A robot that learns movements and stabilization on its own

SRIP ‘2022 Report

Submitted By: Harshvardhan Vala
BTech Computer Science and Engineering,
IIT Gandhinagar

Under the Guidance of:
Professor Harish Palanthandalam-Madapusi
Sujay Kadam

Abstract

Name: Harshvardhan Vala
Institution: IIT Gandhinagar
Advisor: Professor Harish Palanthandalam-Madapusi

Robotics is changing the world rapidly around us. Automated systems have become a huge part of our life and are widely being used across industries. When robotics is concerned, learning is an integral part of it. A robot that can learn movements and sense and react to its environment in order to complete certain tasks, can achieve a major breakthrough that will further help us realize larger goals in future. Self learning robots are on the verge of becoming the new normal.

This project deals with working out with an algorithm that will help a robot to self learn. For this project, we have a 2R robotic manipulator that has a simple reaching task from one point to the other. We realized this by implementing the learnings of Reinforcement Learning, Control Theory and Learning Control. We performed extensive simulations on the 2R robot manipulator using a model-based iterative learning controller (MBIL) and RL controller. We analyzed its behavior by introducing perturbations, and changing the various parameters of the controller and the environment around the robot. The controller used is developed based on basics of human motor learning that can show features such as retention, savings and generalization. We performed simulations by setting up position and velocity based force fields on the 2R manipulator, and analyzed their effects. We also performed the reaching task by setting up an RL based controller. At the end we did some basic comparisons between the MBIL and RL controller on parameters discussed above (savings, retention, interference etc.).

Table of Content

Model Based Iterative Learning Controller

The model based iterative learning controller is an iteration based controller, that is governed by the following controlling equations:

1. $u_{ILC,k}(t) = \lambda_{ILC} u_{ILC,k-1}(t) + \gamma_{ILC} \tanh(\varepsilon_{TE,k-1}(t))$,
   where,
   $\varepsilon_{TE,k-1}(t) \triangleq y_{ref,k-1}(t) - y_{k-1}(t)$

2. $u_{FF,k(t)} = \lambda_{FF} u_{FF,k-1}(t) + \gamma_{FF} M_{\rho}^{\dagger} \tanh(\varepsilon_{MOE,k-1}(t))$
   where,
   $\varepsilon_{MOE,k-1}(t) \triangleq M_{\rho} u_{k-1}(t) - y_{pred,k-1}(t)$ and,
   $u_k(t) = u_{ILC,k}(t) + u_{FF,k}(t)$,

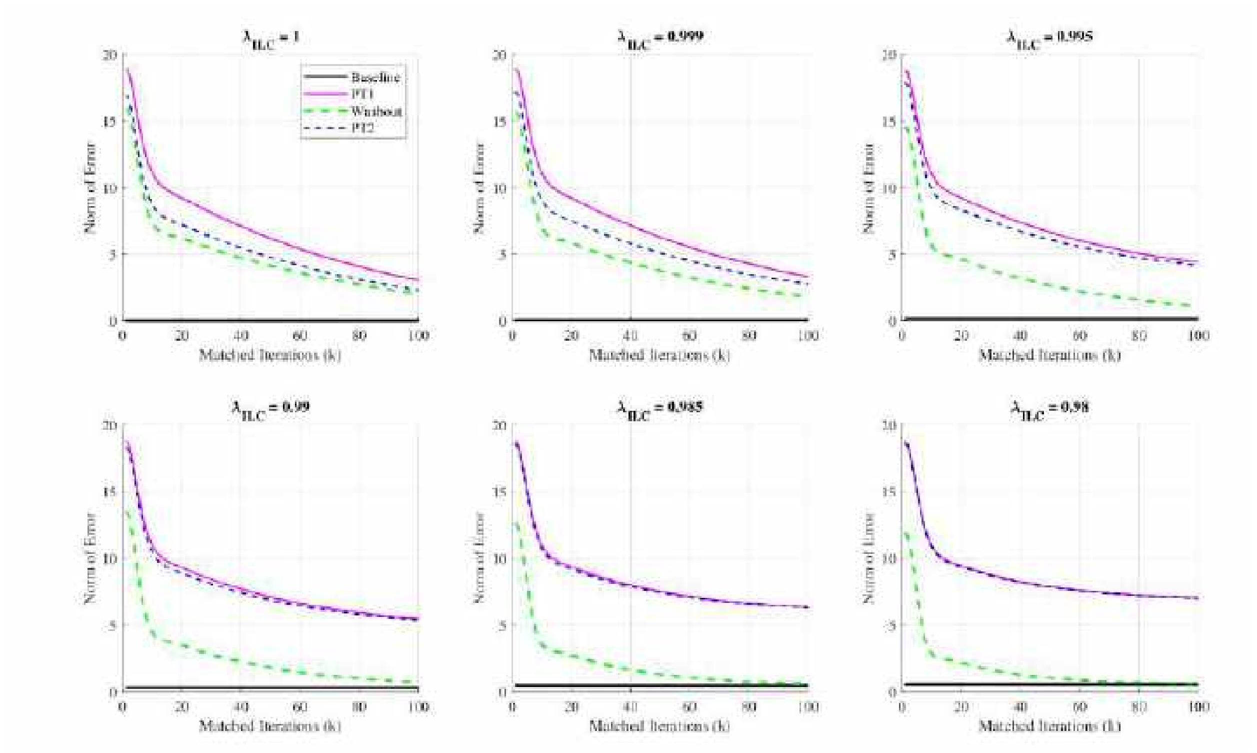
3. $\varepsilon_{SPE,k-1}(t) \triangleq y_{k-1}(t) - y_{pred,k-1}(t)$
   where,
   $y_{pred,k}(t) = \lambda_{pred} y_{pred,k-1}(t) + \gamma_{pred} \tanh(\varepsilon_{SPE,k-1}(t))$

where the parameters are defined as:

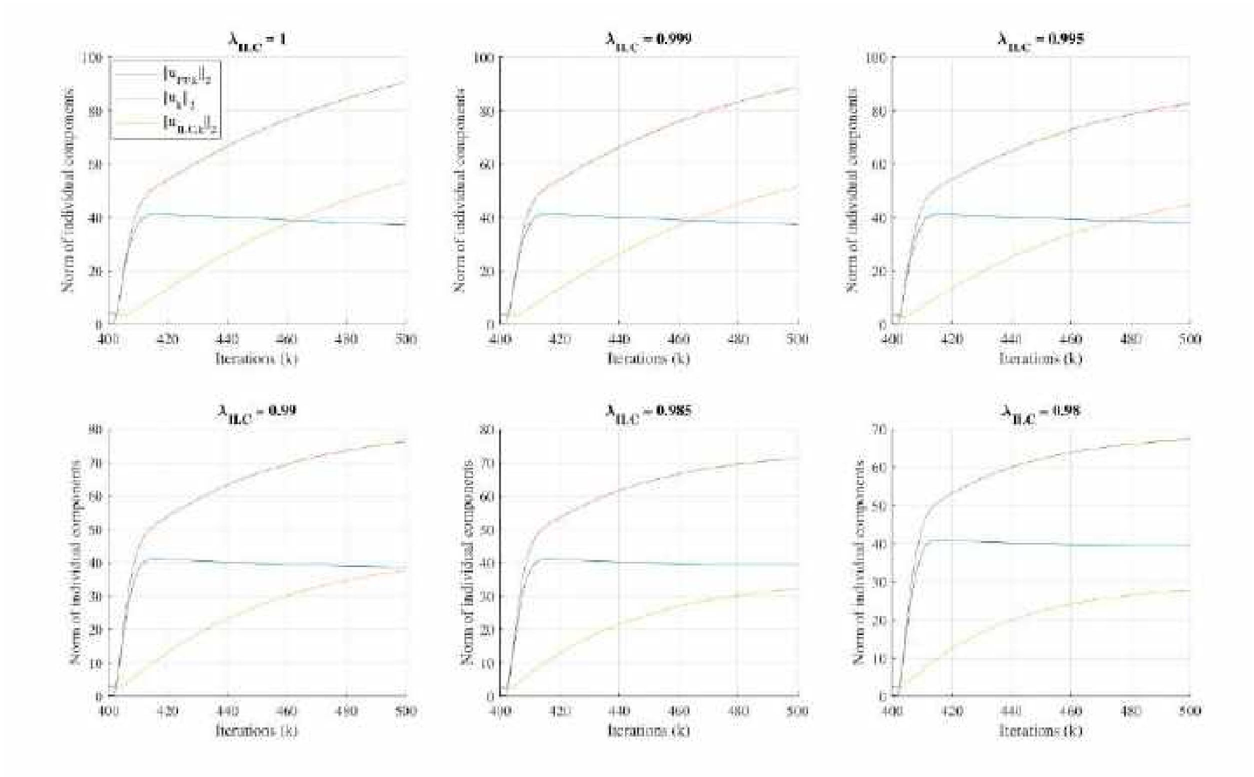
λ_{ILC} : scalar memory factor
γ_{ILC} :scalar learning rate
$\varepsilon_{TE,\, k-1}(t) \in R^{l}$: vector task error (TE) from the previous trial
$y_{ref,k-1}(t) \in R^{l}$: vector reference trajectory for the given task
$y_{k-1}(t) \in R^{l}$: the actual vector task trajectory achieved
$u_{FF,k}(t) \in R^{m}$:the vector feedforward input
$M_{\rho} \in R^{l\times m}$: simplified representation of the matrix input-output map (forward model without accounting for any environmental interaction or dynamics)
$\varepsilon_{MOE,k-1}(t+\rho) \in R^{l}$: denotes a vector model output error (MOE)
$\varepsilon_{SPE,k-1}(t)$:sensory prediction error (SPE)
y_{pred}: predicted sensory output

The first task was to vary the parameters of the controller equation, and analyze how it affects the learning of the 2R robotic manipulator, and trying to understand the effects based on what the parameter signifies.
The following are the results of the simulations:

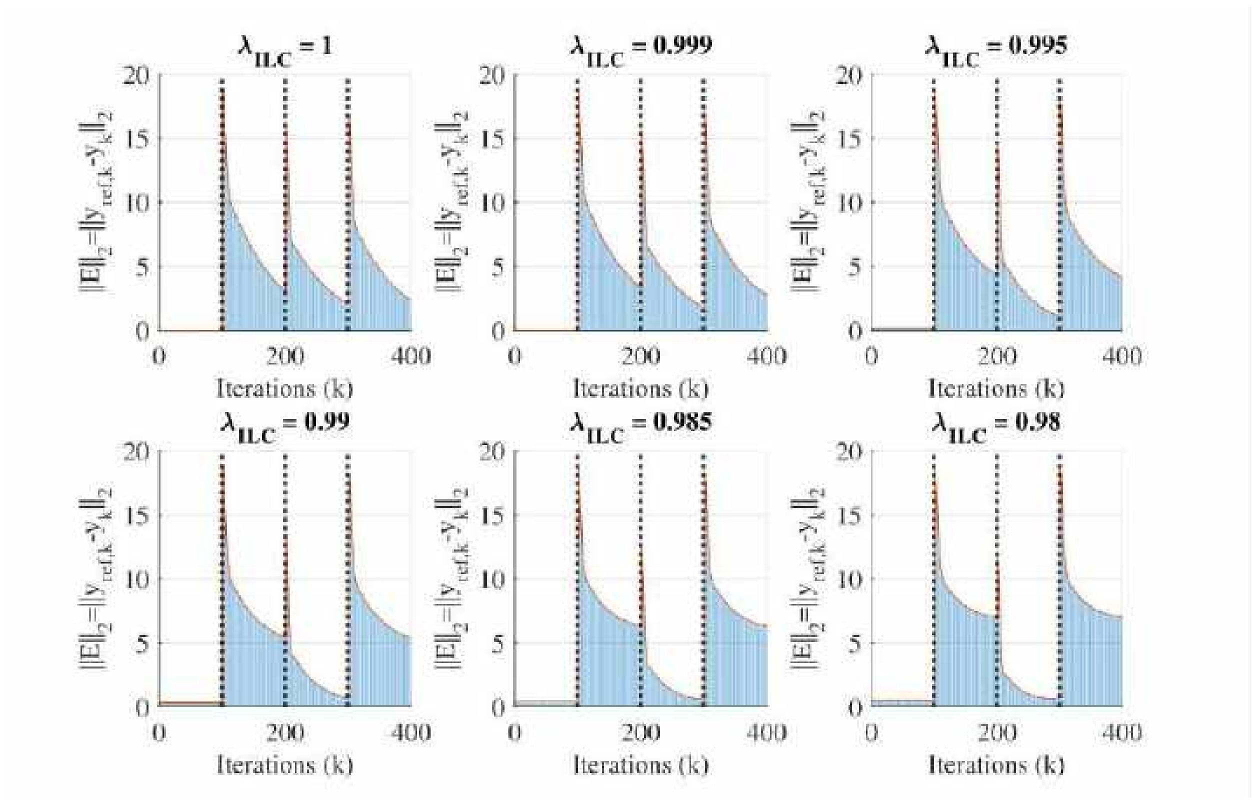
1. Changing λ_{ILC}



Observation: Savings and retention are lower for lower values of λ_{ILC}.

