|  |  |
| --- | --- |
|  |  |
| Pipe Capacity & Utilization | |
| User Manual | |
| May 1, 2018 | |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rev | Description | Date | Prep | Chk'd | Apprv'd |
| 01 | New Document |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Revision History

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

CONTENTS

[1 INTRODUCTION 4](#_Toc2448565)

[2 THEORY 5](#_Toc2448566)

[3 Program Outline 5](#_Toc2448567)

[3.1 General Enhancements 5](#_Toc2448568)

[3.2 Geometry 5](#_Toc2448569)

[3.3 Material Properties 5](#_Toc2448570)

[4 VM Stress (Riser Code, API-RP-2RD) 6](#_Toc2448571)

[4.1.1 Combined Stresses 6](#_Toc2448572)

[4.1.2 Allowable Stresses 7](#_Toc2448573)

[4.1.3 Allowable Stress in Plain Pipe 7](#_Toc2448574)

[5 API STD 2RD 8](#_Toc2448575)

[5.1.1 Burst Pressure 8](#_Toc2448576)

[5.1.2 Collapse Due to External Pressure 8](#_Toc2448577)

[5.1.3 Tension 10](#_Toc2448578)

[5.1.4 Internal Pressure 10](#_Toc2448579)

[5.1.5 Temperature Derating 13](#_Toc2448580)

[5.1.6 APISTD2RD.yml 14](#_Toc2448581)

[5.1.7 APISTD2RDMethods.py 17](#_Toc2448582)

[5.1.8 APISTD2RD.py 20](#_Toc2448583)

[5.1.9 Summary.py 21](#_Toc2448584)

[6 API RP 2RD 23](#_Toc2448585)

[6.1 Combined Stresses 23](#_Toc2448586)

[6.2 Allowable Stresses 23](#_Toc2448587)

[6.3 Allowable Stress in Plain Pipe 23](#_Toc2448588)

[6.4 API RP 2RD.ini 24](#_Toc2448589)

[6.5 APIRP2RD.yml 25](#_Toc2448590)

[6.6 APIRP2RD\_Methods.py 26](#_Toc2448591)

[6.7 APIRP2RD.py 26](#_Toc2448592)

[7 API 6A 30](#_Toc2448593)

[8 ASME B31.8 31](#_Toc2448594)

[8.1.1 Steel Pipe Design Formula 31](#_Toc2448595)

[8.1.2 B31.yml 33](#_Toc2448596)

[8.1.3 Summary\_ASMEB31.yml 35](#_Toc2448597)

[8.1.4 ASMEMethods.py 39](#_Toc2448598)

[8.1.5 ASMEB31Sizing.py 43](#_Toc2448599)

[9 API 1111 45](#_Toc2448600)

[9.1.1 Specified Minimum Burst Pressure 45](#_Toc2448601)

[9.1.2 MOP Limits 45](#_Toc2448602)

[9.1.3 Internal Pressure (Burst) Design 46](#_Toc2448603)

[9.1.4 Longitudinal Load Design 47](#_Toc2448604)

[9.1.5 Combined Load Design 48](#_Toc2448605)

[10 DNV-OS-F101 48](#_Toc2448606)

[10.1 Units 48](#_Toc2448607)

[10.2 DNV-OS-F101.ini 49](#_Toc2448608)

[10.3 Rigid Pipe 51](#_Toc2448609)

[10.3.1 Characteristic Material Properties 58](#_Toc2448610)

[10.3.2 Pressure containment (bursting) 59](#_Toc2448611)

[11 DNV-OS-F201 (Dynamic risers) 62](#_Toc2448612)

[11.1.1 Geometrical Parameters 62](#_Toc2448613)

[11.1.2 Design Load Effects 62](#_Toc2448614)

[11.1.3 Load Effect Factors 63](#_Toc2448615)

[11.1.4 Resistance Factors 64](#_Toc2448616)

[11.2 Bursting 64](#_Toc2448617)

[11.3 System Hoop Buckling (Collapse) 64](#_Toc2448618)

[12 Input Data Specification 65](#_Toc2448619)

[12.1.1 Overview 65](#_Toc2448620)

[12.1.2 Design Matrix for Rigid Risers 66](#_Toc2448621)

[13 References. 67](#_Toc2448622)

[Appendix A – PROGRAM HISTORY 68](#_Toc2448623)

[Appendix B - ERROR LOG for Programs 69](#_Toc2448624)

[B.1 Calculation Errors 70](#_Toc2448625)

[Appendix C -test cases. 70](#_Toc2448626)

[Appendix D validation 70](#_Toc2448627)

# INTRODUCTION

The codes addressed in the document are:

* API RP 2RD
* API STD 2RD
* DNV OS F1O1
* DNV OS F2O1
* API RP 1111
* API 5C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Codes | Burst | Collapse | Combined Loading |  |
| API RP 2RD | ? | ? | ? |  |
| API STD 2RD |  |  | Method1:    Method2:    Method 3:  based on DNV-OS-F201  Method 4: |  |
| DNV OS F1O1 |  |  |  |  |
| DNV OS F2O1 |  |  |  |  |
| API RP 1111 |  |  |  |  |
| API 5C |  |  |  |  |

# THEORY

# Program Outline

## General Enhancements

* Set-up Logging. Important for checking and audit purposes
* ~~Folder structure:~~
  + ~~Consume a file in a remote\_folder~~
  + ~~Save files to following folders if remote\_folder:~~
    - ~~remote\_folder/results~~
    - ~~remote\_folder/logs~~
  + ~~Save files to following folder if default\_folder~~
    - ~~Results//default\_folder//~~
    - ~~Logs//default\_folder//~~
* ~~Filename should be associated with basename.~~
* Save copy of COD for audit purposes at end of project (Part of project close-out)a

## Geometry

## Material Properties

# VM Stress (Riser Code, API-RP-2RD)

* This section applies to metal risers made of steel or titanium.

|  |  |  |
| --- | --- | --- |
| **Primary** | Any normal or shear stress that is necessary to have static equilibrium of the imposed forces and moments. A primary stress is not self-limiting. Thus, if a primary stress substantially exceeds the yield strength, either failure or gross structural yielding will occur. | |
| **Membrane** | σp is the average value across the thickness of a solid section excluding the effects of discontinuities and stress concentrations. For example, the general primary membrane stress in a pipe loaded in pure tension is the tension divided by the cross-sectional area. Σp may include global bending as in the case of a simple pipe loaded by a bending moment. |
| **Bending** | σb is the portion of primary stress proportional to the distance from the centroid of a cross section, excluding the effects of discontinuities and stress concentrations. |
| **Secondary** | σq is any normal or shear stress that develops as a result of material restraint. This type of stress is self-limiting which means that local yielding can relieve the conditions that cause the stress, and a single application of load will not cause failure. | |

Table 1 –Stresses To Consider

### Combined Stresses

Principal stress components at each critical section should be combined using the von Mises yield criterion defined by the following equation.

(1)

Where

σ*e* = von Mises equivalent stress

σ1, σ2, σ3 =Principal stresses.

### Allowable Stresses

The von Mises equivalent stresses should be less than the allowable stresses defined by the right hand side of the following inequalities.

(2)  (3)

(4)

Where

σ*a* = *Ca*σ*y* = basic allowable combined stress.

*Ca* = allowable stress factor, *Ca* = 2/3

σ*y* = material minimum yield strength, defined for

Steel or titanium as the tensile stress required to

Produce a total elongation of 0.5 percent of the

Test specimen gage length.

*Cf* = design case factor

### Allowable Stress in Plain Pipe

For plain round pipe, where transverse shear and torsion are negligible, the three principal stress components of primary membrane stress (average stress across pipe wall) are σpr, σpθ, and σpz, where r, θ, and z refer to radial, hoop, and axial stresses. Thus,

(5)

For Thick Wall Pipe

(6)

(7)

(8)

Where:

Pi = internal pressure.

Po = external pressure.

Do, Di = outside, inside diameters.

t = pipe wall thickness.

A = .

T = true wall tension in pipe at section being analyzed.

M = global bending moment in pipe.

I = moment of inertia = .

Note that the criteria for (σp + σb)e and (σp + σb + σq)e are never controlling for this case.

Substituting Equations 6, 7, and 8 into Equation 5 gives, following a little algebra,

Where

Teff = T – PiAi + PoAo

And

Ai = πDi2 / 4

Ao = πDo2 / 4.

# API STD 2RD

### Burst Pressure

The minimum burst pressure of pipe is determined by Equation 1:

Where

k is a parameter to account for variability in mechanical properties and wall thickness, and is equal to 0.45 for API 5L or API SeT pipe;

D is the outside diameter of the pipe;

T is the nominal thickness of the pipe reduced for corrosion, wear and/or erosion as appropriate;

S is the specified minimum yield strength of the pipe;

U is the specified minimum ultimate strength of the pipe.

### Collapse Due to External Pressure

The collapse pressure for pipes can be calculated as a function of nominal wall thickness as per

Equation 2.

Where

P y is the yield collapse pressure, given by Equation 3;

Pel is the elastic collapse pressure, given by Equation 4.

Where

E is Young's modulus;

v is Poisson's ratio.

The collapse pressure may alternatively be calculated as a function of the elastic capacity, plastic

Capacity and the Ovality of the pipe.

Where

is pipe Ovality

Pp is the plastic collapse pressure of a pipe

Where

αfab is the fabrication factor

1.0 seamless pipe

αfab = 0.85 UOE pipe

0.925 UO/TRB pipe

The initial departure from circularity of pipe and pipe ends (i.e. the initial ovality).

The initial ovality shall not be taken less than 0.0025 (0.25 %). Ovalization caused during the construction and installation phase shall be included in the ovality. The ovalization due to external pressure or moment in the as-installed position shall not be included.

Collapse Due to Pure Bending

Pure bending can result in wrinkling of the pipe wall or flattening of the cross section. Both conditions are known as buckling due to bending. The bending strain that can result in such deformation can be calculated

### Tension

The tension capacity may be calculated by Equation 10.

Where

A is the pipe cross-section area, given by Equation 11.

### Internal Pressure

The casing pressure equal to the pressure caused by a tubing leak, extreme pressure in a drilling riser, hydrostatic test pressure, incidental pressure, and/or design pressure shall not exceed the pressure determined by Equation 14.

Where

Pe is the external pressure;

Pi is the internal pressure;

FD is a design factor and is given by Equation 15.

0.81 Production casing with tubing leak

0.81 Drilling riser with extreme pressure

FD = 0.90 Hydrostatic test

0.67 Incidental pressure

0.60 Design pressure

**Method 1**

The load combinations and associated design factors for Method 1 are given in Equations 18 and 19.

Where

T = Ta - PiAj + PeAo is the effective tension in the pipe;

Ta = σaA

σa is the axial stress in the pipe wall;

Ao, is the external cross section of the pipe

Ai is the internal cross section of the pipe

M is the moment in the pipe;

FD is a design factor, given by Equation 20.

**Method 2**

Method 2 limits axial load based on yield tension including the effect of internal pressure. The load combinations and associated design factors

Where

FD is a design factor,

0.80 SLS, ULS

FD =

1. ALS

**Method 3**

Method 3 is based on DNV-OS-F201, which uses a load and resistance factor design (LRFD) format. For load controlled conditions, refer to the section in F201 on Combined Load Criteria for Riser Pipes. For displacement controlled conditions, refer to the F201 section labeled Displacement Controlled Conditions. When the calculated moment exceeds 90 % of the plastic moment,

**Method 4**

Method 4 sets limits on combined axial load and pressure, without considering bending. Refer to API 1111 for further explanation. Bending limits are set based on bending strain.

**NOTE** there are cases such that the plastic moment limit set by Method 2 does not result in excessive bending strain (Le. the loading condition is displacement controlled).

The load combinations and associated criteria are given by

Where

FD is a design factor,

0.80 SLS, ULS

FD =

1. ALS

For sizing, limiting the load combination in Equation 24 to 0.67 (F D = 0.67) using static effective tension and normal operating pressure will usually provide a design that meets SLS, ULS and ALS conditions.

Method 4 sets limits for bending strain for cases of internal or external overpressure. The criteria are given by

ε ≤ F Dεb (internal overpressure)

Where

c is the bending-induced strain in the pipe;

F D is a design factor;

fc is the collapse factor for use with combined pressure and bending loads; see Equation 29.

