In [2]:

```python
### Benjamin Tollison ###
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy
import sympy as sp
from IPython.display import Latex, Math, display
from sympy import (
    Eq,
    Function,
    Matrix,
    cos,
    cosh,
    exp,
    integrate,
    lambdify,
    pi,
    sin,
    sinh,
    symbols,
)
from decimal import Decimal
from sympy.solvers.pde import pdsolve
from sympy.solvers.solveset import linsolve
def displayEquations(LHS,RHS):
    left = sp.latex(LHS)
    right = sp.latex(RHS)
    display(Math(left + '=' + right))
    np.set_printoptions(suppress=True)
def displayVariable(variable:str,RHS):
    left = sp.latex(symbols(variable))
    right = sp.latex(RHS)
    display(Math(left + '=' + right))
def displayVariableWithUnits(variable:str,RHS,units):
    left = sp.latex(symbols(variable))
    right = sp.latex(RHS)
    latexUnit = sp.latex(symbols(units))
    display(Math(left + '=' + right + '\\;' +'\\left['+ latexUnit + '\\right]'))
def format_scientific(number:float):
    a = '%E' % number
    return a.split('E')[0].rstrip('0').rstrip('.') + 'E' + a.split('E')[1]
deg2rad = np.pi/180
rad2deg = 180/np.pi
```

In [33]:

```python
inlet_area = np.pi*(.25**2-.1**2)
rotation_speed = 7200*(2*np.pi/60)
displayVariableWithUnits('\\sigma',inlet_area,'m^2')
displayVariableWithUnits('u_1',rotation_speed,'\\frac{m}{s}')
inlet_total_pressure = 1.02e5 # Pa
inlet_total_temperature = 335 # K
mass_flow_rate = 5 # kg/s
demensionaless_mass_flow = (mass_flow_rate*(287*inlet_total_temperature)**0.5)/(inlet_t
def MachFlow(machnumber):
  kappa = demensionaless_mass_flow
  gamma = 1.4
  M = machnumber
  return (gamma)**0.5 * M * (1 + ((gamma-1)*M**2)/2 )**((-gamma-1)/(2*gamma-2)) - kappa
def MachFlowPrime(machnumber):
  gamma = 1.4
  M = machnumber
  first_part = (gamma)**0.5*(1 + ((gamma-1)*M**2)/2 )**((-gamma-1)/(2*gamma-2))
```

```python
  second_part = (gamma)**0.5 *M*((-gamma-1)/(2*gamma-2))*(1 + ((gamma-1)*M**2)/2 )**((-
  return first_part + second_part
intial_mach_guess = 1.2
increment_cutoff = 100
increment_count = 0
while abs(MachFlow(intial_mach_guess))>1e-8:
  increment_count += 1
  if increment_count == increment_cutoff:
    print('scheme didn\'t converge')
    displayVariable('e_r',format_scientific(abs(MachFlow(intial_mach_guess))))
    break
  intial_mach_guess = intial_mach_guess - MachFlow(intial_mach_guess)/MachFlowPrime(int
final_mach = intial_mach_guess
displayVariable('M_i',final_mach)
displayVariable('e_r',format_scientific(abs(MachFlow(intial_mach_guess))))
displayVariable('i',increment_count)
```

$\sigma = 0.164933614313464 \; \left[m^2\right]$

$u_1 = 753.98223686155 \; \left[\frac{m}{s}\right]$

$M_i = 3.59709624357504$

$e_r = $ 5.800749E-09

$i = 5$

In [32]:

```python
virtual_speed = final_mach*(1.4*287*inlet_total_temperature)**0.5
displayVariableWithUnits('w_1',virtual_speed,'\\frac{m}{s}')
inducer_angle = np.arccos(rotation_speed/virtual_speed)
displayVariableWithUnits('\\beta_1',inducer_angle,'rad')
displayVariableWithUnits('\\beta_1',inducer_angle*rad2deg,'deg')
```

$w_1 = 1319.71279595148 \; \left[\frac{m}{s}\right]$

$\beta_1 = 0.962679419542757 \; [rad]$

$\beta_1 = 55.1574677639039 \; [deg]$