a)

b)

= J(θ) =

Because we do not need to give the part of the Jacobian that deals with orientations, so

J(θ) = =

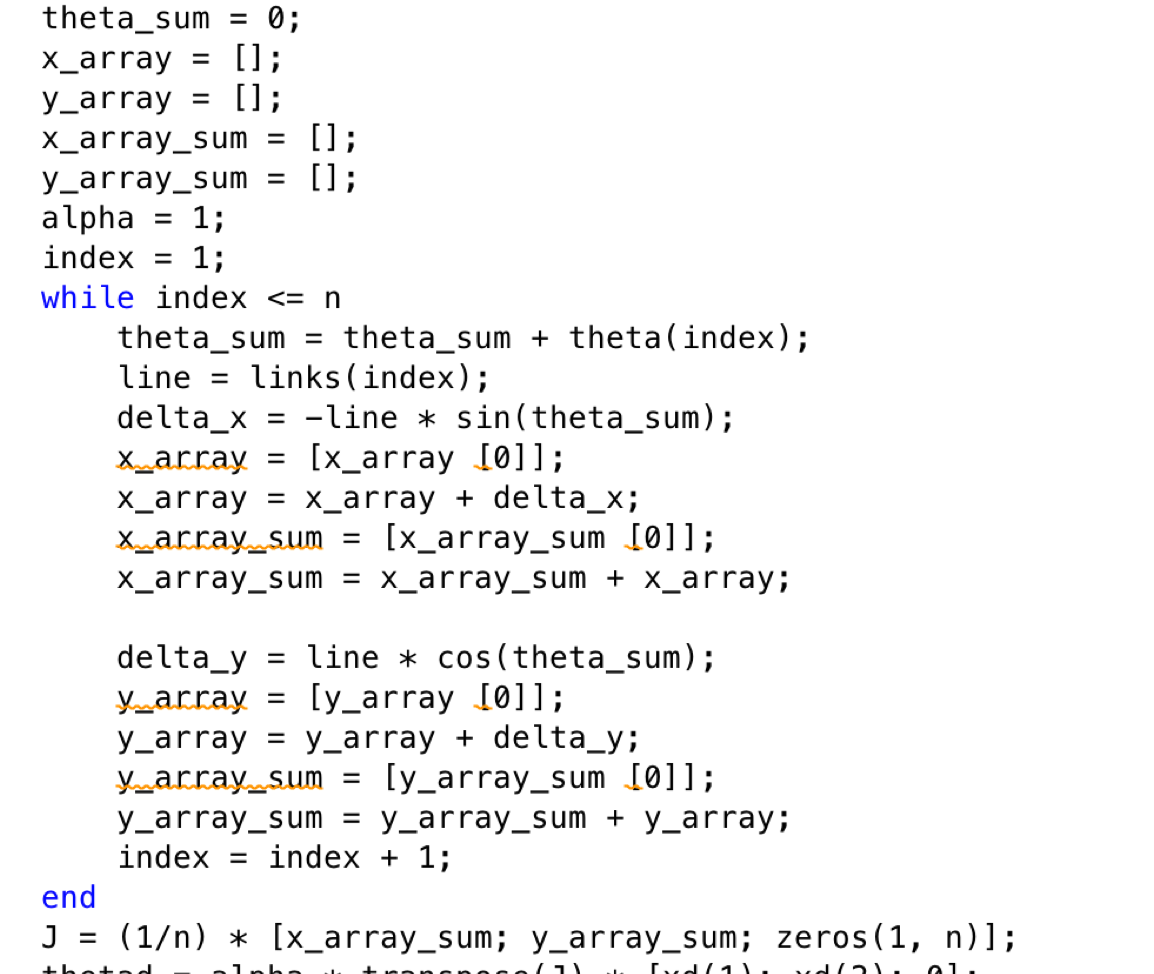
where, is the position of joint , and is the unit axis of joint .

c)

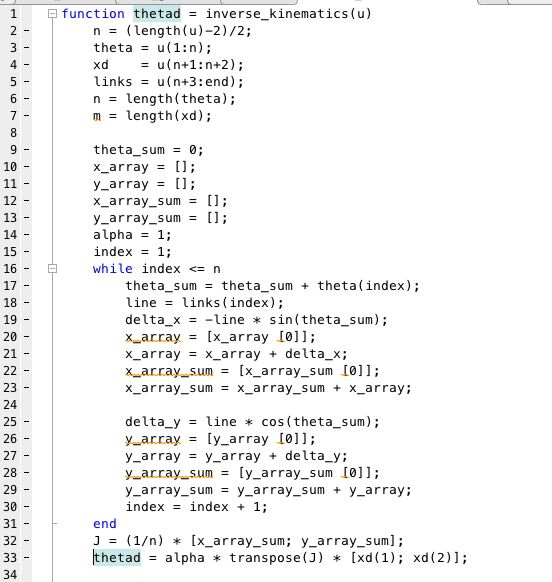
d)

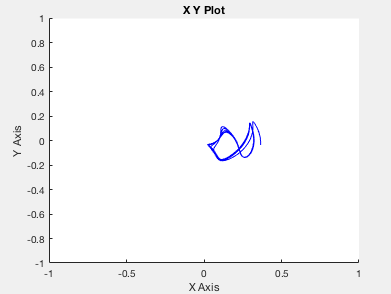
e)