0.5 SLS, ULS

FD =

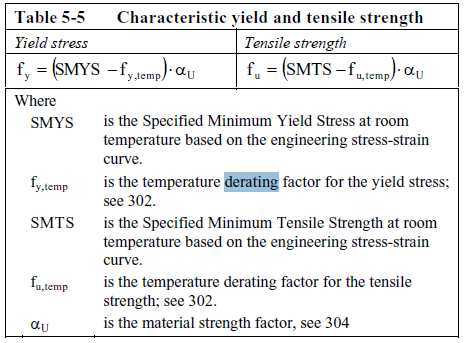
1. ALS

0.6 cold expanded pipe (e.g. DSAW)

Fc = 0.7 seamless pipe

### Temperature Derating

Temperature derating conducted based on the DNV-OS-F201



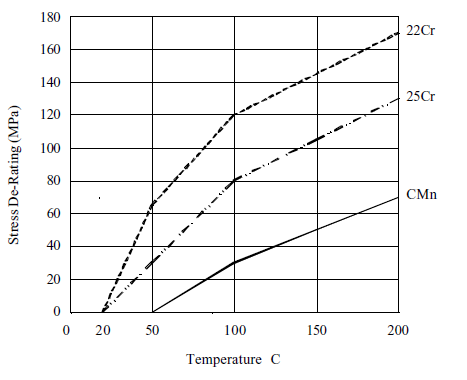


Figure 4‑1 Derating Value for Yield Strength

### APISTD2RD.yml

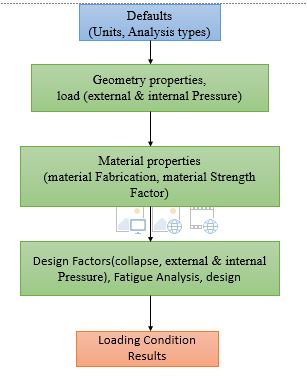
****

Figure 4‑1 APISTD2RD.yml

|  |  |  |
| --- | --- | --- |
|  | Inputs | Description |
| Defaults | Variable WT Burst : True  Variable WT Collapse : True  nominalWTAPISTD2RDMethod1 : True  nominalWTAPISTD2RDMethod2 : False  nominalWTAPISTD2RDMethod4 : False | We considering type of analysis for s  By giving the inputs statement True |
| Geometry Properties | Nominal ID: NULL  Nominal OD: 16  Design WT : 1.9  Nominal WT: - 1.0 to - 2.0  Corrosion Allowance: 0.236 Allowable Stress Factor: 0.667  **load:**  external Pressure: 0  internal Pressure: 8000  bending Moment: 0 | It is the given general geometry for pipe structure I.e. ID and OD and wall thickness |
| Material Properties | E: 30000000.00  rho: 7800  Poisson’s ratio: 0.30  SMYS: 65300  SMUS: 77600  k: 0.45  alphafab: 1  ovality: 0.01  material Fabrication:  Seamless: 1.0  UOE: 0.850  UO: 0.925  TRB: 0.925  Material Strength Factor: Normal: 0.96  Supplementary Requirement U: 1 |  |
| Design Factors | collapse: SLS: 0.8  ULS: 0.8  ALS: 1.0  Cold Expanded Pipe: 0.8  Seamless Pipe : 0.7  Internal Pressure :SLS: 0.8  ULS: 0.8  ALS: 1.0  Production Casing With Tubing Leak: 0.81  Drilling Riser With Extreme Pressure: 0.81 design: 0.600  incidental Pressure: 0.675  hydro Static Test: 0.750  external Pressure: SLS: 0.6  ULS: 0.6  ALS: 1.0  **Fatigue Analysis :**  Service Life: 0.1  Single ULS Event: 0.1  Single ALS Event: 1.0  **Design :**  pressure: 8000  temperature: 250  wall Thickness: 1.9 |  |
| Loading Condition Results | Label: 100yr Wave, 10yr Current, 750 psi  Data Labels: [Near, Far, Cross, Transverse]  Tension: [10578, 12635, 11558, 10838]  Bending Moment: [216, 406, 641, 594]  Plot Settings: Color: 'blue'  Marker: "o"  Marker fill style: 'blue”  Label: 100yr Current, 10yr Wave, 750 psi  Data Labels: [Near, Far, Cross, Transverse]  Tension: 1674  Tension: [12598, 10427, 9937, 12633]  Bending Moment: [181, 195, 441, 449]  Plot Settings: Color: 'blue'  Marker: "D"  Marker fill style: 'none'  Label: 100yr Wave, 10yr Current, 8000 psi  Data Labels: [Far]  Tension: [12633]  Bending Moment: [406]  Plot Settings: Color: 'red'  Marker: "o"  Marker fill style: 'none'  Label: 1yr Wave, 10000 psi  Data Labels: [Far]  Tension: [13762]  Bending Moment: [322]  Plot Settings: Color: 'green'  Marker: "v"  Marker fill style: 'green' |  |

### APISTD2RDMethods.py

|  |  |  |
| --- | --- | --- |
| **Inputs** | **Python Code** | **Description** |
| APISTD2RDBurst | def APISTD2RDBurst(data):  # No corrosion allowance considered for test pressure  Pb = 0.45\*(data["S"] + data["U"])\*math.log(data["OD"]/data["ID"])  Pt = data["designFactors"]["internalPressure"]["hydroStaticTest"] \* Pb  # Corrosion allowance considered for design and accidental pressure design  Pb=0.45\*(data["S"]data["U"])\*math.log(data["OD"]/(data["ID"]+2\*data["CorrosionAllowance"]))  Pd = data["designFactors"]["internalPressure"]["design"] \* PbPa  = data["designFactors"]["internalPressure"]["incidentalPressure"]\* Pb  Data.update({"Pb": Pb, "Pd": Pd, "Pa": Pa, "Pt": Pt})return(data) |  |
| Variable WT Burst | def variableWTBurst(cfg):  # Prepare input/output dataframeheader\_row = ['OD', 'ID', 't', 'Pd', 'Pa', 'Pt', 'Pb'] dataDF = pd.DataFrame(columns = header\_row)  if cfg['geometry']['NominalOD'] !=None:  for t in range(0, len(cfg['geometry']['NominalWT'])):newDFRow = [cfg['geometry']['NominalOD'],cfg['geometry']['NominalOD'] - 2\*cfg['geometry']['NominalWT'][t],cfg['geometry']['NominalWT'][t], None, None, None, None]dataDF.loc[len(dataDF)] = newDFRowelif cfg['geometry']['NominalID'] !=None:  for t in range(0, len(cfg['geometry']['NominalWT'])):newDFRow = [cfg['geometry']['NominalID']+2\*cfg['geometry']['NominalWT'][t],cfg['geometry']['NominalID'], cfg['geometry']['NominalWT'][t], None, None, None, None]dataDF.loc[len(dataDF)] = newDFRow  # Run the program row by row  for i in range(0,len(dataDF)): inputData = dataDF.loc[i].to\_dict() inputData.update({"k": cfg['material']['k'], "S": cfg['material']['SMYS'], \"U":cfg['material']['SMUS'],"CorrosionAllowance":cfg['geometry']['CorrosionAllowance'],\"designFactors":cfg['designFactors'] \})  output = APISTD2RDBurst(inputData)  dataDF.loc[i]["Pd"] = output["Pd"]  dataDF.loc[i]["Pa"] = output["Pa"]  dataDF.loc[i]["Pt"] = output["Pt"]  dataDF.loc[i]["Pb"] = output["Pb"]  logging.critical(dataDF.to\_string())  return (cfg, dataDF) |  |
| Section Properties | def sectionProperties(data):  A = (math.pi/4)\*(data["OD"]\*\*2-data["ID"]\*\*2)  Ai = (math.pi/4)\*(data["ID"]\*\*2)  Ao = (math.pi/4)\*(data["OD"]\*\*2)  I = (math.pi/64)\*(data["OD"]\*\*4-data["ID"]\*\*4)  data.update({"A": A, "Ai": Ai, "Ao": Ao, "I": I})  return data |  |
| APISTD2RDCollapse | def APISTD2RDCollapse(data):  #Collapse Pressure Check  Py = 2\*data["S"]\*(data["t"]/data["ID"])  #print(data)  Pel = 2\*data["E"]\*(data["t"]/data["ID"])\*\*3/(1-data["Poissionsratio"]\*\*2)  Pp = 2\*(data["t"]/data["OD"])\*data["S"]\*data["alphafab"]  Pc = Py\*Pel/math.sqrt(Py\*\*2+Pel\*\*2)data.update({"Py": Py, "Pel": Pel, "Pp": Pp, "Pc": Pc})  return(data) |  |
| APISTD2RDMethod1 | def APISTD2RDMethod1(data):  #check for internal over or external over pr if i.p to--  # Moment Used in method 1  #SigmaA = (data["AllowableStressFac"])\*(data["S"])  # Tension Check  Ty = (data["S"]\*data["A"])/1000\*4.448222 #kN  #Ta = SigmaA\*data["A"]/1000  #Te=(Tadata["internalPressure"]\*data["Ai"]+data["externalPressure"]\*data["Ao"])/1000 # Unit lbs  My=math.pi/4\*(data["S"]\*(data["OD"]data["t"])\*\*2\*data["t"])/1000/12\*1.355818 # Unit kN.mMp = (4/math.pi)\*My  #Method 1 Calculation  # Action: Check if internal overpressure  FDPressureCorrected  math.sqrt((data["designFactors"]["internalPressure"][data['LimitState']]\*\*2)-((data["internalPressure"]-data["externalPressure"])/data["Pb"])\*\*2)  Tlimiting = (FDPressureCorrected\*Ty)  #Method1 calculation  Mlimiting1 = FDPressureCorrected\*My  #Method2 calculation  Mlimiting2 = (FDPressureCorrected\*Mp)\*math.cos(0)  #Method 4 Calculation  TensionLimitCheck=(math.sqrt(((data["internalPressure"]data["externalPressure"])/data["Pb"])\*\*2)+(Tlimiting/Ty)\*\*2)/(data["designFactors"]["internalPressure"][data['LimitState']])  if TensionLimitCheck <= 1:  print ("TensionLimitCheckPass = True")  else:  print ("TensionLimitCheckPass = False")  # Action: Convert to Data Frame TypeError  #1 indicates Method1 calculation  #2 indicates Method2 calculation  # Method 1& 2 plotting calculation  TArray = np.arange(-Tlimiting, Tlimiting, 1)  #print(TArray)  MarrayPositive1 = []  MarrayNegative1 = []  MarrayPositive2 = []  MarrayNegative2 = []  for i in range(0, len(TArray)):  MarrayPositive1.append(My\*(FDPressureCorrected - abs(TArray[i]/Ty)))  MarrayNegative1.append(-My\*(FDPressureCorrected - abs(TArray[i]/Ty)))  #Method 2 Plotting  MarrayPositive2.append(Mp\*(FDPressureCorrected)\*math.cos((math.pi/2)\*((TArray[i]/Ty)/FDPressureCorrected)))MarrayNegative2.append(-Mp\*(FDPressureCorrected)\*math.cos((math.pi/2)\*((TArray[i]/Ty)/FDPressureCorrected)))  resultMatrix=np.stack((TArray,MarrayPositive1, MarrayNegative1,MarrayPositive2, MarrayNegative2) , axis=-1)  resultDF = pd.DataFrame(resultMatrix, columns = ['TArray','MarrayPositive1', 'MarrayNegative1','MarrayPositive2', 'MarrayNegative2'])  return(data, resultDF) |  |
| nominalWTAPISTD2RDMethod1 | def nominalWTAPISTD2RDMethod1(cfg, inputData):  inputData = sectionProperties(inputData)  inputData = APISTD2RDBurst(inputData)  inputData = APISTD2RDCollapse(inputData)resultData, resultDF = APISTD2RDMethod1(inputData)  return (resultData, resultDF) |  |
| APISTD2RDMethod1Utilization | def APISTD2RDMethod1Utilization(cfg):  #check for internal over or external over pr if i.p to--#  Ty(cfg['Method1SLS']["S"]\*cfg['Method1SLS']["A"])/1000\*4.448222 # Unit kN  Mymath.pi/4\*(cfg['Method1SLS']["S"]\*(cfg['Method1SLS']["OD"]cfg['Method1SLS']["t"])\*\*2\*cfg['Method1SLS']["t"])/1000/12\*1.355818 # Unit kN  for i in range(0, len(cfg['LoadingConditionResults'])): loadCategory=cfg['LoadingConditionResults'][i]['LoadCategory']  if (cfg['LoadingConditionResults'][i]['InternalPressure'] >= cfg['LoadingConditionResults'][i]['ExternalPressure']): FDPressureCorrected=math.sqrt((cfg["designFactors"]["internalPressure"][loadCategory]\*\*2)((cfg['LoadingConditionResults'][i]['InternalPressure']-cfg['LoadingConditionResults'][i]['ExternalPressure'])/cfg['Method1SLS']['Pb'])\*\*2)  cfg['LoadingConditionResults'][i]['PressureCondition']= "Internal OverPressure"  else:  FDPressureCorrected=math.sqrt((cfg["designFactors"]["externalPressure"][loadCategory]\*\*2)-((cfg['LoadingConditionResults'][i]['InternalPressure']-cfg['LoadingConditionResults'][i]['ExternalPressure'])/cfg['Method1SLS']['Pc'])\*\*2)  cfg['LoadingConditionResults'][i]['PressureCondition']= "Internal OverPressure"  LHSTensionTerm=cfg['LoadingConditionResults'][0]['Tension']/Ty  LHSBendingTerm=cfg['LoadingConditionResults'][0]['BendingMoment']/My  LHS = abs(LHSTensionTerm) + abs(LHSBendingTerm)  Utilization = LHS/FDPressureCorrected  cfg['LoadingConditionResults'][i]['Method1Utilization']= Utilization cfg['LoadingConditionResults'][i]['DesignFactor']= cfg["designFactors"]["internalPressure"][loadCategory]  return(cfg) |  |

### APISTD2RD.py

|  |  |  |
| --- | --- | --- |
| **Inputs** | **Python Code** | **Description** |
| APISTD2RD.py | # Data preparation  defaultYml = "dataManager\\API\_STD\_2RD\\APISTD2RD.yml"  # Get updateYML file  try:  if (sys.argv[1] != None):  updateYml = "dataManager\\API\_STD\_2RD\\" + sys.argv[1]  logging.critical("Updating default values with contents in file {0}" .format(updateYml) )  except:  updateYml = None  logging.critical("No update values file is provided. Running program default values")  # Get updated configuration file for Analysis  cfg = ymlInput(defaultYml, updateYml)  try:  cfg['FileName'] = updateYml.split('\\')[1].split('.')[0]  except:  cfg['FileName'] = defaultYml.split('\\')[1].split('.')[0]  cfg = customUpdate(cfg)  # Set logging  setLogging(cfg['default']['logLevel'])  if cfg['default']['Analysis']['variableWTBurst'] == True:  (cfg, dataDF) = variableWTBurst(cfg) plotDefault(cfg, dataDF)  saveData(cfg, dataDF)  if cfg['default']['Analysis']['variableWTCollapse'] == True:pass  # variableWTCollapse(cfg)  if cfg['default']['Analysis']['nominalWTAPISTD2RDMethod1'] == True:  UtilizationDF = []  forMethod1Indexinrange(0,len(cfg['nominalWTAPISTD2RDMethod1']['data'])):  # Prepare input data  if cfg['geometry']['NominalOD'] !=None:  cfg['geometry']['NominalID'] = cfg['geometry']['NominalOD'] - 2\*cfg['geometry']['DesignWT']elif cfg['geometry']['NominalID'] !=None:  cfg['geometry']['NominalOD'] = cfg['geometry']['NominalID'] + 2\*cfg['geometry']['DesignWT']  inputData = ({"OD": cfg['geometry']['NominalOD'],"ID": cfg['geometry']['NominalID'],  "t": cfg['geometry']['DesignWT'], "E": cfg['material']['E'],  "Poissionsratio": cfg['material']['Poissionsratio'], "alphafab": cfg['material']['alphafab'],  "k": cfg['material']['k'], "S": cfg['material']['SMYS'],  "U": cfg['material']['SMUS'],  "CorrosionAllowance": cfg['nominalWTAPISTD2RDMethod1']['data'][Method1Index]['CorrosionAllowance'],  "designFactors": cfg['designFactors'],  # "AllowableStressFac": cfg['geometry']['AllowableStressFac'],  "externalPressure": cfg['nominalWTAPISTD2RDMethod1']['data'][Method1Index]['ExternalPressure'],  "internalPressure": cfg['nominalWTAPISTD2RDMethod1']['data'][Method1Index]['InternalPressure'],  "LimitState": cfg['nominalWTAPISTD2RDMethod1']['data'][Method1Index]['LimitState']  })  resultData, DF = nominalWTAPISTD2RDMethod1(cfg, inputData)  UtilizationDF.append(DF)  cfg['Method1SLS'] = resultData  # Prepare plot  plotDefaultM1(cfg, UtilizationDF)  # cfg = APISTD2RDMethod1Utilization(cfg)  # saveData(cfg, dataDF) |  |