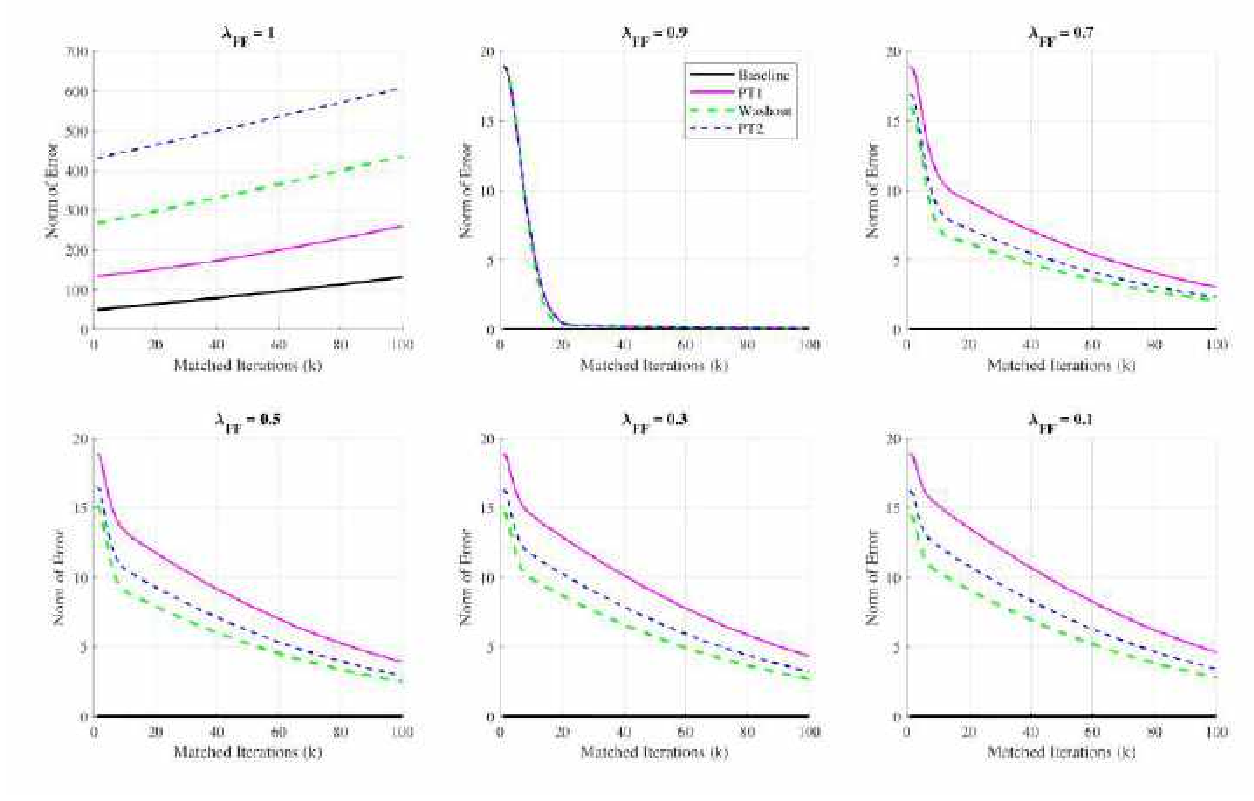


Observation: Combined contribution is decreasing due to decrease in U_{ILC}. U_{FF} however remains the same.

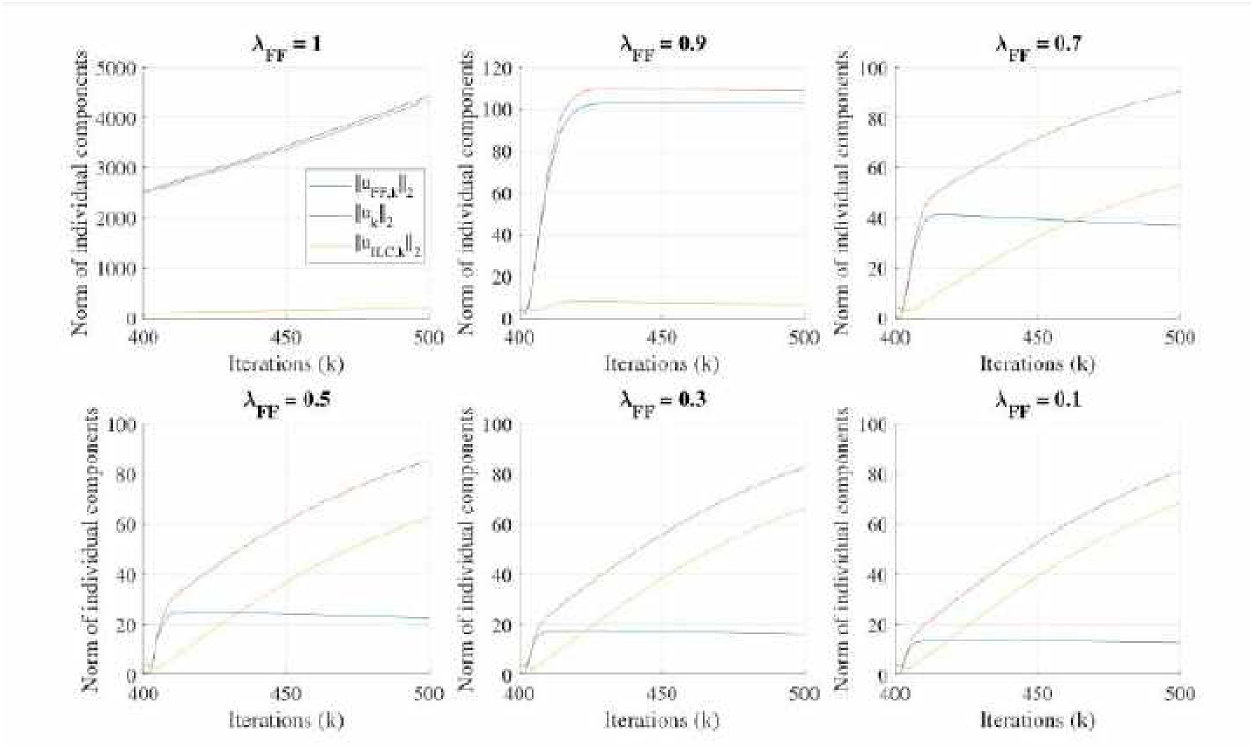


Observation :
Washout is faster when we have a smaller value of lambda ilc. Error in tasks Perturbation 1 and Perturbation 2 however increase.

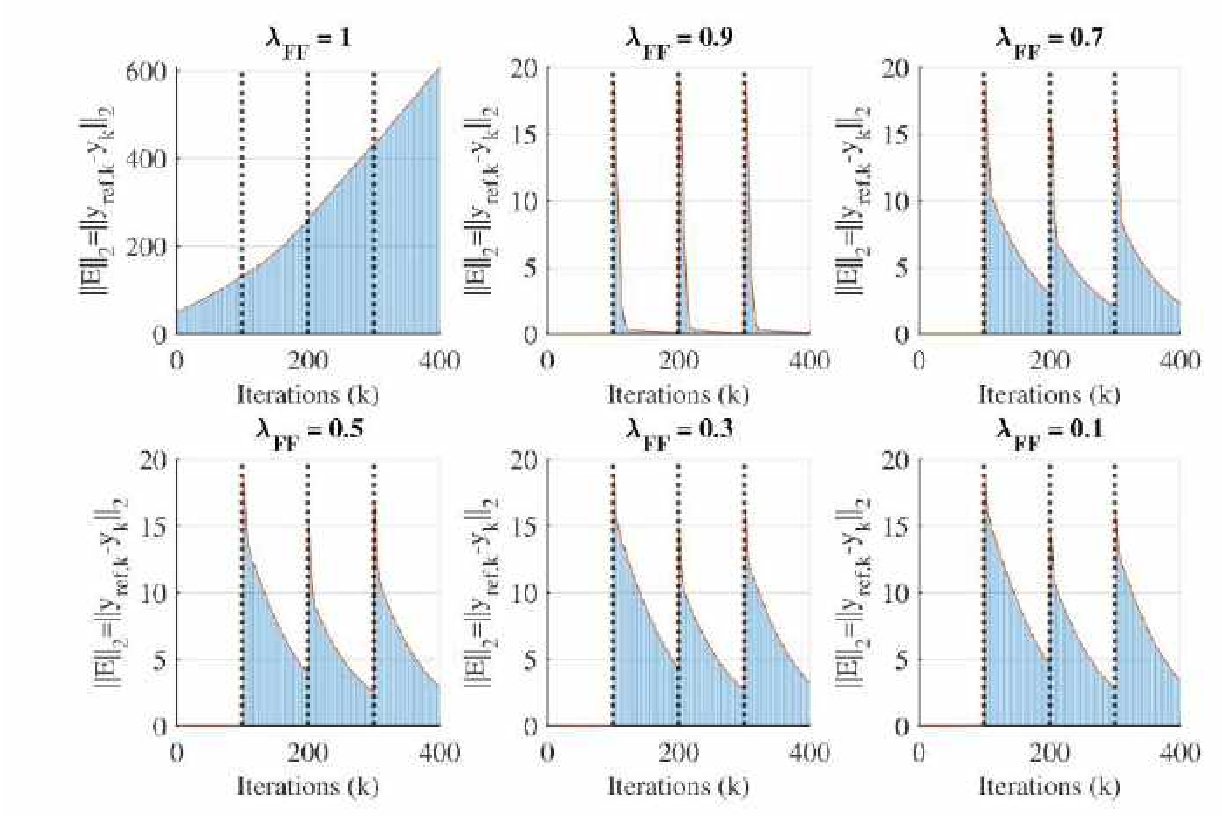
2. Changing λ_{FF}



Observation: For λ_{FF}=1 the simulation becomes unstable. For smaller values of λ_{FF} larger values of savings (and retention) are obtained.

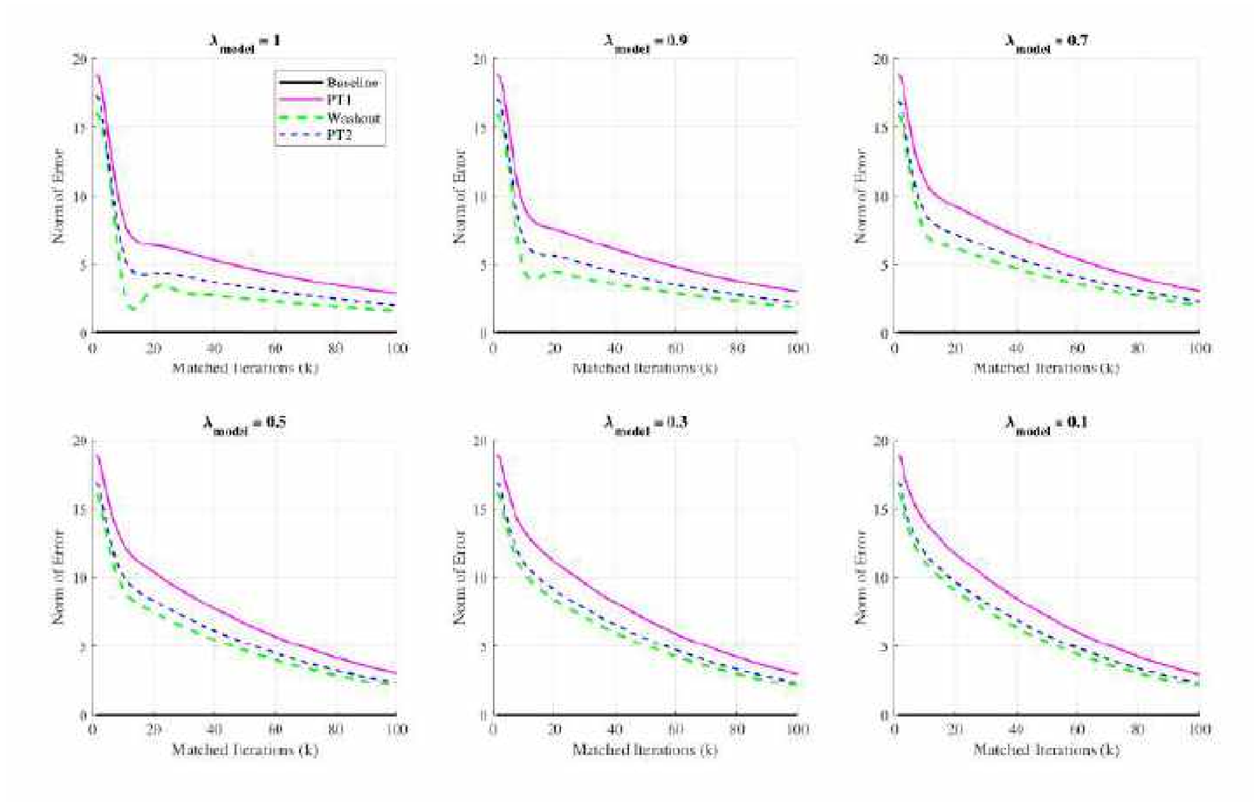


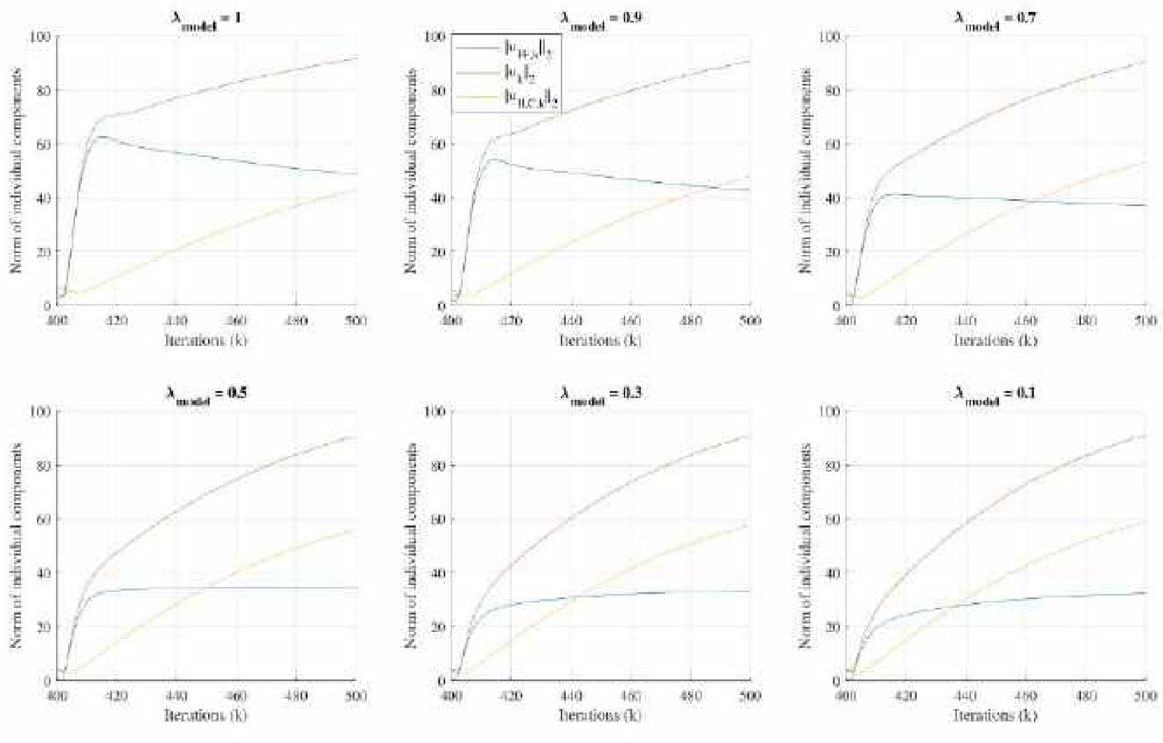
Observation: As λ_{FF} decreases, the contribution of U_{FF} decreases.



