|  |  |
| --- | --- |
|  |  |
| Catenary Programme | |
| User Manual | |
| December 22, 2018 | |



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

Revision History

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

CONTENTS

[1 INTRODUCTION 3](#_Toc535102305)

[2 Catenary 3](#_Toc535102306)

[2.1 Catenary.yml 4](#_Toc535102307)

[2.2 Catenary Methods.py 8](#_Toc535102308)

[2.3 CatenaryRiser.py 16](#_Toc535102309)

[2.4 Pipe Properties 20](#_Toc535102310)

[2.5 CatenaryRiserSummary.py 23](#_Toc535102311)

[3 Post- Process 25](#_Toc535102312)

[3.1 Strength Analysis 25](#_Toc535102313)

[3.1.1 OrcaFlex.yml 25](#_Toc535102314)

[3.1.2 OrcaFlexAnalysis.py 26](#_Toc535102315)

[3.1.3 Post Process.py 28](#_Toc535102316)

[3.1.4 Post Process Plotting.py 28](#_Toc535102317)

[3.2 Modal Analysis 28](#_Toc535102318)

[3.2.1 Model Analysis.yml 28](#_Toc535102319)

[3.2.2 Model Analysis.py 29](#_Toc535102320)

[3.3 Fatigue Analysis 33](#_Toc535102321)

[3.3.1 OrcaFlex\_Fatigue.yml 33](#_Toc535102322)

[3.3.2 OrcaFlexAnalysis\_Fatigue.py 34](#_Toc535102323)

[3.4 RAO’s 37](#_Toc535102324)

[3.4.1 RAO.yml 37](#_Toc535102325)

[3.4.2 RAO’s Analysis .py 39](#_Toc535102326)

[3.4.3 RAO’s plotting 41](#_Toc535102327)

[4 Common Code 43](#_Toc535102328)

[4.1 Virtual Environment 43](#_Toc535102329)

[4.2 Program Handling 44](#_Toc535102330)

[4.3 Set Logging 44](#_Toc535102331)

[4.4 Yaml 45](#_Toc535102332)

[4.5 DataFrame to Image 46](#_Toc535102333)

[4.6 DataFrame to Xlsx 47](#_Toc535102334)

[4.7 Line Plot 49](#_Toc535102335)

[4.8 Counter Plot 49](#_Toc535102336)

[5 references 49](#_Toc535102337)

[Appendix A – PROGRAM HISTORY 49](#_Toc535102338)

[Appendix B - ERROR LOG 49](#_Toc535102339)

[B.1 Calculation Errors 49](#_Toc535102340)

[Appendix C -test cases. 49](#_Toc535102341)

# INTRODUCTION

TBA

# Summary

## Program Improvements

* Lazy wave equations should be improved so that each catenary should work very similar to how a simple catenary riser works.
* Lazy wave shape is determined based on using a particular internal fluid inside the riser. This internal fluid (eg. Production fluid) is defined in input using “LazyWaveCatenaryDefinition” and "configurationFluid"
  + The new lazy wave configuration for other internal fluids (seawater, empty, heavier production fluid etc.) should be determined by the program
  + A single plot showing all fluids will be very good plot.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | Name | Description | Scope | Status |
| 1 | Input customization with infile dictionary |  | All algorithms |  |
| 2 | \*\*kwargs |  | All algorithms |  |
| 3 | Wire up to common.config |  | catenary |  |
| 4 | Correct technical errors | Diameters,  Pipe types for taper jt and equivalent pipe  Axial drag formulas?  Is buoyancy diameter fixed? | Catenary |  |
| 5 | Pipe properties | Obtain from pipe.py | Catenary |  |
| 6 | Incorporate orcaflex 10.2 updates? |  | Catenary |  |
|  |  |  |  |  |

