function for calculating fatigue constant for slopes are as follows.

* def constant(value):

if value == "-":

return (np.nan)

elif 1 < value > 4 :

return (10\*\*value)

elif value == " ":

return (np.nan)

The remaining second, third and the fourth slope calculations script has been done as per

the below lines.

* ### second slope calculations.
  + slope1 = fatigueRawdata["m2"]
  + slope2 = slope1.map(constant)
  + fatigueRawdata["m2"] = slope2

The reading the column data for slope2 and applying condition for fatigue constant for slope.

* + fatigueConstant2 = fatigueRawdata["Log a2~"]
  + fatigue = fatigueConstant2.map(constant)

The updating the col with calculated values.

* + fatigueRawdata["Log a2~"] = fatigue

The conditional statement using numpy (np.where(if condition, true, false)

* + transferCycle2 = np.where(refernceConstantCycleandStressdata > 2,

The updating column with calculated values.

* + fatigueRawdata["Transfer Cycles 2"], np.nan )
  + fatigueRawdata["Transfer Cycles 2"] = transferCycle2

The conditional statement using numpy (np.where(if condition, true, false) and the updating column with calculated values.

* + transferStress2 = np.where(refernceConstantCycleandStressdata > 2,
  + fatigueRawdata["Transfer Stress 2 (Mpa)"],np.nan)
  + fatigueRawdata["Transfer Stress 2 (Mpa)"] = transferStress2
* ### third slope calculations
  + fatigueConstant3 = fatigueRawdata["Log a3~"]
  + fatigue1 = fatigueConstant3.map(constant)
  + fatigueRawdata["Log a3~"] = fatigue1
  + slope3 = np.where(refernceConstantCycleandStressdata > 2,
  + fatigueRawdata["m3"], np.nan )
  + fatigueRawdata["m3"] = slope3
  + transferCycle3 = np.where(refernceConstantCycleandStressdata > 3,
  + fatigueRawdata["Transfer Cycles 3"], np.nan )
  + fatigueRawdata["Transfer Cycles 3"] = transferCycle3
  + CAFL = np.where(refernceConstantCycleandStressdata > 3,
  + fatigueRawdata["CAFL (Mpa)"], np.nan )
  + fatigueRawdata["CAFL (Mpa)"] = CAFL
* ### fourth slope calculations
  + fatigueConstant4 = fatigueRawdata["Log a4~"]
  + fatigue2 = fatigueConstant4.map(constant)
  + fatigueRawdata["Log a4~"] = fatigue2
  + slope4 = np.where(refernceConstantCycleandStressdata > 3,
  + fatigueRawdata["m4"], np.nan )
  + fatigueRawdata["m4"] = slope4
* plt.loglog(transferCycle1,transferStress1)

## Linear slope data.

The accessing of linear slope data for high stress range calculation are prescribed below. Assigning the variable for getting linear slope data and reading using panda’s data frame.

* linearslopeData = fatigueRawdata.loc[:, 'Curve Type':'m1']
* pd.DataFrame(linearslopeData)

The high stress range calculation at different ranges are determined below.

* # high stress range calulation at 900
  + linearslopeData["High Stress Range 2 (Mpa)"] = 900
  + cal = (linearslopeData["High Stress Range 2 (Mpa)"])\*\*linearslopeData["m1"]
  + highStressRange = linearslopeData["Log a1"]/cal
  + linearslopeData["N - High Stress Range 2"] = highStressRange
* # high stress range calulation at 500
  + linearslopeData["High Stress Range 2 (Mpa)1"] = 500
  + cal1 = (linearslopeData["High Stress Range 2 (Mpa)1"])\*\*linearslopeData["m1"]
  + highStressRange1 = linearslopeData["Log a1"]/cal1
  + linearslopeData["N - High Stress Range 21"] = highStressRange1
* # high stress range calulation at 350
  + linearslopeData["High Stress Range 2 (Mpa)2"] = 350
  + cal2 = (linearslopeData["High Stress Range 2 (Mpa)2"])\*\*linearslopeData["m1"]
  + highStressRange2 = linearslopeData["Log a1"]/cal2
  + linearslopeData["N - High Stress Range 22"] = highStressRange2
* # high stress range calulation at 300
  + linearslopeData["High Stress Range 2 (Mpa)3"] = 300
  + cal3 = (linearslopeData["High Stress Range 2 (Mpa)3"])\*\*linearslopeData["m1"]
  + highStressRange3 = linearslopeData["Log a1"]/cal3
  + linearslopeData["N - High Stress Range 23"] = highStressRange3
* # high stress range calulation at 250
  + linearslopeData["High Stress Range 2 (Mpa)4"] = 250
  + cal4 = (linearslopeData["High Stress Range 2 (Mpa)4"])\*\*linearslopeData["m1"]
  + highStressRange4 = linearslopeData["Log a1"]/cal4
  + linearslopeData["N - High Stress Range 24"] = highStressRange4
* # high stress range calulation at 200
  + linearslopeData["High Stress Range 2 (Mpa)5"] = 200
  + cal5 = (linearslopeData["High Stress Range 2 (Mpa)5"])\*\*linearslopeData["m1"]
  + highStressRange5 = linearslopeData["Log a1"]/cal5
  + linearslopeData["N - High Stress Range 25"] = highStressRange5
  + linearslopeData.to\_excel("File Directory")

## Shear data calculation.

The following shows accessing the fatigue basic curve modules.