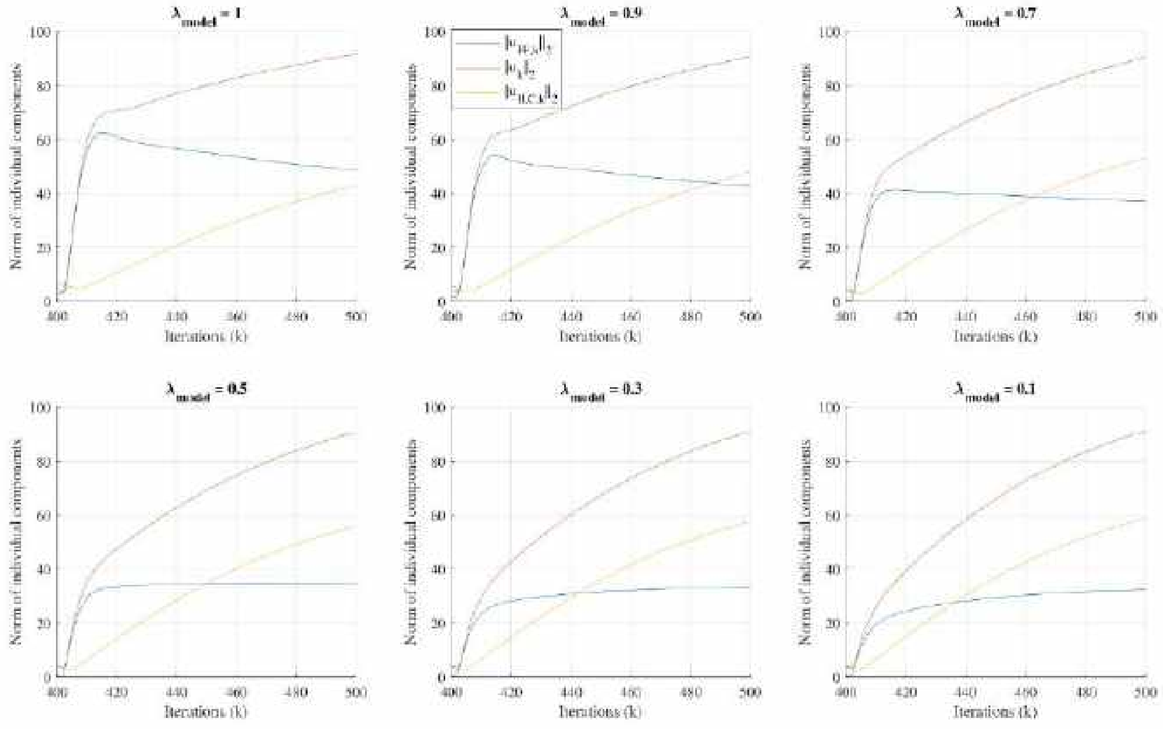
Observation: There isn’t much difference in initial norm except for really high values of λ_{FF} (close to 1). Retention increases as λ_{FF} decreases. Knee point is lower for higher values of λ_{FF} implying it’s using more of its savings for lower values of λ_{FF} for response .

3. Changing λ_{model}

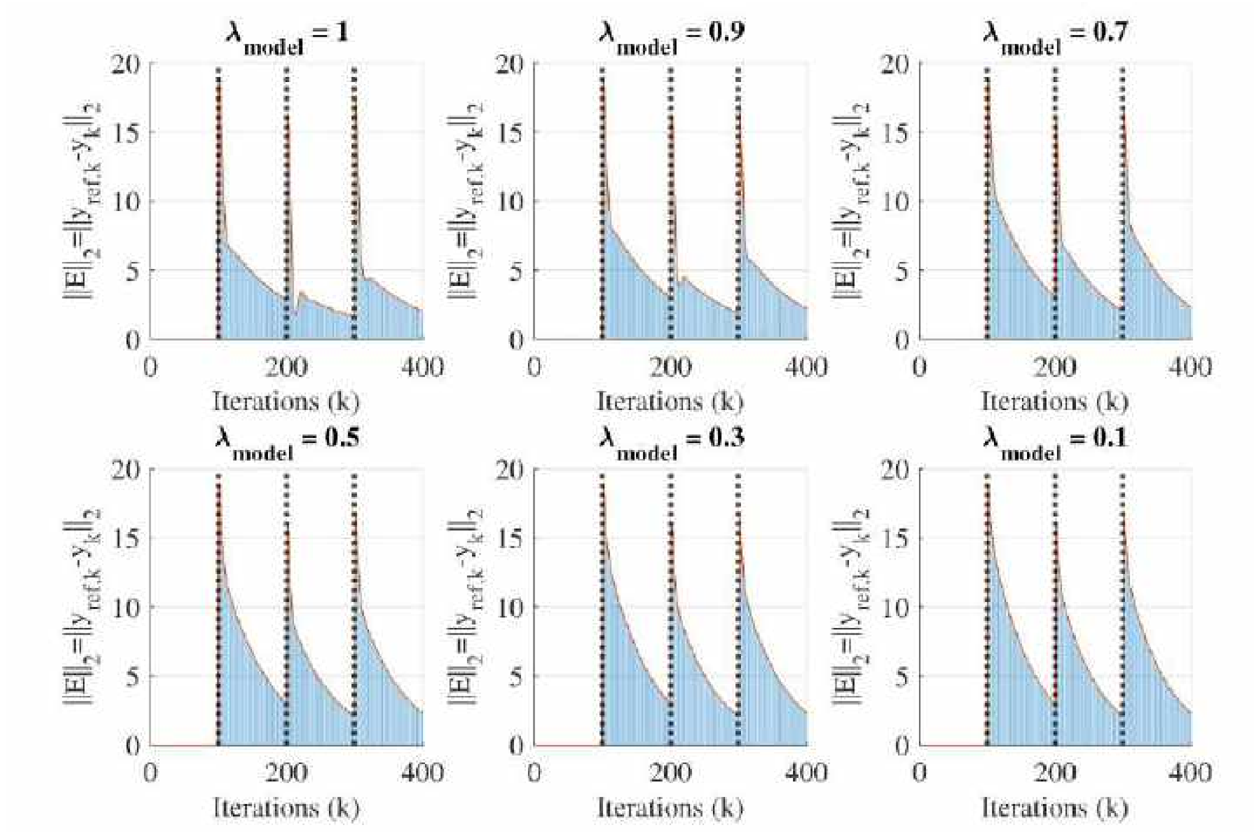




$\lambda_{model} = 1$
$\lambda_{model} = 0.9$
$\lambda_{model} = 0.7$
$\lambda_{model} = 0.5$
$\lambda_{model} = 0.3$
$\lambda_{model} = 0.1$
$\|u_{ff,k}\|_2$
$\|u_k\|_2$
$\|u_{ILC,k}\|_2$
Norm of individual components
Iterations (k)

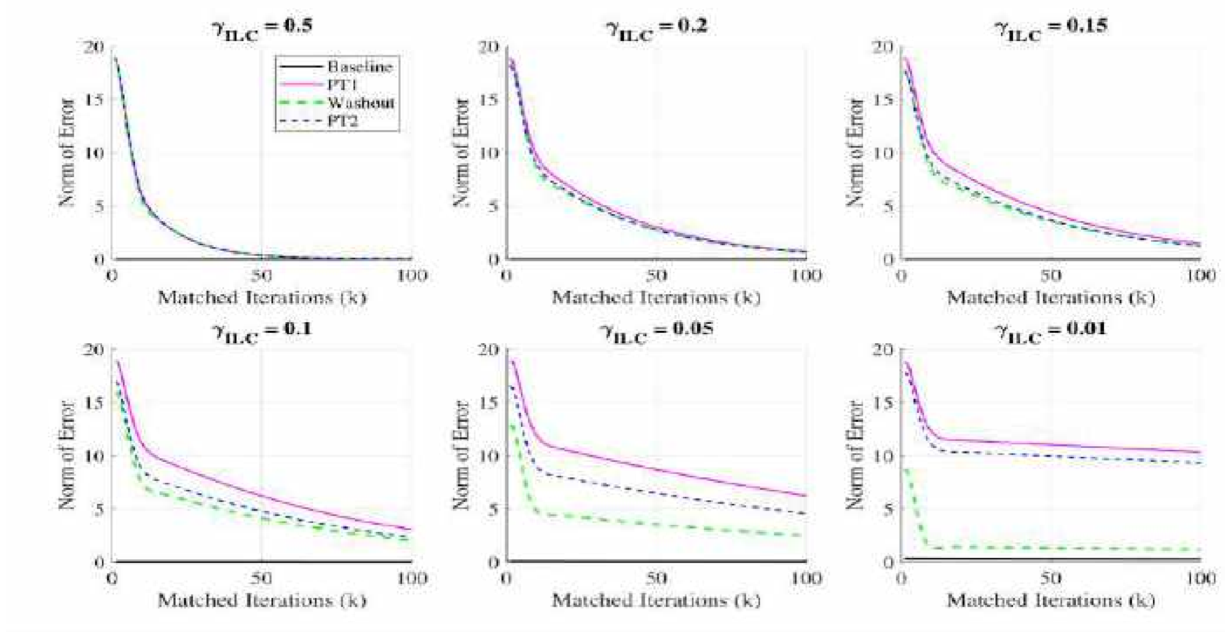


$\lambda_{model} = 1$
$\lambda_{model} = 0.9$
$\lambda_{model} = 0.7$
$\lambda_{model} = 0.5$
$\lambda_{model} = 0.3$
$\lambda_{model} = 0.1$
$\|u_{ff,k}\|_2$
$\|u_k\|_2$
$\|u_{ILC,k}\|_2$
Norm of individual components
Iterations (k)

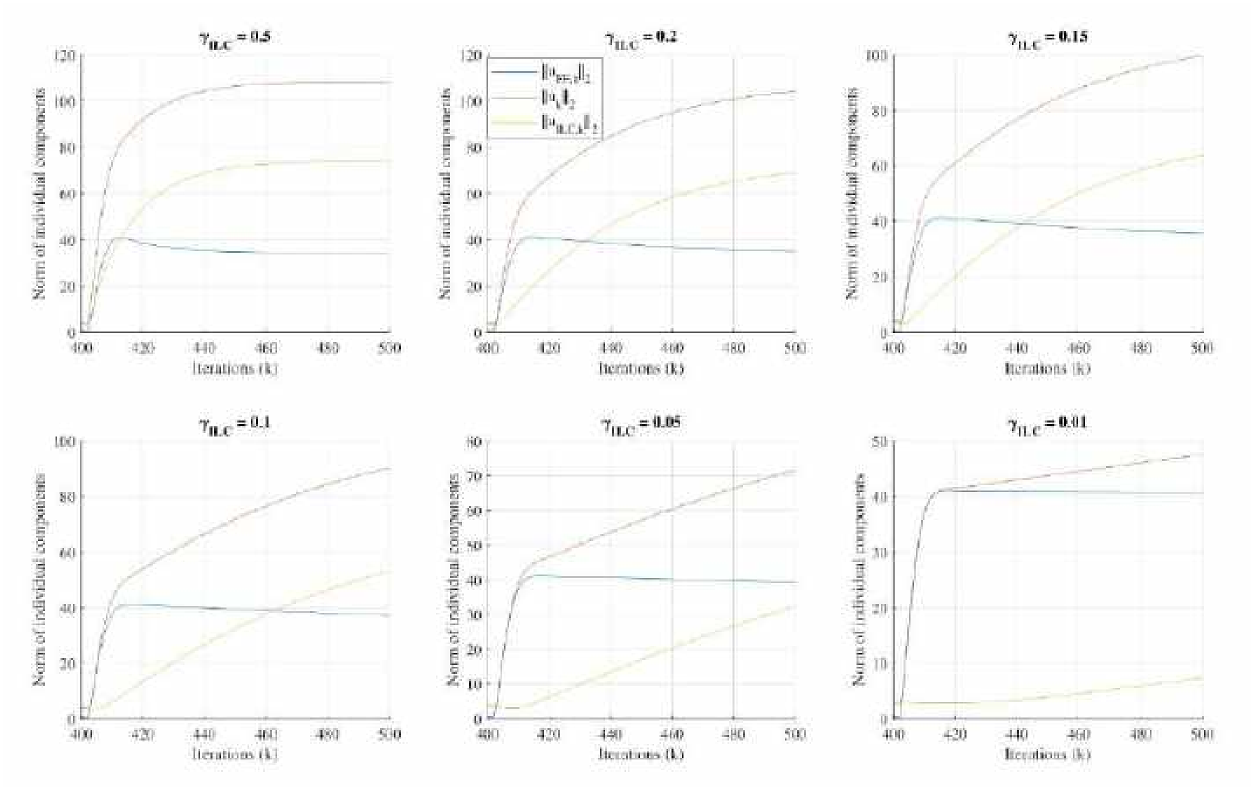


Observation: y_{model} represents how well the controller understands the environment. For higher values of the λ_{model}, the controller understands/ remembers the environment well and emphasizes the action of U_{FF} leading to a sharper knee curve.