# Catenary

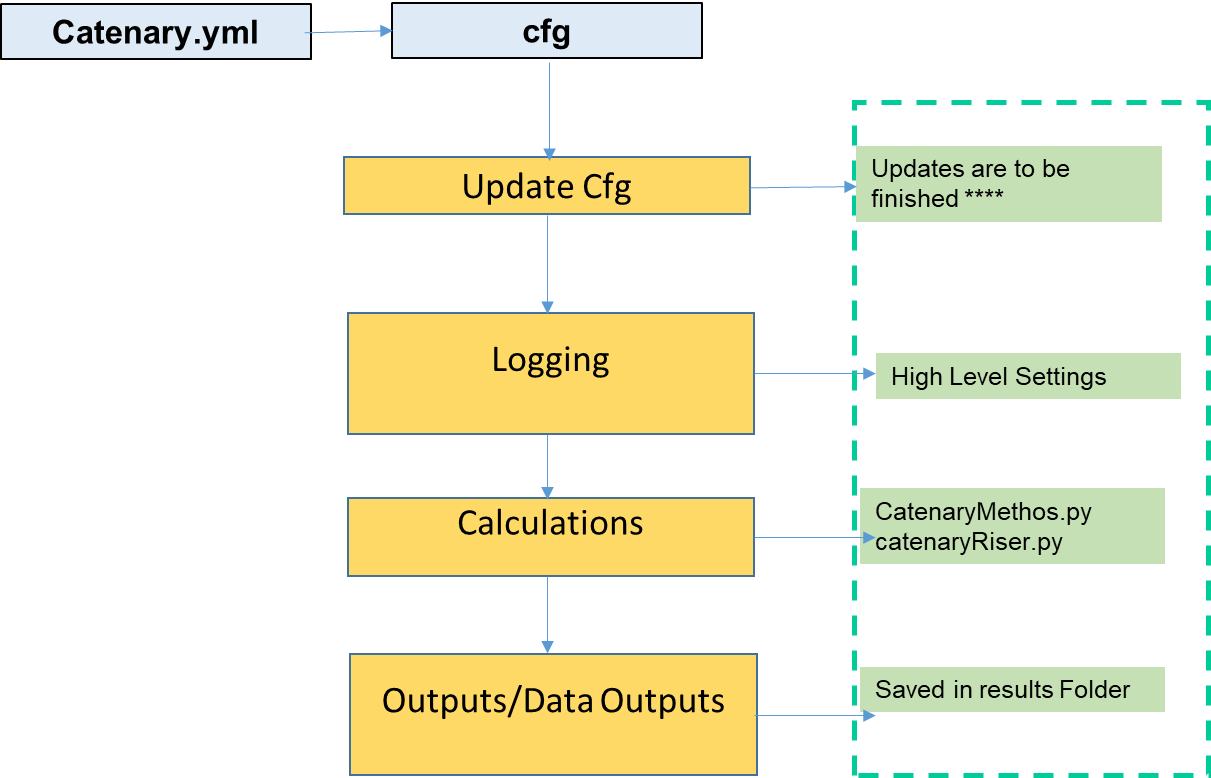


Figure 2‑1 Catenary Program Handling

## Catenary.yml

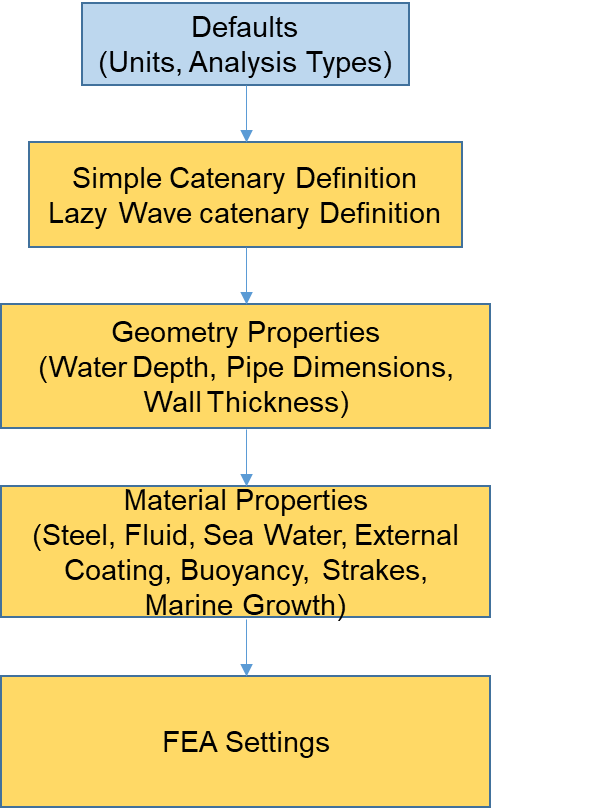


Figure 2‑2 Catenary.yml

|  |  |  |
| --- | --- | --- |
|  | Inputs | Description |
| Defaults | Analysis:  SCR : False  Jumper : False  SLWR : True  Extreme: True  Fatigue: False | We considering type of analysis for slwr  By giving the inputs statement True |
| Geometry | geometry:  waterDepth: 1347.2  NominalID: NULL  NominalOD: 8.625  DesignWT : 1.000  CorrosionAllowance: 0.236  ExternalCoating:  Thickness: 2.953  Strakes:  BaseThickness: 0.5906  Area: NULL  MarineGrowth:  Thickness: NULL | It is the given general geometry for pipe structure I.e. ID and OD and wall thickness  For pipe.in a particular depth towards the sea bed |
| Material Properties | Steel:  Rho: 7850  PoissionsRatio: 0.30  SMYS: 65300  SMUS: 77600  E: 30000000.00  G: 11538461.54  Fluid:  Rho: 400.4616  SeaWater:  Rho: 1025  ExternalCoating:  Rho: 800.9  Buoyancy:  Rho: 650  Strakes:  MassPerUnitLength: 21.5787  WeightPerUnitLength: -42.3375  Density: NULL  MarineGrowth:  Density: NULL | The type of material considered for the pipe  By taking the pre-defined values as inputs |
| Simple Catenary Definition | simpleCatenaryDefinition:  axialLineForce : NULL  verticalDistance : 1874.64  declinationAngle : 8 | Defining The Simple catenary Definition |
| Lazy Wave Catenary | LazyWaveCatenaryDefinition:  VerticalDistance : 1874.64  TDPToAnchor : 1500  HangoffBelowMeanSeaLevel : 0  HangoffAboveKeel : 11.21  SagBendElevationAboveSeabed : 731.52  HogBendAboveSeabed : 1097.28  declinationAngle : 8  UniformBuoyancy :  Thickness: NULL  BuoyancyFactor: 2  DiscreteBuoyancy:  Thickness: NULL  BuoyancyFactor: NULL  BuoyancyCoverage: 50  Spacing:  10  TaperJoint:  L: 9.144  ThickendThickness: 3.00 | Defining the lazy wave by giving the buoyancy parameters |

|  |  |  |
| --- | --- | --- |
| FEA Settings | FEASettings:  AnchorAdjustment: 54.36  Seabed:  FrictionCoefficient:  Normal: 0.6  Axial: 0.45  Stiffness:  Extreme: 10000  Fatigue: 1000  DesignPressure:  Surface: 750  TDP: NULL  Hydrodynamic:  Extreme:  MainPipe:  Ca: [1.5, '~', 0]  Cd: [1.5, '~', 0.008]  BuoyPipe:  Ca: [1, '~', 0.001]  Cd: [1, '~', 0.001]  BuoyPipeStartEnd:  Ca: [1, '~', 0.053570901]  Cd: [1, '~', 0.080356352]  Fatigue:  MainPipe:  Ca: [1.5, '~', 0]  Cd: [1.35, '~', 0.008]  BuoyPipe:  Ca: [1, '~', 0.001]  Cd: [0.9, '~', 0.001]  BuoyPipeStartEnd:  Ca: [1, '~', 0.053570901]  Cd: [1, '~', 0.080356352]  CorrosionAllowance:  Extreme: 0.236  Fatigue: 0.118  Damping:  Extreme:  Name: SteelDamping  Mode: Stiffness Proportional  DampingRatio: 0.5  Period1: '~'  ApplyToGeometricStiffness: No  Fatigue:  Name: SteelDamping  Mode: Stiffness Proportional  DampingRatio: 0.5  Period1: '~'  ApplyToGeometricStiffness: No  Mesh:  Top:  L: 25  Size: 0.5  Buoyancy:  L: NULL  Size: 2  BeforeTDP:  L: 250  Size: 1  AfterTDP:  L: 250  Size: 1  Stage:  Extreme:  Duration:  - 8  - 120  TargetLogSampleInterval: 0.3  ImplicitConstantTimeStep: 0.02  Fatigue:  Duration:  - 8  - 600  TargetLogSampleInterval: 0.3  ImplicitConstantTimeStep: 0.2  FlexJointStiffness: ConnectionLabel: FJProfile | Defining The FEA Settings |

## Catenary Methods.py

|  |  |  |
| --- | --- | --- |
| Inputs | Python Code | Description |
| Catenary Equation | def catenaryEquation(data):  if(data["F"] != None):  S = data["d"]\*(2\*data["F"]/data["w"]-data["d"])  #Horizontal Distance.  X=(((data["F"]/data["w"])-data["d"])\*math.log((S+(data["F"]/data["w"]))/((data["F"]/data["w"])-data["d"])))  #weight of the suspended chain  W =(data["w"]\*S)  #normalized horizontal tension component  THorizontal= (data["F"]\*X/math.sqrt(S\*\*2+X\*\*2) #catenary shape parameter  b=(data["w"]\*9.81/THorizontal)  data.update({"S": S, "X": X, "W": W, "THorizontal": THorizontal} return data  elif(data["q"] != None):  tanq = math.tan(math.radians(90 - data["q"]))  BendRadius = data["d"]\*(math.cos(math.radians(90 - data["q"])))/(1-math.cos(math.radians(90 - data["q"])))  S = BendRadius \* tanq  X = BendRadius \* math.asinh(tanq)  data.update({"S": S, "X": X, "BendRadius": BendRadius})  return data | Defining the catenary equation |
| Catenary Force | def catenaryForces(data):  # Vertical load on vessel  Fv = data['weightPerUnitLength']\*data['S']  # Total force along catenary  F = Fv/(math.sin(math.radians(90-data['q'])))  # Horizontal force along catenary  Fh = F\*math.cos(math.radians(90-data['q']))  data.update({"Fv": Fv, "F": F, "Fh": Fh})  return data |  |
| Sag Elevation | def sagHogEquation(data):  # Sag to Buoyancy Configuration  BendRadius = data['HangOff']["BendRadius"]  d = (data['HogBendAboveSeabed']-data['SagBendElevationAboveSeabed'])\*abs(data['WeightPerUnitLengthWithBuoyancy'])/(abs(data['WeightPerUnitLengthWithBuoyancy']) + abs(data['WeightPerUnitLengthWithOutBuoyancy']))  X = BendRadius\*math.acosh(d/BendRadius + 1)  S = BendRadius\*math.sinh(X/BendRadius)  data['SagToBuoyancy'] = {"d": d, "S": S, "X": X, "BendRadius" : BendRadius}  # Buoy to Hog Configuration  BendRadius = BendRadius\*data['WeightPerUnitLengthWithOutBuoyancy']/(abs(data['WeightPerUnitLengthWithBuoyancy']))  d = (data['HogBendAboveSeabed']-data['SagBendElevationAboveSeabed'])\*abs(data['WeightPerUnitLengthWithOutBuoyancy'])/(abs(data['WeightPerUnitLengthWithBuoyancy']) + abs(data['WeightPerUnitLengthWithOutBuoyancy']))  X = BendRadius\*math.acosh(d/BendRadius + 1)  S = BendRadius\*math.sinh(X/BendRadius)  data['BuoyancyToHog'] = {"d": d, "S": S, "X": X, "BendRadius" : BendRadius}  # Hog to Buoyancy Configuration  d = data['HogBendAboveSeabed']\*abs(data['WeightPerUnitLengthWithOutBuoyancy'])/(abs(data['WeightPerUnitLengthWithBuoyancy']) + abs(data['WeightPerUnitLengthWithOutBuoyancy']))  X = BendRadius\*math.acosh(d/BendRadius + 1)  S = BendRadius\*math.sinh(X/BendRadius)  data['HogToBuoyancy'] = {"d": d, "S": S, "X": X, "BendRadius" : BendRadius}  # Buoyancy to TouchDown Configuration  BendRadius = data['HangOff']["BendRadius"]  d = data['HogBendAboveSeabed']\*abs(data['WeightPerUnitLengthWithBuoyancy'])/(abs(data['WeightPerUnitLengthWithBuoyancy']) + abs(data['WeightPerUnitLengthWithOutBuoyancy']))  X = BendRadius\*math.acosh(d/BendRadius + 1)  S = BendRadius\*math.sinh(X/BendRadius)  data['BuoyancyToTouchDown'] = {"d": d, "S": S, "X": X, "BendRadius" : BendRadius}  return data |  |
| Lazy wave Catenary Equation | def lazyWaveCatenaryEquation(data):  # Hang-off to Sag Section  catenaryResult = catenaryEquation(data['HangOff'])  data['HangOff'].update(catenaryResult)  data = sagHogEquation(data)  # Summarize the lazy wave catenary  Fh = data['HangOff']["BendRadius"]\*data['WeightPerUnitLengthWithOutBuoyancy']  Fv = Fh + data['WeightPerUnitLengthWithOutBuoyancy']\*data['HangOff']["S"]  HangOffToBuoyancy = {"S" : data['HangOff']["S"] + data['SagToBuoyancy']["S"],  "X" : data['HangOff']["X"] + data['SagToBuoyancy']["X"]}  Buoyancy = {"S" : data['BuoyancyToHog']["S"] + data['HogToBuoyancy']["S"],  "X" : data['BuoyancyToHog']["X"] + data['HogToBuoyancy']["X"]}  BuoyancyToTouchDown = {"S" : data['BuoyancyToTouchDown']["S"], "X" : data['BuoyancyToTouchDown']["X"]}  HangoffToTDP = {"S" : HangOffToBuoyancy["S"] + Buoyancy["S"] + BuoyancyToTouchDown["S"],  "X": HangOffToBuoyancy["X"] + Buoyancy["X"] + BuoyancyToTouchDown["X"]}  data['Summary'] = { "HangOffToBuoyancy": HangOffToBuoyancy, "Buoyancy": Buoyancy, "BuoyancyToTouchDown": BuoyancyToTouchDown, "HangoffToTDP": HangoffToTDP, "Fh": Fh, "Fv": Fv}  # data['Summary'] = { "HangOffToBuoyancy": HangOffToBuoyancy}  return data |  |
| Lazy Wave Plot | def lazyWavePlot(data, spacing = 10):  # HangoffToBuoyancy Section  X=[]  Y=[]  BendRadius = data['lazyWaveCatenaryResult']['HangOff']['BendRadius']  HorizontalDistance = data['lazyWaveCatenaryResult']['HangOff']['X']  XRange = data['lazyWaveCatenaryResult']['Summary']["HangOffToBuoyancy"]['X']  XCoordinateAdjustment = 0  YCoordinateAdjustment = (BendRadius + data['lazyWaveCatenaryResult']['HangOff']['d'] \  + data['LazyWaveCatenaryDefinition']['HangoffBelowMeanSeaLevel'])  for i in range(0, spacing+1, 1):  X.append(XCoordinateAdjustment + i\*XRange/spacing)  Y.append(BendRadius\*(math.cosh((X[i] - HorizontalDistance)/BendRadius)) \  - YCoordinateAdjustment)  data['lazyWaveCatenaryResult']['Summary']["HangOffToBuoyancy"]['PlotData'] = {"X": X, "Y": Y}  PlotData = {"X": X, "Y": Y, "label": "Hang-off to Buoyancy", "SaveAndShow": False} genericPlot(PlotData) |  |