### Summary.py

|  |  |  |
| --- | --- | --- |
| **Inputs** | **Python Code** | **Description** |
| **Summary.py** | **# Data preparation**  **defaultYml = "dataManager\\API\_STD\_2RD\\summary.yml"**  **# Get updateYML file**  **try:**  **if (sys.argv[1] != None):**  **updateYml = "dataManager\\API\_STD\_2RD\\" + sys.argv[1]**  **logging.critical("Updating default values with contents in file {0}".format(updateYml))**  **except:**  **updateYml = None**  **logging.critical("No update values file is provided. Running program default values")**  **# Get updated configuration file for Analysis**  **cfg = ymlInput(defaultYml, updateYml)**  **# Set logging**  **setLogging(cfg['default']['logLevel'])**  **# Get file List**  **data = {"Folder": cfg['ymlFiles']['Folder'], "FileNameFilter": cfg['ymlFiles']['FileNameFilter']}fileList = fileList(data)**  **# Get data**  **dataDF = extractData(fileList, cfg)**  **# Save Data**  **fileName = cfg['dataFrame']['label']**  **saveDataFrame(dataDF, fileName)** |  |

# API RP 2RD

## Combined Stresses

Principal stress components at each critical section should be combined using the von Mises yield criterion defined by the following equation.

Where

σe = von Mises equivalent stress.

σ1, σ2, σ3 = principal stresses.

## Allowable Stresses

The von Mises equivalent stresses should be less than the allowable stresses defined by the right hand side of the following inequalities.

where

σa = Caσy = basic allowable combined stress.

Ca = allowable stress factor, Ca = 2/3

σy = material minimum yield strength, defined for steel or titanium as the tensile stress required to produce a total elongation of 0.5 percent of the test specimen gage length.

Cf = design case factor.

## Allowable Stress in Plain Pipe

For plain round pipe, where transverse shear and torsion are negligible, the three principal stress components of primary membrane stress (average stress across pipe wall) are σpr, σpθ, and σpz, where r, θ, and z refer to radial, hoop, and axial stresses.

Moreover, for a thick walled pipe,

Where

Pi = internal pressure.

Po = external pressure.

Do, Di = outside, inside diameters.

t = pipe wall thickness.

A =

T = true wall tension in pipe at section being analysed.

M = global bending moment in pipe.

I = moment of inertia =

Note that the criteria for (σp + σb)e and (σp + σb + σq)e are never controlling for this case.

## API RP 2RD.ini

|  |  |  |  |
| --- | --- | --- | --- |
| **Python Code** | Code | Reference/ Notes | Units |
| **[codeParameterInput]**  **AllowableStressFac=0.666**  **DesignCaseFac=1** |  |  |  |
| **[pipeInput]**  **NominalOD=0.24765**  **NominalWT=0.034925**  **MinimumWT=0.03492**  **NominalID=0.1778** |  |  |  |
| **[loadingData]**  **ExternalPressure=0**  **InternalPressure=0**  **TensionEffective=0**  **Moment=-460000** |  |  |  |
| **[Material Properties]**  **YieldStrength = 80** |  |  |  |

## APIRP2RD.yml

|  |  |  |
| --- | --- | --- |
|  | Inputs | Description |
| Code Parameter | Allowable Stress Fac: 0.666  Design Case Fac: 1 |  |
| Pipe properties | NominalOD: 0.24765  NominalWT: 0.034925  MinimumWT: 0.034925  NominalID: 0.1778 |  |
| load | ExternalPressure: 0  InternalPressure: 0  EffectiveTension: 0  BendingMoment: -460000 |  |
| Material Properties | YieldStrength = 5.52E+08 |  |
| units | codeParameterInput: #  pipeInput: m  loadingData: kNm  TensionEffective: N  Moment: Nm  YieldStength : pa |  |

## APIRP2RD\_Methods.py

|  |  |  |
| --- | --- | --- |
| **Inputs** | **Python Code** | **Description** |
| **\_\_init\_\_** | **class APIRP:**  **def\_\_init\_\_(self,APIRPNominalOD\_m,APIRPNominalWT\_m,APIRPMinimumWT\_m,APIRPExternalPressure,APIRPInternalPressure,APIRPTensionEffective,APIRPMoment,APIRPYieldStrength):**  **self.APIRPNominalOD\_m = APIRPNominalOD\_m # Outside diameter**  **self.APIRPNominalWT\_m = APIRPNominalWT\_m # Nominal Wall Thickness**  **self.APIRPMinimumWT\_m = APIRPMinimumWT\_m # APIRP wall Thickness**  **self.APIRPExternalPressure = APIRPExternalPressure # External Pressure self.APIRPInternalPressure= APIRPInternalPressure # Internal Pressure**  **self.APIRPTensionEffective = APIRPTensionEffective # N**  **self.APIRPMoment = APIRPMoment # Global bending moment in APIRP N self.APIRPYieldStrength = APIRPYieldStrength # pa**  **## We can define the APIRP values by creating a object called 'APIRP'**  **APIRP1 = APIRP(0.24765,0.034925,0.034925,0,0,0,0,5.52E+08)**  **APIRP2 = APIRP(0.24765,0.034925,0.034925,0,0,0,10E+04,5.52E+08)**  **APIRP3 = APIRP(0.24765,0.034925,0.034925,0,0,0,16E+04,5.52E+08)**  **APIRP4 = APIRP(0.24765,0.034925,0.034925,0,0,0,24E+04,5.52E+08)**  **APIRP5 = APIRP(0.24765,0.034925,0.034925,0,0,0,30E+04,5.52E+08)**  **APIRP6 = APIRP(0.24765,0.034925,0.034925,0,0,0,36E+04,5.52E+08)**  **APIRP7 = APIRP(0.24765,0.034925,0.034925,0,0,0,42E+04,5.52E+08)**  **APIRP8 = APIRP(0.24765,0.034925,0.034925,0,0,0,46.5E+04,5.52E+08)** |  |

## APIRP2RD.py

|  |  |  |
| --- | --- | --- |
| **APIRP2RDCalculation** | **def APIRP2RDCalculation(APIRP2RD):**  **fromcustom.API\_RP\_2RD.APIRP2RD\_MethodsimportAPIRP1,APIRP2,APIRP3,APIRP4,APIRP5,APIRP6,APIRP7,APIRP8**  **## APIRP Calculation**  **APIRPNominalID=APIRP2RD.APIRPNominalOD\_m2\*APIRP2RD.APIRPNominalWT\_m**  **# inside diameter**  **print("APIRP\_NominalInsideDiameter : ","{:.4e}".format(APIRPNominalID))**  **APIRPMinimumID\_m = APIRP2RD.APIRPNominalOD\_m -(2\*APIRP2RD.APIRPMinimumWT\_m)**  **print('APIRP\_MinimumID\_m : ',APIRPMinimumID\_m )**    **APIRPA = (math.pi/4)\*(APIRP2RD.APIRPNominalOD\_m\*\*2 - APIRPMinimumID\_m\*\*2) # Area print("APIRP\_Area : ","{:.3e}".format(APIRPA))**  **APIRPAi = (math.pi/4)\*(APIRPMinimumID\_m\*\*2print("APIRP\_Ai : ","{:.3e}".format(APIRPAi))**  **APIRPAo = (math.pi/4)\*(APIRP2RD.APIRPNominalOD\_m\*\*2) print("APIRP\_Ao : ","{:.3e}".format(APIRPAo))**  **APIRPI = (math.pi/64)\*(APIRP2RD.APIRPNominalOD\_m\*\*4 - APIRPMinimumID\_m\*\*4) # Moment of Intertiaprint("APIRP\_Intertia : ","{:.3e}".format(APIRPI))**  **APIRPTensionTrueT=APIRP2RD.APIRPTensionEffective+(APIRP2RD.APIRPInternalPressure\*APIRPAi)-(APIRP2RD.APIRPExternalPressure\*APIRPAo print("APIRP\_TensionTrue\_T : ","{0:.2f}".format(APIRPTensionTrueT))**  **#Stress Calculation**  **SigmaA = APIRP2RD.AllowableStressFac\*APIRP2RD.APIRPYieldStrength**  **# Basic**  **print("SigmaA : ","{:.3e}".format(SigmaA))**  **SigmaACf = APIRP2RD.DesignCaseFac\*SigmaA # R.H.S of Equation**  **#print("SigmaACf in ksi : ","{:.3e}".format(SigmaACf))**  **#print("SigmaACf in pa : ","{:.3e}".format(SigmaACf\*6894757.293178))**  **print("Stress(R.H.S) in kPa : ","{0:.2f}".format(SigmaACf\*1E-3))**  **print("Stress(R.H.S) in ksi: ","{0:.2f}".format(SigmaACf\*1.45038E-07))**    **sigmaRadial\_1=((APIRP2RD.APIRPExternalPressure\*APIRP2RD.APIRPNominalOD\_m)+(APIRP2RD.APIRPInternalPressure\*APIRPNominalID))/(APIRP2RD.APIRPNominalOD\_m+APIRPNominalID)**  **print("sigmaRadial\_1 : ","{:.3f}".format(sigmaRadial\_1))**  **sigmaCircumferential\_1=(((APIRP2RD.APIRPInternalPressureAPIRP2RD.APIRPExternalPressure)\*((APIRP2RD.APIRPNominalOD\_m)/(2\*APIRP2RD.APIRPMinimumWT\_m)))-APIRP2RD.APIRPInternalPressure)**  **print("sigmaCircumferential\_1 : ", "{:.3f}".format(sigmaCircumferential\_1))**  **sigmaAxial\_1=(APIRPTensionTrueT/APIRPA)+((APIRP2RD.APIRPMoment)/(2\*APIRPI))\*(APIRP2RD.APIRPNominalOD\_mAPIRP2RD.APIRPMinimumWT\_m)**  **print("sigmaAxial\_1 : ","{:.3e}".format(sigmaAxial\_1))**  **sigmaAxial\_2=(APIRPTensionTrueT/APIRPA)-((APIRP2RD.APIRPMoment)/(2\*APIRPI))\*(APIRP2RD.APIRPNominalOD\_m-APIRP2RD.APIRPMinimumWT\_m)**  **print("sigmaAxial\_2 : ","{:.3e}".format(sigmaAxial\_2))**  **AllowableStress=((sigmaRadial\_1sigmaCircumferential\_1)\*\*2)+((sigmaCircumferential\_1-sigmaAxial\_1)\*\*2)+((sigmaAxial\_1-sigmaRadial\_1)\*\*2)**  **#print("AllowableStress ","{:.3e}".format((1/sqrt(2))\*(sqrt(AllowableStress))))**  **#print("AllowableStress\_LHS : ","{:.2f}".format((1/sqrt(2))\*(sqrt(AllowableStress)\*0.000000145037738007)))**  **LHS\_Squared\_1=(sigmaRadial\_sigmaCircumferential\_1)\*\*2+(sigmaCircumferential\_1 - sigmaAxial\_2)\*\*2 + (sigmaAxial\_1 - sigmaRadial\_1)\*\*2**  **#print("LHS\_Squared\_1 : ","{:.3e}".format(LHS\_Squared\_1))**  **LHS\_Squared\_2 = (sigmaRadial\_1 - sigmaCircumferential\_1)\*\*2+(sigmaCircumferential\_1 - sigmaAxial\_2)\*\*2 + (sigmaAxial\_2 - sigmaRadial\_1)\*\*2**  **#print("LHS\_Squared\_2 : ","{:.3e}".format(LHS\_Squared\_2))**  **VMStress\_1\_Pa = (1/math.sqrt(2))\*(math.sqrt(LHS\_Squared\_1))**  **print("VMStress\_1\_Pa : ","{:.3e}".format(VMStress\_1\_Pa))**  **VMStress\_2\_Pa = (1/math.sqrt(2))\*(math.sqrt(LHS\_Squared\_2))**  **print("VMStress\_2\_Pa : ","{:.3e}".format(VMStress\_2\_Pa))**  **VMStress\_Pa = max(VMStress\_2\_Pa,VMStress\_1\_Pa)**  **VMStress\_MPa = VMStress\_Pa \* 1E-6**  **VMStress\_ksi = VMStress\_Pa \* 1.45038E-007**  **trueWallTensionLimit = SigmaACf\*APIRPA**  **print("Tension Limit in kN : ","{0:.2f}".format(trueWallTensionLimit\*1E-3))**  **bendingMoment=APIRP2RD.DesignCaseFac\*APIRP2RD.APIRPYieldStrength\*APIRPI/(APIRP2RD.APIRPNominalOD\_m - APIRP2RD.APIRPNominalWT\_m)**  **bendingMomentLimit = math.sqrt(bendingMoment) print("Bending Moment Limit in kN.m : ","{0:.2f}".format(bendingMomentLimit\*1E-3))**  **internalPressureLimit = (SigmaACf\*math.sqrt(2))/math.sqrt(((-APIRPNominalID/(APIRPNominalID + APIRP2RD.APIRPNominalOD\_m))-((APIRP2RD.APIRPNominalOD\_m/(2\*APIRP2RD.APIRPMinimumWT\_m))-1))\*\*2 +(((APIRP2RD.APIRPNominalOD\_m/(2\*APIRP2RD.APIRPMinimumWT\_m))-1)-(APIRPAi/APIRPA))\*\*2 +((APIRPAi/APIRPA)+(APIRPNominalID/(APIRPNominalID+ APIRP2RD.APIRPNominalOD\_m)))\*\*2)**  **print("internal Pressure Limit in MPa : ","{0:.2f}".format(internalPressureLimit\*1E-6))**  **externalPressureLimit=(SigmaACf\*math.sqrt(2))/math.sqrt(((APIRP2RD.APIRPNominalOD\_m/(2\*APIRP2RD.APIRPMinimumWT\_m))-(APIRP2RD.APIRPNominalOD\_m/(APIRPNominalID + APIRP2RD.APIRPNominalOD\_m)))\*\*2 +((APIRP2RD.APIRPNominalOD\_m/(2\*APIRP2RD.APIRPMinimumWT\_m))+(APIRPAo/APIRPA))\*\*2+((APIRPAo/APIRPA)+(APIRP2RD.APIRPNominalOD\_m/(APIRPNominalID + APIRP2RD.APIRPNominalOD\_m)))\*\*2)**  **print("external Pressure Limit in MPa : ","{0:.2f}".format(externalPressureLimit\*1E-6))**  **Tension\_LHS = 2\*(((APIRP2RD.APIRPMoment/(2\*APIRPI))\*(APIRP2RD.APIRPNominalOD\_m-APIRP2RD.APIRPMinimumWT\_m))\*\*2)**  **#Tension\_LHS = 2\*(((APIRP2RD.APIRPMoment/(2\*APIRPI))\*(APIRP2RD.APIRPNominalOD\_m- APIRP2RD.APIRPMinimumWT\_m))\*\*2)**  **#print("Tension\_LHS\_"+str(count)+" : ","{:.2f}".format(Tension\_LHS))**  **Tension\_RHS = (2\*(SigmaACf\*\*2)-(sigmaRadial\_1 -sigmaCircumferential\_1)\*\*2-Tension\_LHS)/2**  **#print("Tension\_RHS\_"+str(count)+" : ","{:.2f}".format(Tension\_RHS))**  **## #calculating the Equation**    **count =0**  **x = range(8)**  **list1 = [APIRP1, APIRP2,APIRP3,APIRP4,APIRP5,APIRP6,APIRP7,APIRP8]**  **TensionList1 = []**  **TensionList2 = []**  **BendMomentList = []**  **for i in list1:**  **count = 1+count**  **print("BendingMoment\_APIRP"+str(count)+" : ","{:.2f}".format(i.APIRPMoment))**  **#print("BendingMoment\_APIRP"+str(count)+" : ","{:.2f}".format(APIRP2RD.APIRPMoment))**  **Tension\_LHS = 2\*(((i.APIRPMoment/(2\*APIRPI))\*(i.APIRPNominalOD\_m-i.APIRPMinimumWT\_m))\*\*2)**  **#Tension\_LHS = 2\*(((APIRP2RD.APIRPMoment/(2\*APIRPI))\*(APIRP2RD.APIRPNominalOD\_m- APIRP2RD.APIRPMinimumWT\_m))\*\*2)**  **print("Tension\_LHS\_"+str(count)+" : ","{:.2f}".format(Tension\_LHS))**  **Tension\_RHS = (2\*(SigmaACf\*\*2)-(sigmaRadial\_1 -sigmaCircumferential\_1)\*\*2-Tension\_LHS)/2**  **print("Tension\_RHS\_"+str(count)+" : ","{:.2f}".format(Tension\_RHS))**  **a = (1/APIRPA)\*\*2**  **b = (4\*((i.APIRPMoment/(2\*APIRPI))\*(APIRP2RD.APIRPNominalOD\_m- APIRP2RD.APIRPMinimumWT\_m)))\*(1/APIRPA)**  **c = -Tension\_RHS**  **Tension\_positive1 = (-b+(math.sqrt(b\*\*2-(4\*a\*c))))/(2\*a)**  **Tension\_negative1 = (-b-(math.sqrt(b\*\*2-(4\*a\*c))))/(2\*a)**    **TensionList1.append(Tension\_positive1/1000)**  **TensionList2.append(Tension\_negative1/1000)**  **BendMomentList.append(i.APIRPMoment/1000)**  **#print(TensionList1)**  **#print(TensionList2)**  **print(BendMomentList)**  **x1 = [-a for a in BendMomentList]**  **x2 = x1[::-1] # Reversing x1 values**  **x3 = BendMomentList**  **x4 = x3[::-1]**  **x\_axis = []**  **x\_axis += x1+x2+x3+x4 # Combining all values in one set like x1,x2,x3 and x4**  **print(x\_axis)**  **y1 = TensionList1**  **y2 = y1[::-1]**  **y2 = [-b for b in y2]**  **y3 = [-c for c in y1]**  **y4 = y1[::-1]**  **y\_axis = []**  **y\_axis += y1+y2+y3+y4**  **print(y\_axis)**    **#it creates plot tittle, color and size**  **plt.title('Effective Tension vs Bending Moment plot', fontsize=14, fontweight='bold', color='black')**  **#it creates plot X-axis name, fontsize and colour**  **plt.xlabel('Bending Moment (kips-ft)', fontsize=12, fontweight='bold', color='black')**  **#it creates plot Y-axis name, fontsize and colour**  **plt.ylabel('Effective Tension (kips)', fontsize=12, fontweight='bold', color='black')**  **#it converted Static max.effective tension value KN to Te and it plots these values along with arc length**  **plt.plot(x\_axis, y\_axis)**  **plt.savefig("results\\APIRP2RD\\Tension vs Moment.png",dpi=800)**  **plt.xlim(0, 1.1\*max(x3))**  **plt.ylim(0, 1.1\*max(y1))**  **plt.plot(x\_axis, y\_axis)**  **plt.savefig("results\\APIRP2RD\\Tension vs Moment\_Quad1.png",dpi=800)**  **APIRP2RDCalculation(configParams.APIRP2RD)** |  |