* import FatigueBasiccurve
* from FatigueBasiccurve import\*

The assign of variable to read the data from the given spread sheet and reading the data using the pandas dataframe.

* fatigueBasicurve=pd.read\_excel("File Directory")
* shear7Data = fatigueBasicurve.loc[:, 'Curve Type':'S0 (Mpa)']
* pd.DataFrame(shear7Data)

The high and low stress range calculation for shear7data are as follows.

* shear7Data["High Stress Range (Mpa)"] = 1000
* cal = shear7Data["High Stress Range (Mpa)"]\*\*(slope)
* highStressRange = fatigueConstant1 /cal
* shear7Data["N - High Stress Range"] = highStressRange
* ##low stress range calculation for shear7 data
  + shear7Data["Low Stress Range(Mpa)"] = 1
  + shear7Data["Low Stress Range(Mpa)"].iloc[30:45] = np.arange(2,17)
  + lowStressRange = shear7Data["Low Stress Range(Mpa)"]
* def dat(refernceConstantCycleandStressdata,y,v,z,a,c,d,f,e,h):

for Numofslopes in refernceConstantCycleandStressdata:

if Numofslopes == 1:

return(z/v\*\*(y))

elif Numofslopes == 2:

return(c/v\*\*(a))

elif Numofslopes == 3:

return(f/v\*\*(d))

elif Numofslopes == 4:

return(h/v\*\*(e))

* fatigueData= dat(refernceConstantCycleandStressdata,slope,lowStressRange,fatigueConstant1,slope2,fatigue,slope3,fatigue1,slope4,fatigue2)
* shear7Data["N - Low Stress Range"] = fatigueData
* shear7Data.to\_excel("File Directory")
* linearslopeData = fatigueRawdata.loc[:, 'Curve Type':'m1']
* pd.DataFrame(linearslopeData)
* highStressRangevalues = [900,500,350,300,250,200]
* highStresscal = linearslopeData["Log a1"]
* slope = linearslopeData["m1"]
* y = []
* for data,value in enumerate(highStressRangevalues):
* y.append(highStresscal/value\*\*(slope))

# Standard Reffernce.

The standard reference for different curves of fatigue analysis are as follows.

|  |  |
| --- | --- |
| **S.no** | **Reference** |
| 1 | American Petroleum Institute - "Recommended Practise for Planning, Designing and Constructing Fixed Offshore Platforms – Load and Resistance Factor Design". API-RP-2A-LRFD, Second Edition, Apr 1994. |
| 2 | Debt Norske Veritas (DnV) - "Fatigue Strength Analysis for Mobile Offshore Units", Aug 1984. |
| 3 | Norsok Standard - "Design of Steel Structures". N-004, Rev 1, Dec 1998. |
| 4 | Det Norske Veritas (DnV) - "Fatigue Design of Offshore Steel Structures" (DnV-RP-C203), Aug 2005. |
| 5 | Det Norske Veritas (DnV) - "Fatigue Design of Offshore Steel Structures" (DnV-RP-C203), Aug 2000. |
| 6 | Values for titanium supplied for Agbami (job 1378) - originally from OTC 8409 "Advances in Titanium Risers for FPSO's"; Carl Baxter et al 1997. |
| 7 | Value used on Asgard project (1570) - Based on data from Marintek 1998. |
| 8 | Values for titianium supplied for Kristin (job 1599) - based on data from Marintek; From OMAE papers "Fatigue strength of titanium riser welds effects of material grade and weld method", OMAE 2002/MAT-28576 & "Fatigue of 28-inch titanium riser – sn data and defect assessment", OMAE 2002/MAT-28577. |
| 9 | Det Norske Veritas (DnV) - "Fatigue Design of Offshore Steel Structures" (DnV-RP-C203), Apr 2008. |
| 10 | BP Engineering Technical Practises, "Riser Fatigue Calculation Guidance Note", GN65-704, Revision 2, Jun 2008 |
| 11 | Det Norske Veritas (DnV) - "Fatigue Design of Offshore Steel Structures" (DnV-RP-C203), Apr 2010. |
| 12 | Det Norske Veritas (DnV) - "Fatigue Design of Offshore Steel Structures" (DnV-RP-C203), Oct 2011. |