Estimasi Model Data

ARMA dan MLE di Python

Disusun oleh:

Achmadi's Team

seluruh material dalam diktat ini tersedia di URL:

https://github.com/mekatronik-achmadi/tugas_kuliah/tree/master/Model_Estimation/Python

Kata Pengantar

Hari ini telah diketahui bersama, perkembangan komputer telah mencapai taraf yang sangat tinggi. Penggunaan komputer telah begitu dekat dengan kehidupan manusia, mulai dari ukuran genggaman tangan hingga sebesar bangunan pabrik. Salah satu penggunaan komputer yang banyak ditemui adalah dalam bidang rancang-bangun atau rekayasa. Dalam bidang ini, komputer digunakan untuk memperkirakan (*estimasi*) suatu sistem berdasarkan data input-output tanpa perlu mengetahui pemodelan sistem sebenarnya secara fisis. Penjelasan singkat ini berusaha memberikan satu contoh bagaimana langkah-langkah memperkirakan model suatu sistem yang dijelaskan dengan menitikberatkan kepada sisi teknis ketimbang sisi teori.

Daftar Isi

Dasar Teori

ARMA

ARMA adalah salah satu model matematika yang digunakan sebagai dasar untuk memperkirakan model suatu sistem. Model matematika ini termasuk model yang banyak dipakai apabila data yang digunakan adalah data statistik yang stasioner. Model terdiri dari 2 bagian, yaitu AR (*Auto-Regresive*) dan MA (*Moving-Average*). Secara umum, bentuk matematisnya adalah sebagai berikut:

$$y_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

Dengan **c** adalah konstanta awal (seringkali bernilai 0), **p** dan **q** adalah tingkat (*order*), **x** adalah input, dan **ε** adalah gangguan yang disebut *white-noise.* Dalam banyak software, seringkali model ARMA cukup sebagai fungsi **ARMA(p,q)**, sedangkan **x** dan **ε** didapat melalui proses estimasi.

Bentuk lain model ARMA adalah dengan mengganti *white-noise* dengan output langkah sebelumnya, sebab *white-noise* juga dapat dianggap sebagai respon sistem dimana perubahannya dapat pula merepresentasikan output sistem. Sehingga bentuk ini dapat ditulis sebagai berikut:

$$y_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{i=1}^{q} \theta_i y_{t-i}$$

MLE

Mendapatkan model matematika saja tidak cukup untuk memperkirakan model suatu sistem. Diperlukan suatu proses perkiraan (*estimasi*) yang umumnya adalah proses berulang (*iteration*). Sehingga model matematika yang telah didapat memiliki arti matematis yang kemudian model dapat digunakan untuk perancangan (rekayasa) atau memperkirakan output di masa depan (*forecasting*). Terdapat banyak metode estimasi diantaranya yang banyak digunakan adalah BJ (*Box-Jenkins*) dan MLE (*Maximum Likehood Estimation*).

Disini digunakan MLE yang dibangun berdasarkan penentuan tingkat persamaan satu anggota terhadap anggota lain dalam satu data statistik. Tingkat persamaan ini sering disebut *log-likelihood* yang ditentukan oleh suatu fungsi yang disebut *log-likelihood function*, yang ditulis sebagai berikut:

$$\ln L(\theta; x_1, x_2, , , x_n) = \ln f(x_i|\theta)$$

Instalasi Software

Tentang Software

Disini digunakan software atau tepatnya bahasa pemrograman yang telah digunakan sangat luas, yaitu Python. Bahasa Python adalah bahasa interpretatif tingkat tinggi yang memiliki banyak keunggulan antara lain:

- Modular.
- Dinamis.
- Berbasis objek
- Tidak spesifik pada penggunaan tertentu (*general-purpose*).
- Tingkat *readability* tinggi.
- Perkembangannya yang pesat hampir disemua bidang teknologi.

Selain karena keunggulan tersebut, alasan lain mengapa dipilih Python dibandingkan software lain yang berbayar seperti Matlab, Mathematica, atau MathCad adalah karena Python sifatnya yang bebas terbuka (*Free Open-Sources*), sehingga tidak diperlukan biaya untuk membeli software atau melakukan pelanggaran berupa pembajakan software.