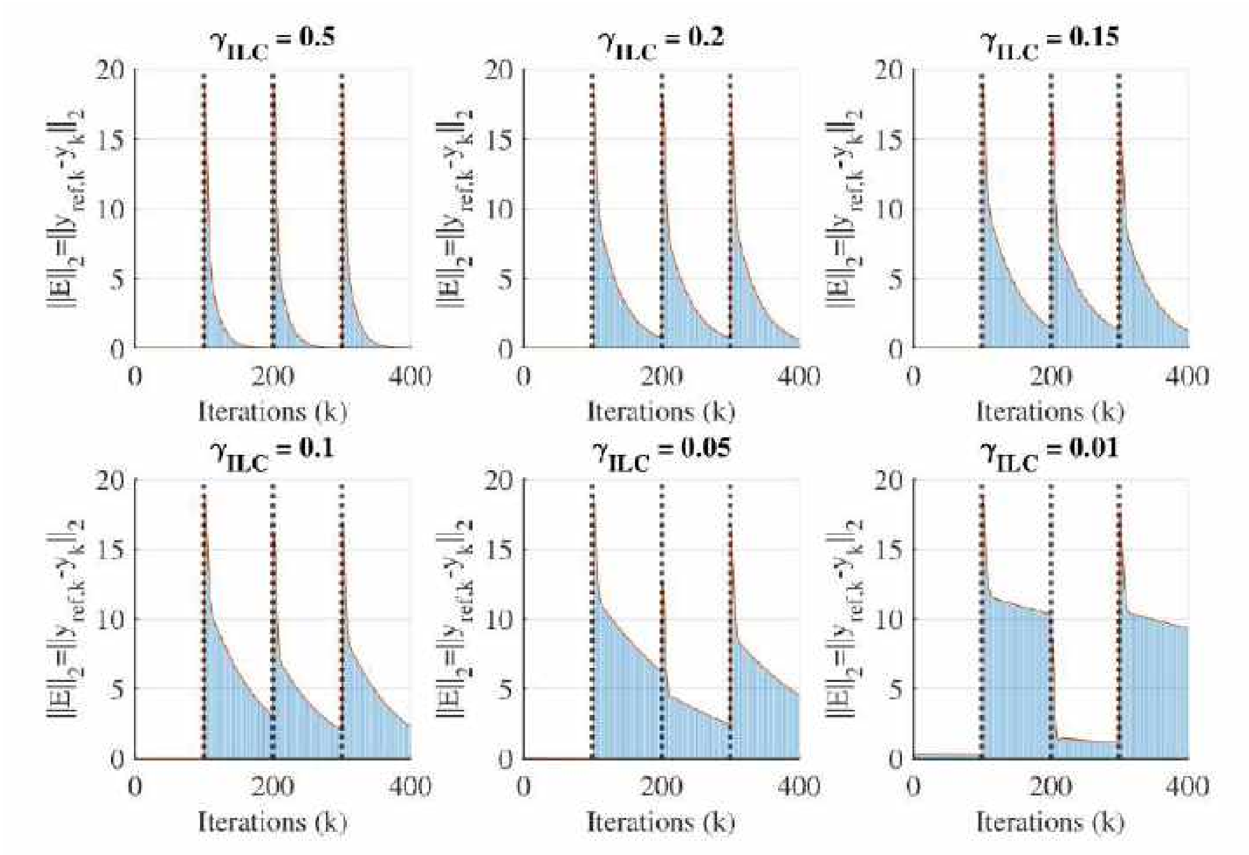
4. Changing γ_{ILC}:



Observation: For higher values of $\gamma_{\mathbf{ILC}}$, savings and retention are low because it gives more priority to the actual conditions and error obtained while performing the task. For lower values of $\gamma_{\mathbf{ILC}}$, higher savings and retention leads to completing tasks based on memory and not taking into account the feedback much. In general, higher $\gamma_{\mathbf{ILC}}$ implies lower savings and retention. Knee behavior is sharper for $\gamma_{\mathbf{ILC}}$.

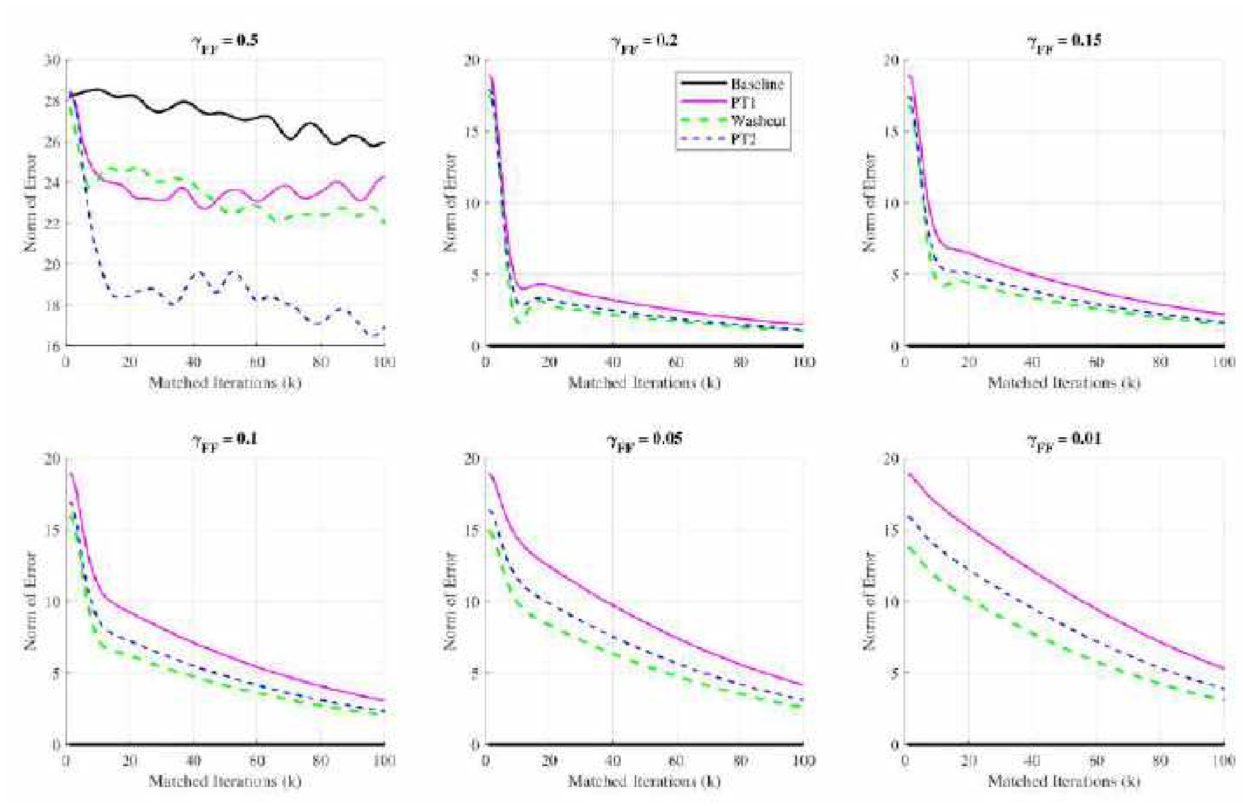


Observation: Contribution of U_{ILC} decreases as $\gamma_{\mathbf{ILC}}$ decreases. U_{FF} remains almost the same. It's not learning to adapt to its new environment since it's not taking feedback into much consideration.

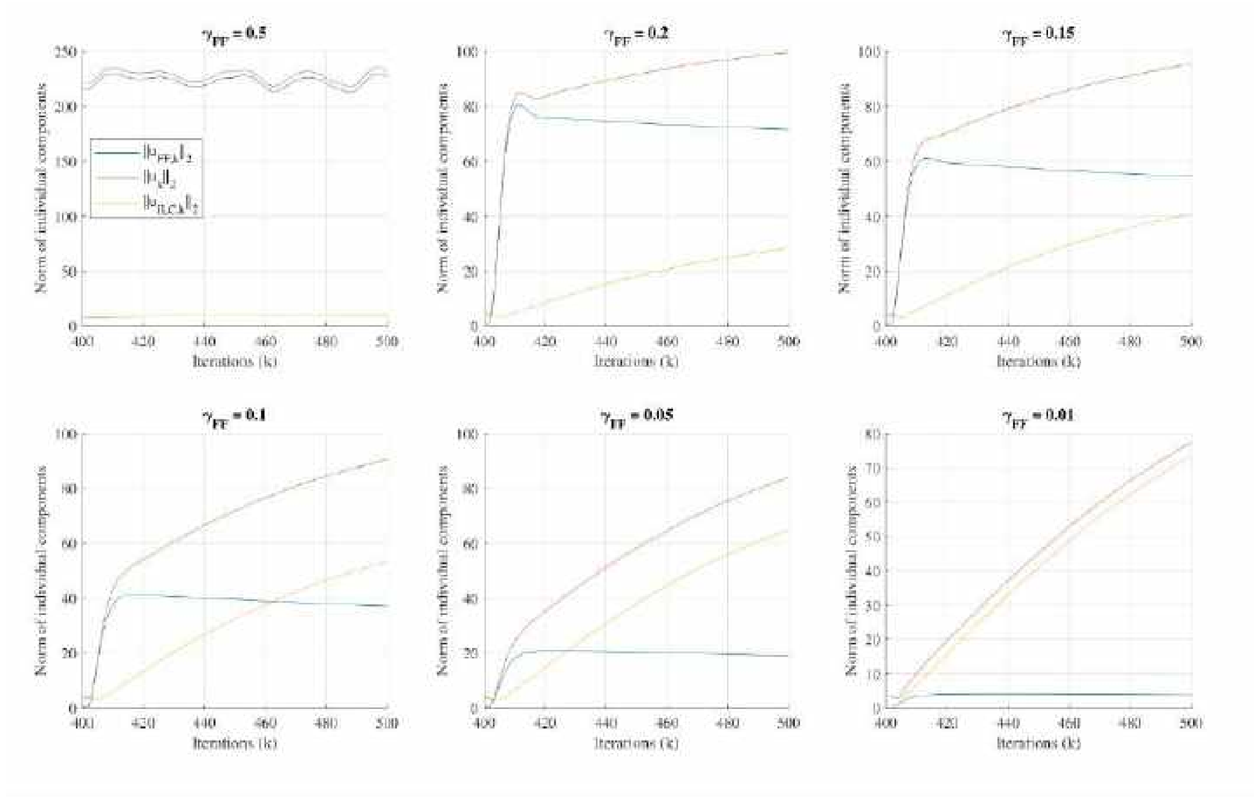


Observations: Initial error in washout trials decreases as γ_{ILC} decreases, because it is not really forgetting anything it learnt during the PT1 trials due to which it acts similarly in washout trials as it did during the baseline task. However, it's adapting faster to the new conditions as γ_{ILC} increases.

