LAMPIRAN

Lampiran 1 Penyelesaian model GP untuk Portofolio 2 (P2)

```
#Sintaks Skripsi Gilang Juniar Azmi_G5401211029
#Portofolio 2 (P2)

from pyomo.environ import *

#Model
model = ConcreteModel()

#Indeks i, j, serta deviasi
model.M = RangeSet(1,34) #indeks i dan j
model.N = RangeSet(1,3) #variabel deviasi

#Variabel keputusan
model.w = Var(model.M, within=NonNegativeReals)
model.d_pos = Var(model.N, within=NonNegativeReals)
model.d_neg = Var(model.N, within=NonNegativeReals)

#Data Parameter
ekspektasi_return = [0.0078820, 0.0015885, 0.0018701, 0.0000454,
      0.0007490, … , 0.0008625, 0.0004547]
dividend_yield = [0.0709742, 0.0000000, 0.0994800, 0.0342685,
      0.0000000, … , 0.0145833, 0.0000000]
risiko = [
    [0.0023818, 0.0000035, 0.0001803, -0.0000754, 0.0000124,
    0.0000841, … , 0.0013119],[…],…,[…]]

#Parameter
model.e = Param(model.M, initialize=lambda model, i:
ekspektasi_return[i-1])
model.d = Param(model.M, initialize=lambda model, i:
dividend_yield[i-1])
model.v = Param(model.M,model.M, initialize=lambda model, i,j:
risiko[i-1][j-1])

#Fungsi objektif
model.objective = Objective(expr=model.d_neg[1] + model.d_neg[2]
+ model.d_pos[3], sense=minimize)

#Kendala
def rule_const1(model):
    return sum(model.w[i] * model.e[i] for i in model.M) -
model.d_pos[1] + model.d_neg[1] == 0.001752
model.const1 = Constraint(rule=rule_const1)

def rule_const2(model):
    return sum(model.w[i] * model.d[i] for i in model.M) -
model.d_pos[2] + model.d_neg[2] == 0.043231
model.const2 = Constraint(rule=rule_const2)

def rule_const3(model):
    return sum(model.w[i] * model.w[j] * model.v[i,j] for i in
model.M for j in model.M) - model.d_pos[3] + model.d_neg[3] ==
0.000088
model.const3 = Constraint(rule=rule_const3)
```

```
def rule_const4(model):
    return sum(model.w[i] for i in model.M) == 1
model.const4 = Constraint(rule=rule_const4)

#Kendala Produk Deviasi Nol
def zero_product_rule(model,N):
    return model.d_pos[N] * model.d_neg[N] == 0
model.zero_product = Constraint(model.N, rule=zero_product_rule)

#Solver
result = SolverFactory('ipopt').solve(model)

#Display output nilai variabel keputusan
print('Nilai fungsi objektif:',model.objective())

#Display variabel keputusan
L1 = list(model.w.keys())
for i in L1:
    print('w',i, '--', model.w[i]())

L2 = list(model.d_pos.keys())
for i in L2:
    print('d_pos',i, '--', model.d_pos[i]())

L3 = list(model.d_neg.keys())
for i in L3:
    print('d_neg',i, '--', model.d_neg[i]())

#Display f1, f2, f3:
print('f1:',  value(sum(model.w[i]  *  model.e[i]  for  i  in
model.M)))
print('f2:',  value(sum(model.w[i]  *  model.d[i]  for  i  in
model.M)))
print('f3:', value(sum(model.w[i] * model.w[j] * model.v[i,j] for
i in model.M for j in model.M)))

#Output:
=  RESTART:  C:/Users/Gilang  Juniar/Documents/Folder  Penting
Departemen Matematika/Semester 7/Karya Ilmiah/Sintaks Portofolio
2.py
Nilai fungsi objektif: 0.0
w 1 -- 0.04058448740316664    w 18 -- 0.04598146548800533
w 2 -- 0.01896367931353353    w 19 -- 0.018750486997532394
w 3 -- 0.03292465646262546    w 20 -- 0.02560976675249379
w 4 -- 0.024245133071600627   w 21 -- 0.042301623003261145
w 5 -- 0.027378352583671387   w 22 -- 0.026463274517248253
w 6 -- 0.025565644001197842   w 23 -- 0.01908146228525968
w 7 -- 0.035982202198964613   w 24 -- 0.05727641165743717
w 8 -- 0.03611053149112539    w 25 -- 0.029932633301632143
w 9 -- 0.018763742267742575   w 26 -- 0.02544346593489553
w 10 -- 0.029765009235385036  w 27 -- 0.03425761349371137
w 11 -- 0.020962056905304907  w 28 -- 0.022680697273230972
w 12 -- 0.02034099671627234   w 29 -- 0.03988374430597316
w 13 -- 0.034004625145881165  w 30 -- 0.030094062467652075
w 14 -- 0.028240991201553965  w 31 -- 0.03757485723211975
w 15 -- 0.02368900561753114   w 32 -- 0.03029032753365052
w 16 -- 0.028223085334141293  w 33 -- 0.023709305877477566
w 17 -- 0.028147830990913815  w 34 -- 0.016776771937807335
```

```
d_pos 1 -- 0.00028339951971154733
d_pos 2 -- 0.0075311260414968415
d_pos 3 -- 0.0
d_neg 1 -- 0.0
d_neg 2 -- 0.0
d_neg 3 -- 1.2257871972661078e-05
f1: 0.0020353995197115475
f2: 0.050762126041496845
f3: 7.574212802733902e-05
```