# API 6A

Calculation for pressure vessels subjected to stress from internal pressure, axial loads and an external bending moment.

|  |  |
| --- | --- |
| Material | API |
| Sy | 80000 psi |
| Syt | 80000 psi |
| Sut | 95000 psi |

Table 2 Material Data

Design Stress intensity at Operating Condition

2/3.Sy otherwise

Maximum Stress Intensity at Hydrotest Condition

5/6.Syt otherwise

Maximum Combined Primary and Secondary Stress Intensity

2.Sy otherwise

# ASME B31.8

### Steel Pipe Design Formula

The design pressure for steel gas piping systems or the nominal wall thickness for a given design pressure shall be determined by the following formula (for limitations).

Where

D nominal outside diameter of pipe, in.

E longitudinal joint factor obtained

F design factor obtained

P design pressure

S specified minimum yield strength, psi, stipulated in the specifications under which the pipe was purchased from the manufacturer or determined in accordance

T temperature derating factor obtained

t nominal wall thickness, in.

**Plastic Pipe and Tubing Design Formula**.

The design pressure for plastic gas piping systems or the nominal wall thickness for a given design pressure

D = Specified Outside Diameter,in

P = Design pressure, psig

S = for thermoplastic pipe and tubing, long-term hydrostatic strength determined in accordance with the listed specification at a temperature equal to 73°F, lOO°F, 120°F, or 140oP; for reinforced thermosetting plastic pipe, use 11,000 psi

t = specified wall thickness,in.

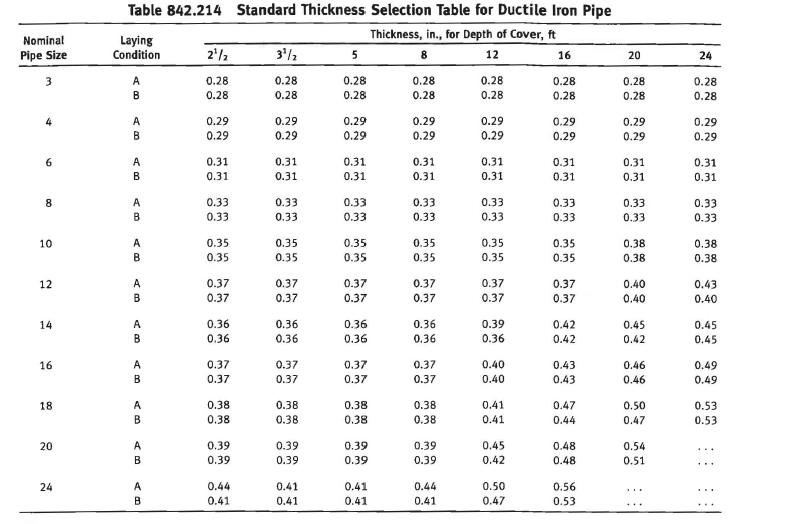


Figure 7‑1 Standard Thickness Selection Table for Ductile Iron Pipe

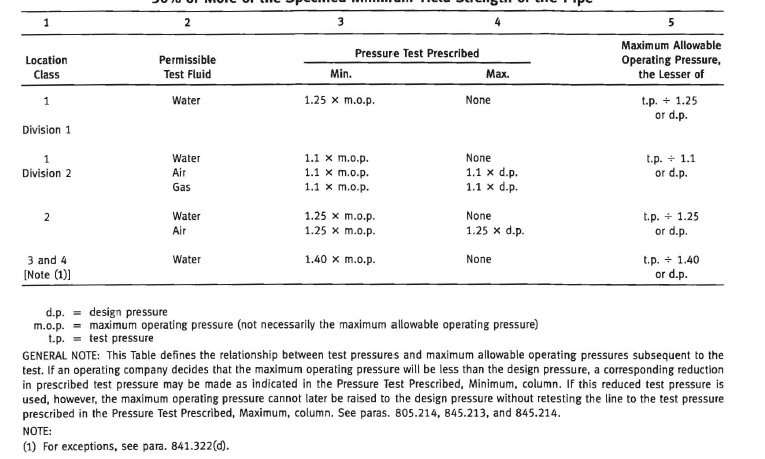


Figure 7‑2 Test Requirements for Pipelines and Mains to Operate at Hoop Stresses of

30% or More of the Specified Minimum Yield Strength of the Pipe

### B31.yml

|  |  |  |
| --- | --- | --- |
|  | Inputs | Description |
| Default | Constants:  g : 9.81  units:  verticalDistance: m  declinationAngle: deg with vertical  Temperature: deg F  Pressure: psi  EffectiveTension: kN  BendingMoment: kNm  YieldStrength: psi  E: psi  NominalID: inch  NominalOD: inch  NominalWT: inch  Rho: kg/m3  MassPerUnitLength: kg/m  WeightPerUnitLength: N/m  SoilStiffness: kN/m/m  BuoyancyCoverage: Percentage  ThermalExpansionCoefficient: in/in/deg F  logLevel: DEBUG  Analysis:  Code : ASME B31.4  Extreme: False  Fatigue: False |  |
| Geometry | NominalID: NULL  NominalOD: 18  DesignWT : 0.75  CorrosionAllowance: 0.0 |  |
| Design: | InternalPressure: 2220  ExternalPressure: 0  Temperature:  Ambient: 50  Operating: 82  Maximum: NULL  BendingMoment: 0  AxialForce: 0  Torsion: 0  Condition: Restrained |  |
| Material: | E: 30000000.00  rho: 7800  Poissionsratio: 0.30  SMYS: 60000  SMUS: NULL  ThermalExpansionCoefficient: 6.5E-6  WeldFactor:  Seamless: 1.0 |  |
| Design Factors: | Pressure : 0.72  Longitudinal : 0.90  EquivalentStress : 0.90 |  |
| Fatigue: | serviceLife: NULL  singleULSEvent: NULL  singleALSEvent: NULL |  |

### Summary\_ASMEB31.yml

|  |  |  |
| --- | --- | --- |
|  | Inputs | Description |
| Default | Constants:  g: 9.81  units:  verticalDistance: m  declinationAngle: deg with vertical  Temperature: deg F  Pressure: psi  EffectiveTension: kN  BendingMoment: kNm  YieldStrength: psi  E: psi  NominalID: inch  NominalOD: inch  NominalWT: inch  Rho: kg/m3  MassPerUnitLength: kg/m  WeightPerUnitLength: N/m  SoilStiffness: kN/m/m  BuoyancyCoverage: Percentage  ThermalExpansionCoefficient: in/in/deg F  logLevel: DEBUG  Analysis:  Code: ASME B31.4  Extreme: true  Fatigue: false |  |
| Geometry: | NominalID: null  NominalOD: 16  DesignWT: 0.942  CorrosionAllowance: 0.0 |  |
| Design: | InternalPressure: 3180  ExternalPressure: 0  Temperature:  Ambient: 50  Operating: 140  Maximum: null  BendingMoment: 0  AxialForce: 0  Torsion: 0  Condition: Restrained |  |
| Material: | E: 30000000.0  rho: 7800  Poissionsratio: 0.3  SMYS: 65300  SMUS: null  ThermalExpansionCoefficient: 6.5e-06  WeldFactor:  Seamless: 1.0 |  |
| DesignFactors: | Pressure: 0.6  Longitudinal: 0.9  EquivalentStress: 0.9  Logitudinal: 0.9 |  |
| Fatigue: | serviceLife: null  singleULSEvent: null  singleALSEvent: null |  |
| Result: | TemperatureDerating: 1ASMEB314:  InternalPressure:  S: 65300  t: 0.942  D: 16  Pi: 3180  Po: 0  F: 0.6  WeldFactor: 1.0  T: 1  MaximumDesignPressure: 4613.445 MinimumWallThickness\_Pressure:0.6239882266372333  LogitudinalStress:  S: 65300  t: 0.942  D: 16  Pi: 3180  Po: 0  M: 0  Fa: 0  F: 0.9  WeldFactor: 1.0  E: 30000000.0  Poissionsratio: 0.3  T: 1  Alpha: 6.5e-06  T1: 140  T2: 50  Condition: Restrained  MinimumWallThickness\_Longitudinal:0.09937888198757763  Stress\_Elongation: -17550.0  Stress\_Logitudinal: 58770.0  Stress\_Hoop: 254400.0  Stress\_Axial: 0  Stress\_Moment: 0  EquivalentStress:  S: 65300  t: 0.942  D: 16  Pi: 3180  Po: 0  M: 0  Fa: 0  F: 0.9  WeldFactor: 1.0  E: 30000000.0  Poissionsratio: 0.3  T: 1  Alpha: 6.5e-06  T1: 140  T2: 50  Condition: Restrained  MinimumWallThickness\_Equivalent: 0.41871949069285974  Stress\_Elongation: -17550.0  Stress\_Hoop: 59166.66541794879  Stress\_Axial: 0  Stress\_Moment: 0  Stress\_Equivalent: 58770.0 |  |