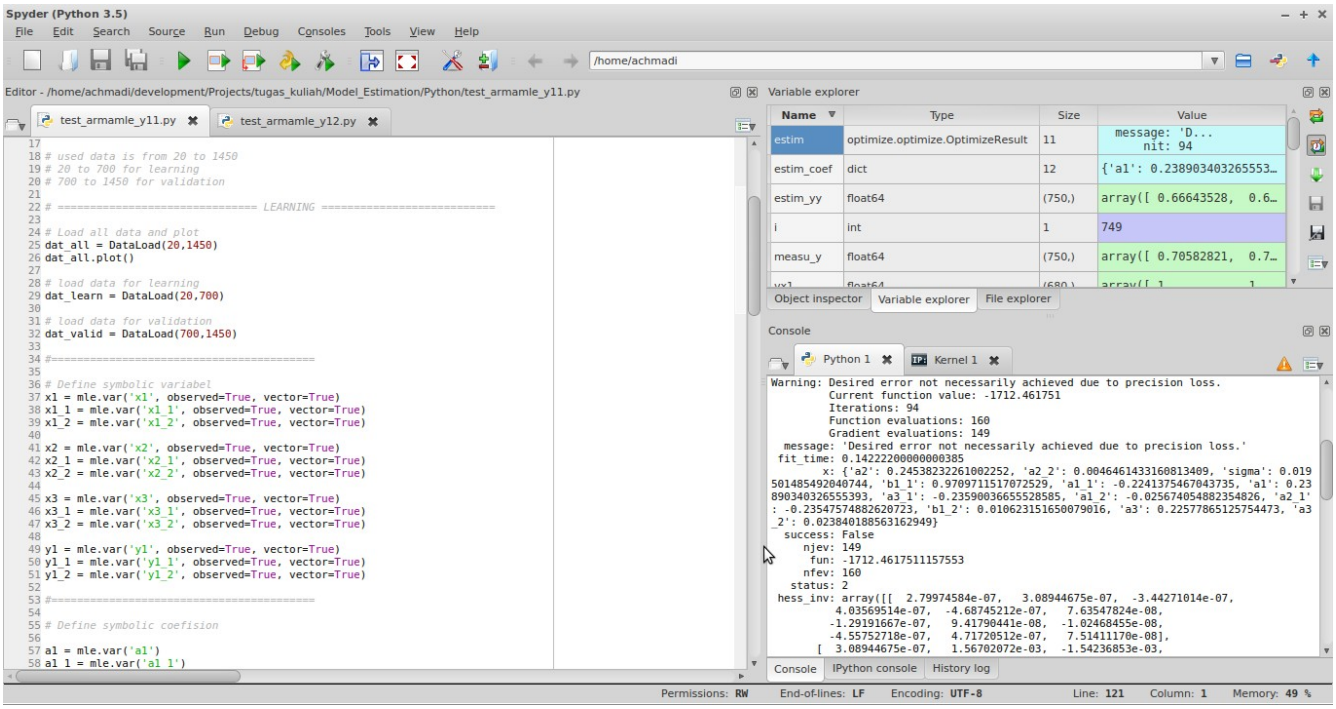
Linux

Secara umum, sistem operasi atau distro berbasis Linux yang terkenal seperti Ubuntu, Debian, LinuxMint, atau Fedora telah terinstal interpreter Python (dalam hal ini Python 3x) sebagai software bawaan. Disini akan dijelaskan contoh instalasi dalam distro Arch Linux mengingat distro ini adalah distro *rolling-release* sehingga memiliki update paket paling cepat dan sangat dekat dengan pengembang tiap paket (*upstream*).

Spyder

Spyder adalah software IDE (*Integrated Development Environment*) untuk Python yang dibuat mirip dengan Matlab dimana terdapat fitur utama seperti Code-Editor, Variable Explorer, dan Console. Berikut perintah instalasi Spyder bersama paket-paket lain yang akan diperlukan disini:

```
$ sudo pacman -S spyder3 python-rpy2 python-matplotlib python-pytables python-sympy python-scipy python-pandas python-h5py python-pillow python-jedi python-patsy python-nose
```



Theano

Theano adalah modul Python yang didesain untuk penggunaan optimisasi dan evaluasi expresi matematika. Modul populer digunakan untuk kebutuhan seperti Machine Learning. Untuk instalasi di Arch Linux, paket modul ini tersedia di repositori AUR (*Arch User Repository*) dengan URL:

https://aur.archlinux.org/packages/python-theano-git/

MLE

MLE disini adalah modul Python yang digunakan untuk proses pemodelan dan estimasi. Untuk dapat berjalan dengan baik, modul ini membutuhkan modul Theano. Paket untuk modul ini belum tersedia baik di repositori resmi maupun AUR. Untuk instalasi paket ini dapat mengunjungi repositori proyek pengembang di URL:

https://github.com/ibab/python-mle/

atau jika membutuhkan PKGBUILD dapat menggunakan PKGBUILD yang sudah penulis buat di URL:

https://github.com/mekatronik-achmadi/my_pkgbuild/blob/master/python-extra/PKGBUILD

Windows

Untuk instalasi di sistem operasi Windows, tidaklah semudah di sistem operasi atau distro berbasis Linux dikarenakan memang sistem operasi Windows tidak dirancang sebagai sistem operasi untuk kebutuhan pemrograman. Meski demikian, pengembang Python tetap menyediakan *binary* dan kode sumber modul baik dalam paket *installer* maupun paket terkompres sehingga Python dapat dijalankan di sistem operasi Windows. Untuk instalasi Python 3x, dapat diunduh dari URL berikut:

https://www.python.org/downloads/windows/

Spyder

Untuk instalasi Spyder di sistem operasi Windows maka dapat diinstal salah satu dari paket-paket software berikut. Paket-paket software ini telah menyertakan Spyder dalam instalasinya. Berikut URL yang dapat dikunjungi:

Python(x,y) : http://python-xy.github.io/

WinPython : https://winpython.github.io/

Anaconda : https://www.continuum.io/

Apabila diperlukan untuk mengecek semua kebutuhan modul Python untuk Spyder, maka dapat dilihat daftar kebutuhan modulnya di URL berikut:

https://github.com/spyder-ide/spyder

Theano

Untuk instalasi Theano dapat digunakan salah satu paket software Spyder di atas. Langkah-langkah instalasi dapat dilihat di URL berikut:

https://blogs.msdn.microsoft.com/lukassteindl/2015/11/23/deep-learning-simple-installation-guide-for-theano-on-windows/

MLE

Untuk instalasi MLE dapat digunakan langkah instalasi sebagai Theano di atas dengan kode sumber modul dapat di download di URL:

https://github.com/ibab/python-mle

Data Loading

Sumber Data

Sumber data yang digunakan adalah data yang di-*generate* dari software lain yang kemudian disimpan dalam format *.m dan *.mat. Disini yang dapat digunakan adalah yang berformat *.mat. Terdiri dari 2 set data yaitu set data **input** dan set data **output**, dimana akan digunakan 3 data input dan 2 data ouput. Kedua set data ini dapat di download di URL:

https://github.com/mekatronik-achmadi/tugas_kuliah/blob/master/Model_Estimation/Python/data_input.mat

https://github.com/mekatronik-achmadi/tugas_kuliah/blob/master/Model_Estimation/Python/data_output.mat