Lampiran 2 Penyelesaian model FGP untuk Portofolio 3 (P3)

```
#Sintaks Skripsi Gilang Juniar Azmi_G5401211029
#Portofolio 3 (P3)

from pyomo.environ import *

#Model
model = ConcreteModel()

#Indeks i dan j
model.M = RangeSet(1,34) #indeks i dan j

#Variabel keputusan
model.w = Var(model.M, within=NonNegativeReals)
model.lmbda = Var(within=NonNegativeReals, bounds=(0,1))

#Data Parameter
ekspektasi_return = [0.0078820, 0.0015885, 0.0018701, 0.0000454,
      0.0007490, … , 0.0008625, 0.0004547]
dividend_yield = [0.0709742, 0.0000000, 0.0994800, 0.0342685,
      0.0000000, … , 0.0145833, 0.0000000]
risiko = [
    [0.0023818, 0.0000035, 0.0001803, -0.0000754, 0.0000124,
    0.0000841, … , 0.0013119],[…],…,[…]]

#Parameter
model.e = Param(model.M, initialize=lambda model, i:
ekspektasi_return[i-1])
model.d = Param(model.M, initialize=lambda model, i:
dividend_yield[i-1])
model.v = Param(model.M,model.M, initialize=lambda model, i,j:
risiko[i-1][j-1])

#Fungsi objektif
model.objective = Objective(expr=model.lmbda, sense=maximize)

#Kendala
def rule_const1(model):
    return sum(model.w[i] * model.e[i] for i in model.M) -
(0.003504 * model.lmbda) >= 0.001752
model.const1 = Constraint(rule=rule_const1)

def rule_const2(model):
    return sum(model.w[i] * model.d[i] for i in model.M) -
(0.043231 * model.lmbda) >= 0.043231
model.const2 = Constraint(rule=rule_const2)
```

```
def rule_const3(model):
    return sum(model.w[i] * model.w[j] * model.v[i,j] for i in
model.M for j in model.M) + (0.000044 * model.lmbda) <= 0.000088
model.const3 = Constraint(rule=rule_const3)

def rule_const4(model):
    return sum(model.w[i] for i in model.M) == 1
model.const4 = Constraint(rule=rule_const4)

#Solver
result = SolverFactory('ipopt').solve(model)

#Display output nilai variabel keputusan
print('Nilai fungsi objektif:',model.objective())

#Display variabel keputusan
L1 = list(model.w.keys())
for i in L1:
    print('w',i, '--', model.w[i]())

#Display f1, f2, f3:
print('f1:',  value(sum(model.w[i]  *  model.e[i]  for  i  in
model.M)))
print('f2:',  value(sum(model.w[i]  *  model.d[i]  for  i  in
model.M)))
print('f3:', value(sum(model.w[i] * model.w[j] * model.v[i,j] for
i in model.M for j in model.M)))

#Output:
=  RESTART:  C:\Users\Gilang  Juniar\Documents\Folder  Penting
Departemen Matematika\Semester 7\Karya Ilmiah\Sintaks Portofolio
3.py
Nilai fungsi objektif: 0.39404904394738094
w 1 -- 0.07404603985962764       w 18 -- 0.09106058114303639
w 2 -- 0.0                       w 19 -- 3.604490413988201e-10
w 3 -- 0.012907609210300729      w 20 -- 0.013941666070068795
w 4 -- 1.1753261265514713e-08    w 21 -- 0.02937148126198716
w 5 -- 3.112530748783932e-07     w 22 -- 0.0
w 6 -- 0.03229308414202354       w 23 -- 0.0
w 7 -- 4.7585230165896034e-08    w 24 -- 0.25269480438765196
w 8 -- 0.055223137415623874      w 25 -- 7.826449760708463e-08
w 9 -- 0.0                       w 26 -- 0.016117137687821945
w 10 -- 0.0324331453745308       w 27 -- 0.07121237896627636
w 11 -- 0.0                      w 28 -- 1.0058035690404787e-09
w 12 -- 0.0                      w 29 -- 0.11778460789086988
w 13 -- 1.0649820281604734e-08   w 30 -- 0.05917430513234006
w 14 -- 2.7499619173435656e-08   w 31 -- 0.05709003753983974
w 15 -- 0.0                      w 32 -- 0.05062717519768954
w 16 -- 0.0                      w 33 -- 0.007818115133983304
w 17 -- 0.02620423648782136      w 34 -- 0.0
lambda: 0.39404904394738094
f1: 0.0031327378940477313
f2: 0.06026612619859115
f3: 7.067184691439171e-05
```