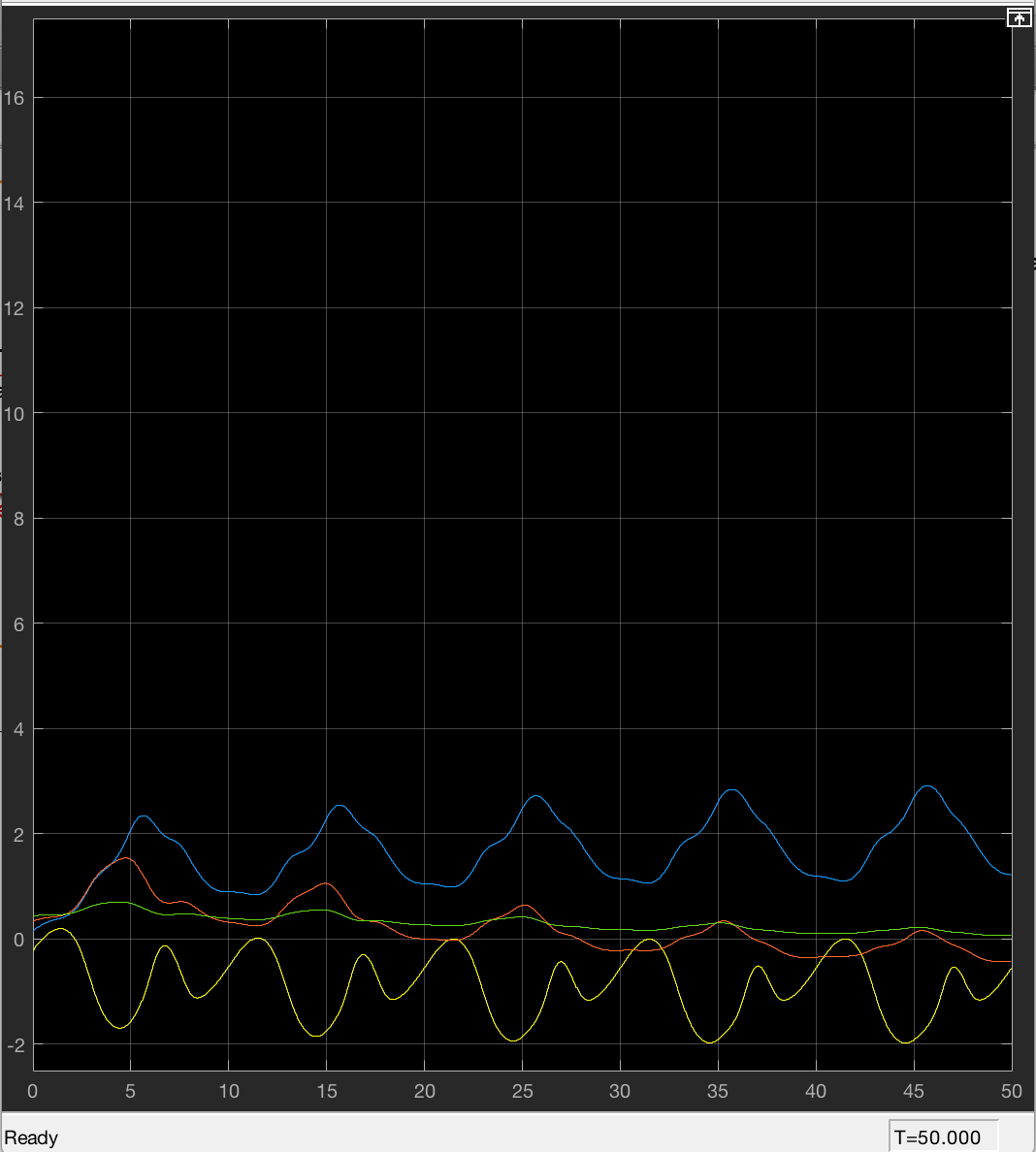
f)



(g)



