Delft University of Technology

Manual: MoC / Radial Stator Code

Author : Nitish Anand
Electronic mail : nitish.ug2@gmail.com

Date : 16th August 2016

Abstract

Method of characteristics(MoC) is a marching type method which solves hyperbolic partial differential equations. This MoC code designs a de Laval Nozzle. The output of the MoC has been used to design a radial stator for ORC application. Details on the method can be found in the Master Thesis [1].

Contents

[1] email author

1 MoC Code

MOC_Config.in is the default configuration file for the MOC code. However, a different configuration file name can be given as argument during function call:

1.1 RUN SYNTAX:

```
nitish@NITISH:./MAIN <input_file_name>
```

1.2 Configuration file inputs:

1. **PLANAR/AXISSYM:** Specify if the nozzle is Axis-symmetric or Planar (Note: At the moment the code is only for Planar cases, but axis-symmetric is open for future application)

```
NozzleType <args>
#<args> = PLANAR / AXISSYM
```

2. **EoS:** Equation of state to be used for thermodynamic calculations.

PRFT	:Perfect Gas
EoS	:Van der Waals
EoS_TAB	:Van der Waals with tables
RefProp	:FluidProp
RefProp_TAB	:FluidProp with tables (not recommended)

```
GAS_EQU <args>
# <args> = PRFT, EoS, EoS_TAB, RefProp, RefProp_TAB
```

3. **Total Conditions:** Specify total conditions. Required in all Gas Models
   To: Total temperature in Kelvin
   Po: Total pressure in Pa

```
To = <args>              # K
Po = <args>              # Pa
```

4. **GAS_EQN = PRFT:** Parameters required to use Perfect Gas Model.

```
gamma = <args>           # cnst
R = 8.314                # mol/Kg/K
M = <args>               # kg/mol
```

5. **GAS_EQN = RefProp:** Specify Fluid name from FluidProp file. Make sure *.FLD file is available in FluidProp library. Also specify from where the Throat Properties should be calculated, from 'FILE' or by 'ITER' (iteration).

```
FLDNAME <args>          # Specify *.FLD name
THROAT_PROP <args>      #<args> = ITER / FILE
```

6. **GAS_EQN = EoS :** Specify properties for Van der Waals equation of State.
   ()Guess = Specify guess value for the properties (specific volume, Temperature, Pressure)
   ()c = Specify critical temperature, pressure and specific volume.
   ()o = Specify Total Enthalpy and Entropy.

```
VrhoGuess = <args>      # m^3/mol
TGuess = <args>         # K
PGuess = <args>         # Pa

Tc = <args>             # K
Pc = <args>             # Pa
Vc = <args>             # m^3/mol

Ho = <args>             # J/Kg/K
so = <args>             # J/Kg
```

7. **Table Control:** Specify the limit and delta for the table. LLim = Lower Limit velocity LLim = Upper Limit velocity dV = step of the velocity

```
Table_Oper Calc
LLim = <args>
ULim = <args>
dV = <args>
```

8. **Design Variables:** Refer the Figure 1 for details about the parameters.

```
rho_t = <args>
y_t = <args>
rho_d = <args>
Noz_Design_Mach = <args>
```

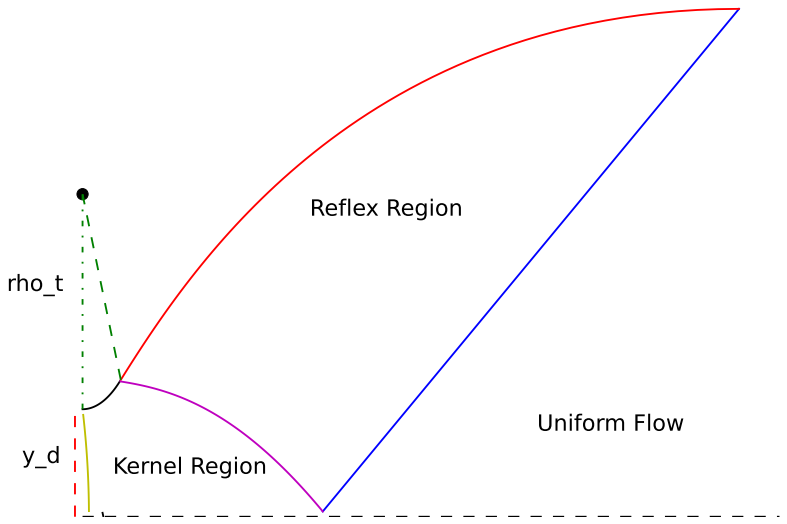

Figure 1: Nozzle Parametrization

- - - - - = Axis of symmetry
_____ = Expansion Curvature
_____ = Reflex Curvature
_____ = Uniform property line
_____ = Sauer Line
_____ = Kernel to Reflex
- - - - - -= Radius along which the expansion region is defined
• = Center of rotation for creating expansion region