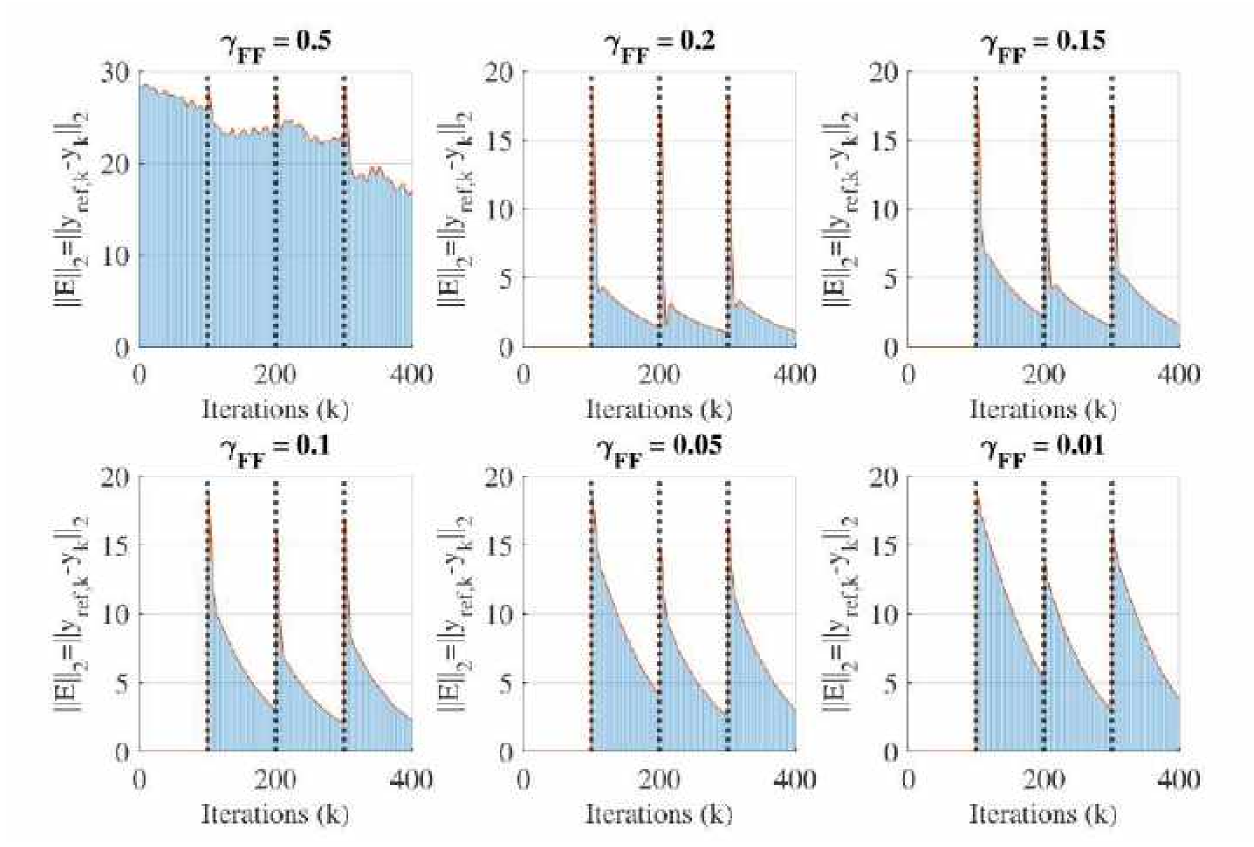
5. Changing γ_{FF}:



Observations: For very high values of γ_{FF}, system becomes unstable. With higher γ_{FF}, savings and retentions become low as it's reacting more to the change in environment rather than relying on its earlier learnings. Knee behavior becomes more significant for higher γ_{FF}.

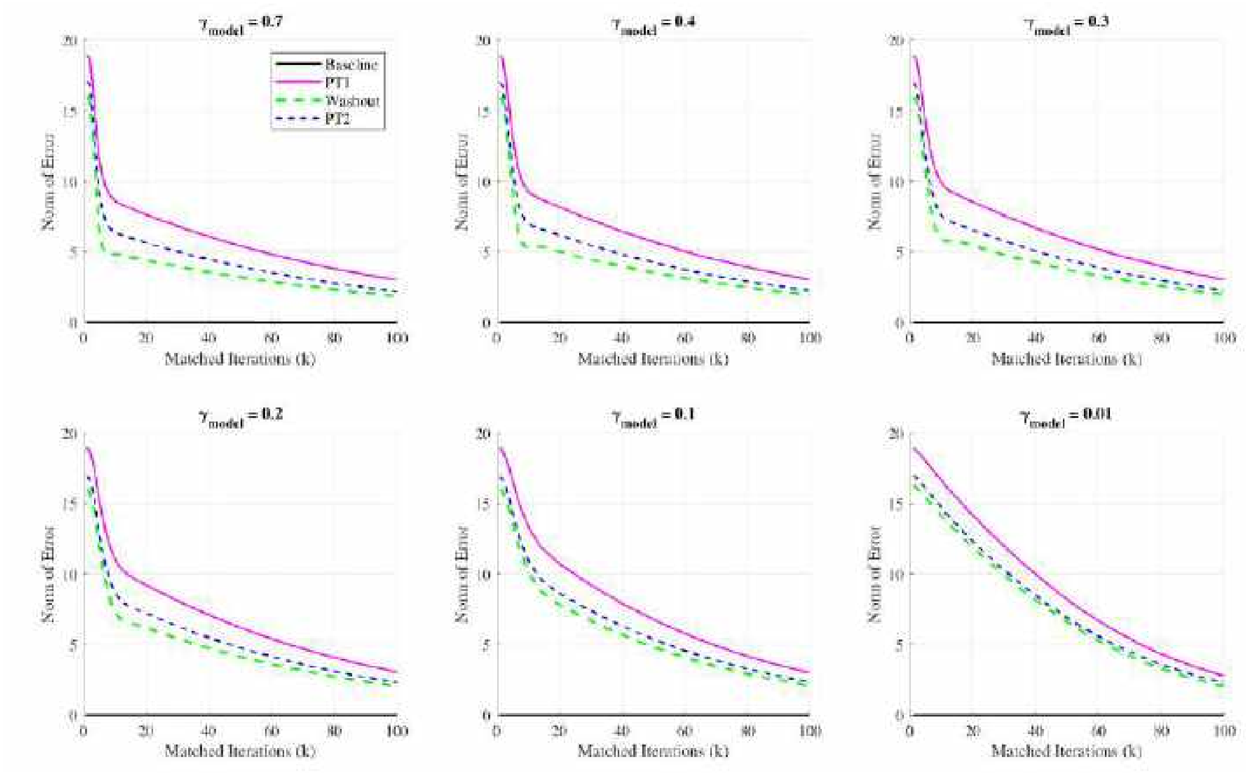


Observations: U_{FF} decreases as $\gamma_{\mathbf{FF}}$ decreases, as it's not reacting much to the change in environment. However, U_{ILC} is increasing now because U_{FF} isn't contributing much to U_K and hence large errors cause U_{ILC} to adapt accordingly.

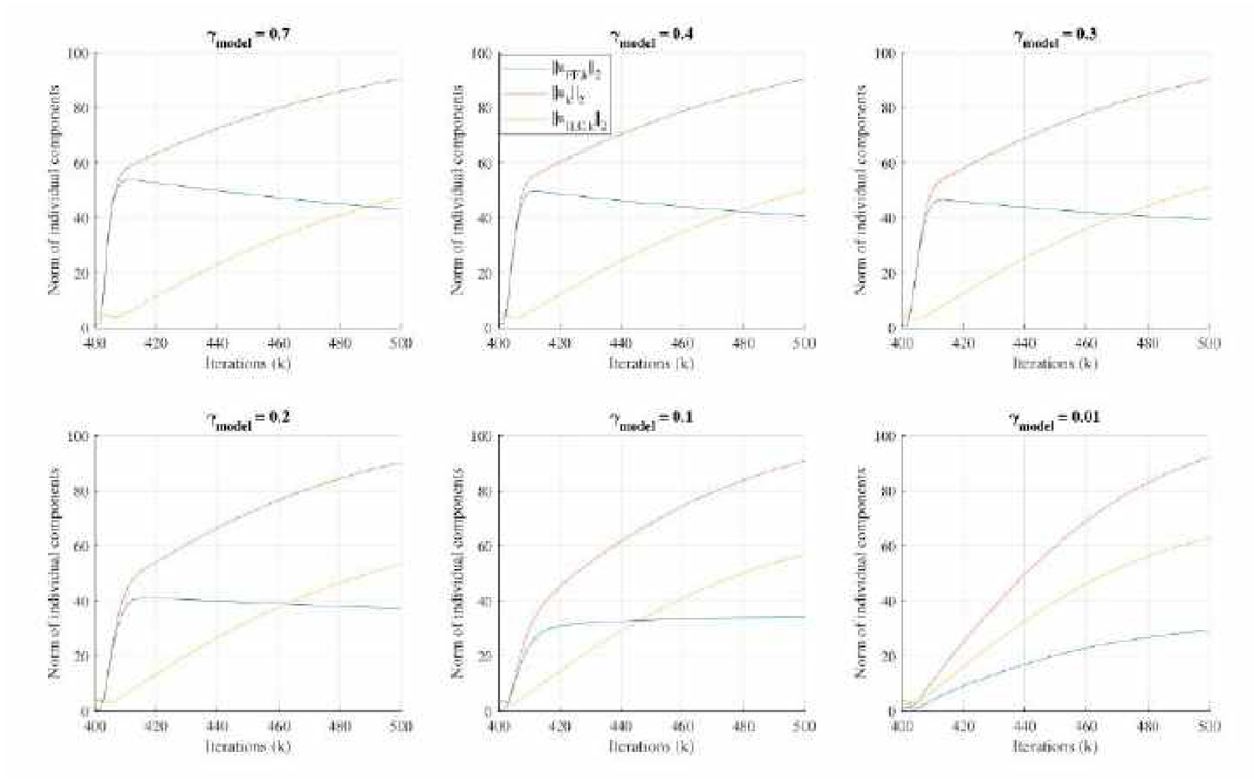


Observations: The slower rate of adaptation is faster for lower values of γ_{FF}. Knee point is lower for higher values of γ_{FF}.

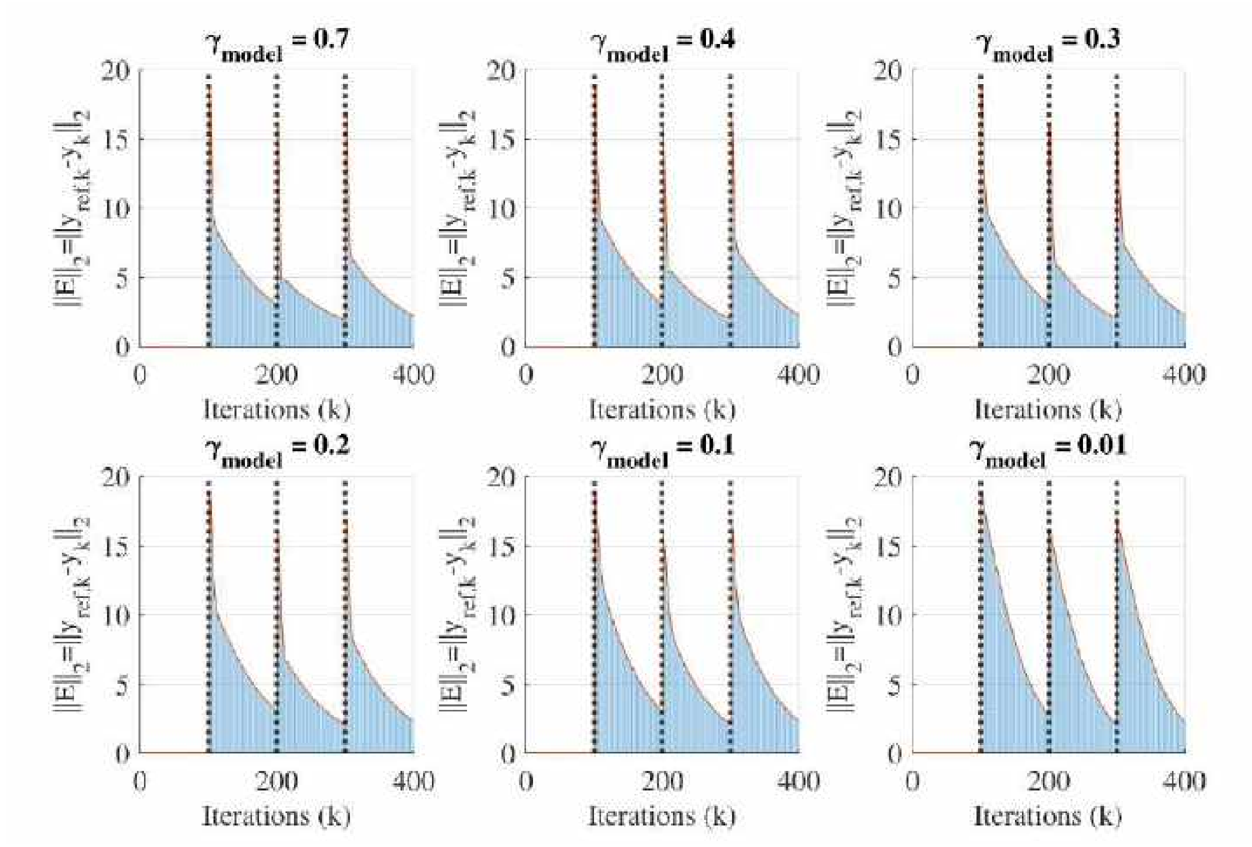
6. Changing γ_{model}:



Observations: γ_{model} doesn't affect the savings and retention much but it affects the knee behavior. Higher values of the γ_{model} show sharper knee behavior.