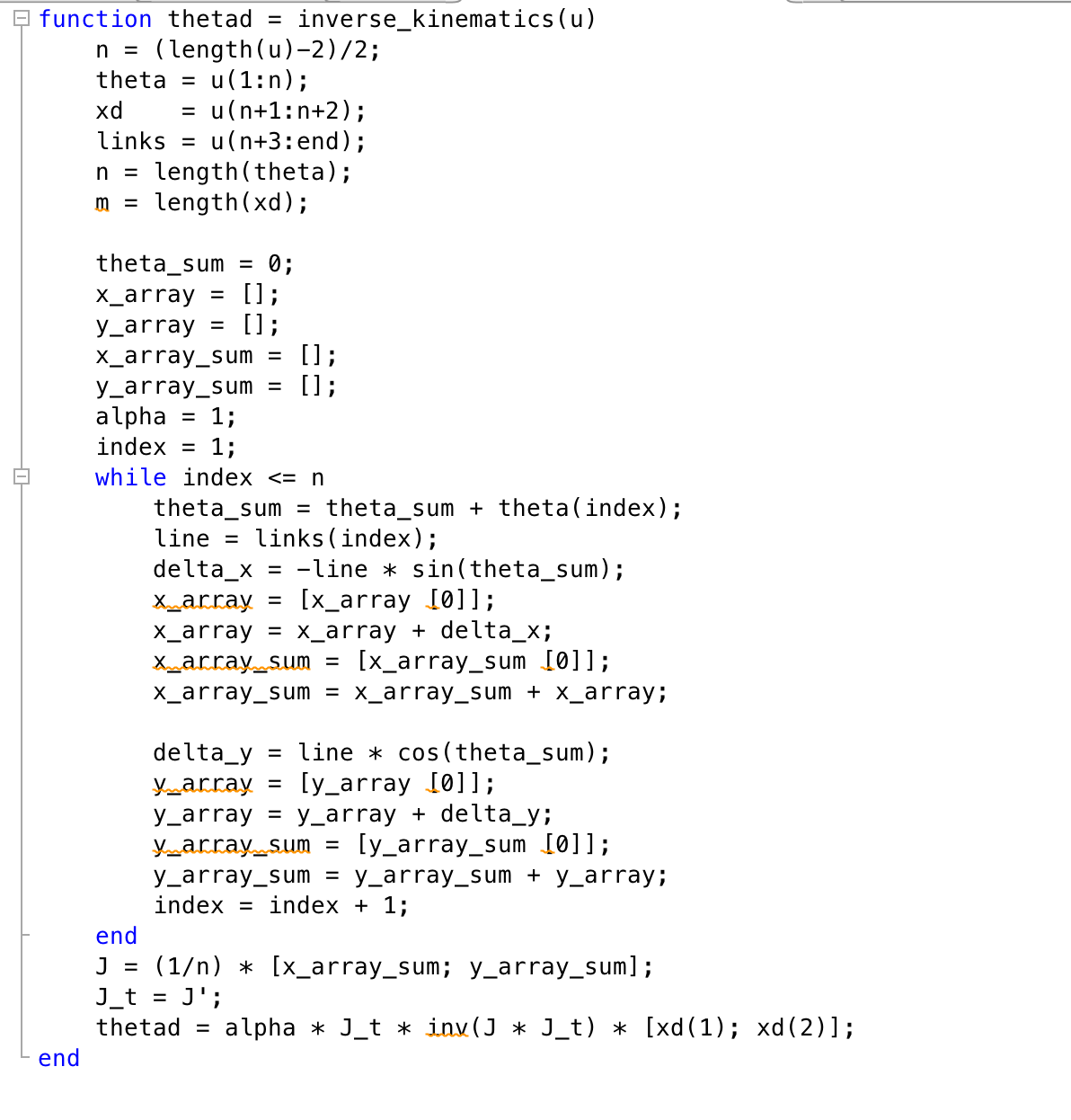


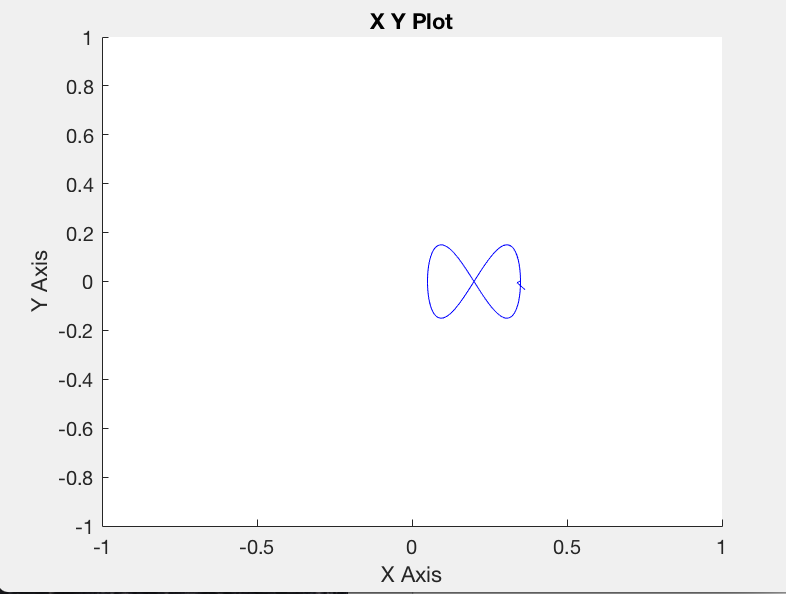


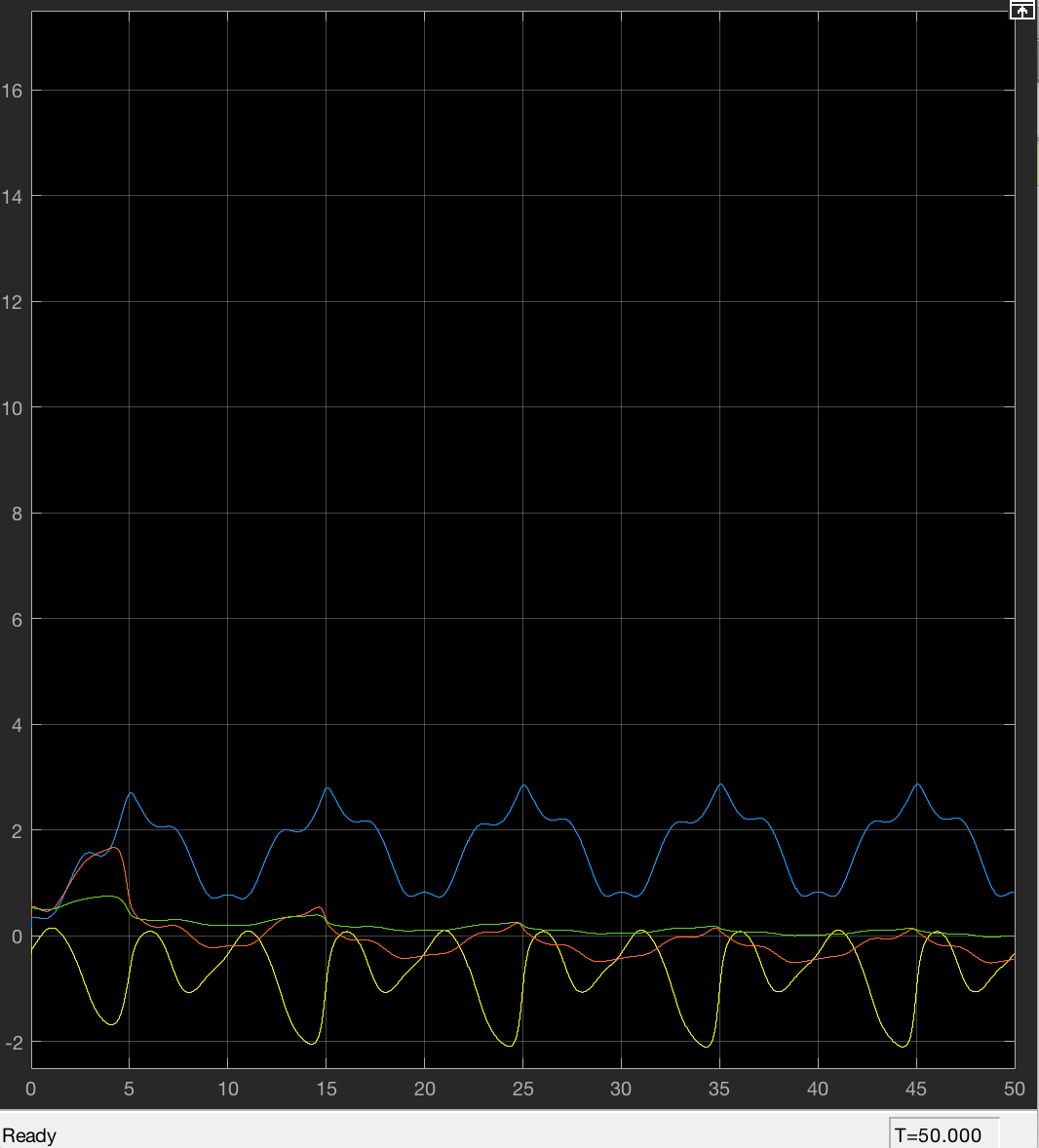
Not very good, it looks barely like “8”. The range of arms represented in blue and yellow are become bigger and bigger.