### ASMEMethods.py

|  |  |  |
| --- | --- | --- |
| **Inputs** | **Python Code** | **Description** |
| **ASMEB314InternalPressure** | **def ASMEB314InternalPressure(data):**  **DesignPressure = 2\*data['S']\*data['t']/data['D']\*data['F']\*data['WeldFactor']\*data['T]**  **Stress\_Hoop = data['S']\*data['F']\*data['WeldFactor']\*data['T']**  **if data['D']/data['t'] >= 30:**  **ThicknessMinimum\_Pressure = (data['Pi']-data['Po'])\*data['D']/(2\*Stress\_Hoop)**  **else:**  **ThicknessMinimum\_Pressure = (data['Pi']-data['Po'])\*data['D']/(2\*Stress\_Hoop + (data['Pi']-data['Po']))**    **data.update({"MaximumDesignPressure": DesignPressure, "MinimumWallThickness\_Pressure": ThicknessMinimum\_Pressure})**  **return data** |  |
| **ASMEB314LogitudinalStress** | **def ASMEB314LogitudinalStress(data):**  **if data['Condition'] == "Restrained":**  **Stress\_Elongation = -data['E']\*data['Alpha']\*(data['T1'] - data['T2'])**  **Stress\_Longitudinal = data['S']\*data['F']\*data['WeldFactor']\*data['T']**  **Stress\_Moment = 0**  **Stress\_Axial = 0**  **Stress\_Hoop = (Stress\_Longitudinal - Stress\_Elongation - Stress\_Moment - Stress\_Axial)/data['Poissionsratio']**    **DbytRatio = data['D']/data['t']**  **if DbytRatio >= 30:**  **ThicknessMinimum\_Longitudinal = (data['Pi']-data['Po'])\*data['D']/(2\*Stress\_Hoop)**  **else:**  **ThicknessMinimum\_Longitudinal = (data['Pi']-data['Po'])\*data['D']/(2\*Stress\_Hoop + (data['Pi']-data['Po']))**  **data.update({"MinimumWallThickness\_Longitudinal": ThicknessMinimum\_Longitudinal, \**  **"Stress\_Elongation": Stress\_Elongation, "Stress\_Logitudinal": Stress\_Longitudinal,**  **"Stress\_Hoop": Stress\_Hoop, "Stress\_Axial": Stress\_Axial,**  **"Stress\_Moment": Stress\_Moment})**  **return data** |  |
| **ASMEB314EquivalentStress** | **def ASMEB314EquivalentStress(data):**  **if data['Condition'] == "Restrained":**  **Stress\_Elongation = -data['E']\*data['Alpha']\*(data['T1'] - data['T2'])**  **Stress\_Equivalent = data['S']\*data['F']\*data['WeldFactor']\*data['T']**  **Stress\_Moment = 0**  **Stress\_Axial = 0**  **Stress\_Torsion = 0**  **Stress\_Hoop = ((-1\*math.sqrt(Stress\_Equivalent\*\*2/2 - Stress\_Torsion)/(data['Poissionsratio']-1)) \**  **- Stress\_Elongation - Stress\_Moment)/(1+data['Poissionsratio'])**  **DbytRatio = data['D']/data['t']**  **if DbytRatio >= 30:**  **ThicknessMinimum\_Equivalent = (data['Pi']-data['Po'])\*data['D']/(2\*Stress\_Hoop)**  **else:**  **ThicknessMinimum\_Equivalent = (data['Pi']-data['Po'])\*data['D']/(2\*Stress\_Hoop + (data['Pi']-data['Po']))**  **data.update({"MinimumWallThickness\_Equivalent": ThicknessMinimum\_Equivalent, \**  **"Stress\_Elongation": Stress\_Elongation,**  **"Stress\_Hoop": Stress\_Hoop, "Stress\_Axial": Stress\_Axial,**  **"Stress\_Moment": Stress\_Moment, "Stress\_Equivalent" : Stress\_Equivalent})**  **return data** |  |
| **ASMEB318InternalPressure** | **def ASMEB318InternalPressure(data):**  **DesignPressure = 2\*data['S']\*data['t']/data['D']\*data['F']\*data['WeldFactor']\*data['T']**  **DbytRatio = data['D']/data['t']**  **if DbytRatio >= 30:**  **ThicknessMinimum\_Pressure = (data['Pi']-data['Po'])\*data['D']/(2\*data['S']\*data['F']\*data['WeldFactor']\*data['T'])**  **else:**  **ThicknessMinimum\_Pressure = (data['Pi']-data['Po'])\*data['D']/(2\*data['S']\*data['F']\*data['WeldFactor']\*data['T'] + (data['Pi']-data['Po']))**  **data.update({"MaximumDesignPressure": DesignPressure,**  **"MinimumWallThickness": ThicknessMinimum\_Pressure})**  **return data** |  |
| **ASMEB318LogitudinalStress** | **def ASMEB318LogitudinalStress(data):**  **if data['Condition'] == "Restrained":**  **Stress\_Elongation = -data['E']\*data['Alpha']\*(data['T1'] - data['T2'])**  **Stress\_Longitudinal = data['S']\*data['F']\*data['WeldFactor']\*data['T']**  **Stress\_Moment = 0**  **Stress\_Axial = 0**  **Stress\_Hoop = (Stress\_Longitudinal - Stress\_Elongation - Stress\_Moment)/data['Poissionsratio']**  **DbytRatio = data['D']/data['t']**  **if DbytRatio >= 30:**  **ThicknessMinimum\_Longitudinal = (data['Pi']-data['Po'])\*data['D']/(2\*Stress\_Hoop)**  **else:**  **ThicknessMinimum\_Longitudinal = (data['Pi']-data['Po'])\*data['D']/(2\*Stress\_Hoop + (data['Pi']-data['Po']))**  **data.update({"MinimumWallThickness\_Longitudinal": ThicknessMinimum\_Longitudinal, \**  **"Stress\_Elongation": Stress\_Elongation, "Stress\_Logitudinal": Stress\_Longitudinal,**  **"Stress\_Hoop": Stress\_Hoop, "Stress\_Axial": Stress\_Axial,**  **"Stress\_Moment": Stress\_Moment})**  **return data** |  |
| **ASMEB318EquivalentStress** | **def ASMEB318EquivalentStress(data):**  **if data['Condition'] == "Restrained":**  **Stress\_Elongation = -data['E']\*data['Alpha']\*(data['T1'] - data['T2'])**  **Stress\_Equivalent = data['S']\*data['F']\*data['WeldFactor']\*data['T']**  **Stress\_Moment = 0**  **Stress\_Axial = 0**  **Stress\_Torsion = 0**  **Stress\_Hoop = ((-1\*math.sqrt(Stress\_Equivalent\*\*2/2 - Stress\_Torsion)/(data['Poissionsratio']-1)) \**  **- Stress\_Elongation - Stress\_Moment)/(1+data['Poissionsratio'])**  **DbytRatio = data['D']/data['t']**  **if DbytRatio >= 30:**  **ThicknessMinimum\_Equivalent = (data['Pi']-data['Po'])\*data['D']/(2\*Stress\_Hoop)**  **else:**  **ThicknessMinimum\_Equivalent = (data['Pi']-data['Po'])\*data['D']/(2\*Stress\_Hoop + (data['Pi']-data['Po']))**  **data.update({"MinimumWallThickness\_Equivalent": ThicknessMinimum\_Equivalent, \**  **"Stress\_Elongation": Stress\_Elongation,**  **"Stress\_Hoop": Stress\_Hoop, "Stress\_Axial": Stress\_Axial,**  **"Stress\_Moment": Stress\_Moment, "Stress\_Equivalent" : Stress\_Equivalent})**  **return data** |  |

### ASMEB31Sizing.py

|  |  |  |
| --- | --- | --- |
| **Inputs** | **Python Code** | **Description** |
|  | **# Data preparation**  **defaultYml = "dataManager\\ASMEB31\\B31.yml"**  **# Get updateYML file**  **try:**  **if (sys.argv[1] != None):**  **updateYml = "dataManager\\ASMEB31\\" + sys.argv[1]**  **logging.critical("Updating default values with contents in file {0}" .format(updateYml) )**  **except:**  **updateYml = None**  **logging.critical("No update values file is provided. Running program default values")**  **# Get updated configuration file for Analysis**  **cfg = ymlInput(defaultYml, updateYml)**  **try:**  **cfg['FileName'] = updateYml.split('\\')[1].split('.')[0]**  **except:**  **cfg['FileName'] = defaultYml.split('\\')[1].split('.')[0]**  **# Set logging**  **setLogging(cfg['Default']['logLevel'])**  **logging.info(cfg)**  **customdata = {"Code": cfg['Default']['Analysis']['Code']}**  **customdata = temperatureDerating(customdata)**  **cfg['Result'] = {"TemperatureDerating": customdata['TemperatureDerating']}**  **customdata = {"S": cfg['Material']['SMYS'],**  **"t" : cfg['Geometry']['DesignWT'],**  **"D" : cfg['Geometry']['NominalOD'],**  **"Pi": cfg['Design']['InternalPressure'],**  **"Po": cfg['Design']['ExternalPressure'],**  **"F" : cfg['DesignFactors']['Pressure'],**  **"WeldFactor" : cfg['Material']['WeldFactor']['Seamless'],**  **"T" : cfg['Result']['TemperatureDerating']**  **}**  **if cfg['Default']['Analysis']['Code'] == "ASME B31.4":**  **from custom.ASMEB31.ASMEMethods import ASMEB314InternalPressure**  **customdata = ASMEB314InternalPressure(customdata)**  **cfg['Result']['ASMEB314'] = {}**  **cfg['Result']['ASMEB314']['InternalPressure'] = customdata**  **if cfg['Default']['Analysis']['Code'] == "ASME B31.8":**  **from custom.ASMEB31.ASMEMethods import ASMEB318InternalPressure**  **customdata = ASMEB318InternalPressure(customdata)**  **cfg['Result']['ASMEB318'] = {}**  **cfg['Result']['ASMEB318']['InternalPressure'] = customdata**  **customdata = {"S": cfg['Material']['SMYS'],**  **"t" : cfg['Geometry']['DesignWT'],**  **"D" : cfg['Geometry']['NominalOD'],**  **"Pi": cfg['Design']['InternalPressure'],**  **"Po": cfg['Design']['ExternalPressure'],**  **"M": cfg['Design']['BendingMoment'],**  **"Fa": cfg['Design']['AxialForce'],**  **"F" : cfg['DesignFactors']['Longitudinal'],**  **"WeldFactor" : cfg['Material']['WeldFactor']['Seamless'],**  **"E" : cfg['Material']['E'],**  **"Poissionsratio" : cfg['Material']['Poissionsratio'],**  **"T" : cfg['Result']['TemperatureDerating'],**  **"Alpha" : cfg['Material']['ThermalExpansionCoefficient'],**  **"T1": cfg['Design']['Temperature']['Operating'],**  **"T2": cfg['Design']['Temperature']['Ambient'],**  **"Condition": cfg['Design']['Condition']**  **}**  **if cfg['Default']['Analysis']['Code'] == "ASME B31.4":**  **from custom.ASMEB31.ASMEMethods import ASMEB314LogitudinalStress**  **customdata = ASMEB314LogitudinalStress(customdata)**  **cfg['Result']['ASMEB314']['LogitudinalStress'] = customdata**  **if cfg['Default']['Analysis']['Code'] == "ASME B31.8":**  **from custom.ASMEB31.ASMEMethods import ASMEB318LogitudinalStress**  **customdata = ASMEB318LogitudinalStress(customdata)**  **cfg['Result']['ASMEB318']['LogitudinalStress'] = customdata**  **customdata = {"S": cfg['Material']['SMYS'],**  **"t" : cfg['Geometry']['DesignWT'],**  **"D" : cfg['Geometry']['NominalOD'],**  **"Pi": cfg['Design']['InternalPressure'],**  **"Po": cfg['Design']['ExternalPressure'],**  **"M": cfg['Design']['BendingMoment'],**  **"Fa": cfg['Design']['AxialForce'],**  **"F" : cfg['DesignFactors']['EquivalentStress'],**  **"WeldFactor" : cfg['Material']['WeldFactor']['Seamless'],**  **"E" : cfg['Material']['E'],**  **"Poissionsratio" : cfg['Material']['Poissionsratio'],**  **"T" : cfg['Result']['TemperatureDerating'],**  **"Alpha" : cfg['Material']['ThermalExpansionCoefficient'],**  **"T1": cfg['Design']['Temperature']['Operating'],**  **"T2": cfg['Design']['Temperature']['Ambient'],**  **"Condition": cfg['Design']['Condition']**  **}**  **if cfg['Default']['Analysis']['Code'] == "ASME B31.4":**  **from custom.ASMEB31.ASMEMethods import ASMEB314EquivalentStress**  **customdata = ASMEB314EquivalentStress(customdata)**  **cfg['Result']['ASMEB314']['EquivalentStress'] = customdata**  **if cfg['Default']['Analysis']['Code'] == "ASME B31.8":**  **from custom.ASMEB31.ASMEMethods import ASMEB318EquivalentStress**  **customdata = ASMEB318EquivalentStress(customdata)**  **cfg['Result']['ASMEB318']['EquivalentStress'] = customdata**  **saveDataYaml(cfg, 'results\\ASMEB31\\' + 'Summary\_' + cfg['FileName'])** |  |

# API 1111

Pipe selection for most offshore pipelines is determined by considering installation and operation loads in addition to the stresses resulting from internal pressure. Design should begin with material selection and pipe sizing for flow considerations and be modified later as a result of design cycles that include the following:

a) Burst due to net internal pressure;

b) Combined bending and tension during installation and operation;

c) Collapse due to external pressure, with the pipe either empty or filled;

d) Buckling and collapse due to combined bending and external pressure;

e) Pipeline stability against horizontal or vertical displacement during construction and operation;

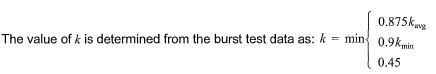
f) Effects of thermal expansion and contraction;

g) in-place and in-service pipeline repair capabilities;

### Specified Minimum Burst Pressure

The specified minimum burst pressure may be written in the form:

Where:



### MOP Limits

The MOP should not exceed any of the following:

a) the design pressure of any component, including pipe, valves, and fittings;

b) 80 % of the applied hydrostatic test pressure in accordance with 8.2.

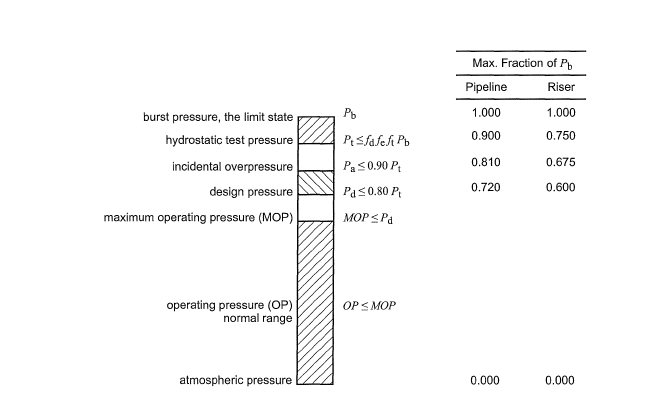


Figure 8‑1 Pressure Level Relations

### Internal Pressure (Burst) Design

The hydrostatic test pressure, the pipeline design pressure, and the incidental overpressure, including both internal and external pressures acting on the pipelines, shall not exceed that determined by the equations.

Where

fd is the internal pressure (burst) design factor, applicable to all pipelines;

0.90 for pipelines;

0.75 for pipeline risers.

fe is the weld joint factor, longitudinal or spiral seam welds. See ASME 831.4 or ASME 831.8. Only materials with a factor of 1.0 are acceptable.

ft is the temperature derating factor, as specified in ASME 831.8 [1.0 for temperatures less than 121°C (250 OF)].

Pa is the incidental over pressure (internal minus external pressure), in N/mm2 (psi).

Pb is the specified minimum burst pressure of pipe, in N/mm2 (psi).

Pd is the pipeline design pressure, in N/mm2 (psi).

Pt is the hydrostatic test pressure (internal minus external pressure), in N/mm2 (psi).

The specified minimum burst pressure (Pb) is determined by one of the following equations:

Where

D is the outside diameter of pipe, in mm (in.);

Di is D - 2t = inside diameter of pipe, in mm (in.);

S is the specified minimum yield strength (SMYS) of pipe, in N/mm2 (psi) (see API 5L, ASME 831.4, or ASME 831.8 as appropriate);

t is the nominal wall thickness of pipe, in mm (in.);

U is the specified minimum ultimate tensile strength of pipe, in N/mm2 (psi);

tn is the natural log.

### Longitudinal Load Design

The effective tension due to static primary longitudinal loads (see 4.6.2) shall not exceed the value given by.

Where

Ty=SA

A is the cross-sectional area of pipe steel, in mm2 (in. 2);

Ai is the internal cross-sectional area of the pipe, mm2 (in. 2);

Ao is the external cross-sectional area of the pipe, mm2 (in. 2);

Pi is the internal pressure in the pipe, in N/mm2 (psi);

Po is the external hydrostatic pressure, in N/mm2 (psi);

Ta is the axial (material) tension in pipe, in N (Ib);

Teff is the effective tension in pipe, in N (Ib);

Ty is the yield tension of the pipe, in N (Ib);

O"a is the axial stress in the pipe wall, in N/mm2 (psi).

The physical meaning of the term "effective" tension relates to the interaction between the pipe and other structures (sleds, anchor points, lay barge hangoff, etc.). The applied force at a boundary condition is always the effective tension. For the on-bottom portion of a pipeline, the effective tension will vary as loading conditions change. Typically, when a pipeline is just installed on the seabed, the effective tension is equal to the residual horizontal lay tension. For

a single pipeline under fully restrained condition (i.e. far from the end), the effective tension Teff is given by:

Where

Tlay is the residual lay tension;

in the case of a pipe-in-pipe (PIP) installed with the inner pipe free-standing inside the outer pipe during construction, the term 11ay will be negative (without mitigation) and additive to pressure and temperature loading.

∆Pi is the internal pressure change since laydown of the pipe;

E is Young's modulus of elasticity;

∝ is the thermal coefficient of expansion of the pipe;

∆T is the temperature change in the pipe since laydown;

v is the poisson ratio.

### Combined Load Design

The combination of primary longitudinal load (static and dynamic) and differential pressure load shall not exceed that given by:

52

# DNV-OS-F101

The DNV-OS-F101 subsea pipeline design code is implemented as python program to help check pipeline design. The code from the design guideline and program implemented are summarized in this section.

## Units

|  |  |
| --- | --- |
| Ovality | # |
| loadTemperature | Deg C |
| loadPressure | Psi |
| loadEffectiveTension | KN |
| loadBendingMoment | kN-m |
|  |  |

**units** = **{'ovality': '#', 'loadTemperature' : 'deg C', "loadPressure": "psi",   
 "loadEffectiveTension": "kN", "loadBendingMoment": "kNm"}**

## DNV-OS-F101.ini

DNV OS F101 input and calculation python code explains in below tables.

|  |  |  |  |
| --- | --- | --- | --- |
| **Python Code** | Code | Reference/ Notes | Units |
| [materialResistanceFactors]  SLS = 1.15  ULS = 1.15  ALS = 1.15  FLS = 1.00 |  | Sec 5  C 203 | # |
| [pressureContainment]  safetyClass = {"low": 1.046, "medium" : 1.138, "high" : 1.308}  [other]  safetyClass = {"low": 1.04, "medium" : 1.14, "high" : 1.26} |  | Sec 5  C 204 | # |
| [materialStrengthFactors]  Normally = 0.96  suplementaryRequirement = 1.00 |  | Sec 5  C306 | # |
| [materialFabrication]  factors = {"Seamless": 0.96, "UO" : 1.00, "TRB" : 0.93, "ERW" : 0.93, "UOE" : 0.85} |  | Sec 5  C307 | # |
| [pressureTestSafetyFactors]  mill = {"low": 1.0, "medium" : 1.088, "high" : 1.251}  system = {"low": 1.03, "medium" : 1.05, "high" : 1.05} |  | Sec 5  D201 | # |
| [loadEffectFactors] values = {"ULS": {"functional": 1.2, "environmental" : 0.7, "interference" : 0.0, "accidental" : 0.0},  "ULS\_Local" : {"functional": 1.1, "environmental" : 1.3, "interference" : 1.1, "accidental" : 0.0},  "FLS" : {"functional": 1.0, "environmental" : 1.0, "interference" : 1.0, "accidental" : 0.0},  "ALS" : {"functional": 1.0, "environmental" : 1.0, "interference" : 1.0, "accidental" : 1.0} } |  | Sec 4  G 303 | # |
| [conditionloadEffectFactors]  Pipelinerestingonunevenseabed = 1.07  reelingonandj-tubepullin = 0.82  systempressuretest = 0.93  otherwise = 1.0 |  | Sec 4  G 304 | # |
| [resistanceFactors]  safetyClass= 1.046  environmental = 1.1  interference = 1.0  accidental = 1.0  gammae = {"low": 2.0, "medium" = 2.5, "high" = 3.3} | ?? | Sec ?  Sec 5  D 609 | # |
| [load]  values = { "effectiveTension" : {"functional" : 0, "environmental" : 0, "interference" : 0, "accidental" : 0},  "bendingMoment" : {"functional" : -100.00, "environmental" : -639.5, "interference" : 0, "accidental" : 0},  "strain" : {"functional" : 0.01, "environmental" : 0.01, "interference" : 0.001, "accidental" : 0.001} } | Input loading | Sec ? |  |
| [loadPressure]  designInternal =3  fluidTemperature = 25  internal = 3  external=0.5  pbt1=1  plt=1.5  ph=1.1  pli=1.75  minimumInternal=0 | Input loading | Sec ? |  |
| [loadTemperature]  fluid = 25 | Input loading | Sec ? |  |