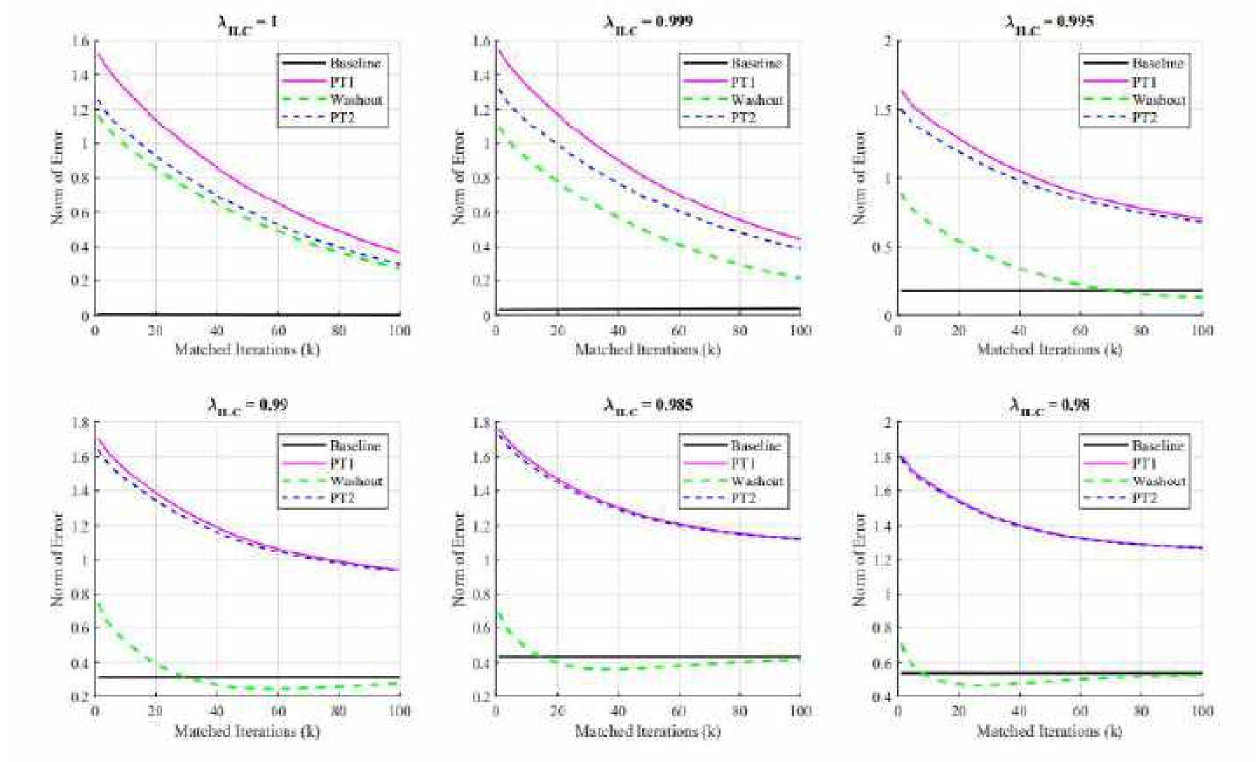
Observations: $\gamma_{\mathbf{model}}$ affects the nature and contribution of U_{FF}. Since y_{model} is a term in ε_{MOE}, higher $\gamma_{\mathbf{model}}$ implies a significant response of U_{FF}. However, for lower values of $\gamma_{\mathbf{model}}$, the lack of contribution of U_{FF} leads to U_{ILC} increasing at a faster rate.

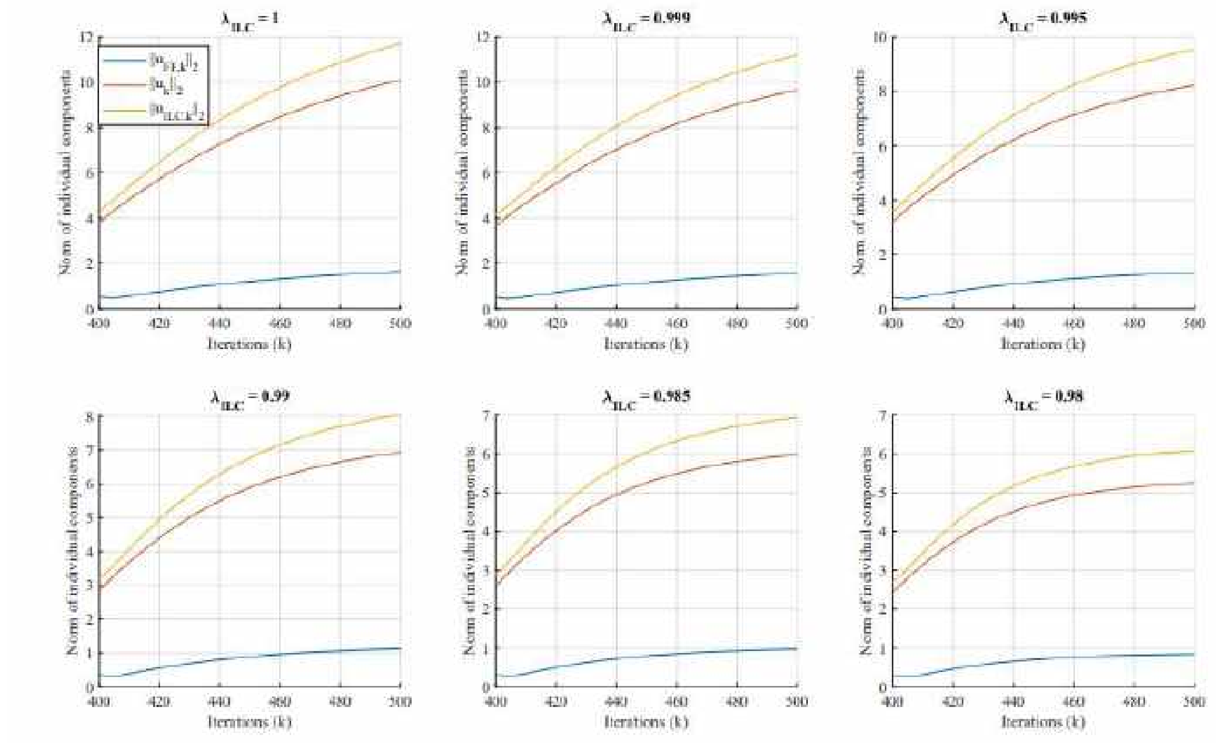


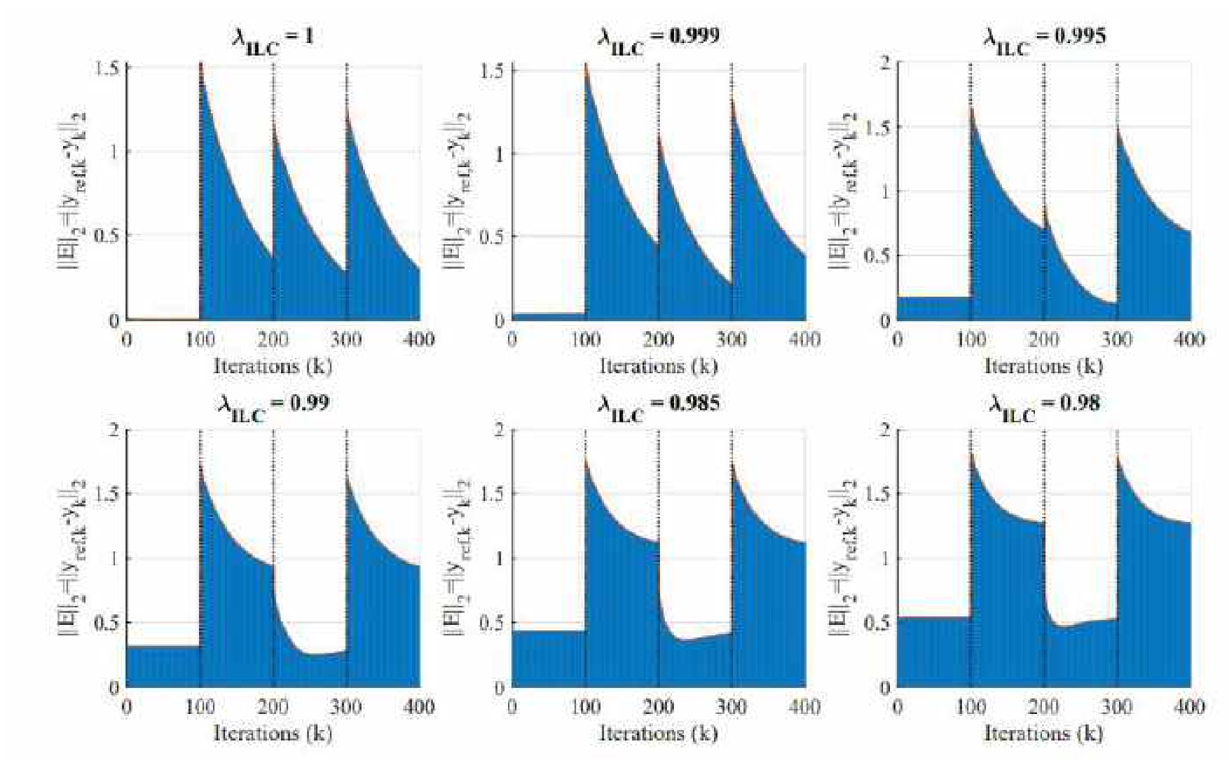
Observation: Slower rate of adaptation dominates most of the decayed response.

The second task was to analyze the same graphs when the destination was changed during the perturbations trials. Similar analysis on varying parameters was done. The following are the obtained results from the simulations:

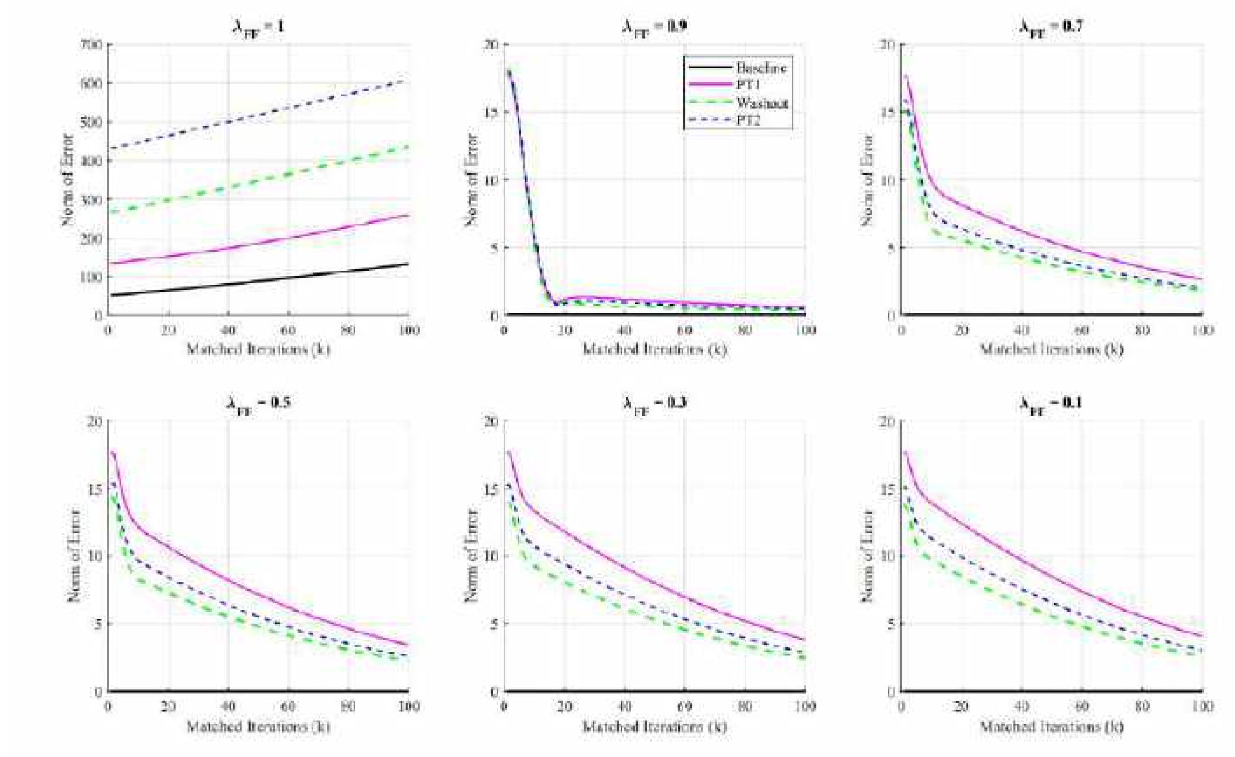
1. Changing λ_{ILC}

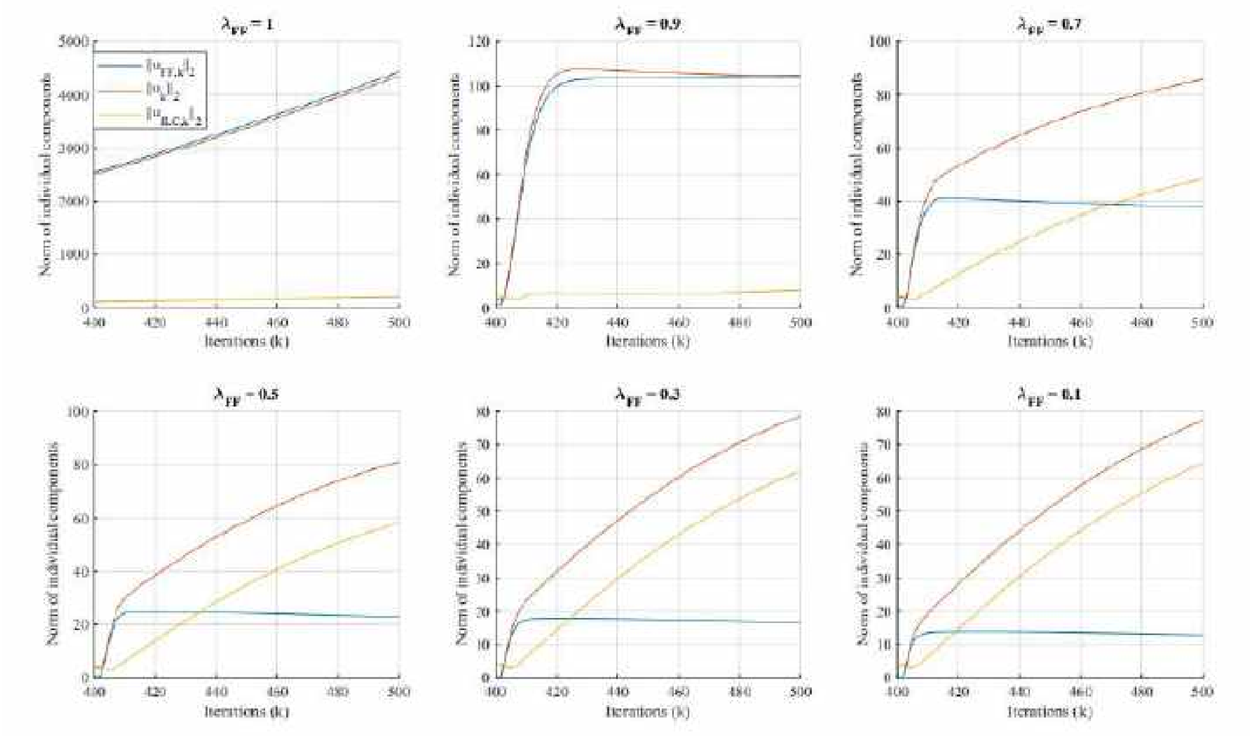




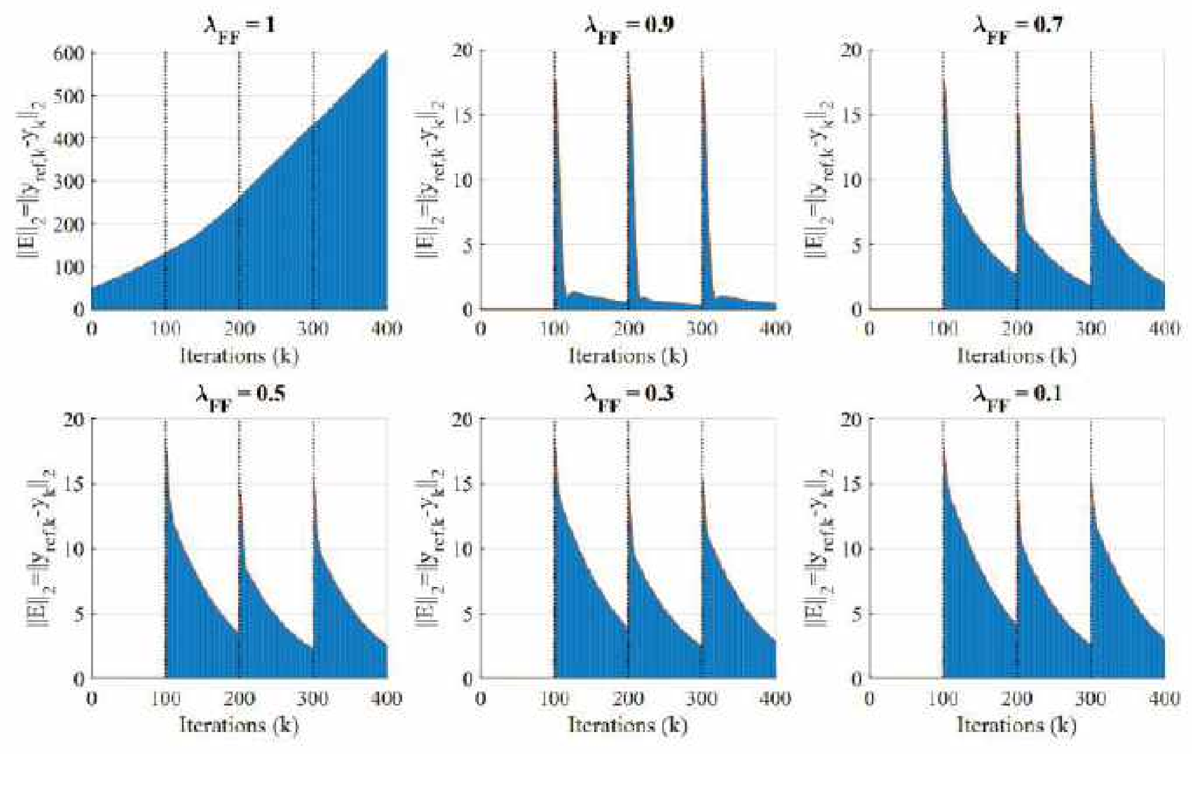


2. Changing λ_{FF}



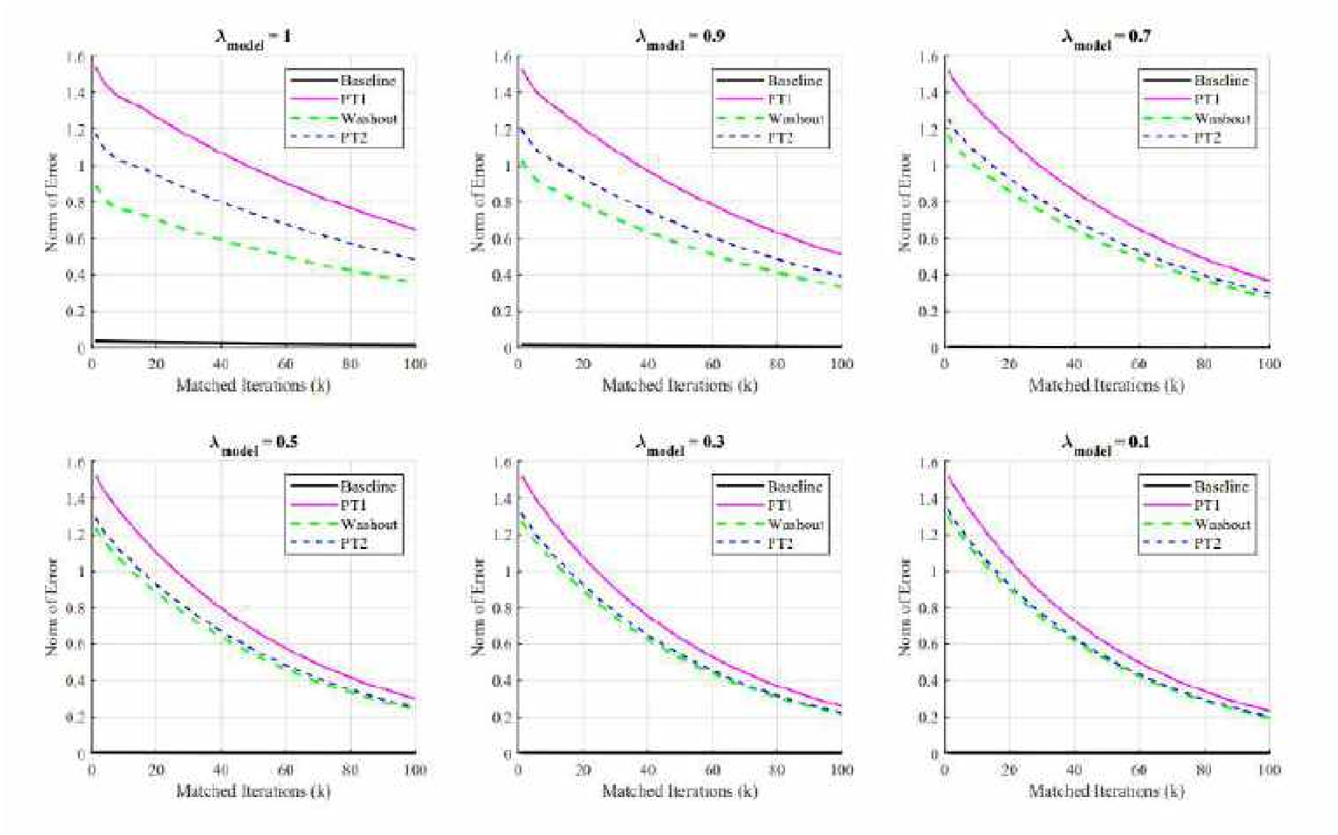


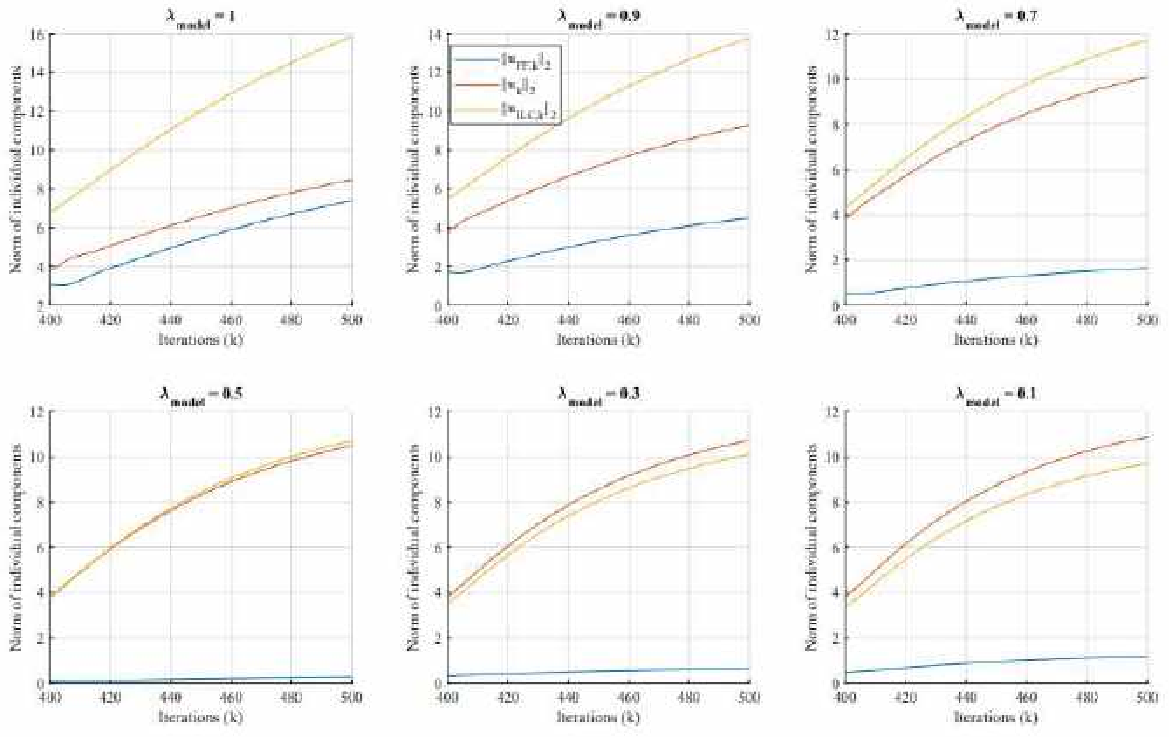
$\lambda_{FF} = 1$
$\lambda_{FF} = 0.9$
$\lambda_{FF} = 0.7$
$\lambda_{FF} = 0.5$
$\lambda_{FF} = 0.3$
$\lambda_{FF} = 0.1$
$\|u_{FF,k}\|_2$
$\|u_k\|_2$
$\|u_{ILC,k}\|_2$
Norm of individual components
Iterations (k)

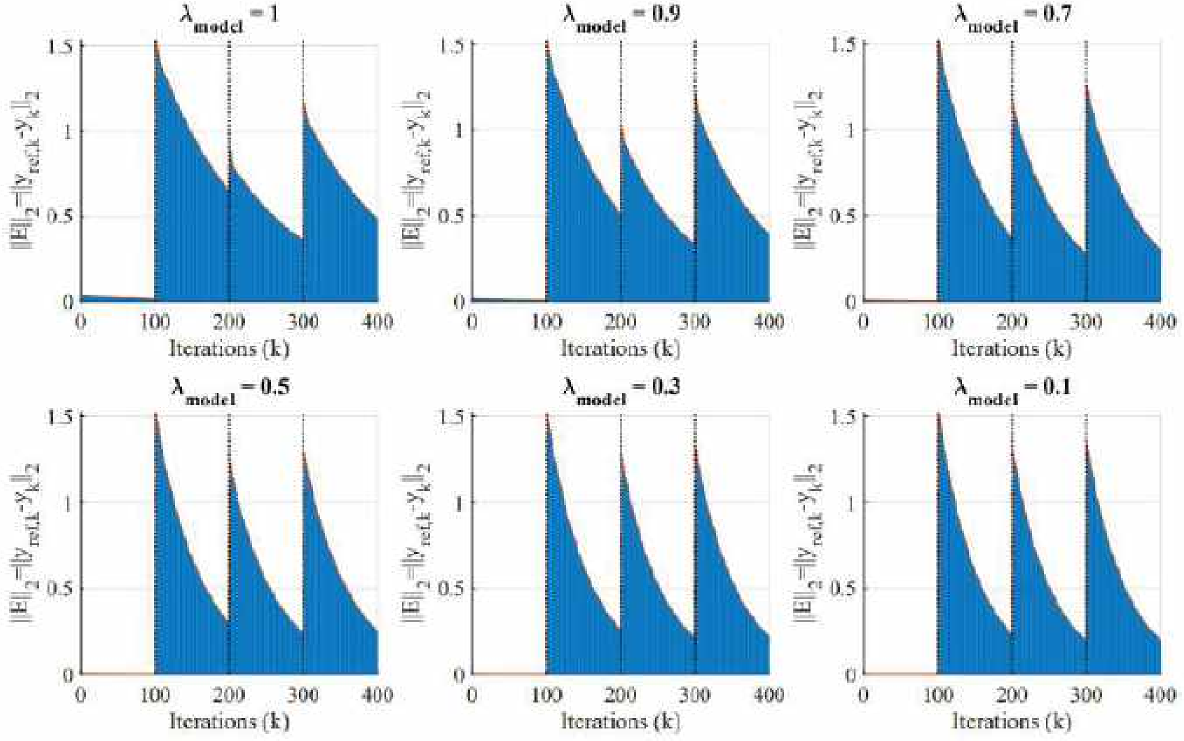


$\lambda_{FF} = 1$
$\lambda_{FF} = 0.9$
$\lambda_{FF} = 0.7$
$\lambda_{FF} = 0.5$
$\lambda_{FF} = 0.3$
$\lambda_{FF} = 0.1$
$\|E\|_2 = \|y_{ref,k} - y_k\|_2$
Iterations (k)