Because Jacobian has very small coefficients, it needs many iterations until convergence in certain configuration. And it is not conservative. And when Jacobian maps endeffector displacement to configuration displacement, there’s some error.

h)



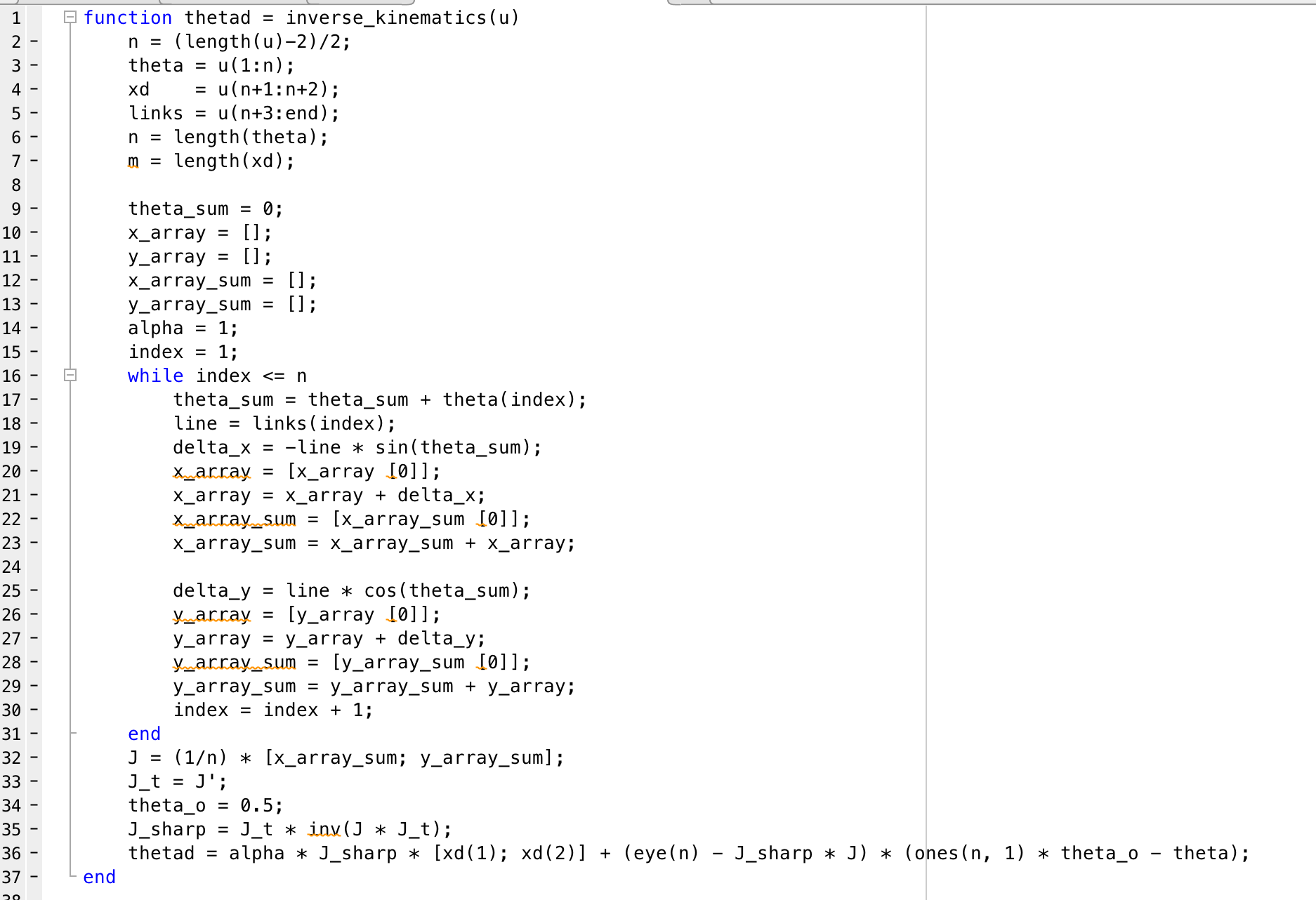


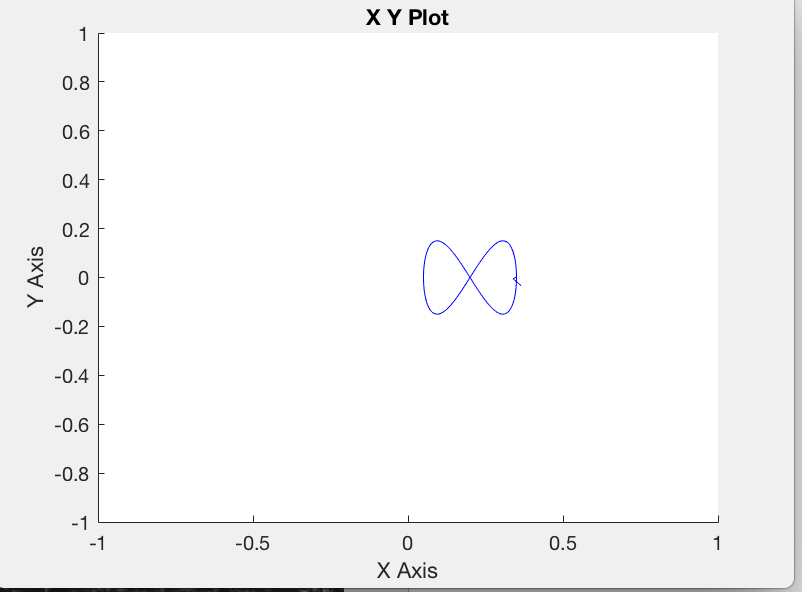


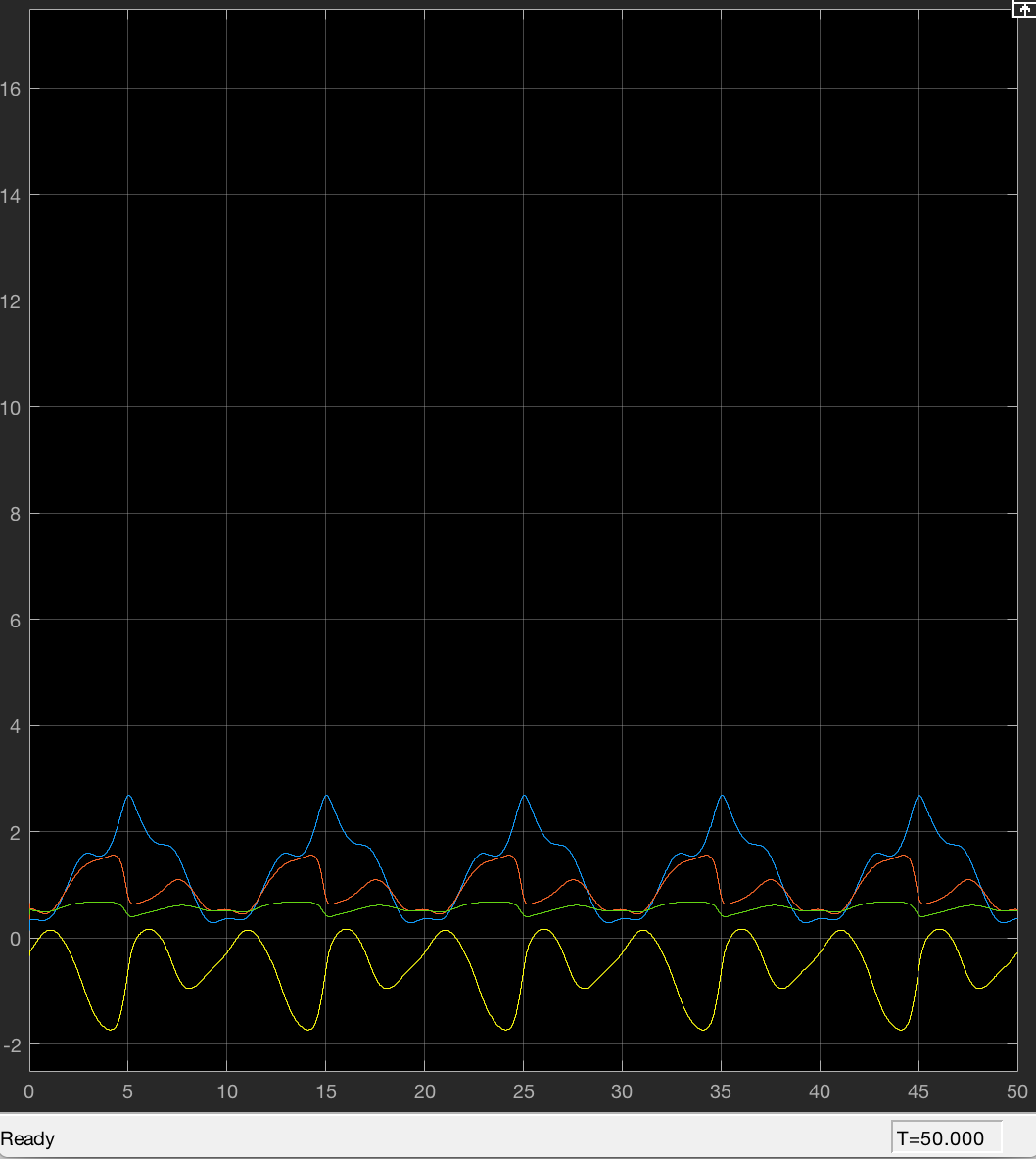
This one is pretty good! It looks like “8”, except the beginning. And the graph is conciding.

Because we have minimized the error, when we use calculate and use J#, so it can get more precise thetad.

i)







This one is awesome! It looks like “8”, except the beginning. And the graph is conciding.

Because it minimizes the error and adjust result with optimization criterion, when it uses J# and theta\_o, so it can get more precise thetad.