| Buoyancy Section | # Buoyancy Section  X=[]  Y=[]  BendRadius = data['lazyWaveCatenaryResult']['BuoyancyToHog']['BendRadius']  HorizontalDistance = data['lazyWaveCatenaryResult']['HangOff']['X'] \  +data['lazyWaveCatenaryResult']["SagToBuoyancy"]['X'] \  + data['lazyWaveCatenaryResult']["BuoyancyToHog"]['X']  XCoordinateAdjustment = data['lazyWaveCatenaryResult']['HangOff']['X'] \  + data['lazyWaveCatenaryResult']["SagToBuoyancy"]['X']  XRange = data['lazyWaveCatenaryResult']['Summary']["Buoyancy"]['X']  YCoordinateAdjustment = -BendRadius + data['lazyWaveCatenaryResult']['HangOff']['d'] \  - (data['LazyWaveCatenaryDefinition']['HogBendAboveSeabed'] \  - data['LazyWaveCatenaryDefinition']['SagBendElevationAboveSeabed']) \  + data['LazyWaveCatenaryDefinition']['HangoffBelowMeanSeaLevel']  for i in range(0, spacing+1, 1):  X.append(XCoordinateAdjustment + XRange\*i/spacing)  Y.append(-BendRadius\*(math.cosh((-X[i] + HorizontalDistance)/BendRadius)) \  - YCoordinateAdjustment)  data['lazyWaveCatenaryResult']['Summary']["Buoyancy"]['PlotData'] = {"X": X, "Y": Y}  PlotData = {"X": X, "Y": Y}  PlotData = {"X": X, "Y": Y, "label": "Buoyancy", "SaveAndShow": False}  genericPlot(PlotData) |  |
| Buoyancy To Touch Down | # BuoyancyToTouchDown Section  X=[]  Y=[]  BendRadius = data['lazyWaveCatenaryResult']["BuoyancyToTouchDown"]['BendRadius']  HorizontalDistance = data['lazyWaveCatenaryResult']['Summary']['HangoffToTDP']['X']  XCoordinateAdjustment = data['lazyWaveCatenaryResult']['Summary']['HangoffToTDP']['X'] \  - data['lazyWaveCatenaryResult']["BuoyancyToTouchDown"]['X']  XRange = data['lazyWaveCatenaryResult']['BuoyancyToTouchDown']['X']  YCoordinateAdjustment = (BendRadius + data['LazyWaveCatenaryDefinition']['HogBendAboveSeabed'] \  + data['LazyWaveCatenaryDefinition']['HangoffBelowMeanSeaLevel'])  for i in range(0, spacing+1, 1):  X.append( XCoordinateAdjustment + XRange\*i/spacing)  Y.append(BendRadius\*math.cosh((X[i] - HorizontalDistance)/BendRadius) \  - BendRadius - data['LazyWaveCatenaryDefinition']['VerticalDistance'] \  - data['LazyWaveCatenaryDefinition']['HangoffBelowMeanSeaLevel'])  data['lazyWaveCatenaryResult']['Summary']["BuoyancyToTouchDown"]['PlotData'] = {"X": X, "Y": Y}  PlotData = {"X": X, "Y": Y, "label": "Buoyancy to TDP", "SaveAndShow": True, "fileName": "results\\" + "SLWR\_" + data['FileName']} genericPlot(PlotData)  return data |  |
| Generic Plot | def genericPlot(data):  if data["label"] == None:  plt.plot(data["X"], data["Y"])  else:  plt.plot(data["X"], data["Y"], label=data["label"])  #it creates plot X-axis name, fontsize and colour  plt.xlabel('Horizontal distance[m]', fontsize=12, fontweight='bold', color='black')  #it creates plot Y-axis name, fontsize and colour  plt.ylabel('Distance from seabed [m]', fontsize=12, fontweight='bold', color='black')  #it creates plot tittle, color and size  plt.title('Catenary Mooring Line Shape',fontsize=14, fontweight='bold', color='black')  # Gridlines  plt.grid()  if(data['SaveAndShow'] == True):  plt.savefig(data["fileName"].replace(".", ""), dpi=800)  plt.legend(loc='upper center', bbox\_to\_anchor=(0.5, -0.05),  fancybox=True, shadow=True, ncol=5) |  |
| Buoyancy properties | def buoyancyProperties(cfg):  if cfg["LazyWaveCatenaryDefinition"]["UniformBuoyancy"]["Thickness"] != None:  cfg = pipeProperties(cfg, FluidDensity=cfg['Material']['Fluid']['Rho'], Buoyancy = True)  WeightPerUnitLengthWithBuoyancy = cfg['equivalentPipe']['weightPerUnitLength']  cfg["LazyWaveCatenaryDefinition"]["UniformBuoyancy"]["BuoyancyFactor"] = 1 - (WeightPerUnitLengthWithBuoyancy/WeightPerUnitLengthWithOutBuoyancy)  # Add buoyancy Ratio and also buoyancy module detailed properties  elif cfg["LazyWaveCatenaryDefinition"]["UniformBoyancy"]["BuoyancyFactor"] != None: BuoyancyOD = math.sqrt(2\*cfg['catenaryResult']['FluidFilled']['weightPerUnitLength']/9.81\*4/math.pi/(cfg['Material']['SeaWater']['Rho'] \-cfg['Material']['Buoyancy']['Rho']) + (cfg['InsulationSection']['OD']\*0.0254)\*\*2)/0.0254 cfg["LazyWaveCatenaryDefinition"]["UniformBuoyancy"]["Thickness"] = (BuoyancyOD - cfg['InsulationSection']['OD'])/2  WeightPerUnitLengthWithBuoyancy = (1 - cfg["LazyWaveCatenaryDefinition"]["UniformBuoyancy"]["BuoyancyFactor"]) \  \* cfg['catenaryResult']['FluidFilled']['weightPerUnitLength']  DiscreteBuoyancyOD = math.sqrt(100/cfg['LazyWaveCatenaryDefinition']['DiscreteBuoyancy']['BuoyancyCoverage'] \* BuoyancyOD\*\*2 - cfg['InsulationSection']['OD']\*\*2)  cfg['LazyWaveCatenaryDefinition']['DiscreteBuoyancy']['Thickness'] = DiscreteBuoyancyOD  cfg['lazyWaveCatenaryResult'] = {}  cfg['lazyWaveCatenaryResult']['WeightPerUnitLengthWithOutBuoyancy'] = cfg['catenaryResult']['FluidFilled']['weightPerUnitLength']  cfg['lazyWaveCatenaryResult']['WeightPerUnitLengthWithBuoyancy'] = WeightPerUnitLengthWithBuoyancy  return cfg |  |