## Rigid Pipe

|  |  |  |
| --- | --- | --- |
|  |  | Reference |
| def DNVThickness(self):  self.thickness1["installation"] = self.nominalWallThickness - self.thicknessFabricationTolerance  self.thickness1["operation"] = self.nominalWallThickness - self.thicknessFabricationTolerance - self.corrosionAllowance  self.thickness2["installation"] = self.nominalWallThickness  self.thickness2["operation"] = self.nominalWallThickness - self.corrosionAllowance  self.thickness3["installation"] = self.nominalWallThickness  self.thickness3["operation"] = self.nominalWallThickness - 0.5 \* self.corrosionAllowance |  | Sec 5  C 401 |
| def updateThicknessFabricationTolerance(self, thicknessFabricationTolerance = None):  if self.thicknessFabricationTolerance is None:  self.thicknessFabricationTolerance = thicknessFabricationTolerance | Update the wall thickness fabriation tolerance with user specific values | NA |
| def updateCorrosionAllowance(self, CorrosionAllowance = None):  if self.corrosionAllowance is None:  self.corrosionAllowance = CorrosionAllowance | Update the Corrosion Allowance with user specific values | NA |
| def updateErosionAllowance(self, ErosionAllowance = None):  if self.erosionAllowance is None:  self.erosionAllowance = ErosionAllowance | Update the Erosion Allowance with user specific values | NA |
| def pressureContainmentResistance(self, thickness = None):  yield\_ContainmentOrBurst = min(self.Parameters.materialProperties.specifiedMinimumYieldStrength, self.Parameters.materialProperties.specifiedMinimumTensileStrength/1.15)  if thickness is None:  self.pressureContainmentResistance = 2 \* self.nominalWallThickness / (self.nominalDiameter - self.nominalWallThickness) \*yield\_ContainmentOrBurst\*2/math.sqrt(3)  else:  self.pressureContainmentResistance = 2 \* thickness / (self.nominalDiameter - thickness) \*yield\_ContainmentOrBurst\*2/math.sqrt(3) |  | Sec 5  D 202 |
| # ACTION : Need formulas for plt and ph  def pressureContainment(self, operatingCondition, safetyClass="high", limitState="ULS"):  thickness = self.thickness1[operatingCondition]  self.pressureContainmentResistance(thickness)  quantity1 = self.pressureContainmentResistance / self.Parameters.pressureTestSafetyFactors.system[safetyClass] / self.Parameters.materialResistanceFactors[limitState]  quantity2 = self.Parameters.loadPressure.plt / self.Parameters.pressureTestSafetyFactors.system[safetyClass] - self.Parameters.loadPressure.external  quantity3 = self.Parameters.loadPressure.ph \* self.Parameters.materialStrengthFactors.Normally / self.Parameters.pressureTestSafetyFactors.mill[safetyClass]  LHS = self.Parameters.loadPressure.pli - self.Parameters.loadPressure.external  RHS = min(quantity1, quantity2, quantity3)  self.incidentPressureburstcheckMargin = (RHS - LHS) / RHS  if LHS <= RHS:  self.incidentPressureburstCheckPass = True  else:  self.incidentPressureburstCheckPass = False  LHS = self.Parameters.loadPressure.plt - self.Parameters.loadPressure.external  RHS = min(quantity1, self.Parameters.loadPressure.ph)  self.incidentPressureburstcheckMargin = (RHS - LHS) / RHS  if LHS <= RHS:  self.testPressureburstCheckPass = True  else:  self.testPressureburstCheckPass = False |  | Sec 5  D 201 |
| # ACTION: Find reference for ovality per DNV standards. Default is 3%  def updateOvality(self, Dmax = None, Dmin = None):  if self.ovality is None:  if (Dmax == None) or (Dmin == None):  self.ovality = self.Parameters.defaults.ovality  else:  self.ovality = (Dmax - Dmin)/self.nominalDiameter  if self.ovality < 0.005:  self.ovality = 0.005 |  | Sec 5  D 402 |
| # Temperature Derating  def temperatureDeratingValues(self, fluidTemperature = None):  if fluidTemperature is None:  if (self.Parameters.loadPressure.fluidTemperature > self.Parameters.materialProperties.temperatureDerateLimit):  self.fy\_temp = 1.14 - self.Parameters.loadPressure.fluidTemperature / 850 # Temperature Derating Factor for the Yield Stress.  # Action: Add ultimate Stress drating  self.fu\_temp = self.fy\_temp  else:  self.fy\_temp = 0 # Temperature Derating Factor for the Yield Stress  self.fu\_temp = self.fy\_temp  else:  if fluidTemperature > self.Parameters.materialProperties.temperatureDerateLimit:  self.fy\_temp = 1.14 - fluidTemperature / 850 # Temperature Derating Factor for the Yield Stress  # Add ultimate Stress drating  self.fu\_temp = self.fy\_temp  else:  self.fy\_temp = 0 # Temperature Derating Factor for the Yield Stress  self.fu\_temp = self.fy\_temp |  | Sec  8.3.1 |
| # Material strength values and derated values  def updateStrengthValues(self,specifiedMinimumYieldStrength=None, specifiedMinimumTensileStrength=None):  if specifiedMinimumYieldStrength is not None:  self.Parameters.materialProperties.specifiedMinimumYieldStrength = specifiedMinimumYieldStrength  self.minimumYieldStrength = (self.Parameters.materialProperties.specifiedMinimumYieldStrength - self.fy\_temp)\*self.Parameters.materialStrengthFactors.Normally  if specifiedMinimumYieldStrength is not None:  self.Parameters.materialProperties.specifiedMinimumTensileStrength = specifiedMinimumTensileStrength  self.minimumTensileStrength = (self.Parameters.materialProperties.specifiedMinimumTensileStrength - self.fu\_temp)\*self.Parameters.materialStrengthFactors.Normally |  | Sec?  C306 |
| # DNV-OS-F101, Section 13 D700, Local Buckling - Collapse Check.  def collapse(self, thickness, pipeFabricationMethod):  self.dOverTRatio = self.nominalDiameter / thickness  self.tOverDRatio = 1/self.dOverTRatio  self.elasticCollapseCapacity = 2\*self.Parameters.materialProperties.youngsModulus\*self.tOverDRatio\*\*3/(1-self.Parameters.materialProperties.poissionsratio\*\*2)  self.plasticCollapseCapacity = self.minimumYieldStrength\*self.Parameters.materialFabrication.factors[pipeFabricationMethod]\*2\*self.tOverDRatio  b = -self.elasticCollapseCapacity  c = -(self.plasticCollapseCapacity\*\*2 + self.plasticCollapseCapacity\*self.elasticCollapseCapacity\*self.ovality\*self.dOverTRatio)  d = self.elasticCollapseCapacity\*(self.plasticCollapseCapacity\*\*2)  u = 1/3 \* (-b\*\*2/3 + c)  v = 1/2 \* (2/27\*(b\*\*3) - b\*c/3 + d)  phi= math.acos(-v/math.sqrt(-u\*\*3))  y = -2 \* math.sqrt(-u) \* math.cos(phi/3+ 60\*math.pi/180)  self.collapsePressure = y-b/3 |  | Sec 5  D 402 |
| def localBucklingUnderExternalPressure(self, thickness, pipeFabricationMethod, safetyClass="high", limitState="ULS"):  # Evalaute collapse pressure  self.collapse(thickness, pipeFabricationMethod)  LHS = self.Parameters.loadPressure.external - self.Parameters.loadPressure.minimumInternal  RHS = self.collapsePressure/self.Parameters.other.safetyClass[safetyClass]/self.Parameters.materialResistanceFactors[limitState]  self.localBucklingExternalPressureCheckMargin = (RHS - LHS) / RHS  if LHS <= RHS:  self.localBucklingExternalPressureCheckPass = True  else:  self.localBucklingExternalPressureCheckPass = False |  | Sec 5  D 401 |
| def localBucklingUnderExternalPressure\_polyMethod(self, thickness, pipeFabricationMethod):  self.collapse(thickness, pipeFabricationMethod)  coefficientX3 = 1  coefficientX2 = -self.elasticCollapseCapacity  coefficientX1 = -(self.plasticCollapseCapacity\*\*2 + self.elasticCollapseCapacity\*self.plasticCollapseCapacity\*self.ovality\*self.dOverTRatio)  coefficientX0 = self.elasticCollapseCapacity \* self.plasticCollapseCapacity\*\*2  polynomialCoefficients = [coefficientX3, coefficientX2, coefficientX1, coefficientX0]  potentialRoots = solvePolynomialEquation(polynomialCoefficients)  print(potentialRoots) | Not used. Another method to find roots for a polynomial equation |  |
| def propagationBuckling(self, thickness, pipeFabricationMethod, safetyClass="high", limitState="ULS", pipeCondition="Cold"):  # Update the wall thickness ratio for provided wall thickness  self.collapse(thickness, pipeFabricationMethod)  if pipeCondition == "Cold":  yieldStrength = self.Parameters.materialProperties.specifiedMinimumYieldStrength  else:  yieldStrength = self.minimumYieldStrength  if self.dOverTRatio < 15 or self.dOverTRatio < 45:  self.propagationPressure = 35\*yieldStrength \* self.Parameters.materialFabrication.factors[pipeFabricationMethod] \* (self.tOverDRatio\*\*2.5)  else:  self.propagationPressure = None  if self.propagationPressure is not None:  LHS = self.Parameters.loadPressure.external - self.Parameters.loadPressure.minimumInternal  RHS = self.propagationPressure / self.Parameters.other.safetyClass[safetyClass]/self.Parameters.materialResistanceFactors[limitState]  self.propagationBucklingCheckMargin = (RHS - LHS) / RHS  if LHS <= RHS:  self.propagationBucklingCheckPass = True  else:  self.propagationBucklingCheckPass = False  else:  self.propagationBucklingCheckMargin = None  self.propagationBucklingCheckPass = "Pipe out of code limits. Detailed FEA may be required to ascertain pipe bending capacity" |  | Sec 5  D 501 |
| # Action : Add buckle arrestor capacity design equations  TBA |  | Sec 5  D 502 |
| def localBucklingCombinedLoadingLoadControlled(self, thickness, pipeFabricationMethod, operatingCondition, safetyClass="high", limitState="ULS", pipeCondition="Cold", conditionEffect ="Other"):  # Update the wall thickness ratio for provided wall thickness  self.collapse(thickness, pipeFabricationMethod)  # self.pressureContainment(operatingCondition, safetyClass)  pInternal = self.Parameters.loadPressure.internal  pExternal = self.Parameters.loadPressure.external  if (pInternal >= pExternal):  logging.critical("Internal Overpressure")  overPressureRatio = (pInternal - pExternal)/self.pressureContainmentResistance  else:  logging.critical("External Overpressure")  overPressureRatio = (pExternal - self.Parameters.loadPressure.minimumInternal)/self.collapsePressure  if pipeCondition == "Cold":  yieldStrength = self.Parameters.materialProperties.specifiedMinimumYieldStrength  else:  yieldStrength = self.minimumYieldStrength  gammac = self.Parameters.conditionLoadEffectFactors.values[conditionEffect]  Msd1 = self.Parameters.load.values["bendingMoment"]["functional"] \* self.Parameters.loadEffectFactors.values[limitState]["functional"] \* gammac + \  self.Parameters.load.values["bendingMoment"]["environmental"] \* self.Parameters.loadEffectFactors.values[limitState]["environmental"] + \  self.Parameters.load.values["bendingMoment"]["interference"] \* self.Parameters.loadEffectFactors.values[limitState]["interference"] \* gammac + \  self.Parameters.load.values["bendingMoment"]["accidental"] \* self.Parameters.loadEffectFactors.values[limitState]["accidental"] \* gammac  Ssd1 = self.Parameters.load.values["effectiveTension"]["functional"] \* self.Parameters.loadEffectFactors.values[limitState]["functional"] \* gammac + \  self.Parameters.load.values["effectiveTension"]["environmental"] \* self.Parameters.loadEffectFactors.values[limitState]["environmental"] + \  self.Parameters.load.values["effectiveTension"]["interference"] \* self.Parameters.loadEffectFactors.values[limitState]["interference"] \* gammac + \  self.Parameters.load.values["effectiveTension"]["accidental"] \* self.Parameters.loadEffectFactors.values[limitState]["accidental"] \* gammac  Msd2 = self.Parameters.load.values["bendingMoment"]["functional"] / self.Parameters.loadEffectFactors.values[limitState]["functional"] \* gammac + \  self.Parameters.load.values["bendingMoment"]["environmental"] \* self.Parameters.loadEffectFactors.values[limitState]["environmental"] + \  self.Parameters.load.values["bendingMoment"]["interference"] \* self.Parameters.loadEffectFactors.values[limitState]["interference"] \* gammac + \  self.Parameters.load.values["bendingMoment"]["accidental"] \* self.Parameters.loadEffectFactors.values[limitState]["accidental"] \* gammac  Ssd2 = self.Parameters.load.values["effectiveTension"]["functional"] / self.Parameters.loadEffectFactors.values[limitState]["functional"] \* gammac + \  self.Parameters.load.values["effectiveTension"]["environmental"] \* self.Parameters.loadEffectFactors.values[limitState]["environmental"] + \  self.Parameters.load.values["effectiveTension"]["interference"] \* self.Parameters.loadEffectFactors.values[limitState]["interference"] \* gammac + \  self.Parameters.load.values["effectiveTension"]["accidental"] \* self.Parameters.loadEffectFactors.values[limitState]["accidental"] \* gammac  self.Msd = max(Msd1, Msd2)  self.Ssd = max(Ssd1, Ssd2)  plasticTensionCapacity = yieldStrength \* math.pi \* (self.nominalDiameter - thickness) \* thickness  plasticMomentCapacity = yieldStrength \* math.pi \* ((self.nominalDiameter - thickness)\*\*2) \* thickness  beta = (60 - self.dOverTRatio) / 90  alphaC = (1-beta) + beta\*(self.Parameters.materialProperties.tensileStrength)/yieldStrength  if overPressureRatio < 2/3:  alphaP = 1 - beta  else:  alphaP = 1 - 3\*beta\*(1-overPressureRatio)  LHS\_Term1a = self.Parameters.other.safetyClass[safetyClass]/self.Parameters.materialResistanceFactors[limitState]\*self.Msd/alphaC/plasticMomentCapacity  LHS\_Term1b = self.Parameters.other.safetyClass[safetyClass]/self.Parameters.materialResistanceFactors[limitState]\*self.Ssd/alphaC/plasticTensionCapacity  if (pInternal >= pExternal):  LHS\_Term2 = alphaP/alphaC \* overPressureRatio  else:  LHS\_Term2 = self.Parameters.other.safetyClass[safetyClass]/self.Parameters.materialResistanceFactors[limitState] \* overPressureRatio  interactionRatio = (LHS\_Term1a + LHS\_Term1b \*\* 2)\*\*2 + LHS\_Term2 \*\*2  RHS = 1  if (self.dOverTRatio <= 15) or (self.dOverTRatio <= 45):  logging.critical("CAUTION: D/t ratio outside of code limits. Adjust platic loads with FEA findings prior to use")  if (self.Ssd/plasticTensionCapacity >= 0.4):  logging.critical("CAUTION: Tension loading ratio outside of code limits. Refer to code")  self.localBucklingCombinedLoadingLoadControlledCheckMargin = (RHS - interactionRatio) / RHS  if interactionRatio <= RHS:  self.localBucklingCombinedLoadingLoadControlledCheckPass = True  else:  self.localBucklingCombinedLoadingLoadControlledCheckPass = False | **Internal Overpressure**      **External Overpressure** | Sec5  D 605  Sec 5  D 607 |
| Point Load considerations  TBA |  | Sec 5  D 606 |
| def localBucklingCombinedLoadingDisplacementControlled(self, thickness, pipeFabricationMethod, operatingCondition, safetyClass="high", limitState="ULS", pipeCondition="Cold", conditionEffect ="Other"):  # Update the wall thickness ratio for provided wall thickness  self.collapse(thickness, pipeFabricationMethod)  logging.critical("Update alphah in .ini file to re-run Displacement controlled method")  if (self.dOverTRatio <= 45):  logging.critical("CAUTION: D/t ratio outside of code limits. Refer to Code")  gammac = self.Parameters.conditionLoadEffectFactors.values[conditionEffect]  esd1 = self.Parameters.load.values["strain"]["functional"] \* self.Parameters.loadEffectFactors.values[limitState]["functional"] \* gammac + \  self.Parameters.load.values["strain"]["environmental"] \* self.Parameters.loadEffectFactors.values[limitState]["environmental"] + \  self.Parameters.load.values["strain"]["interference"] \* self.Parameters.loadEffectFactors.values[limitState]["interference"] \* gammac + \  self.Parameters.load.values["strain"]["accidental"] \* self.Parameters.loadEffectFactors.values[limitState]["accidental"] \* gammac  esd2 = self.Parameters.load.values["strain"]["functional"] / self.Parameters.loadEffectFactors.values[limitState]["functional"] \* gammac + \  self.Parameters.load.values["strain"]["environmental"] \* self.Parameters.loadEffectFactors.values[limitState]["environmental"] + \  self.Parameters.load.values["strain"]["interference"] \* self.Parameters.loadEffectFactors.values[limitState]["interference"] \* gammac + \  self.Parameters.load.values["strain"]["accidental"] \* self.Parameters.loadEffectFactors.values[limitState]["accidental"] \* gammac  esd = max(esd1, esd2)  pInternal = self.Parameters.loadPressure.internal  pExternal = self.Parameters.loadPressure.external  if (pInternal >= pExternal):  ec = 0.78 \* (self.tOverDRatio - 0.01) \* (1 + 5.75 \* (self.Parameters.loadPressure.minimumInternal - pExternal)/self.pressureContainmentResistance ) \* self.Parameters.defaults.alphah\*\*-1.5 \* self.Parameters.defaults.alphagw  logging.critical("Internal Overpressure")  self.localBucklingCombinedLoadingDisplacementControlledCheckMargin = (esd - ec) / ec  if esd <= ec:  self.localBucklingCombinedLoadingDisplacementControlledCheckPass = True  else:  self.localBucklingCombinedLoadingDisplacementControlledCheckPass = False  else:  ec = 0.78 \* (self.tOverDRatio - 0.01) \* alphah\*\*-1.5 \* alphagw  logging.critical("External Overpressure")  LHS = (esd/ec\*self.Parameters.resistanceFactors.gammae[safetyClass]) \*\* 0.8 + \  (pExternal - self.Parameters.loadPressure.minimumInternal)/self.collapsePressure\*self.Parameters.other.safetyClass[safetyClass]/self.Parameters.materialResistanceFactors[limitState]  RHS =1  self.localBucklingCombinedLoadingDisplacementControlledCheckMargin = (LHS - RHS) / RHS  if LHS <= RHS:  self.localBucklingCombinedLoadingDisplacementControlledCheckPass = True  else:  self.localBucklingCombinedLoadingDisplacementControlledCheckPass = False | **Internal Overpressure**      **External Overpressure** | Sec 5  D 608  Sec 5  D 609 |
| # Utilization Circles (TBA)  Combined Effective Bending moment vs. Combined Effective Tension charts. With varying pressures | * Find the limiting BM and Effective Tension for given pressure * Solve polynomial question to find roots and provide solution * Pressures can range from 0 to BurstCapcity | NA |
|  |  |  |

### Characteristic Material Properties

The characteristic material strength fy and fu, values to be used in the limit state criteria are:

Where:

fy,temp and fu,temp are the de-rating values due to the temperature of the yield stress and the tensile strength respectively.

αU is the material strength factor

The material properties shall be selected with due regard to material type and potential temperature and/ or ageing effects and shall include:

— yield stress

— Tensile strength

— Young's modulus

— Temperature expansion coefficient.