Data Loading

Untuk me-*loading* data ke workspace dengan dipanggil oleh skrip Python lain, maka proses *data loading* ini dibentuk sebagai sebuah *Class*. Menulis skrip dalam bentuk class akan memudahkan pekerjaan karena class dapat berisi fungsi maupun data sehingga kode sumber dapat dipisah-pisah dan memudahkan pemrograman. Skrip yang berisi proses data loading ini diberi nama **data_load.py**. Berikut isi class data_load.py:

```
import numpy as np
import scipy.io as sio
import matplotlib.pyplot as plt

class DataLoad:
    def __init__(self,n_start,n_stop):
        input_data=sio.loadmat("data_input.mat")['input_data']
        output_data=sio.loadmat("data_output.mat")['output_data']

        u_11 = input_data[n_start:n_stop,1]
        u_12 = input_data[n_start:n_stop,3]
        u_13 = input_data[n_start:n_stop,2]
        y_11 = output_data[n_start:n_stop,1]
        y_12 = output_data[n_start:n_stop,2]

        self.u11 = u_11/u_11.max()
        self.u12 = u_12/u_12.max()
        self.u13 = u_13/u_13.max()
        self.y11 = y_11/y_11.max()
        self.y12 = y_12/y_12.max()

        self.t = np.arange(0,n_stop-n_start,1)
        self.dim = np.shape(self.t)[0]
```

```
    def plot(self):
        plt.figure()
        plt.title('Measured')
        plt.subplot(5, 1, 1)
        plt.plot(self.u11)
        plt.ylabel('Input 1')
        plt.subplot(5, 1, 2)
        plt.plot(self.u12)
        plt.ylabel('Input 2')
        plt.subplot(5, 1, 3)
        plt.plot(self.u13)
        plt.ylabel('Input 3')
        plt.subplot(5, 1, 4)
        plt.plot(self.y11)
        plt.ylabel('Output 1')
        plt.subplot(5, 1, 5)
        plt.plot(self.y12)
        plt.ylabel('Output 2')

        plt.xlabel('Data Series')
        plt.show()
```

Kode sumber dapat diunduh di URL:

https://github.com/mekatronik-achmadi/tugas_kuliah/blob/master/Model_Estimation/Python/data_load.py

Penjelasan

Berikut penjelasan per baris kode. Perlu diperhatikan bahwa dalam bahasa Python, penulisan indentasi adalah bagian dari bahasa pemrograman.

```
import numpy as np
import scipy.io as sio
import matplotlib.pyplot as plt
```

Baris kode tersebut untuk meng-*import* modul Python yang dibutuhkan.

```
class DataLoad:
```

Baris kode tersebut menjadi nama class yang dapat dipanggil oleh skrip Python lainnya nanti.

```
def __init__(self,n_start,n_stop):
```

Baris kode tersebut menjadi fungsi inisialisasi class dimana fungsi ini akan dijalankan pertama kali saat class dipanggil dan menjadi objek. Disini variabel n_start dan n_stop menjadi argumen dari fungsi ini.

```
input_data=sio.loadmat("data_input.mat")['input_data']
output_data=sio.loadmat("data_output.mat")['output_data']
```

Baris kode tersebut untuk meng-*import* data dari file *.mat sebagai variable NumPy

array

```
u_11 = input_data[n_start:n_stop,1]
u_12 = input_data[n_start:n_stop,3]
u_13 = input_data[n_start:n_stop,2]
y_11 = output_data[n_start:n_stop,1]
y_12 = output_data[n_start:n_stop,2]
```

Baris kode tersebut untuk mengambil dan memisahkan (*parsing*) 3 input dan 2 output ke variabel masing-masing dengan rentang yang telah ditentukan.

```
self.u11 = u_11/u_11.max()
self.u12 = u_12/u_12.max()
self.u13 = u_13/u_13.max()
self.y11 = y_11/y_11.max()
self.y12 = y_12/y_12.max()
```

Baris kode tersebut untuk melakukan normalisasi sehingga semua variabel memiliki rentang nilai yang sama dan tidak terjadi pengabaian data karena nilai satu variabel yang terlalu kecil bagi variabel lain.

```
self.t = np.arange(0,n_stop-n_start,1)
self.dim = np.shape(self.t)[0]
```

Baris kode tersebut untuk men-*generate* index data dan nilai ukuran array.

```
def plot(self):
```

Baris kode tersebut dan seterusnya adalah fungsi untuk plot semua variabel dalam satu grafik.

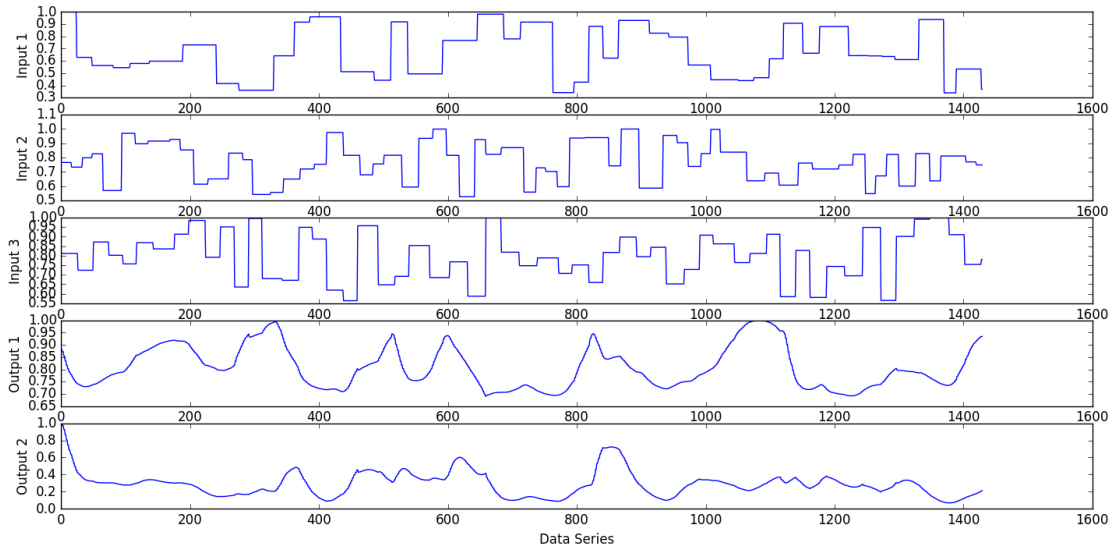
Selanjutnya untuk memanggil class ini dan plot semua data dapat digunakan sebagaimana file di URL:

https://github.com/mekatronik-achmadi/tugas_kuliah/blob/master/Model_Estimation/Python/test_armamle_y11.py