## CatenaryRiser.py

|  |  |  |
| --- | --- | --- |
| Inputs | Python Code | Description |
| Build Model | def BuildModel(FEAType):  if cfg['default']['Analysis']['SLWR']:  for LoadingIndex in range (0, len(cfg['EnvironmentLoad'][FEAType])):  cfg['fileName'] = 'results\\FEA\\' + FEAType + '\\' + 'SLWR\_' + cfg['FileName'] + '\_' + cfg['EnvironmentLoad'][FEAType][LoadingIndex]['Wave']['WaveTrains'][0]['Name'] + '.yml'  if cfg['default']['Analysis'][FEAType] == True:  FEAmodel1, FEAmodel2 = orcaflexModel(cfg, FEAType, LoadingIndex)  if FEAType == 'Extreme':  WriteOrcaflexModel([FEAmodel1, 'dataManager\\VesselTypes\_Extreme.yml', FEAmodel2], cfg, FEAType, LoadingIndex) elif FEAType == 'Fatigue':  WriteOrcaflexModel([FEAmodel1, 'dataManager\\VesselTypes\_Fatigue.yml', FEAmodel2], cfg, FEAType, LoadingIndex)  saveDataYaml(cfg, 'results\\' + 'SLWR\_' + 'Summary\_' + cfg['FileName'], False)  if cfg['default']['Analysis']['SCR']:  for LoadingIndex in range (0, len(cfg['EnvironmentLoad'][FEAType])):  cfg['fileName'] = 'results\\FEA\\' + FEAType + '\\' + 'SCR\_' + cfg['FileName'] + '\_' + cfg['EnvironmentLoad'][FEAType][LoadingIndex]['Wave']['WaveTrains'][0]['Name'] + '.yml'  if cfg['default']['Analysis'][FEAType] == True:  FEAmodel1, FEAmodel2 = orcaflexModel(cfg, FEAType, LoadingIndex)  if FEAType == 'Extreme':  WriteOrcaflexModel([FEAmodel1, 'dataManager\\VesselTypes\_Extreme.yml', FEAmodel2], cfg, FEAType, LoadingIndex)  elif FEAType == 'Fatigue':  WriteOrcaflexModel([FEAmodel1, 'dataManager\\VesselTypes\_Fatigue.yml', FEAmodel2], cfg, FEAType, LoadingIndex) saveDataYaml(cfg, 'results\\' + 'SCR\_' + 'Summary\_' + cfg['FileName']) | To build the extreme and fatigue analysis |
| Data Preparation | # Data preparation  defaultYml = "dataManager\\catenary.yml"  # Get updateYML file  try:  if (sys.argv[1] != None):  updateYml = "dataManager\\" + 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) | Preparing the data preparation from the input yml file and updating the inputs from yml file. |
| Set Logging | # Set logging  setLogging(cfg['default']['logLevel'])  # Evaluate section properties and mass per unit length and related properties.  inputData = {"d": cfg["simpleCatenaryDefinition"]["verticalDistance"],  "F": cfg["simpleCatenaryDefinition"]["axialLineForce"],  "q" : cfg["simpleCatenaryDefinition"]["declinationAngle"]  }  catenaryResult = catenaryEquation(inputData)  cfg['catenaryResult'] = catenaryResult  cfg = pipeProperties(cfg, FluidDensity=cfg['Material']['Fluid']['Rho'], Buoyancy = False)  WeightPerUnitLengthWithOutBuoyancy = cfg['equivalentPipe']['weightPerUnitLength']  inputData = {'weightPerUnitLength' : cfg['equivalentPipe']['weightPerUnitLength'],  'S' : cfg['catenaryResult']['S'],  'q' : cfg['catenaryResult']['q']  }  output = catenaryForces(inputData)  cfg['catenaryResult']['FluidFilled'] = output  cfg = pipeProperties(cfg, FluidDensity=cfg['Material']['SeaWater']['Rho'], Buoyancy = False)  inputData = {'weightPerUnitLength' : cfg['equivalentPipe']['weightPerUnitLength'],  'S' : cfg['catenaryResult']['S'],  'q' : cfg['catenaryResult']['q']  }  output = catenaryForces(inputData)  cfg['catenaryResult']['SeaWaterFilled'] = output  cfg = pipeProperties(cfg, FluidDensity=0, Buoyancy = False)  inputData = {'weightPerUnitLength' : cfg['equivalentPipe']['weightPerUnitLength'],  'S' : cfg['catenaryResult']['S'],  'q' : cfg['catenaryResult']['q']  }  output = catenaryForces(inputData)  cfg['catenaryResult']['Empty'] = output |  |
| Lazy Wave Analysis | # Lazy Wave Analysis  cfg = buoyancyProperties(cfg)  lazyWaveInputs = {"WeightPerUnitLengthWithBuoyancy": cfg['lazyWaveCatenaryResult']['WeightPerUnitLengthWithBuoyancy'],  "WeightPerUnitLengthWithOutBuoyancy": cfg['lazyWaveCatenaryResult']['WeightPerUnitLengthWithOutBuoyancy'],  "SagBendElevationAboveSeabed": cfg['LazyWaveCatenaryDefinition']['SagBendElevationAboveSeabed'],  "HogBendAboveSeabed": cfg['LazyWaveCatenaryDefinition']['HogBendAboveSeabed'],  "HangOff": {"d": cfg['LazyWaveCatenaryDefinition']['VerticalDistance'] - cfg['LazyWaveCatenaryDefinition']['SagBendElevationAboveSeabed'],  "q": cfg["LazyWaveCatenaryDefinition"]["declinationAngle"], "F": None },  }  cfg['lazyWaveCatenaryResult'].update(lazyWaveCatenaryEquation(lazyWaveInputs))  cfg['lazyWaveCatenaryResult']['TotalBuoyancy'] = (cfg['lazyWaveCatenaryResult']['WeightPerUnitLengthWithBuoyancy'] - cfg['lazyWaveCatenaryResult']['WeightPerUnitLengthWithOutBuoyancy'])\*cfg['lazyWaveCatenaryResult']['Summary']['Buoyancy']['S']  cfg = lazyWavePlot(cfg, cfg['LazyWaveCatenaryDefinition']['Spacing'])  # Construct FEA Model  cfg = pipeProperties(cfg, FluidDensity=0, Buoyancy = False)  cfg['MainPipe'] = {'SteelSection': cfg['SteelSection'], 'BuoyancySection': cfg['BuoyancySection'],  'InsulationSection': cfg['InsulationSection'], 'equivalentPipe': cfg['equivalentPipe']}  cfg = pipeProperties(cfg, FluidDensity=0, Buoyancy = True)  cfg['BuoyPipe'] = {'SteelSection': cfg['SteelSection'], 'BuoyancySection': cfg['BuoyancySection'],  'InsulationSection': cfg['InsulationSection'], 'equivalentPipe': cfg['equivalentPipe']}  if cfg['default']['Analysis']['Extreme']:  FEAType = 'Extreme'  BuildModel(FEAType)  if cfg['default']['Analysis']['Fatigue']:  FEAType = 'Fatigue'  BuildModel(FEAType) | Defining the lazy wave analysis |