Lampiran 3 Penyelesaian model FGP untuk Skenario 1 (S1)

```
#Sintaks Skripsi Gilang Juniar Azmi_G5401211029
#Skenario 1 (S1)

from pyomo.environ import *

#Model
model = ConcreteModel()

#Indeks i dan j
model.M = RangeSet(1,34) #indeks i dan j

#Variabel keputusan
model.w = Var(model.M, within=NonNegativeReals)
model.lmbda = Var(within=NonNegativeReals, bounds=(0,1))

#Data Parameter
ekspektasi_return = [0.0078820, 0.0015885, 0.0018701, 0.0000454,
      0.0007490, … , 0.0008625, 0.0004547]
dividend_yield  =  [0.0709742,  0.0000000,  0.0994800,  0.0342685,
      0.0000000, … , 0.0145833, 0.0000000]
risiko = [
    [0.0023818,  0.0000035,  0.0001803,  -0.0000754,  0.0000124,
    0.0000841, … , 0.0013119],[…],…,[…]]

#Parameter
model.e    =    Param(model.M,    initialize=lambda    model,    i:
ekspektasi_return[i-1])
model.d    =    Param(model.M,    initialize=lambda    model,    i:
dividend_yield[i-1])
model.v  =  Param(model.M,model.M,  initialize=lambda  model,  i,j:
risiko[i-1][j-1])

#Fungsi objektif
model.objective = Objective(expr=model.lmbda, sense=maximize)

#Kendala
def rule_const1(model):
    return  sum(model.w[i]  *  model.e[i]  for  i  in  model.M)  -
(0.001752 * model.lmbda) >= 0.001752
model.const1 = Constraint(rule=rule_const1)

def rule_const2(model):
    return  sum(model.w[i]  *  model.d[i]  for  i  in  model.M)  -
(0.021615 * model.lmbda) >= 0.043231
model.const2 = Constraint(rule=rule_const2)

def rule_const3(model):
    return sum(model.w[i] * model.w[j] * model.v[i,j] for i in
model.M for j in model.M) + (0.000022 * model.lmbda) <= 0.000088
model.const3 = Constraint(rule=rule_const3)

def rule_const4(model):
    return sum(model.w[i] for i in model.M) == 1
model.const4 = Constraint(rule=rule_const4)

#Solver
result = SolverFactory('ipopt').solve(model)
```

```
#Display output nilai variabel keputusan
print('Nilai fungsi objektif:',model.objective())

#Display variabel keputusan
L1 = list(model.w.keys())
for i in L1:
    print('w',i, '--', model.w[i]())

#Display f1, f2, f3:
print('f1:',  value(sum(model.w[i]  *  model.e[i]  for  i  in
model.M)))
print('f2:',  value(sum(model.w[i]  *  model.d[i]  for  i  in
model.M)))
print('f3:', value(sum(model.w[i] * model.w[j] * model.v[i,j] for
i in model.M for j in model.M)))

#Output:
=  RESTART:  C:\Users\Gilang  Juniar\Documents\Folder  Penting
Departemen Matematika\Semester 7\Karya Ilmiah\Skenario 1.py
Nilai fungsi objektif: 0.7880996881326815
w 1 -- 0.07404596594953122       w 18 -- 0.09106071899322316
w 2 -- 0.0                       w 19 -- 0.0
w 3 -- 0.012907551714623313      w 20 -- 0.013941981699524182
w 4 -- 8.353384867922559e-10     w 21 -- 0.029371278417249038
w 5 -- 3.041396830771353e-08     w 22 -- 0.0
w 6 -- 0.032293517030416594      w 23 -- 0.0
w 7 -- 1.8852957538998927e-08    w 24 -- 0.25269293623609634
w 8 -- 0.05522272527653708       w 25 -- 2.559785518986113e-08
w 9 -- 0.0                       w 26 -- 0.016117381448551255
w 10 -- 0.03243316067278385      w 27 -- 0.07121247633545306
w 11 -- 0.0                      w 28 -- 0.0
w 12 -- 0.0                      w 29 -- 0.11778497381006264
w 13 -- 3.2461561803672734e-10   w 30 -- 0.05917468177084693
w 14 -- 8.589184298509502e-09    w 31 -- 0.057090152947426705
w 15 -- 0.0                      w 32 -- 0.050627277870712126
w 16 -- 0.0                      w 33 -- 0.007818258874277688
w 17 -- 0.026204946298492408     w 34 -- 0.0
lambda: 0.7880996881326815
f1: 0.0031327407209050047
f2: 0.060265767117543015
f3: 7.067181513843025e-05
```