Atau dicontohkan sebagaimana skrip berikut:

```
from data_load import DataLoad

dat_all = DataLoad(20,1450)
dat_all.plot()
```

Simpan skrip tersebut dengan nama sesukanya dan dalam folder yang sama dengan kedua file *.mat dan file data_load.py. Apabila skrip tersebut dijalankan maka akan menghasilkan grafik sebagai berikut:



Data Late

Model matematika yang digunakan nantinya tidak hanya menggunakan variabel *present* disetiap iterasinya, namun juga variabel di satu langkah mundur dan di dua langkah mundur. Untuk itu diperlukan fungsi atau class untuk membuat variabel array yang berisi nilai array satu langkah mundur dan dua langkah mundur untuk setiap variabel array input-output. Skrip class ini diberi nama **data_late.py**, berikut isi class tersebut:

```
import numpy as np

class DataLate:
    def __init__(self):
        pass

    def late(self,data_input,late_order):
        ndim = np.shape(data_input)[0]
        mat_late = np.zeros(ndim)

        for i in range(late_order,ndim):
            mat_late[i] = data_input[i-late_order]

        return mat_late
```

Kode sumber dapat diunduh di URL:

https://github.com/mekatronik-achmadi/tugas_kuliah/blob/master/Model_Estimation/Python/data_late.py

Penjelasan

Berikut penjelasan per baris kode. Perlu diperhatikan bahwa dalam bahasa Python, penulisan indentasi adalah bagian dari bahasa pemrograman.

```
import numpy as np
```

Baris kode tersebut untuk meng-*import* modul Python yang dibutuhkan.

```
class DataLate:
```

Baris kode tersebut menjadi nama class yang dapat dipanggil oleh skrip Python lainnya nanti.

```
def __init__(self):
        pass
```

Baris kode tersebut menjadi fungsi inisialisasi class dimana fungsi ini akan dijalankan pertama kali saat class dipanggil dan menjadi objek. Disini fungsi ini tidak melakukan apa pun.

```
def late(self,data_input,late_order):
        ndim = np.shape(data_input)[0]
        mat_late = np.zeros(ndim)
```

Baris kode tersebut adalah fungsi sebenarnya yang akan menghitung variabel sejauh nilai argumen late_order. Di baris kode tersebut didapatkan ukuran array dari array data_input kemudian membentuk array bernilai 0 sesuai ukuran array data_input.

```
        for i in range(late_order,ndim):
            mat_late[i] = data_input[i-late_order]

        return mat_late
```

Baris tersebut memasukkan nilai data_input pada index seusai late_order ke array mat_late dan hasil akhir dikembalikan ke skrip yang memangkil class ini.

Selanjutnya untuk memanggil class ini dapat digunakan sebagaimana file di URL:

https://github.com/mekatronik-achmadi/tugas_kuliah/blob/master/Model_Estimation/Python/test_armamle_y11.py

Atau dicontohkan sebagaimana skrip berikut:

```
from data_late import DataLate

lt = DataLate()
vx1 = dat_learn.u11
vx1_1 = lt.late(vx1,1)
vx1_2 = lt.late(vx1,2)
```

Variabel array vx1_1 dan vx1_2 adalah masing-masing variabel dengan nilai terlambat satu langkah dan dua langkah terhadap variabel vx1.

Estimasi Model

Seluruh material untuk bagian bab ini dapat ditemukan untuk masing-masing output dalam URL:

https://github.com/mekatronik-achmadi/tugas_kuliah/blob/master/Model_Estimation/Python/test_armamle_y11.py

https://github.com/mekatronik-achmadi/tugas_kuliah/blob/master/Model_Estimation/Python/test_armamle_y12.py

Variabel

Untuk membangun model maka yang pertama dilakukan adalah membangun sistem variabel. Disini variabel yang akan dibangun terbagi menjadi 3 kelompok, yaitu:

- Variabel untuk Model matematika
- Variabel koefisien untuk Model matematika
- Variabel array untuk estimasi

Berikut adalah kode sumber untuk masing-masing kelompok.

Variabel Model

```
import mle

x1 = mle.var('x1', observed=True, vector=True)
x1_1 = mle.var('x1_1', observed=True, vector=True)
x1_2 = mle.var('x1_2', observed=True, vector=True)

x2 = mle.var('x2', observed=True, vector=True)
x2_1 = mle.var('x2_1', observed=True, vector=True)
x2_2 = mle.var('x2_2', observed=True, vector=True)

x3 = mle.var('x3', observed=True, vector=True)
x3_1 = mle.var('x3_1', observed=True, vector=True)
x3_2 = mle.var('x3_2', observed=True, vector=True)

y1 = mle.var('y1', observed=True, vector=True)
y1_1 = mle.var('y1_1', observed=True, vector=True)
y1_2 = mle.var('y1_2', observed=True, vector=True)
```

Kode sumber tersebut membangun variabel untuk satu output dan tiga input masing-masing beserta variabel terlambat satu dan dua langkah.