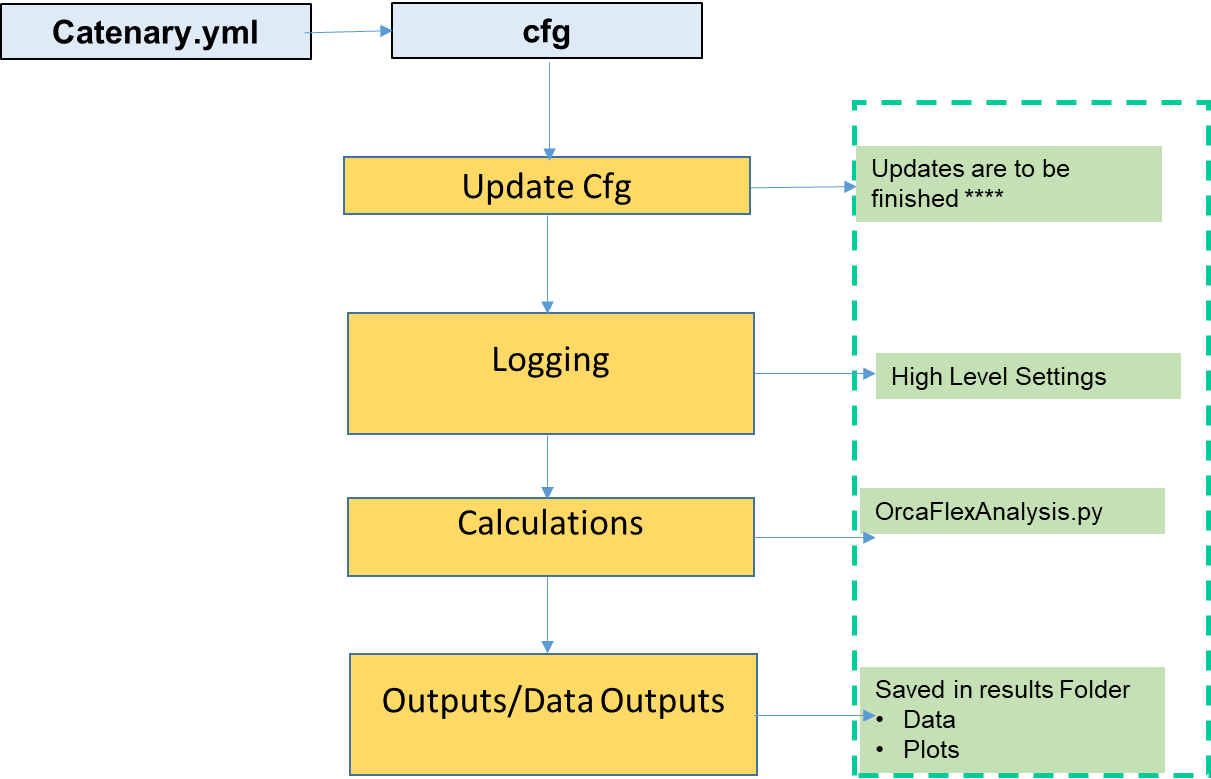
## Pipe Properties

|  |  |  |
| --- | --- | --- |
| Inputs | Python Code | Description |
| Pipe properties For Fluid Density | def pipeProperties(cfg, FluidDensity, Buoyancy = False):  cfg = steelSectionProperties(cfg)  cfg = insulationSectionProperties(cfg)  if Buoyancy == True:  cfg = buoyancySectionProperties(cfg)  else: cfg['BuoyancySection'] = None  cfg = equivalentPipe(cfg, FluidDensity, Buoyancy)  return(cfg) | To define the pipe properties of steel section and insulation section properties. |
| Steel Properties | def steelSectionProperties(cfg):  OD = cfg["geometry"]["NominalOD"]  ID = cfg["geometry"]["NominalID"]  data = {"OD": OD, "ID": ID}  cfg['SteelSection'] = sectionProperties(data)  return cfg | Defining the steel section properties. |
| Insulation Section Properties | def insulationSectionProperties(cfg):  OD=cfg["geometry"]["NominalOD"]+2\*cfg["geometry"]["ExternalCoating"]['Thickness']  ID = cfg["geometry"]["NominalOD"]  data = {"OD": OD, "ID": ID}  cfg['InsulationSection'] = sectionProperties(data) return cfg | Defining the insulation properties. |
| Buoyancy Section Properties | def buoyancySectionProperties(cfg):  OD = cfg['InsulationSection']["OD"]+2\*cfg["LazyWaveCatenaryDefinition"]["UniformBuoyancy"]['Thickness']  ID = cfg['InsulationSection']["OD"]  data = {"OD": OD, "ID": ID}  cfg['BuoyancySection'] = sectionProperties(data)  return cfg | Defining the buoyancy properties |
| 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 | Defining the section properties |
| Equivalent Pipe Properties | def equivalentPipe(cfg, FluidDensity, Buoyancy = False):  if (cfg['geometry']['Strakes']['BaseThickness'] == None and Buoyancy == False):  massPerUnitLength = cfg['SteelSection']['A']\*cfg['Material']['Steel']['Rho']\*(0.0254\*\*2) \  + cfg['InsulationSection']['A']\*cfg['Material']['ExternalCoating']['Rho']\*(0.0254\*\*2) \  + cfg['SteelSection']['Ai']\*FluidDensity\*(0.0254\*\*2)  weightPerUnitLength = (cfg['SteelSection']['A']\*cfg['Material']['Steel']['Rho']\*(0.0254\*\*2) \  + cfg['InsulationSection']['A']\*cfg['Material']['ExternalCoating']['Rho']\*(0.0254\*\*2) \  + cfg['SteelSection']['Ai']\*FluidDensity\*(0.0254\*\*2) \  - cfg['InsulationSection']['Ao']\*cfg['Material']['SeaWater']['Rho']\*(0.0254\*\*2) \ )\*cfg['default']['Constants']['g']  elif(cfg['geometry']['Strakes']['BaseThickness'] != None and Buoyancy == False):  massPerUnitLength = cfg['SteelSection']['A']\*cfg['Material']['Steel']['Rho']\*(0.0254\*\*2) \  + cfg['InsulationSection']['A']\*cfg['Material']['ExternalCoating']['Rho']\*(0.0254\*\*2) \  + cfg['Material']['Strakes']['MassPerUnitLength'] \  + cfg['SteelSection']['Ai']\*FluidDensity\*(0.0254\*\*2)  weightPerUnitLength = (cfg['SteelSection']['A']\*cfg['Material']['Steel']['Rho']\*(0.0254\*\*2) \  + cfg['InsulationSection']['A']\*cfg['Material']['ExternalCoating']['Rho']\*(0.0254\*\*2) \  + cfg['SteelSection']['Ai']\*FluidDensity\*(0.0254\*\*2) \  - cfg['InsulationSection']['Ao']\*cfg['Material']['SeaWater']['Rho']\*(0.0254\*\*2) \  )\*cfg['default']['Constants']['g'] \  + cfg['Material']['Strakes']['WeightPerUnitLength']  elif(Buoyancy == True):  massPerUnitLength = cfg['SteelSection']['A']\*cfg['Material']['Steel']['Rho']\*(0.0254\*\*2) \  + cfg['InsulationSection']['A']\*cfg['Material']['ExternalCoating']['Rho']\*(0.0254\*\*2) \  + cfg['BuoyancySection']['A']\*cfg['Material']['Buoyancy']['Rho']\*(0.0254\*\*2) \  + cfg['SteelSection']['Ai']\*FluidDensity\*(0.0254\*\*2)  weightPerUnitLength = (cfg['SteelSection']['A']\*cfg['Material']['Steel']['Rho']\*(0.0254\*\*2) \  + cfg['InsulationSection']['A']\*cfg['Material']['ExternalCoating']['Rho']\*(0.0254\*\*2) \  + cfg['BuoyancySection']['A']\*cfg['Material']['Buoyancy']['Rho']\*(0.0254\*\*2) \  + cfg['SteelSection']['Ai']\*FluidDensity\*(0.0254\*\*2) \  - cfg['BuoyancySection']['Ao']\*cfg['Material']['SeaWater']['Rho']\*(0.0254\*\*2) \ )\*cfg['default']['Constants']['g'] \  cfg['equivalentPipe'] = {"weightPerUnitLength": weightPerUnitLength, 'massPerUnitLength': massPerUnitLength  return(cfg)  def buoyancyFactorAndDiameter():  pass | Defining the equivalent pipe properties. |

## CatenaryRiserSummary.py

|  |  |  |
| --- | --- | --- |
| Input | Python Code | Description |
| Data Preparation | # Data preparation  defaultYml = "dataManager\\catenarySummary.yml" | Defining the data preparation |
| Update Yaml | # Get updateYML file  try:  if (sys.argv[1] != None):  updateYml = "dataManager\\" + 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") | Updating the yml |
| Update Configuration | # 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 | # Set logging  setLogging(cfg['default']['logLevel'])  # Get file List  fileList = []  for fileIndex in range(0, len(cfg['ymlFiles'])):  fileList.append(cfg['ymlFiles'][fileIndex]['io']) |  |
| Save Data | # Get data  dataDF = extractData(fileList, cfg)  # Save Data  fileName = cfg['dataFrame']['label']  saveDataFrame(dataDF, fileName) |  |
| Plot Data | # Get plot data  dataPlotDF = extractPlotData(fileList, cfg)  colors = ['blue', 'green', 'cyan', 'magenta', 'yellow', 'black']  for fileIndex in range(0, len(dataPlotDF)):  X = dataPlotDF.iloc[fileIndex, 1]["X"]  Y = dataPlotDF.iloc[fileIndex, 1]["Y"]  if cfg['plot']['settings'] == None:  plt.plot(X, Y, color = colors[fileIndex], label = cfg['ymlFiles'][fileIndex]['Label'])  else:  plt.plot(X, Y, color = colors[fileIndex], label = cfg['ymlFiles'][fileIndex]['Label'], linewidth=cfg['plot']['settings']['linewidth'][fileIndex])  X = dataPlotDF.iloc[fileIndex, 2]["X"]  Y = dataPlotDF.iloc[fileIndex, 2]["Y"]  if cfg['plot']['settings'] == None:  plt.plot(X, Y, color = 'red')  else:  plt.plot(X, Y, color = 'red', linewidth=cfg['plot']['settings']['linewidth'][fileIndex]\*2)  X = dataPlotDF.iloc[fileIndex, 3]["X"]  Y = dataPlotDF.iloc[fileIndex, 3]["Y"]  if cfg['plot']['settings'] == None:  plt.plot(X, Y, color = colors[fileIndex])  else:  plt.plot(X, Y, color = colors[fileIndex], linewidth=cfg['plot']['settings']['linewidth'][fileIndex])  plt.xlabel('Horizontal distance[m]', fontsize=12, fontweight='bold', color='black')  plt.ylabel('Distance Below Hang-off [m]', fontsize=12, fontweight='bold', color='black')  plt.title('Lazy Wave Riser Shape',fontsize=14, fontweight='bold', color='black')  plt.grid()  plt.legend()  plt.savefig('results\\' + cfg['FileName'].replace(".", ""), dpi=800) |  |