Lampiran 4 Penyelesaian model FGP untuk Skenario 2 (S2)

```
#Sintaks Skripsi Gilang Juniar Azmi_G5401211029
#Skenario 2 (S2)

from pyomo.environ import *

#Model
model = ConcreteModel()

#Indeks i dan j
model.M = RangeSet(1,14) #indeks i dan j

#Variabel keputusan
model.w = Var(model.M, within=NonNegativeReals)
model.lmbda = Var(within=NonNegativeReals, bounds=(0,1))
```

```
#Data Parameter
ekspektasi_return = [0.0078820, 0.0015885, 0.0018701, 0.0000454,
      0.0007490, … , 0.0008625, 0.0004547]
dividend_yield  =  [0.0709742,  0.0000000,  0.0994800,  0.0342685,
      0.0000000, … , 0.0145833, 0.0000000]
risiko = [
    [0.0023818,  0.0000035,  0.0001803,  -0.0000754,  0.0000124,
    0.0000841, … , 0.0013119],[…],…,[…]]

#Parameter
model.e    =    Param(model.M,    initialize=lambda    model,    i:
ekspektasi_return[i-1])
model.d    =    Param(model.M,    initialize=lambda    model,    i:
dividend_yield[i-1])
model.v  =  Param(model.M,model.M,  initialize=lambda  model,  i,j:
risiko[i-1][j-1])

#Fungsi objektif
model.objective = Objective(expr=model.lmbda, sense=maximize)

#Kendala
def rule_const1(model):
    return  sum(model.w[i]  *  model.e[i]  for  i  in  model.M)  -
(0.006266 * model.lmbda) >= 0.003133
model.const1 = Constraint(rule=rule_const1)

def rule_const2(model):
    return  sum(model.w[i]  *  model.d[i]  for  i  in  model.M)  -
(0.060266 * model.lmbda) >= 0.060266
model.const2 = Constraint(rule=rule_const2)

def rule_const3(model):
    return sum(model.w[i] * model.w[j] * model.v[i,j] for i in
model.M for j in model.M) + (0.000036 * model.lmbda) <= 0.000071
model.const3 = Constraint(rule=rule_const3)

def rule_const4(model):
    return sum(model.w[i] for i in model.M) == 1
model.const4 = Constraint(rule=rule_const4)

#Solver
result = SolverFactory('ipopt').solve(model)

#Display output nilai variabel keputusan
print('Nilai fungsi objektif:',model.objective())

#Display variabel keputusan
L1 = list(model.w.keys())
for i in L1:
    print('w',i, '--', model.w[i]())

#Display f1, f2, f3:
print('f1:',  value(sum(model.w[i]  *  model.e[i]  for  i  in
model.M)))
print('f2:',  value(sum(model.w[i]  *  model.d[i]  for  i  in
model.M)))
print('f3:', value(sum(model.w[i] * model.w[j] * model.v[i,j] for
i in model.M for j in model.M)))
```

```
#Output:
=   RESTART:   C:\Users\Gilang   Juniar\Documents\Folder   Penting
Departemen Matematika\Semester 7\Karya Ilmiah\Skenario 2.py
Nilai fungsi objektif: 0.0010491900663560725
w 1 -- 0.07435392746487528        w 18 -- 0.09135128160843717
w 2 -- 2.8886456315287035e-09     w 19 -- 5.922299804641196e-09
w 3 -- 0.012976198346255992       w 20 -- 0.013753710041071298
w 4 -- 2.3120479909907702e-08     w 21 -- 0.029549749403886518
w 5 -- 9.923062602006596e-08      w 22 -- 2.3249205886877133e-09
w 6 -- 0.032189626820156435       w 23 -- 0.0
w 7 -- 8.027724272510385e-08      w 24 -- 0.2527393863956589
w 8 -- 0.05544304367200361        w 25 -- 9.637310455684612e-08
w 9 -- 0.0                        w 26 -- 0.01586725677335831
w 10 -- 0.032520834449191606      w 27 -- 0.07133826175143118
w 11 -- 3.2915917832075383e-09    w 28 -- 6.848696916336825e-09
w 12 -- 0.0                       w 29 -- 0.11791323897829803
w 13 -- 3.2915917832075383e-09    w 30 -- 0.05918559968420058
w 14 -- 4.6646739312563615e-08    w 31 -- 0.057192441258512874
w 15 -- 2.416747741147049e-10     w 32 -- 0.05068195496341431
w 16 -- 1.638928335447936e-09     w 33 -- 0.007725412030187374
w 17 -- 0.02521769551113791       w 34 -- 0.0
lambda: 0.0010491900663560725
f1: 0.0031395642538585575
f2: 0.0603292224588987
f3: 7.097223010680596e-05
```