Variabel Koefisien

```
import mle

a1 = mle.var('a1')
a1_1 = mle.var('a1_1')
a1_2 = mle.var('a1_2')

a2 = mle.var('a2')
a2_1 = mle.var('a2_1')
a2_2 = mle.var('a2_2')

a3 = mle.var('a3')
a3_1 = mle.var('a3_1')
a3_2 = mle.var('a3_2')

b1_1 = mle.var('b1_1')
b1_2 = mle.var('b1_2')

sigma = mle.var('sigma')
```

Kode sumber tersebut membangun variabel untuk satu output dan tiga input masing-masing beserta variabel terlambat satu dan dua langkah. Variabel sigma adalah variabel nilai variansi yang nantinya akan digunakan dalam proses estimasi namun variabel ini tidak termasuk dalam model matematika hasil proses estimasi.

Variabel Array

```
import numpy as np
from data_load import DataLoad
from data_late import DataLate

dat_learn = DataLoad(20,700)
lt = DataLate()

vx1 = dat_learn.u11
vx1_1 = lt.late(vx1,1)
vx1_2 = lt.late(vx1,2)

vx2 = dat_learn.u12
vx2_1 = lt.late(vx2,1)
vx2_2 = lt.late(vx2,2)

vx3 = dat_learn.u13
vx3_1 = lt.late(vx3,1)
vx3_2 = lt.late(vx3,2)

vy1 = dat_learn.y11
vy1_1 = lt.late(vy1,1)
vy1_2 = lt.late(vy1,2)
```

Kode sumber tersebut membangun variabel untuk satu output dan tiga input

masing-masing beserta variabel terlambat satu dan dua langkah. Variabel ini yang akan digunakan untuk proses estimasi.

Model Matematika

Secara matematis model matematika yang akan dibangun dengan input-output terlambat pada satu dan dua langkah adalah sebagai berikut:

$$y_t = b_{11}\, y_{t-1} + b_{12}\, y_{t-2} + a_1\, x_{(1)(t)} + a_{11}\, x_{(1)(t-1)} + a_{12}\, x_{(1)(t-2)} + a_2\, x_{(2)(t)} + a_{21}\, x_{(2)(t-1)} + a_{22}\, x_{(2)(t-2)} + a_3\, x_{(3)(t)} + a_{31}\, x_{(3)(t-1)} + a_{32}\, x_{(3)(t-2)}$$

Proses estimasi nanti adalah mendapatkan semua nilai koefisien berdasarkan data x dan y menggunakan metode estimasi MLE. Model persamaan tersebut apabila di tulis dalam Python adalah sebagai berikut:

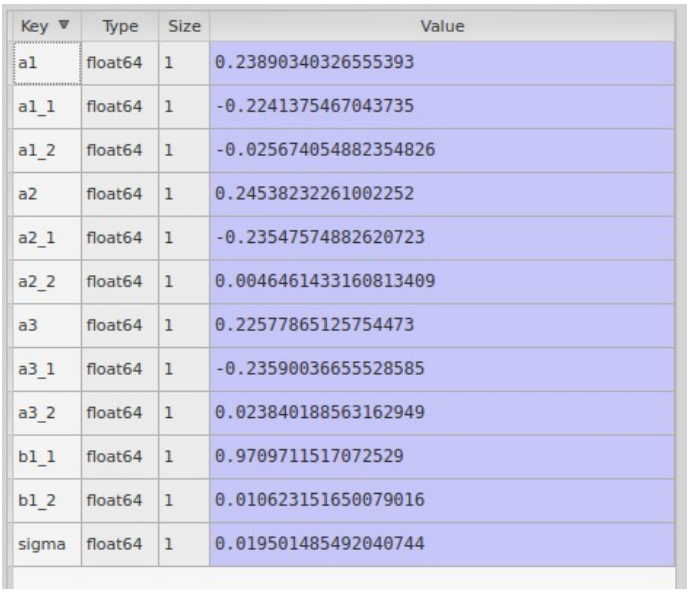
```
import mle

model = mle.Normal(y1,
                   b1_1*y1_1 + b1_2*y1_2 +
                   a1*x1 + a1_1*x1_1 + a1_2*x1_2 +
                   a2*x2 + a2_1*x2_1 + a2_2*x2_2 +
                   a3*x3 + a3_1*x3_1 + a3_2*x3_2,
                   sigma)
```

Estimasi MLE

Langkah selanjutnya adalah melakukan proses estimasi. Variabel model di atas dibangun sebagai objek dari MLE, maka untuk proses estimasi dapat dengan fungsi fit() yang merupakan fungsi member dari objek model dalam MLE. Dalam fungsi fit ini, yang menjadi argumen adalah semua nilai variabel array input-output dan nilai inisial semua koefisien dimana semua bernilai 1.

```
import mle

estim = model.fit({
                    'y1':vy1, 'y1_1':vy1_1, 'y1_2':vy1_2,
                    'x1':vx1, 'x1_1':vx1_1, 'x1_2':vx1_2,
                    'x2':vx2, 'x2_1':vx2_1, 'x2_2':vx2_2,
                    'x3':vx3, 'x3_1':vx3_1, 'x3_2':vx3_2
                    },{
                    'b1_1':1,'b1_2':1,
                    'a1':1,'a1_1':1,'a1_2':1,
                    'a2':1,'a2_1':1,'a2_2':1,
                    'a3':1,'a3_1':1,'a3_2':1,
                    'sigma':1
                  })
estim_coef = estim['x']
```

Selanjutnya akan didapat semua nilai koefisien model dari Variabel Explorer. Berikut hasilnya:



Key	Type	Size	Value
a1	float64	1	0.23890340326555393
a1_1	float64	1	-0.2241375467043735
a1_2	float64	1	-0.025674054882354826
a2	float64	1	0.24538232261002252
a2_1	float64	1	-0.23547574882620723
a2_2	float64	1	0.0046461433160813409
a3	float64	1	0.22577865125754473
a3_1	float64	1	-0.23590036655528585
a3_2	float64	1	0.023840188563162949
b1_1	float64	1	0.9709711517072529
b1_2	float64	1	0.010623151650079016
sigma	float64	1	0.019501485492040744

sehingga didapat model matematis sistem adalah:

$$y_t = 0.97\,y_{t-1} + 0.01\,y_{t-2} + 0.23\,x_{(1)(t)} - 0.22\,x_{(1)(t-1)} - 0.02\,x_{(1)(t-2)} + 0.25\,x_{(2)(t)} - 0.24\,x_{(2)(t-1)} + 0.005\,x_{(2)(t-2)} + 0.23\,x_{(3)(t)} - 0.024\,x_{(3)(t-1)} + 0.024\,x_{(3)(t-2)}$$

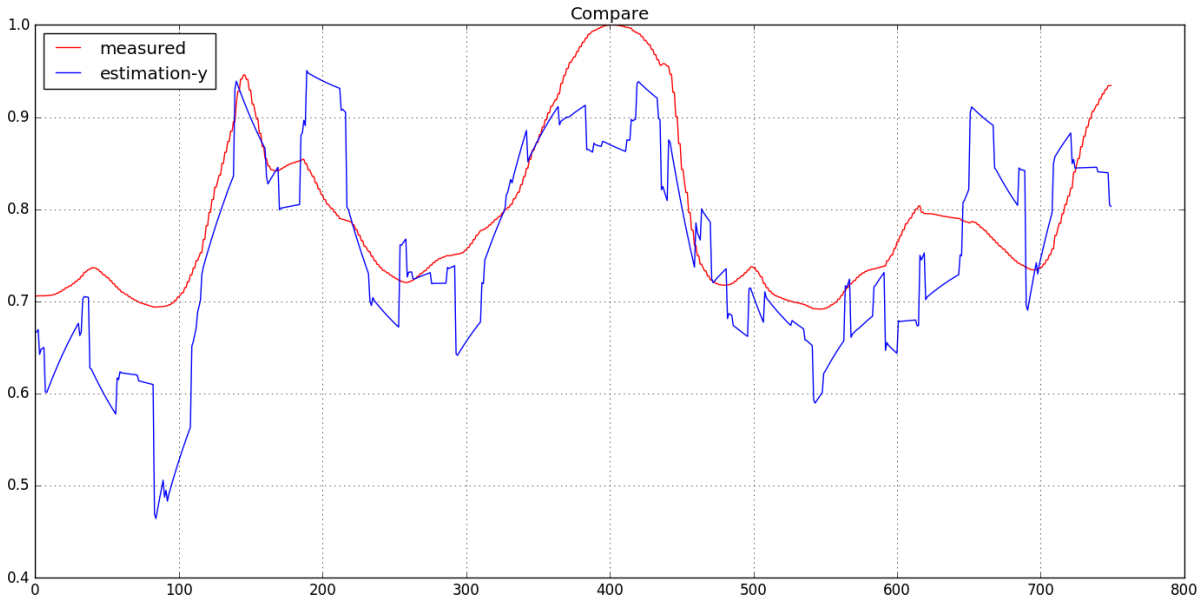
Validasi Model

Langkah selanjutnya adalah melakukan pengujian terhadap model yang didapat terhadap set data yang berbeda dengan set data untuk estimasi. Berikut kode sumber untuk pengambilan set data baru.

```
import numpy as np
from data_late import DataLate

dat_valid = DataLoad(700,1450)
lt = DataLate()

wx1 = dat_valid.u11
wx1_1 = lt.late(wx1,1)
wx1_2 = lt.late(wx1,2)

wx2 = dat_valid.u12
wx2_1 = lt.late(wx2,1)
wx2_2 = lt.late(wx2,2)

wx3 = dat_valid.u13
wx3_1 = lt.late(wx3,1)
wx3_2 = lt.late(wx3,2)
```

Proses iterasi untuk mendapatkan hasil dari model dalam 2 tahap, 1 tahap melakukan perhitungan dengan indeks telah ditentukan sebanyak 3 langkah dan tahap selanjutnya adalah perhitungan dengan looping.

Selanjutnya dengan plot hasil akhir dan variabel array output, dapat dibandingkan tingkat akurasi hasil estimasi.



Perbandingan AR

Untuk membandingkan hasil, maka digunakan model AR (tanpa MA). Berikut bentuk matematisnya:

$$y_t = a_1 x_{(1)(t)} + a_{11} x_{(1)(t-1)} + a_{12} x_{(1)(t-2)} + a_2 x_{(2)(t)} + a_{21} x_{(2)(t-1)} + a_{22} x_{(2)(t-2)} + a_3 x_{(3)(t)} + a_{31} x_{(3)(t-1)} + a_{32} x_{(3)(t-2)}$$

Dalam bentuk Python ditulis sebagai:

```
import mle
model_ar = mle.Normal(y1,
                      a1*x1 + a1_1*x1_1 + a1_2*x1_2 +
                      a2*x2 + a2_1*x2_1 + a2_2*x2_2 +
                      a3*x3 + a3_1*x3_1 + a3_2*x3_2,
                      sigma)
```

dan untuk estimasinya:

```
estim_ar = model_ar.fit({
                      'y1':vy1,
                      'x1':vx1, 'x1_1':vx1_1, 'x1_2':vx1_2,
                      'x2':vx2, 'x2_1':vx2_1, 'x2_2':vx2_2,
                      'x3':vx3, 'x3_1':vx3_1, 'x3_2':vx3_2
                      },{
                      'a1':1,'a1_1':1,'a1_2':1,
                      'a2':1,'a2_1':1,'a2_2':1,
                      'a3':1,'a3_1':1,'a3_2':1,
```

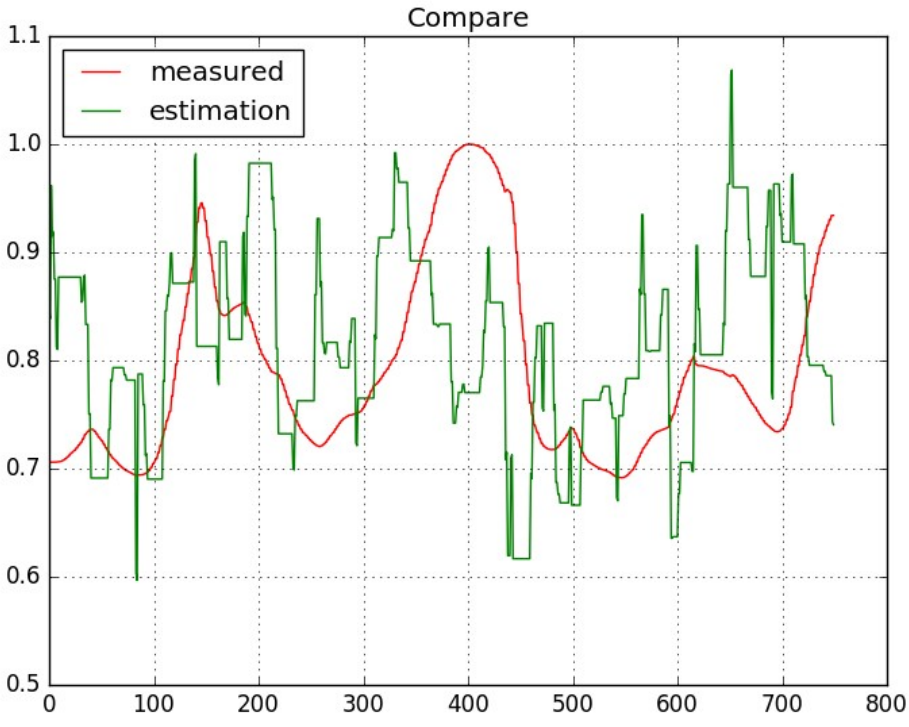
```
                    'sigma':1
                })
estim_coef_ar = estim_ar['x']
```

maka hasil koefisiennya:

Key	Type	Size	Value
a1	float64	1	0.30398173497164249
a1_1	float64	1	0.011722384458460253
a1_2	float64	1	-0.32528052615333014
a2	float64	1	0.26616397231540245
a2_1	float64	1	-0.00087317888178393283
a2_2	float64	1	0.19103480484849017
a3	float64	1	0.32572301828260969
a3_1	float64	1	-0.020903189345624188
a3_2	float64	1	0.29150048181669308
sigma	float64	1	0.12498944887227431

maka hasil persamaannya:

$$y_t = 0.3\,x_{(1)(t)} + 0.1\,x_{(1)(t-1)} - 0.32\,x_{(1)(t-2)} + 0.26\,x_{(2)(t)} - 0.0008\,x_{(2)(t-1)} + 0.19\,x_{(2)(t-2)} + 0.32\,x_{(3)(t)} - 0.021\,x_{(3)(t-1)} + 0.29\,x_{(3)(t-2)}$$



Terlihat bahwa menggunakan ARMA lebih mendekati nilai terukur dengan AR untuk orde polinomial yang sama.

Kode Sumber

Berikut adalah kode sumber Python yang digunakan. Semua file ini dapat di akses di URL:

https://github.com/mekatronik-achmadi/tugas_kuliah/tree/master/Model_Estimation/Python

data_load.py

```
import numpy as np
import scipy.io as sio
import matplotlib.pyplot as plt

class DataLoad:
    def __init__(self,n_start,n_stop):
        input_data=sio.loadmat("data_input.mat")['input_data']
        output_data=sio.loadmat("data_output.mat")['output_data']

        u_11 = input_data[n_start:n_stop,1]
        u_12 = input_data[n_start:n_stop,3]
        u_13 = input_data[n_start:n_stop,2]

        y_11 = output_data[n_start:n_stop,1]
        y_12 = output_data[n_start:n_stop,2]

        self.u11 = u_11/u_11.max()
        self.u12 = u_12/u_12.max()
        self.u13 = u_13/u_13.max()
        self.y11 = y_11/y_11.max()
        self.y12 = y_12/y_12.max()

        self.t = np.arange(0,n_stop-n_start,1)
        self.dim = np.shape(self.t)[0]

    def plot(self):
        plt.figure()
        plt.title('Measured')
        plt.subplot(5, 1, 1)
        plt.plot(self.u11)
        plt.ylabel('Input 1')
        plt.subplot(5, 1, 2)
        plt.plot(self.u12)
        plt.ylabel('Input 2')
        plt.subplot(5, 1, 3)
        plt.plot(self.u13)
        plt.ylabel('Input 3')
        plt.subplot(5, 1, 4)
        plt.plot(self.y11)
        plt.ylabel('Output 1')
        plt.subplot(5, 1, 5)
        plt.plot(self.y12)
        plt.ylabel('Output 2')
```

```
        plt.xlabel('Data Series')
        plt.show()
```

data_late.py

```
import numpy as np

class DataLate:
    def __init__(self):
        pass

    def late(self,data_input,late_order):
        ndim = np.shape(data_input)[0]
        mat_late = np.zeros(ndim)

        for i in range(late_order,ndim):
            mat_late[i] = data_input[i-late_order]

        return mat_late
```