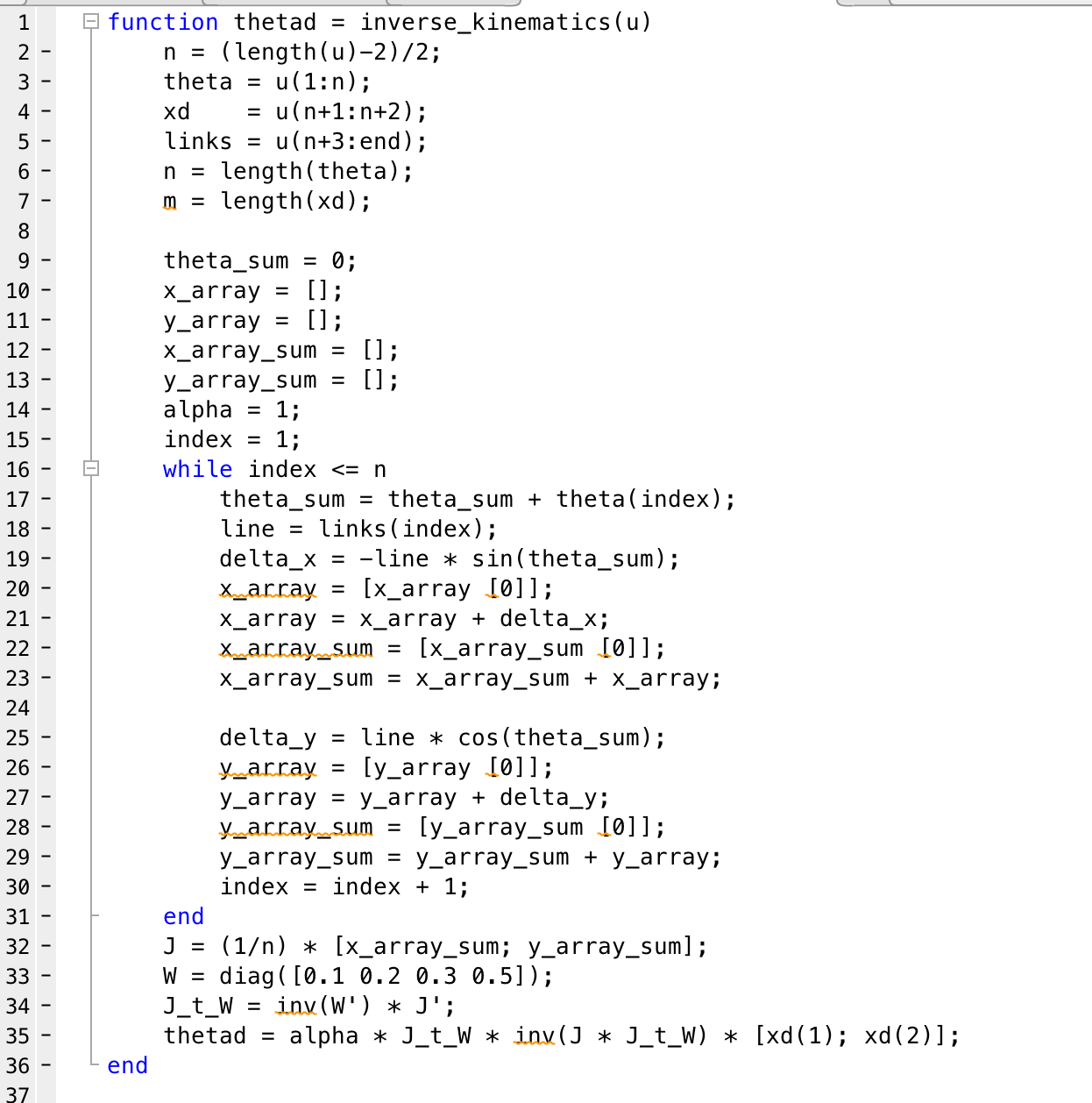
j)

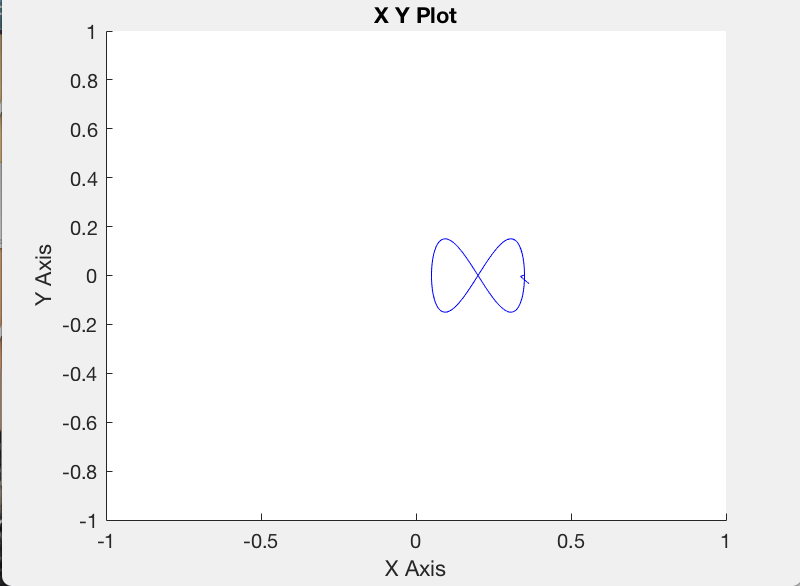
where is a vector of Lagrange multipliers.