Lampiran 5 Penyelesaian model FGP untuk Skenario 3 (S3)

```
#Sintaks Skripsi Gilang Juniar Azmi_G5401211029
#Skenario 3 (S3)

from pyomo.environ import *

#Model
model = ConcreteModel()

#Indeks i dan j
model.M = RangeSet(1,14) #indeks i dan j

#Variabel keputusan
model.w = Var(model.M, within=NonNegativeReals)
model.lmbda = Var(within=NonNegativeReals, bounds=(0,1))

#Data Parameter
ekspektasi_return = [0.0078820, 0.0015885, 0.0000454, 0.0005355,
      0.0016066,  0.0032845,  0.0001520,  0.0004802,  0.0002109,
      0.0075363, 0.0030822, 0.0006463, 0.0000514, 0.0004548]
dividend_yield  =  [0.070974,  0.000000,  0.034268,  0.154751,
      0.103352, 0.027737, 0.114632, 0.121777, 0.015094, 0.000000,
      0.153228, 0.158574, 0.001238, 0.000000]
risiko = [
    [0.0023818,  0.0000035,  -0.0000754,  0.0000471,  -0.0000331,
    0.0001412, … , 0.0013119],[…],…,[…]]
```

```
#Parameter
model.e    =    Param(model.M,    initialize=lambda    model,    i:
ekspektasi_return[i-1])
model.d    =    Param(model.M,    initialize=lambda    model,    i:
dividend_yield[i-1])
model.v  =  Param(model.M,model.M,  initialize=lambda  model,  i,j:
risiko[i-1][j-1])

#Fungsi objektif
model.objective = Objective(expr=model.lmbda, sense=maximize)

#Kendala
def rule_const1(model):
    return  sum(model.w[i]  *  model.e[i]  for  i  in  model.M)  -
(0.003936 * model.lmbda) >= 0.001968
model.const1 = Constraint(rule=rule_const1)

def rule_const2(model):
    return  sum(model.w[i]  *  model.d[i]  for  i  in  model.M)  -
(0.063839 * model.lmbda) >= 0.063839
model.const2 = Constraint(rule=rule_const2)

def rule_const3(model):
    return sum(model.w[i] * model.w[j] * model.v[i,j] for i in
model.M for j in model.M) + (0.000069 * model.lmbda) <= 0.000137
model.const3 = Constraint(rule=rule_const3)

def rule_const4(model):
    return sum(model.w[i] for i in model.M) == 1
model.const4 = Constraint(rule=rule_const4)

#Solver
result = SolverFactory('ipopt').solve(model)

#Display output nilai variabel keputusan
print('Nilai fungsi objektif:',model.objective())

#Display variabel keputusan
L1 = list(model.w.keys())
for i in L1:
    print('w',i, '--', model.w[i]())

#Display f1, f2, f3:
print('f1:',   value(sum(model.w[i]   *   model.e[i]   for   i   in
model.M)))
print('f2:',   value(sum(model.w[i]   *   model.d[i]   for   i   in
model.M)))
print('f3:', value(sum(model.w[i] * model.w[j] * model.v[i,j] for
i in model.M for j in model.M)))

#Output:
=   RESTART:   C:\Users\Gilang   Juniar\Documents\Folder   Penting
Departemen Matematika\Semester 7\Karya Ilmiah\Skenario 3.py
```

```
Nilai fungsi objektif: 0.21928943501834666
w 1 -- 0.11417864193063139        w 8 - 0.0
w 2 -- 0.03931818160022426        w 9 -- 0.13221299394564473
w 3 -- 3.211637703635788e-06      w 10 -- 0.14530975372059782
w 4 -- 0.0                        w 11 -- 0.040595724640641O4
w 5 -- 0.1540065408263872         w 12 -- 0.3064888317289419
w 6 -- 0.053083641668219286       w 13 -- 0.014802485065734403
w 7 -- 5.057004610693733e-09      w 14 -- 0.0
lambda: 0.21928943501834666
f1: 0.002831113226290898
f2: 0.0823286006498685
f3: 0.00012187903228543664
```

