Contents

1 3D Solver

```
# -*- coding: utf-8 -*-
"""
Created on Fri July 17 12:31:26 2020

@author: nastavirs
"""
import tensorflow as tf
import numpy as np
def net_NS3D(self, x, y, z, t):
        Re = 1;
        V_P = self.neural_net(tf.concat([x,y,z,t], 1), self.weights, self.biases) #"neuralnet data processing"
        u = V_P[:, 0:1] # categorizing data
        v = V_P[:, 1:2]
        w = V_P[:, 2:3]
        p = V_P[:, 3:4]

        u_t = tf.gradients(u, t)[0] #"gradient of u using automatic differentiation"
        u_x = tf.gradients(u, x)[0]
        u_y = tf.gradients(u, y)[0]
        u_z = tf.gradients(u, z)[0]
        u_xx = tf.gradients(u_x, x)[0]
        u_yy = tf.gradients(u_y, y)[0]
        u_zz = tf.gradients(u_z, z)[0]

        v_t = tf.gradients(v, t)[0] #"gradient of v using automatic differentiation"
        v_x = tf.gradients(v, x)[0]
        v_y = tf.gradients(v, y)[0]
        v_z = tf.gradients(v, z)[0]
        v_xx = tf.gradients(v_x, x)[0]
        v_yy = tf.gradients(v_y, y)[0]
        v_zz = tf.gradients(v_z, z)[0]

        w_t = tf.gradients(w, t)[0] #"gradient of w using automatic differentiation"
        w_x = tf.gradients(w, x)[0]
        w_y = tf.gradients(w, y)[0]
        w_z = tf.gradients(w, z)[0]
        w_xx = tf.gradients(w_x, x)[0]
        w_yy = tf.gradients(w_y, y)[0]
        w_zz = tf.gradients(w_z, z)[0]

        p_x = tf.gradients(p, x)[0] #"gradient of p using automatic differentiation"
        p_y = tf.gradients(p, y)[0]
        p_z = tf.gradients(p, z)[0]

```

```
45          #"minimum squared error for N-S equations"
46         f_u = u_t + (u*u_x+v*u_y+w*u_z) + p_x - (1/Re)*(u_xx+u_yy+
    u_zz)
47         f_v = v_t + (u*v_x+v*v_y+w*v_z) + p_y - (1/Re)*(v_xx+v_yy+
    v_zz)
48         f_w = w_t + (u*w_x+v*w_y+w*w_z) + p_z - (1/Re)*(w_xx+w_yy+
    w_zz)
49         f_c = u_x + v_y + w_z
50 
51         return u, v, w, p, f_u, f_v, f_w, f_c
```

Listing 1: Solver

Line 7-8: import numpy and tensorflow for calculations

Line 9 : define net_NS3D function with arguments class x,y,z,t

Line 10 : Initialize Reynolds number.

Line 11 : define variable psi_and_p as the return of the function neuralnet with tensorflow variable which is the result of concatenation of x,y and t and 1 implies along column axis, weights, biases which are passed as arguments for the class

Line 12-15 : slice V_P columns wise and assign it to u,v,w,p.

Line 17 : calculating the gradient of u wrt t and assign it to u_t

Line 18 : calculating the gradient of u wrt x and assign it to u_x

Line 19 : calculating the gradient of u wrt y and assign it to u_y

Line 20 : calculating the gradient of u wrt z and assign it to u_z

Line 21 : calculating the gradient of u_x wrt x and assign it to u_xx

Line 22 : calculating the gradient of u_y wrt y and assign it to u_yy

Line 23 : calculating the gradient of u_y wrt y and assign it to u_yy

Line 25 : calculating the gradient of v wrt t and assign it to v_t

Line 26 : calculating the gradient of v wrt x and assign it to v_x

Line 27 : calculating the gradient of v wrt y and assign it to v_y

Line 28 : calculating the gradient of v wrt z and assign it to v_z

Line 29 : calculating the gradient of v_x wrt x and assign it to v_xx

Line 30 : calculating the gradient of v_y wrt y and assign it to v_yy

Line 31 : calculating the gradient of v_z wrt z and assign it to v_zz

Line 33 : calculating the gradient of w wrt t and assign it to w_t

Line 34 : calculating the gradient of w wrt x and assign it to w_x

Line 35 : calculating the gradient of w wrt y and assign it to w_y

Line 36 : calculating the gradient of w wrt z and assign it to w_z

Line 37 : calculating the gradient of w_x wrt x and assign it to w_xx

Line 38 : calculating the gradient of w_y wrt y and assign it to w_yy

Line 39 : calculating the gradient of w_z wrt z and assign it to w_zz

```
Line 41 : calculating the gradient of p wrt x and assign it to p_x

Line 42 : calculating the gradient of p wrt y and assign it to p_y

Line 43 : calculating the gradient of p wrt z and assign it to p_z

Line 45 : calculate the nonlinear partial differentiation equation (N-S) f as
           u_t+(u*u_x+v*u_y+w*u_z)+p_x-1/Re*(u_xx+u_yy+u_zz) and assign it to variable f_u

Line 46 : calculate the nonlinear partial differentiation equation (N-S) f as
           v_t+(u*v_x+v*v_y+w*v_z)+p_y-1/Re*(v_xx+v_yy+v_zz) and assign it to variable f_v

Line 47 : calculate the nonlinear partial differentiation equation (N-S) f as
           w_t+(u*w_x+v*w_y+w*w_z)+p_w-1/Re*(w_xx+w_yy+w_zz) and assign it to variable f_w

Line 48 : calculate the nonlinear partial differentiation equation (N-S) f as
          (u_x+v_y+w_z) and assign it ti variable f_e

Line 50 : returns u,v,w,p,f_u,f_v,f_w.f_e as the result of function net_NS
```