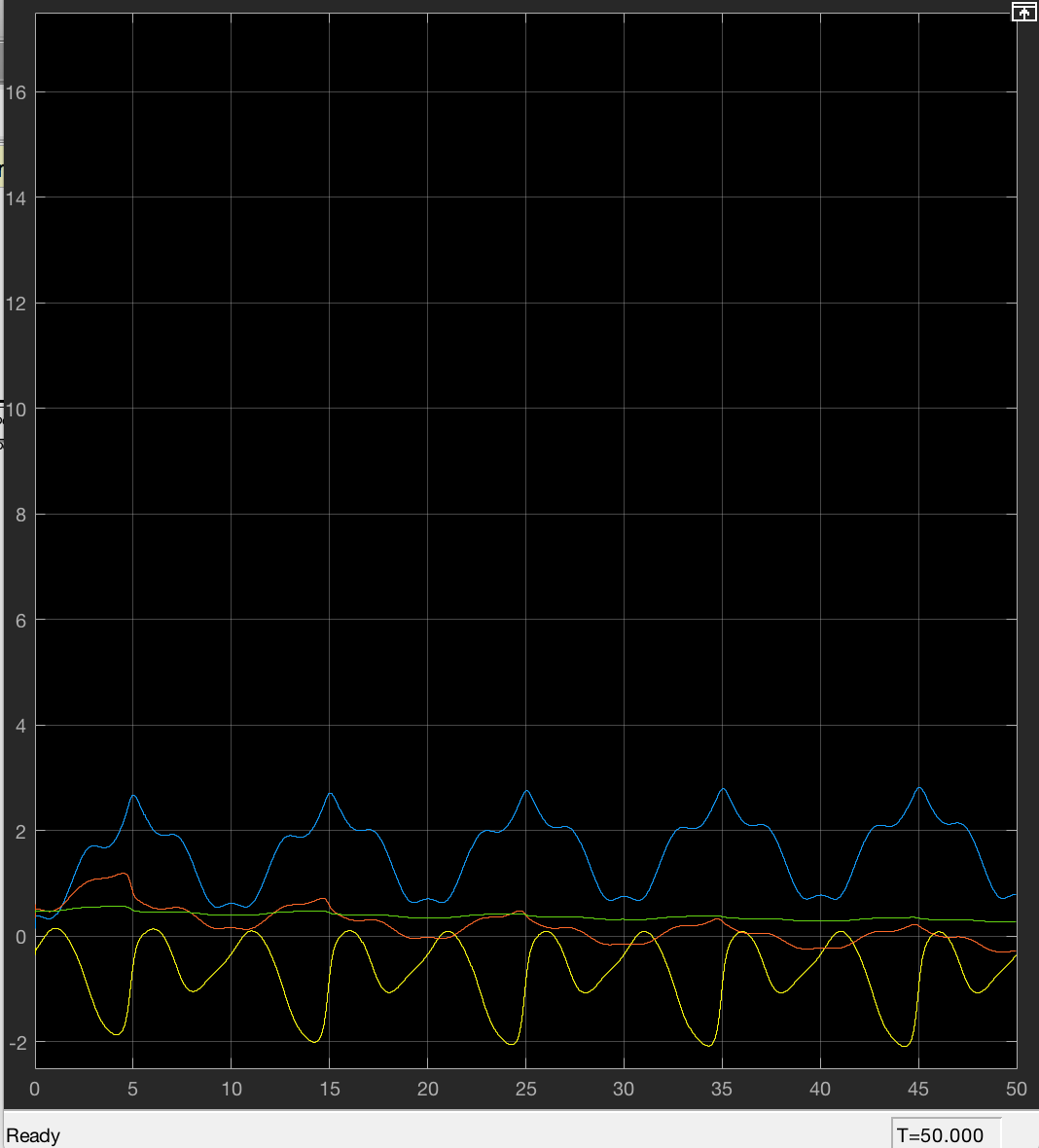


Combine (1) and (2):

Insert (3) to (2):







This one is awesome! It looks like “8”, except the beginning. And the graph is conciding. And the arms represented by yellow and red move more gentle than (h).

Because we have minimized the error, when we use calculate and use J#, so it can get more precise thetad.

Besides, because it uses different weight of each arms, so it can save energy based on the weight, so make yellow line and red line smoother than (h).

k)

where is a vector of Lagrange multipliers.



==>

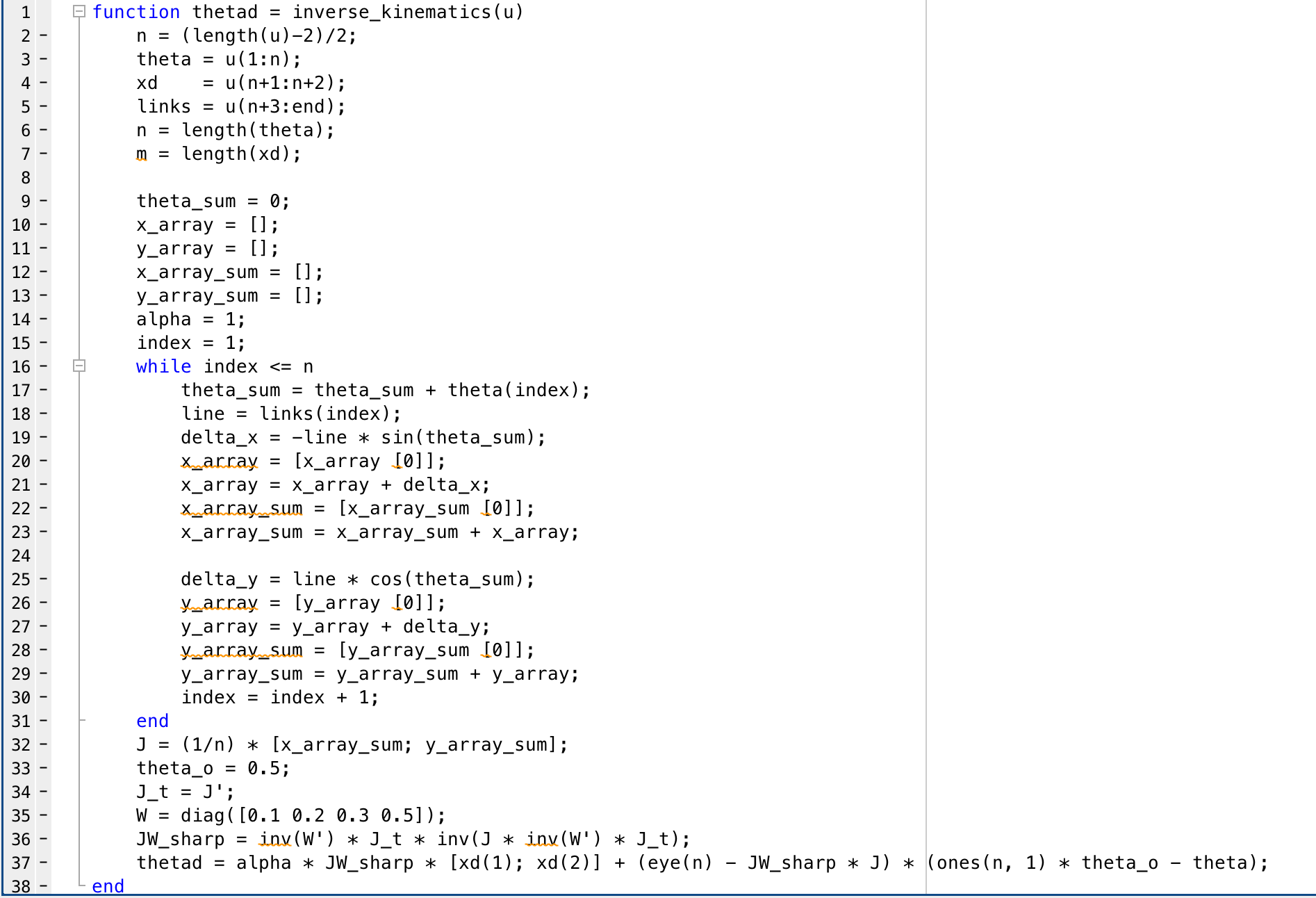
Combine (1) and (2):