For C-Mn steel and 13Cr this shall be considered for temperatures above 50°C, and for 22Cr and 25Cr for temperatures above 20°C.

Guidance note 1:

Field joint coating application during installation may also impose temperatures in excess of the above and shall be considered.

Guidance note 2:

If no other information of de-rating effects on the yield stress exist the recommendations for C-Mn steel, 22Cr and 25Cr Figure 2 below may be used. For 13Cr testing is normally required.

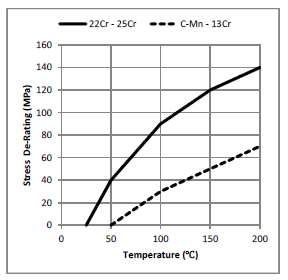


Figure 9‑1 Proposed de-rating values for yield stress of C-Mn, 13Cr, 22Cr and 25Cr

### Pressure containment (bursting)

The pressure containment shall fulfil the following criteria:

The pressure containment resistance pb(t) is given by:

Where:

Local Buckling – external over pressure only (system collapse): The external pressure at any point along the pipeline shall fulfil the following criterion (system collapse check):

Where

pmin is the minimum internal pressure that can be sustained. This is normally taken as zero for as-laid pipeline.

The characteristic resistance for external pressure (pc) (collapse) shall be calculated as:

Where:

The local pressure is the internal pressure at a specific point based on the reference pressure adjusted for the fluid column weight due to the difference in elevation. It can be expressed as:

where

pli is the local incidental pressure

pinc is the incidental reference pressure at the reference elevation

ρcont is the density of the relevant content of the pipeline

g is the gravity

href is the elevation of the reference point (positive upwards)

hl is the elevation of the local pressure point (positive upwards)

plt is the local system test pressure

pt is the system test reference pressure at the reference elevation

ρt is the density of the relevant test medium of the pipeline

pd is the design pressure at the pressure reference elevation

γinc is the incidental to design pressure ratio

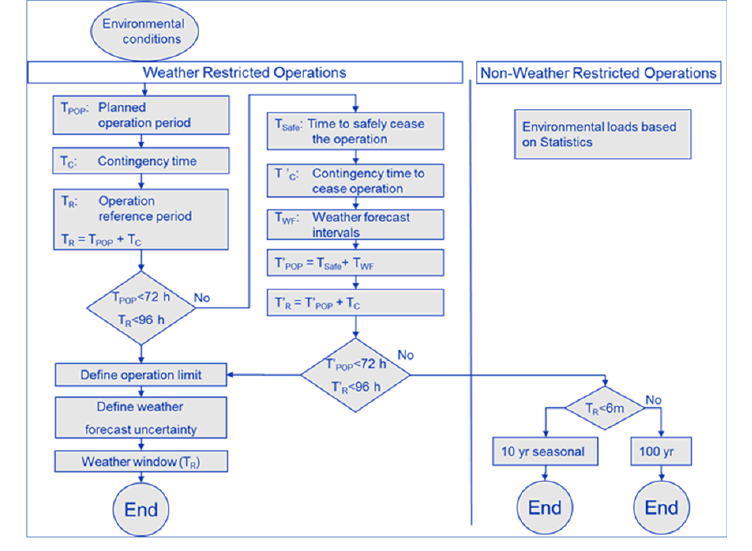
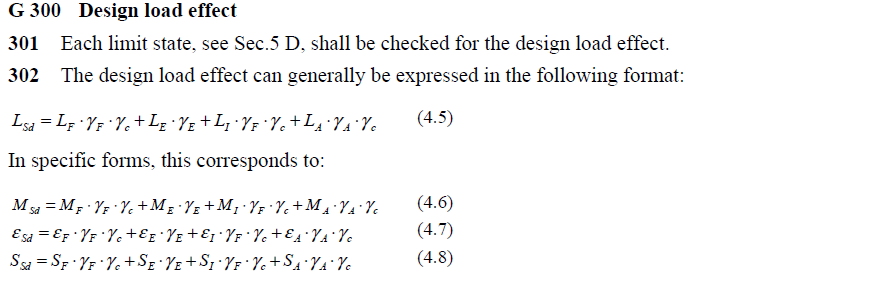
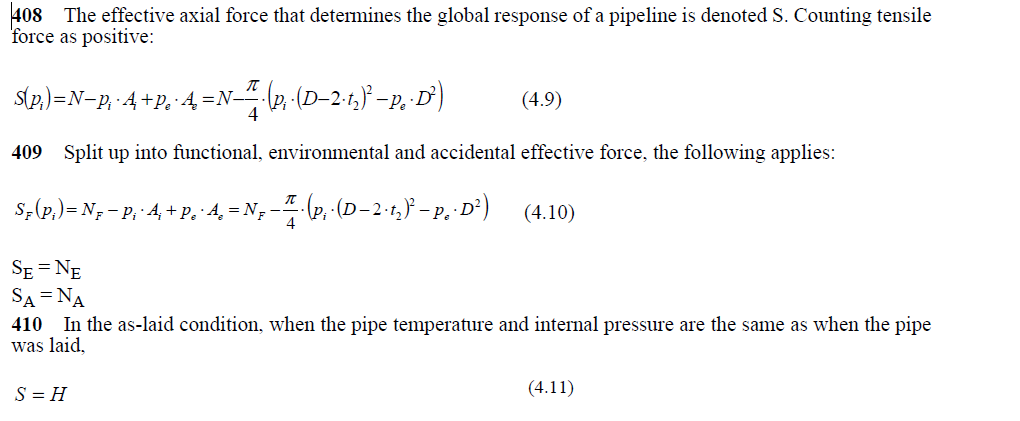


Figure 9‑2 Determination of characteristic environmental load



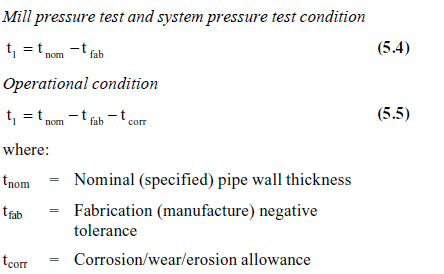


# DNV-OS-F201 (Dynamic risers)

### Geometrical Parameters

For burst and collapse pressure design checks the resistance shall be calculated based on

Wall thickness as follows.



### Design Load Effects

Design load effects are obtained by multiplying the load effect of each category by their corresponding load effect factor. Specific examples are given below for bending moment and effective tension.

Design bending moment for functional and environmental induced load effects:

Where:

MF = Bending moment from functional loads

ME = Bending moment from environmental loads

MA = Bending moment from accidental loads

Design effective tension for functional and environmental induced load effects:

Where

TeF = Effective tension from functional loads

TeE = Effective tension from environmental loads

TeA = Effective tension from accidental loads

The effective tension, Te is given by

Where

Tw = True wall tension (i.e. axial stress resultant found by integrating axial stress over the

cross-section)

pi = Internal (local) pressure

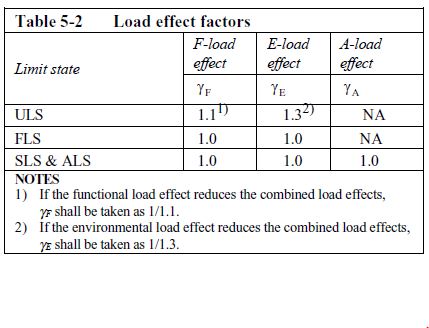
pe = External (local) pressure

Ai = Internal cross-sectional area

Ae = External cross-sectional area

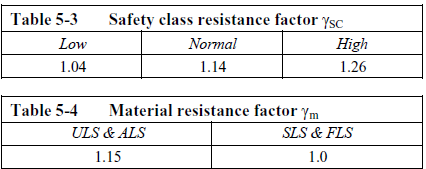
### Load Effect Factors

The design load effect is used in the design checks. Several combinations may have to be checked when load effects from several load categories enter one design check. The load effect factors shown in Table



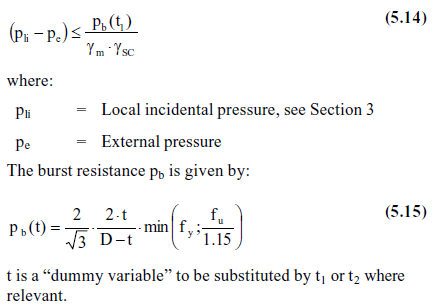
### Resistance Factors

* safety class factor γSC linked to the actual safety class and accounts for the failure consequence.
* material resistance factor γm to account for material and resistance uncertainties.
* A condition factor γc to account for special conditions specified explicitly at the different limit states where relevant.



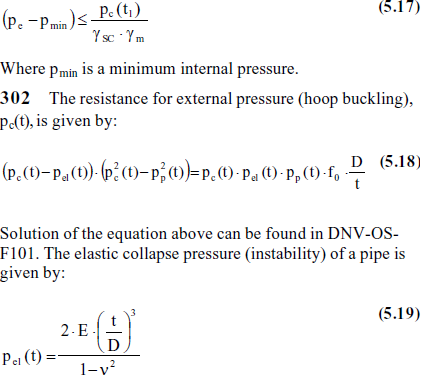
## Bursting

Pipe members subjected to net internal overpressure shall be designed to satisfy the following condition at all cross sections.



## System Hoop Buckling (Collapse)

Pipe members subjected to external overpressure shall be designed to satisfy the following condition.



# Input Data Specification

### Overview

The input data required for the program is given below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Units** | Internal Pressure | |
| **Geometry** |  |  |  |  |
| pipeNominalOD |  | m | 0.24765 | 0.24765 |
| pipeNominalWT |  | m | 0.034925 | 0.034925 |
| pipeMinimumWT |  | m | 0.034925 | 0.034925 |
| pipeNominalID |  |  | 0.1778 | 0.1778 |
|  |  |  |  |  |
| **Material Properties** |  |  |  |  |
| Yield Strength |  | ksi | 80 |  |
|  |  | Pa | 5.52E+08 |  |
|  |  |  |  |  |
| pipeA |  | m^2 | 0.023340213 |  |
| pipeAi | Ai | m^2 | 0.024828666 |  |
| pipeAo | Ao | m^2 | 0.04816888 |  |
| pipeI | I | m^4 | 0.000135582 |  |
|  |  |  |  |  |
| **Loading** |  |  |  |  |
| pipeExternalPressure | Po |  | 0 |  |
| pipeInternalPressure | Pi | ksi | 0 |  |
| pipeTensionEffective |  | N | 0 |  |
| pipeMoment |  | Nm | -4.60E+05 |  |
|  |  |  |  |  |
| PipeTrueTension |  |  | pipeTensionEffective + (pipeInternalPressure\*pipeAi) - (pipeExternalPressure\*pipeAo) |  |
| T |  |  | Pi\*Ai |  |

### Design Matrix for Rigid Risers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | Load Category | Environmental Condition | Pressure | Reduced ensioner  Capacity or One Mooring Line Broken | C a,b f |
| 1 | Operating | Maximum operating | Design | No | 1.0 |
| 2 | Extreme | Extreme | Design | No | 1.2 |
| 3 | Extreme | Maximum operating | Extreme | No | 1.2 |
| 4 | Extreme | Maximum operating | Design | Yes | 1.2 |
| 5 | Temporary | Temporary | Associated | No | 1.2 |
| 6 | Testd | Maximum operating | Testd | No | 1.35 |
| 7 | Survival | Survival | Associated | No | 1.5 |
| 8 | Survival | Extreme | Associated | Yes | 1.5 |
| 9 | Fatigue | Fatigue | Operating | No | Notec |

# References.

1. API RP 2RD 1st Ed with errata (2009) Design of Risers for FPSs and TLPs (June 2009)
2. Dropbox\0119 Programming\001 VMStress2RD\Ref
3. API STD 2RD 2nd Ed (2013) Dynamic Risers for Floating Production Systems
4. API RP 1111 4th Ed (2009) Design, Construction, Operation, and Maintenance of Offshore Hydrocarbon Pipelines (Limit State Design)
5. ASME B31.8 (2007) Gas Transportation and Distribution Piping Systems
6. DNV OS F201 (2001) Dynamic Risers
7. DNV OS F201 (2010) Dynamic Risers

– PROGRAM HISTORY

TBA

|  |  |  |
| --- | --- | --- |
| Revision Date | Features |  |
|  |  |  |

Table 4‑1.Program History

- ERROR LOG for Programs

Table 4‑2. Code Error.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Program** | **Env\_Bat** | **Run\_Instructions** | **(Default) OUTPUT/Error** | **(Custom) OUTPUT/Error** |
| FatigueBasiccurve.bat | FatigueBasiccurve.bat | 1)change directory = "F:\aceengineer”  Call python "F:\aceengineer\FatigueBasiccurve.py"  call python "F:\aceengineer\custom\Fatigue\_Curves\FatiguecurveRawdata.py"  call python "F:\aceengineer\custom\Fatigue\_Curves\LinearslopeCal.py"  call python "F:\aceengineer\custom\Fatigue\_Curves\ll2.py"  call python "F:\aceengineer\custom\Fatigue\_Curves\Shear7dataCal.py" | Working without any Error  C:\Users\AceEngineer-002\Desktop\1.JPG  Working without any Error  C:\Users\AceEngineer-002\Desktop\2.JPG  Working without any Error | No  No  No  No  No |
| Finance.bat | Finance.bat | 1)change directory = "F:\aceengineer”  2)Run the program = "F:\aceengineer\ Finance.bat” | C:\Users\AceEngineer-002\Desktop\11.JPG | no |
| OrcaFlexAnalysis.bat | OrcaFlexAnalysis.bat | 1)change directory = "F:\aceengineer”  2)Run the program = "F:\aceengineer\ OrcaFlexAnalysis.bat” | C:\Users\AceEngineer-002\Desktop\a.JPG | C:\Users\AceEngineer-002\Desktop\b.JPG |
| projectScheduleD01.bat | projectScheduleD01.bat | 1)change directory = "F:\aceengineer”  2)Run the program = "F:\aceengineer\ projectScheduleD01.bat” | C:\Users\AceEngineer-002\Desktop\sss.JPG | no |

Calculation Errors

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

-test cases.

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

Table 5.Test Cases.

validation