Python\functions.py

```
import numpy as np
import pandas as pd
import pysid
import control

def mean_squared_error(x, y):
  return np.square(x - y).mean()

def transferFunction(num, den):
  z = control.TransferFunction.z
  num_z = 0
  den_z = 0
  for i, n in enumerate(num):
    num_z += n * z**(-i)
  for i, d in enumerate(den):
    den_z += d * z**(-i)
  return control.minreal(num_z/den_z, verbose=False)

def predict(u, y, G, H):
  L_u = control.minreal(G/H,     verbose=False)
  L_y = control.minreal(1 - 1/H, verbose=False)

  delay = int(max(len(L_u.den[0][0]), len(L_y.den[0][0])) - 1)
  assert(delay >= 1)

  y_u = control.forced_response(sys=L_u, U=u, return_x=False)[1]
  y_y = control.forced_response(sys=L_y, U=y, return_x=False)[1]
  assert(len(y_u) == len(y_y))

  return y_u[delay:] + y_y[delay:], delay

def models_frame():
  return pd.DataFrame(columns=
['model','na','nb','nc','nd','nf','nk','Jp','A','B','C','D','F','G','H','zG','pG',

def arx(u_i, y_i, u_v, y_v, na_range, nb_range, nk_range):
  models = pd.DataFrame()
  for na in na_range:
    for nb in nb_range:
      for nk in nk_range:
        id = pysid.arx(na=na, nb=nb, nk=nk, u=u_i, y=y_i)
        A = id.A[0][0]
        B = id.B[0][0]

        assert(A[0] == 1)

        G = transferFunction(B, A)
        H = transferFunction([1], A)

        y_p, delay = predict(u_v, y_v, G, H)
        J_p = mean_squared_error(y_v[delay:], y_p)

        models = pd.concat([models, pd.DataFrame({
          'model': 'arx',
          'na': [na],
          'nb': [nb],
```

```python
          'nk': [nk],
          'A': [A],
          'B': [B],
          'G': [G],
          'zG': [G.zeros()],
          'pG': [G.poles()],
          'kG': [G.dcgain()],
          'H': [H],
          'zH': [H.zeros()],
          'pH': [H.poles()],
          'kH': [H.dcgain()],
          'yp': [y_p],
          'Jp': [J_p],
          'delay': [delay],
        })])

  return models

def armax(u_i, y_i, u_v, y_v, na_range, nb_range, nc_range, nk_range):
  models = pd.DataFrame()
  for na in na_range:
    for nb in nb_range:
      for nc in nc_range:
        for nk in nk_range:
          id = pysid.armax(na=na, nb=nb, nc=nc, nk=nk, u=u_i, y=y_i)
          A = id.A[0][0]
          B = id.B[0][0]
          C = id.C[0]

          assert(A[0] == 1)
          assert(C[0] == 1)

          G = transferFunction(B, A)
          H = transferFunction(C, A)

          y_p, delay = predict(u_v, y_v, G, H)
          J_p = mean_squared_error(y_v[delay:], y_p)

          models = pd.concat([models, pd.DataFrame({
            'model': 'armax',
            'na': [na],
            'nb': [nb],
            'nc': [nc],
            'nk': [nk],
            'A': [A],
            'B': [B],
            'C': [C],
            'G': [G],
            'zG': [G.zeros()],
            'pG': [G.poles()],
            'kG': [G.dcgain()],
            'H': [H],
            'zH': [H.zeros()],
            'pH': [H.poles()],
            'kH': [H.dcgain()],
            'yp': [y_p],
            'Jp': [J_p],
            'delay': [delay],
          })])

  return models

def oe(u_i, y_i, u_v, y_v, nb_range, nf_range, nk_range):
  models = pd.DataFrame()
  for nb in nb_range:
    for nf in nf_range:
      for nk in nk_range:
        id = pysid.oe(nb=nb, nf=nf, nk=nk, u=u_i, y=y_i)
        B = id.B[0][0]
        F = id.F[0][0]

        assert(F[0] == 1)

        G = transferFunction(B, F)
        H = transferFunction([1], [1])

        y_p, delay = predict(u_v, y_v, G, H)
        J_p = mean_squared_error(y_v[delay:], y_p)

        models = pd.concat([models, pd.DataFrame({
          'model': 'oe',
          'nb': [nb],
          'nf': [nf],
          'nk': [nk],
          'B': [B],
          'F': [F],
          'G': [G],
          'zG': [G.zeros()],
          'pG': [G.poles()],
          'kG': [G.dcgain()],
          'H': [H],
          'zH': [H.zeros()],
          'pH': [H.poles()],
          'kH': [H.dcgain()],
          'yp': [y_p],
          'Jp': [J_p],
          'delay': [delay],
        })])

  return models

def bj(u_i, y_i, u_v, y_v, nb_range, nc_range, nd_range, nf_range, nk_range):
  models = pd.DataFrame()
  for nb in nb_range:
    for nc in nc_range:
      for nd in nd_range:
        for nf in nf_range:
          for nk in nk_range:
            try:
              id = pysid.bj(nb=nb, nc=nc, nd=nd, nf=nf, nk=nk, u=u_i, y=y_i)
              B = id.B[0][0]
              C = id.C[0]
              D = id.D[0]
              F = id.F[0][0]

              assert(C[0] == 1)
              assert(D[0] == 1)
              assert(F[0] == 1)

              G = transferFunction(B, F)
              H = transferFunction(C, D)

              y_p, delay = predict(u_v, y_v, G, H)
              J_p = mean_squared_error(y_v[delay:], y_p)

              models = pd.concat([models, pd.DataFrame({
                'model': 'bj',
                'nb': [nb],
                'nc': [nc],
                'nd': [nd],
                'nf': [nf],
                'nk': [nk],
                'B': [B],
                'C': [C],
                'D': [D],
                'F': [F],
                'G': [G],
                'zG': [G.zeros()],
                'pG': [G.poles()],
                'kG': [G.dcgain()],
                'H': [H],
                'zH': [H.zeros()],
                'pH': [H.poles()],
                'kH': [H.dcgain()],
                'yp': [y_p],
                'Jp': [J_p],
                'delay': [delay]
              })])
            except Exception as e:
              # display(str(e))
              models = pd.concat([models, pd.DataFrame({
                'model': 'bj',
                'nb': [nb],
                'nc': [nc],
                'nd': [nd],
                'nf': [nf],
                'nk': [nk],
              })])

  return models

```