9. **Throat Parameter:** Specify the number of points on the throat. (Recommended: 20)

```
n = <args>
```

10. **Kernel Parameter:** Specify the differential angle at which the wall pressure waves should be calculated. (Recommended: 0.25)

```
dtau = <args>
```

11. Reflex Parameter: Specify the number of divisions in the reflex zone.

```
n_ref = <args>
```

12. **Convergence Criteria:** Specify the space and velocity convergence values. (Recommended: 1e-6)

```
tolerance_v <args>
tolerance_x <args>
```

13. **File Writing:** Specify the co-ordinate and property file name.

```
File_Name_NProp <name>.out
File_Name_NCods <name>.out
```

Sample Configuration file is available at: "Link: Coming Soon"

1.3 Special Comments:

1.3.1 RefProp Runs:

When doing a RefProp run the properties at the Throat are determined using an iteration. Hence, it might take longer to initialize MoC compared to the other gas models.

2 Stator Design Code:

The stator code has the function Optimize(args), which calls the main function named: CentripetalSupersonicStator(). The function returns a set of value which is minimized by the minimization function. The arguments required to run the Stator tools are:

1. Flow angle : flow angle in degree
2. Number of Blades : n
3. Inlet Radius : radius in meters
4. Outlet Radius Achieved : Optimized by the code (args[0])
5. MOC input file : Name of the MOC input file
6. Kernel Radius Ratio (ρ_t/y_d) : 5
7. Angle of converging circle 1 : angle in degree (80-220)
8. Angle of converging circle 2 : angle in degree (5-20)
9. Conv. circles connect radius : 1 ideally (can be increased or decreased to improve the shape)
10. Nozzle Scale : Optimized by the code (args[1])
11. Design Outlet Radius : radius in meters
12. Type of converging section curve : Bspline / Circle
13. Type of converging section : Concave / Convex
14. weight for Converging Circle : tweak the value to make the stator within the other limits
15. weight for Connecting Circle : tweak the value to make the stator look more uniform
16. print & write : yes/no (use 'yes' for final runs and 'no' for Optimizaiton)

When you run the code for Optimization make sure to give the reasonable input values and if the blades are small use 'convex' connecting circles (parameter 13 in teh above list). Set the design parameters in the function Optimize as:

```
def Optimize(args):
        try: disp=args[2]
        except: disp='no'
        test = CentripetalSupersonicStator(80.4,18, 0.1685,
    args[0],'Nozzle_coords_4.out',10.0,220,5,2.5,args[1],
    0.11908,'CIRC','concave',1.2,1.1,disp)
```

```
        print('Tip␣Radius:␣',args[0],'Scaling␣Factor:␣',args[1],
    'RESIDUAL␣:',test.d)
return abs(test.d)
```

To start the optimization process, call:

```
Res=minimize(Optimize,[0.1199738,  7.90084817],
method='Nelder-Mead')
```

where, arguments should be outlet radius and a reasonable scaling value. The scaling values should be selected such that the 'RESIDUAL' in the optimization process is less than 1. Optimization will give output like:

```
Tip Radius: 0.12 Scaling Factor: 7.72550 RESIDUAL: 0.00109
Tip Radius: 0.12 Scaling Factor: 7.72539 RESIDUAL: 0.00112
Tip Radius: 0.12 Scaling Factor: 7.72541 RESIDUAL: 0.00112
Tip Radius: 0.12 Scaling Factor: 7.72560 RESIDUAL: 0.00109
```

Once the optimization is over use the converged radius and scale values to call the function in the next step (comment out the Res=minimize Call).

```
Optimize([0.05416807,  5.95574489])
```

3 Installing FluidProp wrapper for Python:

In the folder FluidProp_py you will find install file. Run the file in with bash. In case you want to install in a Python version other than 3.0 then make appropriate changes in the install.sh file.

Note: Make sure the /.bashrc file contains

```
export fluidprop=/home/<USERNAME>/fp/api/
```

SYNTAX:

```
bash install.sh
```

Run the test file in python to check if the test file prints legitimate value on the screen.