# ai20btech11006_tut2

January 21, 2022

[ ]:
```
import numpy as np
import cvxpy as cp
```

[ ]:
```
# Exchange rate data.
tickers = ["USD", "EUR", "GBP", "CAD", "JPY", "CNY", "RUB", "MXN", "INR", "BRL"]
n = len(tickers)
F = np.zeros((n, n))
# USD
data = ([1.0, 0.87, 0.76, 1.31, 108.90, 6.72, 65.45, 19.11, 71.13, 3.69],
# EUR
[1.0, 0.88, 1.51, 125.15, 7.72, 75.23, 21.96, 81.85, 4.24],
# GBP
[1.0, 1.72, 142.94, 8.82, 85.90, 25.08, 93.50, 4.84],
# CAD
[1.0, 82.93, 5.11, 49.82, 14.54, 54.23, 2.81],
# JPY
[1.0, 0.062, 0.60, 0.18, 0.65, 0.034],
# CNY
[1.0, 9.74, 2.85, 10.61, 0.55],
# RUB
[1.0, 0.29, 1.09, 0.056],
# MXN
[1.0, 3.73, 0.19],
# INR
[1.0, 0.052],
# BRL
[1.0])
for i in range(n):
    F[i,i:] = data[i]
for j in range(n):
    for i in range(j+1,n):
        F[i,j] = 1.035/F[j,i]

# Initial and final portfolios.
c_req = np.arange(1,n+1)
c_req = 1e4*c_req/c_req.sum()
c_init = c_req[::-1]
```

```
# discussed in class
def cost(F,c_init,c_final,n):
    return np.sum((c_init-c_final)@np.sqrt(F[:,0]/F[0,:]))
```

[ ]:
```
X = cp.Variable((10,10))
c_final = cp.Variable(10)

# constraints discussed in class
constraints = [ c_final == c_init+(X/F)@np.ones(10)-X.T@np.ones(10) , X>=0 , cp.
 ↪diag(X)==0, c_final >= c_req, X.T@np.ones(10)<=c_init  ]

# cost function defined in prev cell
objective = cp.Minimize(cost(F,c_init, c_final, n))

prob = cp.Problem( objective, constraints)

prob.solve()
```

[ ]:
```
7.720060114618434
```

[ ]:
```
print("X:",np.around(X.value, decimals=6))
```

```
X: [[0.00000000e+00 0.00000000e+00 0.00000000e+00 1.00000000e-06
  3.60000000e-05 2.00000000e-06 1.60000000e-05 5.00000000e-06
  1.20000000e-05 1.00000000e-06]
 [1.00000000e-06 0.00000000e+00 1.00000000e-06 2.00000000e-06
  1.21000000e-04 3.00000000e-06 3.20000000e-05 9.00000000e-06
  1.70000000e-05 1.00000000e-06]
 [1.00000000e-06 0.00000000e+00 0.00000000e+00 5.45454545e+02
  1.90000000e-04 4.00000000e-06 3.80000000e-05 1.10000000e-05
  1.90000000e-05 1.00000000e-06]
 [1.00000000e-06 0.00000000e+00 0.00000000e+00 0.00000000e+00
  4.34000000e-04 5.00000000e-06 4.70000000e-05 1.30000000e-05
  2.20000000e-05 1.00000000e-06]
 [1.00000000e-06 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 4.00000000e-06 1.41000000e-04 5.00000000e-06
  3.70000000e-05 1.00000000e-06]
 [1.65029400e+01 0.00000000e+00 0.00000000e+00 0.00000000e+00
  4.40000000e-05 0.00000000e+00 7.27272289e+02 6.30000000e-05
  4.08000000e-04 2.00000000e-06]
 [1.00000000e-06 0.00000000e+00 0.00000000e+00 0.00000000e+00
  2.70000000e-05 1.00000000e-06 0.00000000e+00 3.69090819e+02
  3.70000000e-05 2.00000000e-06]
 [1.56860410e+01 0.00000000e+00 0.00000000e+00 0.00000000e+00
  1.81817752e+02 2.00000000e-06 1.30000000e-05 0.00000000e+00
  6.34000000e-04 1.81818170e+02]
```

```
[1.85192440e+01 0.00000000e+00 0.00000000e+00 1.00000000e-06
 3.00000000e-05 2.00000000e-06 1.70000000e-05 5.00000000e-06
 0.00000000e+00 2.00000000e-06]
[5.09977825e+02 0.00000000e+00 0.00000000e+00 1.00000000e-06
 4.10000000e-05 2.00000000e-06 1.30000000e-05 3.00000000e-06
 1.20000000e-05 0.00000000e+00]]
```

[ ]: