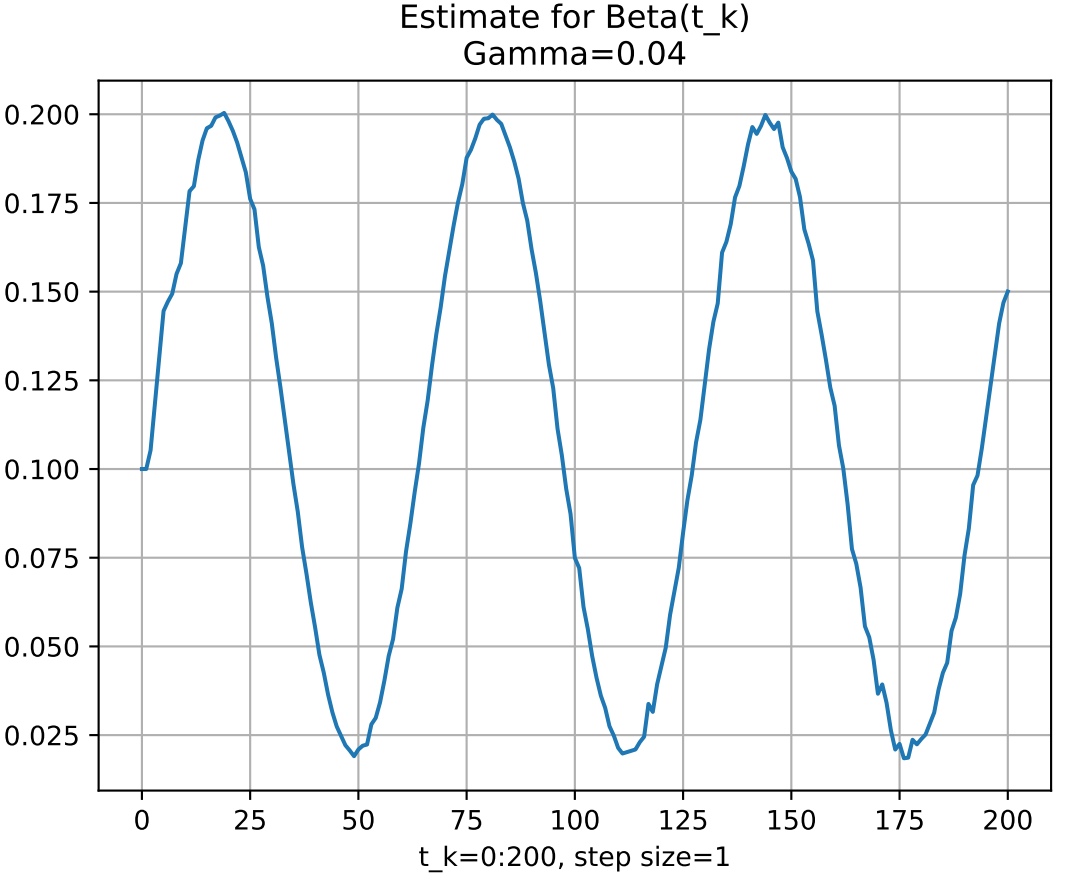
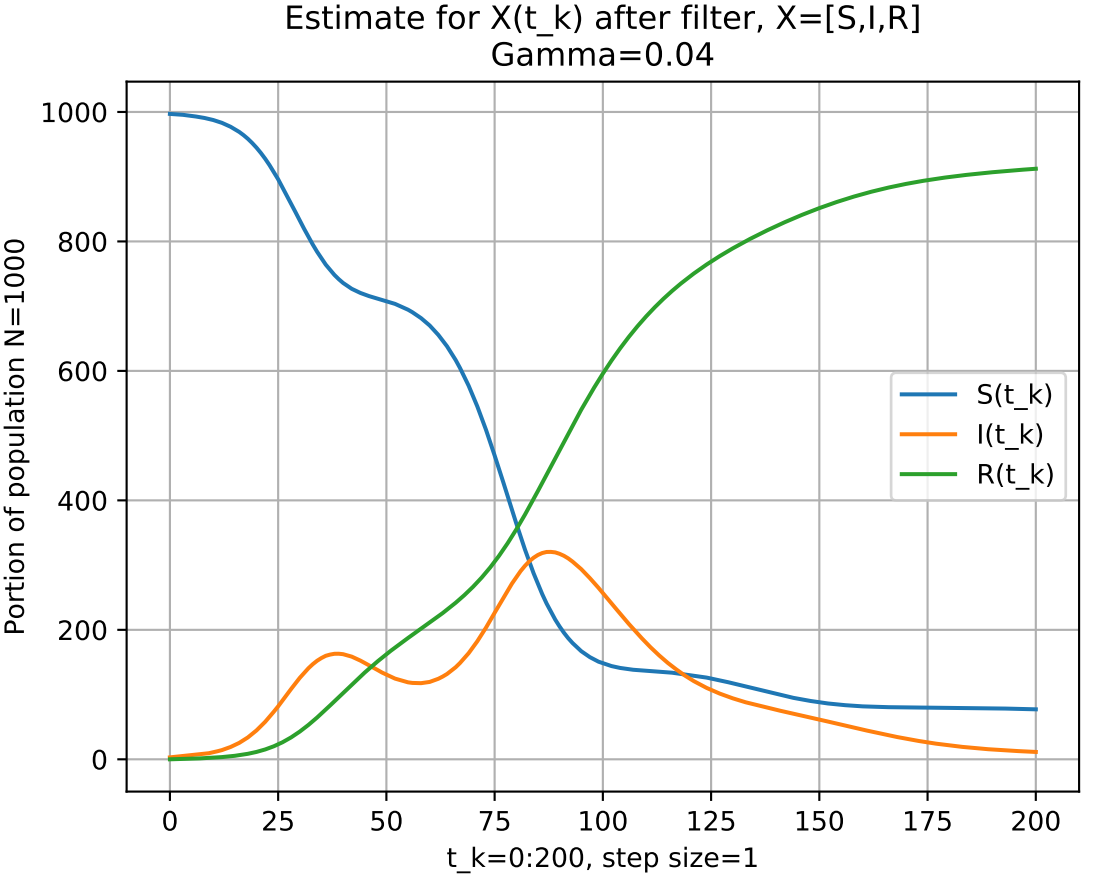
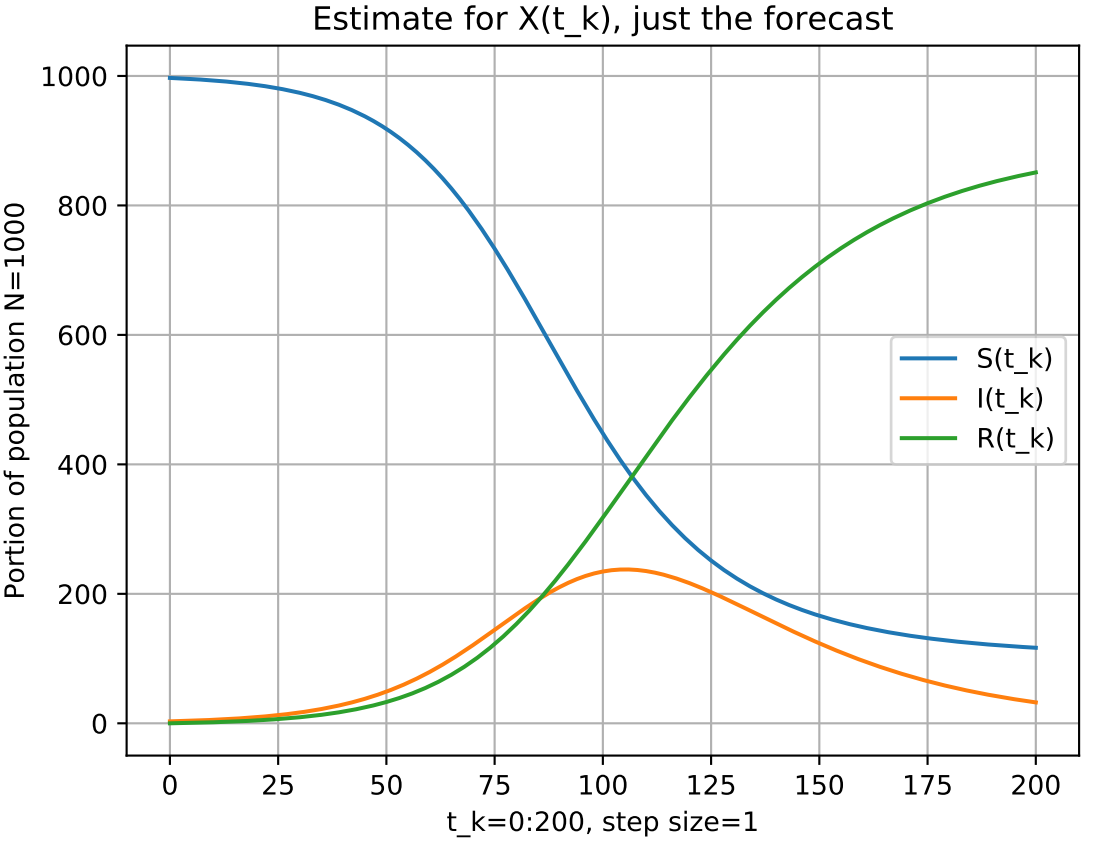

Estimate for Beta(t_k)
Gamma=0.04
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0
25
50
75
100
125
150
175
200
t_k=0:200, step size=1


Estimate for X(t_k) after filter, X=[S,I,R]
Gamma=0.04
Portion of population N=1000
1000
800
600
400
200
0
0
25
50
75
100
125
150
175
200
t_k=0:200, step size=1
S(t_k)
I(t_k)
R(t_k)


Estimate for X(t_k), just the forecast
Portion of population N=1000
t_k=0:200, step size=1
S(t_k)
I(t_k)
R(t_k)
0
200
400
600
800
1000
0
25
50
75
100
125
150
175
200

Branch: master ▾ Find file Copy path

610_Optimization / HW3 / **hw3.py** / <> Jump to ▾

JimVargas5 Did hw3, will format a single pdf nicely

3c74d71 5 minutes ago

1 contributor

Raw Blame History

160 lines (120 sloc) 3.53 KB

```
# Jim Vargas
# MTH 610
# HW 3

# Remember that vectors are actually 2D at least, nx1 or 1xn matrices


import numpy as np
from matplotlib import pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
pdf=PdfPages('raw_graphs_n_such.pdf')




''' constants '''
h=1
t0=0
tf=200
N=1000
t=np.linspace(t0, tf, int(tf/h)+1)
Gamma=0.04

stand_dev=0.001
Q=np.diag(np.array([0.01, 0.01, 0.01, 0.04]))

DATA=np.loadtxt('obsdata_txt.txt')




''' data structures and models '''
Xa_0=np.array([[997, 3, 0, 0.1]]) # [S I R Beta]
Xa_0=Xa_0.transpose() # [S I R Beta]^T
Xa=Xa_0
Xa_storage=np.zeros((4, tf+1))
Xa_storage[:,0]=Xa.reshape(4)
Xf_just_forecast=Xa
Xf_storage=np.zeros((4, tf+1))
Xf_storage[:,0]=Xa.reshape(4)

Pa_0=np.diag(np.array([10, 10, 0.01, 0.04]))
Pa=np.array(Pa_0)

def Mf(X): # 4x1
    'input is augmented analysis Xa_k-1'
    'output is forecast Xf_k'
    S=X[0]
    I=X[1]
    R=X[2]
    Beta=X[3]
```

```
    c=h/N

    S_next= S - c*Beta*S*I
    I_next= I + c*Beta*S*I - h*Gamma*I
    R_next= R + h*Gamma*I
    Beta_next= Beta
    return np.array([S_next, I_next, R_next, Beta_next]).reshape((4,1))

def Ma(X): # 4x4
    'input is augmented analysis Xa_k-1'
    'output is {dMf/dXa}_k-1, for decluttering code'
    S=X[0]
    I=X[1]
    Beta=X[3]
    c=h/N
    return np.array([
        [1-c*Beta*I, -1*c*Beta*S, 0, -1*c*S*I],
        [c*Beta*I, 1+c*Beta*S-h*Gamma, 0, c*S*I],
        [0, h*Gamma, 1, 0],
        [0, 0, 0, 1]
    ])

def Pf(P, X):
    'input is augmented analysis Pa_k-1, augmented analysis Xa_k-1'
    'output is forecast covariance Pf_k'
    return np.matmul(
        Ma(X), np.matmul( P, Ma(X).transpose() )
    ) + Q

H=np.array([[0,0,1,0]]) # observe R only
r_constant=stand_dev**2

def Xa_k(X, P, K, y):
    'input is forecast Xf_k, forecast covariance Pf_k'
    'output is augmented analysis Xa_k'
    return X + K*(y - np.matmul(H,X))

def Pa_k(P,K):
    'input is forecast covariance Pf_k'
    'output is analysis covariance Pa_k'
    return np.matmul(
        (np.identity(4) - np.matmul(K,H)), P
    )




''' main loop '''
for k in range(1, tf+1):
    Xf_just_forecast=Mf(Xf_just_forecast)
    Xf_k=Mf(Xa)
    Pf_k=Pf(Pa, Xa)

    K=np.matmul( Pf_k, H.transpose() )
    c=np.matmul(
        H, np.matmul( Pf_k, H.transpose() )
    )
    K=((c+r_constant)**(-1))*K

    y=DATA[k-1]
    Xa=Xa_k(Xf_k, Pf_k, K, y)
    Pa=Pa_k(Pf_k, K)

    Xf_storage[:,k]=Xf_just_forecast.reshape(4)
    Xa_storage[:,k]=Xa.reshape(4)

```

```
118
119
120
121 ''' plots '''
122 fig_Beta=plt.figure()
123 plt.plot(t, Xa_storage[3,:], label='Beta(t_k)')
124 plt.title("Estimate for Beta(t_k)"+'\n'+"Gamma="+str(Gamma))
125 plt.xlabel("t_k=0:"+str(tf)+", step size="+str(h))
126 plt.grid(True)
127 plt.close()
128 pdf.savefig(fig_Beta)
129
130 fig_state=plt.figure()
131 plt.plot(t, Xa_storage[0,:], label='S(t_k)')
132 plt.plot(t, Xa_storage[1,:], label='I(t_k)')
133 plt.plot(t, Xa_storage[2,:], label='R(t_k)')
134 plt.title("Estimate for X(t_k) after filter, X=[S,I,R]"+'\n'+"Gamma="+str(Gamma))
135 plt.xlabel("t_k=0:"+str(tf)+", step size="+str(h))
136 plt.ylabel("Portion of population N="+str(N))
137 plt.legend(loc='best')
138 plt.grid(True)
139 plt.close()
140 pdf.savefig(fig_state)
141
142 fig_forecasted=plt.figure()
143 plt.plot(t, Xf_storage[0,:], label='S(t_k)')
144 plt.plot(t, Xf_storage[1,:], label='I(t_k)')
145 plt.plot(t, Xf_storage[2,:], label='R(t_k)')
146 plt.title("Estimate for X(t_k), just the forecast")
147 plt.xlabel("t_k=0:"+str(tf)+", step size="+str(h))
148 plt.ylabel("Portion of population N="+str(N))
149 plt.legend(loc='best')
150 plt.grid(True)
151 plt.close()
152 pdf.savefig(fig_forecasted)
153
154 pdf.close()
155
156
157
158
159 # My function names are confusing, maybe change them to 'compute_Xa_k' or something...
```