Lampiran 6 Pembagian dividen per saham

i	Kode Saham	DPS_i	Harga Saham i	i	Kode Saham	DPS_i	Harga Saham i
1	PTRO	49.54	698	18	DSSA	0.00	26827
2	BUMI	0.00	102	19	BIPI	0.00	76
3	KKGI	49.74	500	20	WINS	8.00	500
4	MEDC	29.32	1310	21	GEMS	232.21	6000
		15.75	1325			410.42	6500
5	HITS	0.00	353			397.27	13500
6	DOID	10.64	580			243.73	11100
7	PTBA	397.71	2570	22	ABMM	295.00	3560
8	PGAS	148.31	1435	23	TOBA	0.00	354
9	ENRG	0.00	219	24	BSSR	118.39	3790
10	RAJA	38.00	1370			345.15	3960
11	IATA	0.00	46			179.19	4460
12	DEWA	0.00	73	25	TPMA	45.00	583
13	ITMG	1747.00	25850	26	SHIP	20.00	955
		1228.00	26100	27	DWGL	0.00	130
14	ELSA	27.57	416	28	TCPI	10.00	8075
15	INDY	92.13	1355	29	SURE	0.00	2139
16	ADRO	209.31	2690	30	BESS	0.00	200
		106.84	2430	31	UNIQ	3.98	560
17	BYAN	146.39	18650	32	SUNI	4.40	464
		144.90	20000	33	HILL	35.00	2400
				34	CUAN	0.00	7945

Lampiran 7 Ekspektasi *return* saham *blue chip*

i	Kode Saham	$\bar{R}_i$	i	Kode Saham	$\bar{R}_i$
1	PTRO	0.007882	8	ADRO	0.000480
2	BUMI	0.001588	9	BYAN	0.000211
3	MEDC	0.000045	10	DSSA	0.007536
4	PTBA	0.000536	11	GEMS	0.003082
5	PGAS	0.001607	12	BSSR	0.000646
6	RAJA	0.003284	13	TCPI	0.000051
7	ITMG	0.000152	14	CUAN	0.000455