test_armamle_y11.py

```
import numpy as np
import matplotlib.pyplot as plt
import mle

from data_load import DataLoad
from data_late import DataLate

dat_all = DataLoad(20,1450)
dat_all.plot()

dat_learn = DataLoad(20,700)

dat_valid = DataLoad(700,1450)

x1 = mle.var('x1', observed=True, vector=True)
x1_1 = mle.var('x1_1', observed=True, vector=True)
x1_2 = mle.var('x1_2', observed=True, vector=True)

x2 = mle.var('x2', observed=True, vector=True)
x2_1 = mle.var('x2_1', observed=True, vector=True)
x2_2 = mle.var('x2_2', observed=True, vector=True)

x3 = mle.var('x3', observed=True, vector=True)
x3_1 = mle.var('x3_1', observed=True, vector=True)
x3_2 = mle.var('x3_2', observed=True, vector=True)

y1 = mle.var('y1', observed=True, vector=True)
y1_1 = mle.var('y1_1', observed=True, vector=True)
y1_2 = mle.var('y1_2', observed=True, vector=True)

a1 = mle.var('a1')
a1_1 = mle.var('a1_1')
```

```
a1_2 = mle.var('a1_2')

a2 = mle.var('a2')
a2_1 = mle.var('a2_1')
a2_2 = mle.var('a2_2')

a3 = mle.var('a3')
a3_1 = mle.var('a3_1')
a3_2 = mle.var('a3_2')

b1_1 = mle.var('b1_1')
b1_2 = mle.var('b1_2')

sigma = mle.var('sigma')

lt = DataLate()

vx1 = dat_learn.u11
vx1_1 = lt.late(vx1,1)
vx1_2 = lt.late(vx1,2)

vx2 = dat_learn.u12
vx2_1 = lt.late(vx2,1)
vx2_2 = lt.late(vx2,2)

vx3 = dat_learn.u13
vx3_1 = lt.late(vx3,1)
vx3_2 = lt.late(vx3,2)

vy1 = dat_learn.y11
vy1_1 = lt.late(vy1,1)
vy1_2 = lt.late(vy1,2)

model = mle.Normal(y1,
                   b1_1*y1_1 + b1_2*y1_2 +
                   a1*x1 + a1_1*x1_1 + a1_2*x1_2 +
                   a2*x2 + a2_1*x2_1 + a2_2*x2_2 +
                   a3*x3 + a3_1*x3_1 + a3_2*x3_2,
                   sigma)

estim = model.fit({
                    'y1':vy1, 'y1_1':vy1_1, 'y1_2':vy1_2,
                    'x1':vx1, 'x1_1':vx1_1, 'x1_2':vx1_2,
                    'x2':vx2, 'x2_1':vx2_1, 'x2_2':vx2_2,
                    'x3':vx3, 'x3_1':vx3_1, 'x3_2':vx3_2
                    },{
                    'b1_1':1,'b1_2':1,
                    'a1':1,'a1_1':1,'a1_2':1,
                    'a2':1,'a2_1':1,'a2_2':1,
                    'a3':1,'a3_1':1,'a3_2':1,
                    'sigma':1
                  })
print(estim)
estim_coef = estim['x']
```

```
wx1 = dat_valid.u11
wx1_1 = lt.late(wx1,1)
wx1_2 = lt.late(wx1,2)

wx2 = dat_valid.u12
wx2_1 = lt.late(wx2,1)
wx2_2 = lt.late(wx2,2)

wx3 = dat_valid.u13
wx3_1 = lt.late(wx3,1)
wx3_2 = lt.late(wx3,2)

measu_y = dat_valid.y11
estim_yy = np.zeros(dat_valid.dim)

estim_yy[0] = estim_coef['b1_1']*estim_yy[0] + estim_coef['b1_2']*estim_yy[0] +
estim_coef['a1']*wx1[0] + estim_coef['a1_1']*wx1_1[0] + estim_coef['a1_2']*wx1_2[0]
+ estim_coef['a2']*wx2[0] + estim_coef['a2_1']*wx2_1[0] +
estim_coef['a2_2']*wx2_2[0] + estim_coef['a3']*wx3[0] + estim_coef['a3_1']*wx3_1[0]
+ estim_coef['a3_2']*wx3_2[0]

estim_yy[1] = estim_coef['b1_1']*estim_yy[0] + estim_coef['b1_2']*estim_yy[0] +
estim_coef['a1']*wx1[1] + estim_coef['a1_1']*wx1_1[1] + estim_coef['a1_2']*wx1_2[1]
+ estim_coef['a2']*wx2[1] + estim_coef['a2_1']*wx2_1[1] +
estim_coef['a2_2']*wx2_2[1] + estim_coef['a3']*wx3[1] + estim_coef['a3_1']*wx3_1[1]
+ estim_coef['a3_2']*wx3_2[1]

estim_yy[2] = estim_coef['b1_1']*estim_yy[1] + estim_coef['b1_2']*estim_yy[0] +
estim_coef['a1']*wx1[2] + estim_coef['a1_1']*wx1_1[2] + estim_coef['a1_2']*wx1_2[2]
+ estim_coef['a2']*wx2[2] + estim_coef['a2_1']*wx2_1[2] +
estim_coef['a2_2']*wx2_2[2] + estim_coef['a3']*wx3[2] + estim_coef['a3_1']*wx3_1[2]
+ estim_coef['a3_2']*wx3_2[2]

for i in range(3,dat_valid.dim):
    estim_yy[i] = estim_coef['b1_1']*estim_yy[i-1] + estim_coef['b1_2']*estim_yy[i-
2] + estim_coef['a1']*wx1[i] + estim_coef['a1_1']*wx1_1[i] +
estim_coef['a1_2']*wx1_2[i] + estim_coef['a2']*wx2[i] + estim_coef['a2_1']*wx2_1[i]
+ estim_coef['a2_2']*wx2_2[i] + estim_coef['a3']*wx3[i] +
estim_coef['a3_1']*wx3_1[i] + estim_coef['a3_2']*wx3_2[i]

plt.figure()
plt.title('Compare')
plt.plot(dat_valid.t,measu_y, 'r', label='measured')
plt.plot(dat_valid.t,estim_yy, 'b', label='estimation-y')
plt.legend(loc='best')
plt.grid()
plt.show()
```