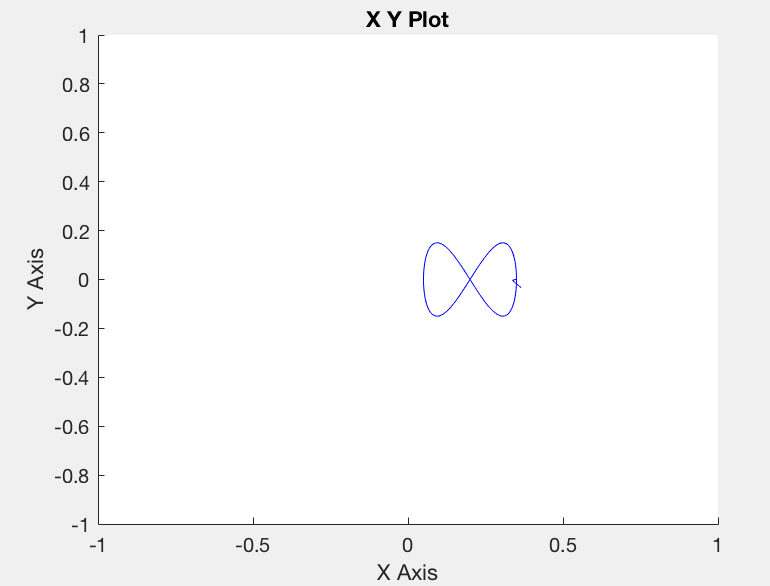
Insert (3) to (2):

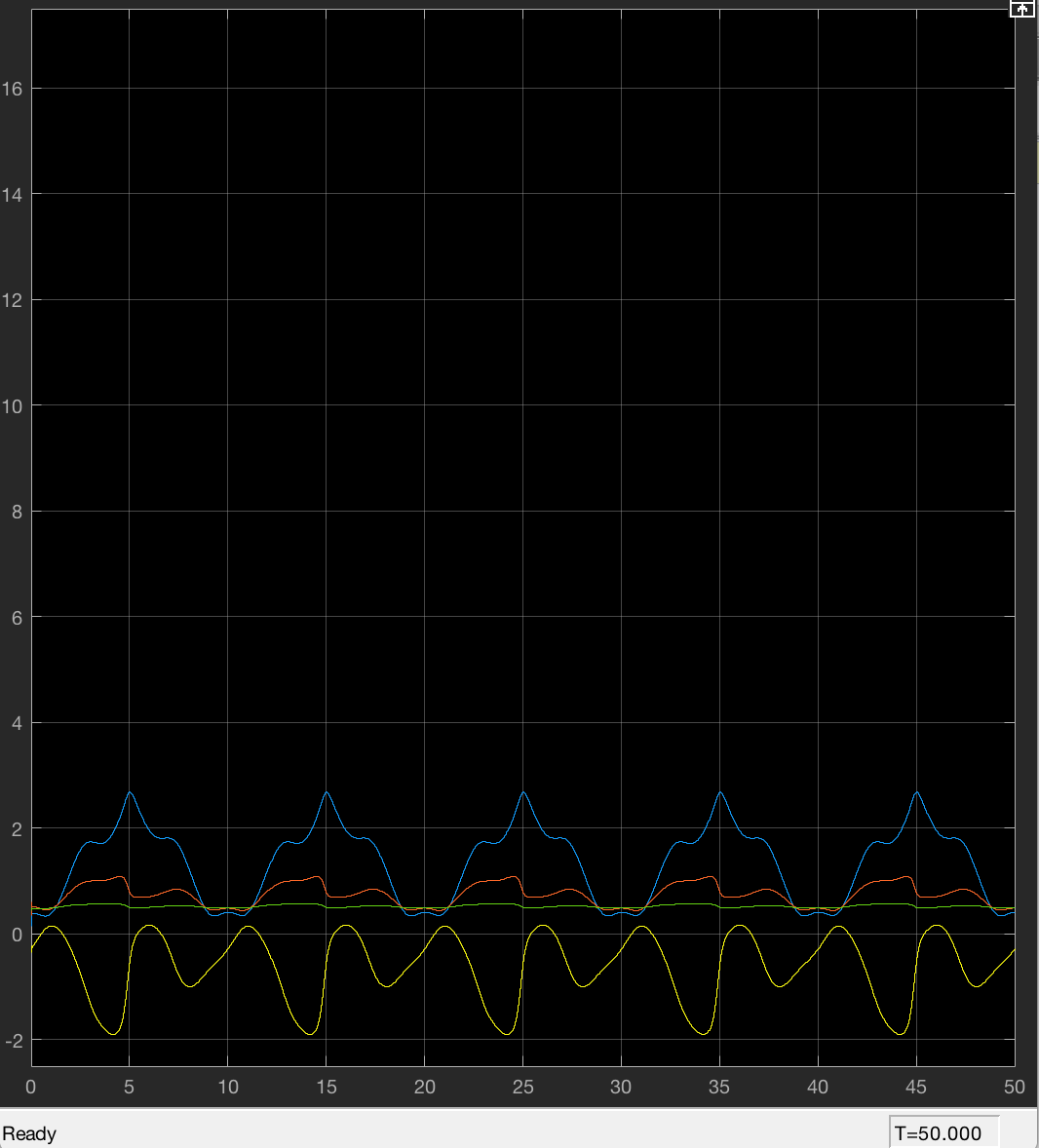
=

=

where,







This one is awesome! It looks like “8”, except the beginning. And the graph is conciding. And the arms represented by yellow and red move more gentle than (h).

Because it minimizes the error and adjust result with optimization criterion, when it uses J# and theta\_o, so it can get more precise thetad.

Besides, because it uses different weight of each arms, so it can save energy based on the weight, so make yellow line and red line smoother than (h).

2.