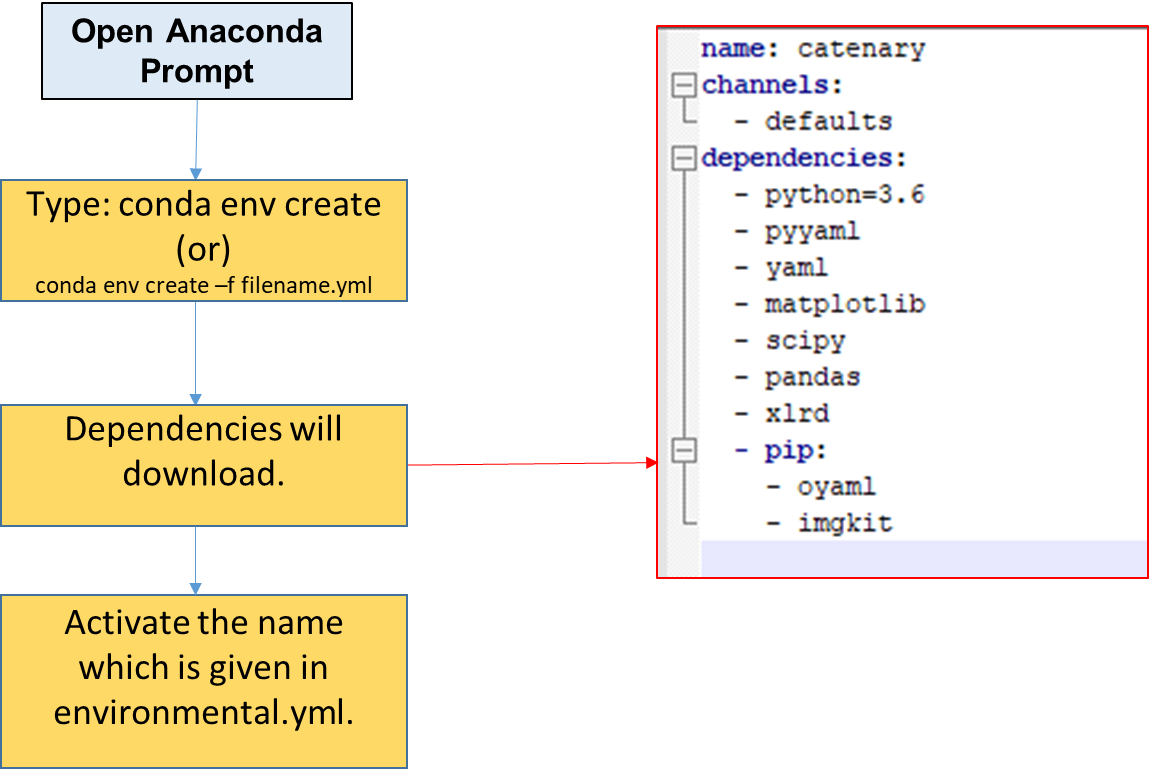
# Post- Process



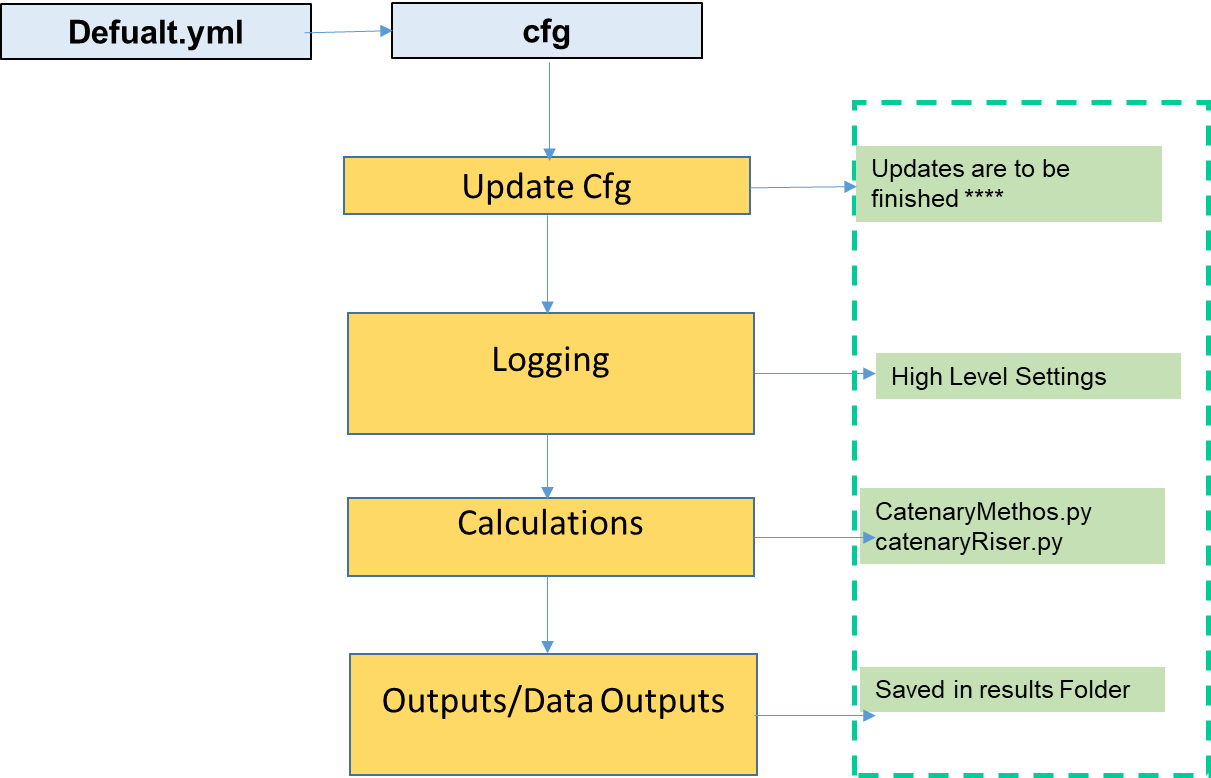
Refer section15: " Dropbox\Engineering\0026 Technical Manuals\0026-ORC-0001-01 Orcaflex Analysis Manual (Draft2).doc".

# Common Code

## Virtual Environment

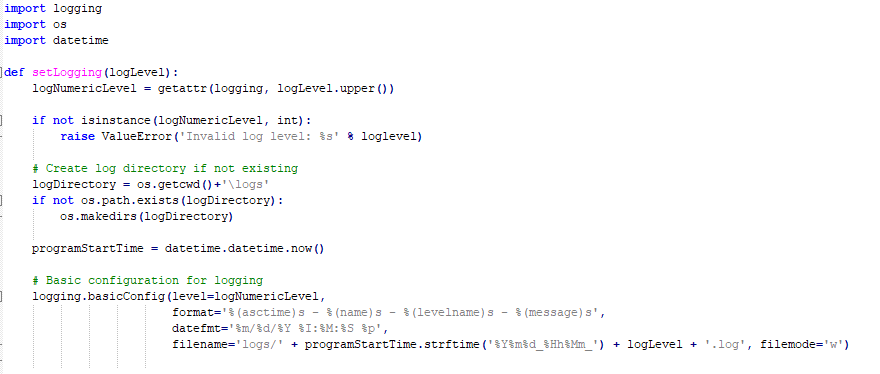


## Program Handling



## Set Logging

Objective: To set logging level. logLevels can be 'DEBUG', 'INFO', 'WARNING', 'ERROR', 'CRITICAL‘



## Yaml

import oyaml as yaml

import collections

from collections import Mapping

def ymlInput(defaultYml, updateYml):

with open(defaultYml, 'r') as ymlfile:

cfg = yaml.load(ymlfile)

if updateYml != None :

# Update values file

try:

with open(updateYml, 'r') as ymlfile:

cfgUpdateValues = yaml.load(ymlfile)

# Convert to logs

# print(cfgUpdateValues)

cfg = update\_deep(cfg, cfgUpdateValues)

except:

print("Update Input file could not be loaded successfully. Running program default values")

return cfg

def update\_deep(d, u):

for k, v in u.items():

# this condition handles the problem

if not isinstance(d, Mapping):

d = u

elif isinstance(v, Mapping):

r = update\_deep(d.get(k, {}), v)

d[k] = r

else:

d[k] = u[k]

return d

## DataFrame to Image

The following code is used to create the image using data frames.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import six

df = pd.DataFrame()

df['date'] = ['2016-04-01', '2016-04-02', '2016-04-03']

df['calories'] = [2200, 2100, 1500]

df['sleep hours'] = [2200, 2100, 1500]

df['gym'] = [True, False, False]

def render\_df\_table(data, col\_width=3.0, row\_height=0.625, font\_size=14,

header\_color='#40466e', row\_colors=['#f1f1f2', 'w'], edge\_color='w',

bbox=[0, 0, 1, 1], header\_columns=0,

ax=None, \*\*kwargs):

if ax is None:

size = (np.array(data.shape[::-1]) + np.array([0, 1])) \* np.array([col\_width, row\_height])

fig, ax = plt.subplots(figsize=size)

ax.axis('off')

mpl\_table = ax.table(cellText=data.values, bbox=bbox, colLabels=data.columns, \*\*kwargs)

mpl\_table.auto\_set\_font\_size(False)

mpl\_table.set\_fontsize(font\_size)

for k, cell in six.iteritems(mpl\_table.\_cells):

cell.set\_edgecolor(edge\_color)

if k[0] == 0 or k[1] < header\_columns:

cell.set\_text\_props(weight='bold', color='w')

cell.set\_facecolor(header\_color)

else:

cell.set\_facecolor(row\_colors[k[0]%len(row\_colors) ])

return ax, fig

def DataFrame\_To\_Image(df, FileName):

(ax, fig) = render\_df\_table(df, header\_columns=0, col\_width=2.0, font\_size=8)

fig.savefig('results//' + FileName)

## DataFrame to Xlsx

|  |  |  |
| --- | --- | --- |
| Input | Description | Referance |
| Data Frame to Xlsx | def DataFrame\_To\_xlsx\_xlsxwriter(df, data):  import xlrd  writer = pd.ExcelWriter(data['FileName'], engine='xlsxwriter')  try:  # WorkSheet = wb.get\_sheet\_by\_name(data['SheetName'])  WorkSheet = wb[data['SheetName']]  except:  wb.create\_sheet(data['SheetName'])  WorkSheet = wb[data['SheetName']]  df.to\_excel(writer, data['SheetName'])  writer.save() | <https://xlsxwriter.readthedocs.io/working_with_pandas.html> |
| Data Frame to Xlsx to Openpyxl | def DataFrame\_To\_xlsx\_openpyxl(df, data):  try:  wb = load\_workbook(data['FileName'])  writer = pd.ExcelWriter(data['FileName'], engine = 'openpyxl')  writer.wb = wb  except:  writer = pd.ExcelWriter(data['FileName'])  try:  # WorkSheet = wb.get\_sheet\_by\_name(data['SheetName'])  WorkSheet = wb[data['SheetName']]  except:  wb.create\_sheet(data['SheetName'])  WorkSheet = wb[data['SheetName']]  # For xlsxwriter  # WorkSheet = wb.add\_worksheet(data['SheetName'])  df.to\_excel(writer, data['SheetName'])  writer.save() | <https://stackoverflow.com/questions/42370977/how-to-save-a-new-sheet-in-an-existing-excel-file-using-pandas/42371251>  <https://stackoverflow.com/questions/36814050/openpyxl-get-sheet-by-name> |
| DataFrameArray\_To\_xlsx\_openpyxl | def DataFrameArray\_To\_xlsx\_openpyxl(dfArray, data):  # try:  # # Load existing workbook  # wb = load\_workbook(filename = data['FileName'])  # writer = pd.ExcelWriter  # writer.wb = wb  # except:  print("Opening new workbook")  writer = pd.ExcelWriter(data['FileName'], engine = 'openpyxl')  for dfIndex in range(0, len(dfArray)):  dfArray[dfIndex].to\_excel(writer, data['SheetNames'][dfIndex])  # property cell.border should be used instead of cell.style.border  # if data['thin\_border']:  # ws = wb.active  # thin\_border = Border(  # left=Side(border\_style=BORDER\_THIN, color='00000000'),  # right=Side(border\_style=BORDER\_THIN, color='00000000'),  # top=Side(border\_style=BORDER\_THIN, color='00000000'),  # bottom=Side(border\_style=BORDER\_THIN, color='00000000')  # )  # rows = len(dfArray[dfIndex])+1  # columns = len(dfArray[dfIndex].columns)+1  # ws.cell(row=rows, column=columns).border = thin\_border  writer.save()  if \_\_name\_\_ == '\_\_main\_\_':  import numpy as np  df = pd.DataFrame(np.random.random([3, 3]),  columns=['A', 'B', 'C'], index=['first', 'second', 'third'])  data = {"FileName" : "example.xlsx",  "SheetName": 'Sheet1'}  DataFrame\_To\_xlsx(df, data) | <https://stackoverflow.com/questions/24917201/applying-borders-to-a-cell-in-openpyxl> |

## Line Plot

## Counter Plot

# references

– PROGRAM HISTORY

## Revision History

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

Figure 4‑1.Program History

- ERROR LOG

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

Figure 4‑2.Error Log.

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.

Figure 4‑3.Test Cases.