Lampiran 8 *Dividend yield* saham *blue chip*

i	Kode Saham	D_i	i	Kode Saham	D_i
1	PTRO	0.070974	8	ADRO	0.121777
2	BUMI	0.000000	9	BYAN	0.015094
3	MEDC	0.034268	10	DSSA	0.000000
4	PTBA	0.154751	11	GEMS	0.153228
5	PGAS	0.103352	12	BSSR	0.158574
6	RAJA	0.027737	13	TCPI	0.001238
7	ITMG	0.114632	14	CUAN	0.000000

Lampiran 9 Tabel kovarian saham *blue chip*

σ_{ij}	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7	σ_8
σ_1	0.0023818	0.0000035	-0.0000754	0.0000471	-0.0000331	0.0001412	-0.0000132	-0.0000241
σ_2	0.0000035	0.0014079	0.0000743	0.0001796	0.0000896	0.0000882	0.0001157	0.0001086
σ_3	-0.0000754	0.0000743	0.0005847	0.0001168	0.0000860	0.0000389	0.0000680	0.0001174
σ_4	0.0000471	0.0001796	0.0001168	0.0003622	0.0001163	0.0000675	0.0001125	0.0001770
σ_5	-0.0000331	0.0000896	0.0000860	0.0001163	0.0003940	0.0001265	0.0000677	0.0001057
σ_6	0.0001412	0.0000882	0.0000389	0.0000675	0.0001265	0.0017086	0.0000899	0.0001135
σ_7	-0.0000132	0.0001157	0.0000680	0.0001125	0.0000677	0.0000899	0.0001998	0.0001433
σ_8	-0.0000241	0.0001086	0.0001174	0.0001770	0.0001057	0.0001135	0.0001433	0.0010823
σ_9	0.0000381	-0.0000136	0.0000080	0.0000116	-0.0000067	0.0000777	0.0000204	0.0000175
σ_{10}	-0.0001016	-0.0000427	0.0000561	0.0000116	0.0000342	-0.0000029	0.0000711	0.0000847
σ_{11}	0.0003594	0.0000615	0.0000680	0.0001375	0.0000330	0.0000456	0.0000795	0.0001418
σ_{12}	0.0000027	0.0000373	0.0000408	0.0000671	0.0000319	-0.0000184	0.0000384	0.0000793
σ_{13}	0.0000791	0.0001677	0.0000019	0.0000931	0.0000594	0.0000242	0.0000416	-0.0000868
σ_{14}	0.0013119	0.0000595	-0.0000692	0.0001241	0.0000265	-0.0000330	0.0000211	-0.0000046

σ_{ij}	σ_9	σ_{10}	σ_{11}	σ_{12}	σ_{13}	σ_{14}
σ_1	0.0000381	-0.0001016	0.0003594	0.0000027	0.0000791	0.0013119
σ_2	-0.0000136	-0.0000427	0.0000615	0.0000373	0.0001677	0.0000595
σ_3	0.0000080	0.0000561	0.0000680	0.0000408	0.0000019	-0.0000692
σ_4	0.0000116	0.0000116	0.0001375	0.0000671	0.0000931	0.0001241
σ_5	-0.0000067	0.0000342	0.0000330	0.0000319	0.0000594	0.0000265
σ_6	0.0000777	-0.0000029	0.0000456	-0.0000184	0.0000242	-0.0000330
σ_7	0.0000204	0.0000711	0.0000795	0.0000384	0.0000416	0.0000211
σ_8	0.0000175	0.0000847	0.0001418	0.0000793	-0.0000868	-0.0000046
σ_9	0.0002905	0.0000129	0.0000314	-0.0000309	0.0000105	0.0000472
σ_{10}	0.0000129	0.0017998	0.0000457	0.0000452	-0.0000231	-0.0000129
σ_{11}	0.0000314	0.0000457	0.0011341	0.0000686	0.0000670	0.0003174
σ_{12}	-0.0000309	0.0000452	0.0000686	0.0001368	-0.0000006	0.0000033
σ_{13}	0.0000105	-0.0000231	0.0000670	-0.0000006	0.0005061	0.0001870
σ_{14}	0.0000472	-0.0001619	0.0003174	0.0000033	0.0001870	0.0025203