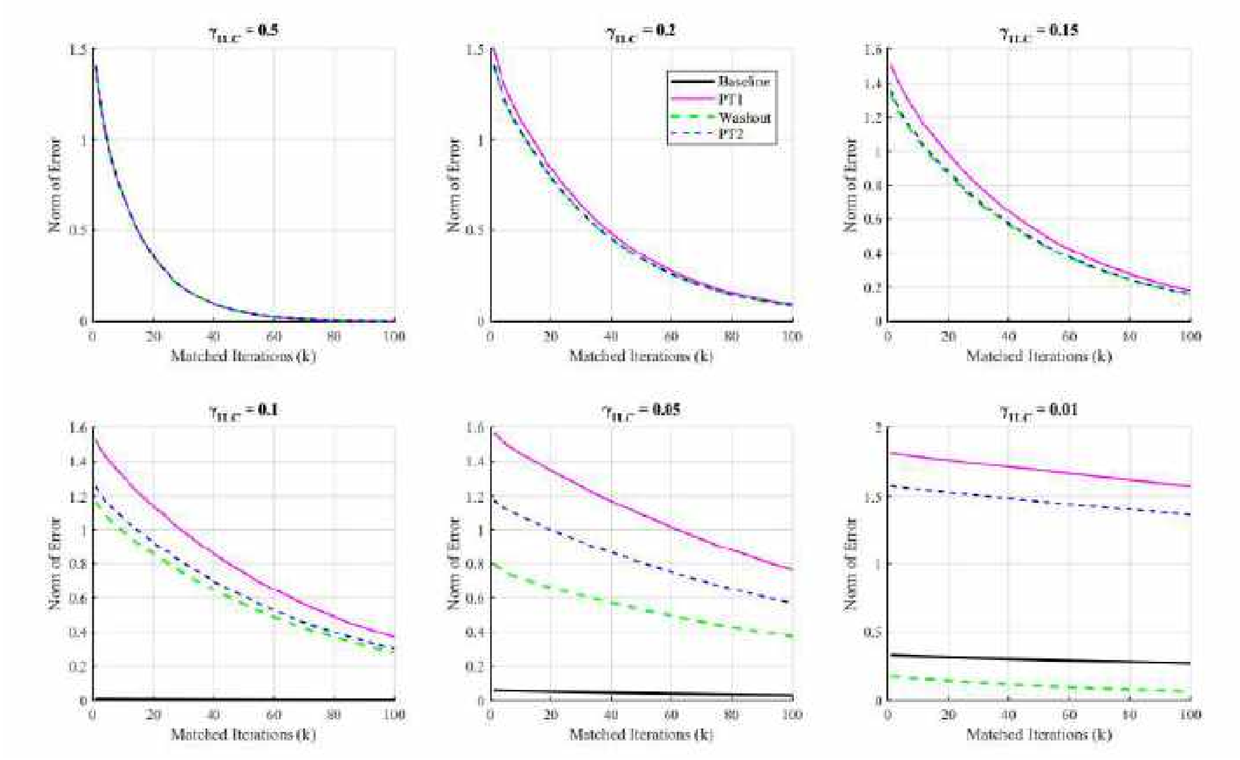
3. Changing λ_{model}

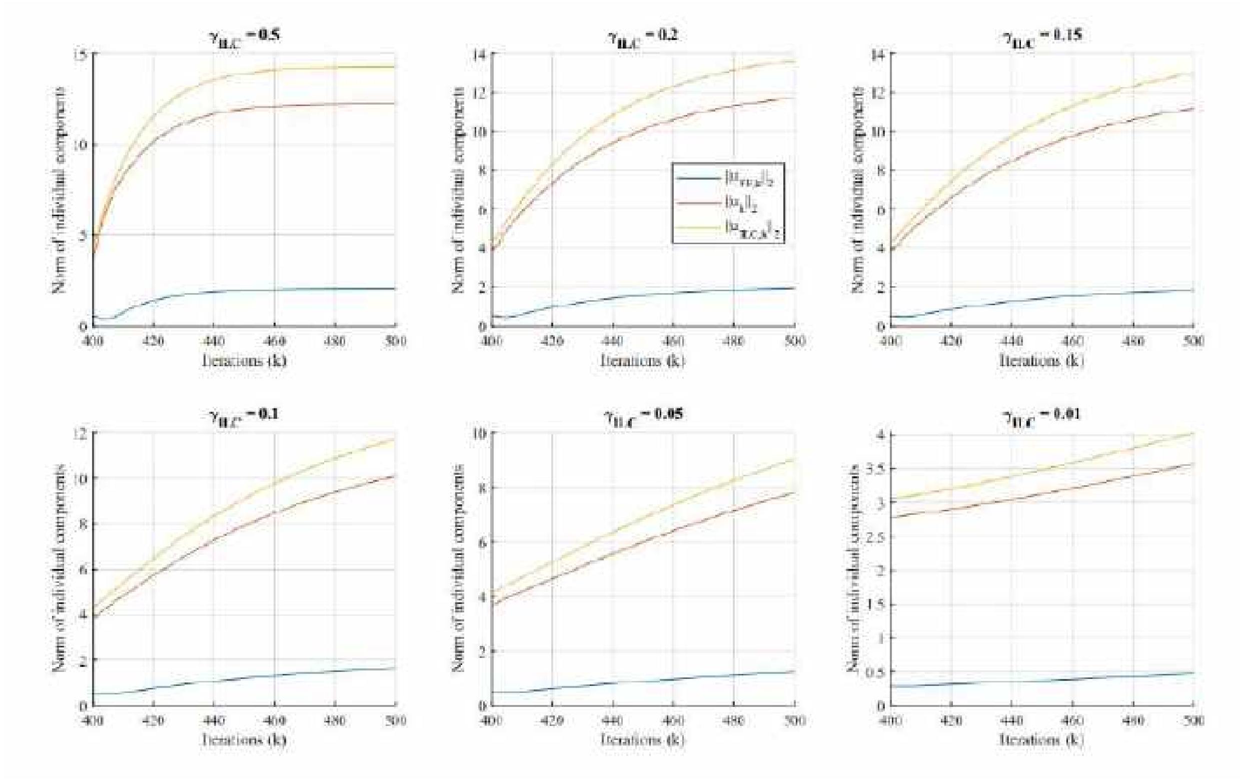


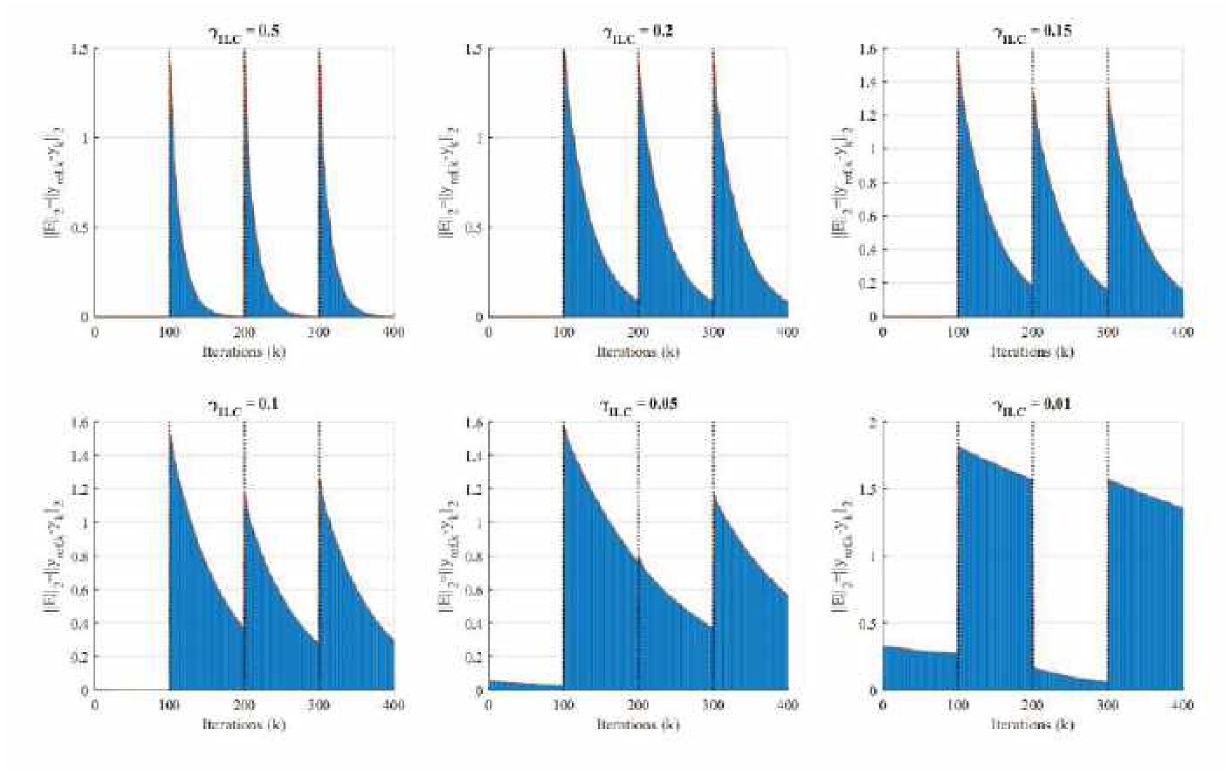

$\lambda_{model} = 1$
$\lambda_{model} = 0.9$
$\lambda_{model} = 0.7$
$\lambda_{model} = 0.5$
$\lambda_{model} = 0.3$
$\lambda_{model} = 0.1$
$\|u_{FF,k}\|_2$
$\|u_k\|_2$
$\|u_{ILC,k}\|_2$
Norm of individual components
Iterations (k)


$\lambda_{model} = 1$
$\lambda_{model} = 0.9$
$\lambda_{model} = 0.7$
$\lambda_{model} = 0.5$
$\lambda_{model} = 0.3$
$\lambda_{model} = 0.1$
$\|E\|_2 = \|y_{ref,k} - y_k\|_2$
Iterations (k)

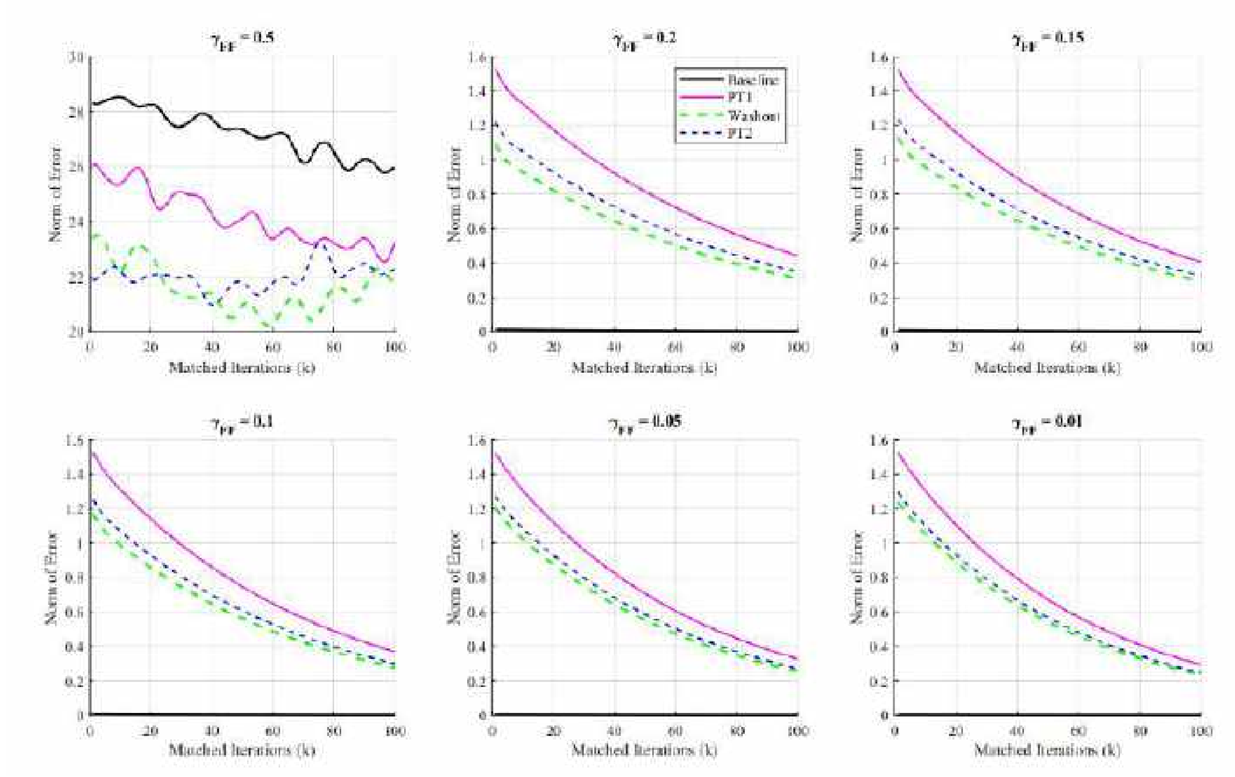
4. Changing γ_{ILC}:

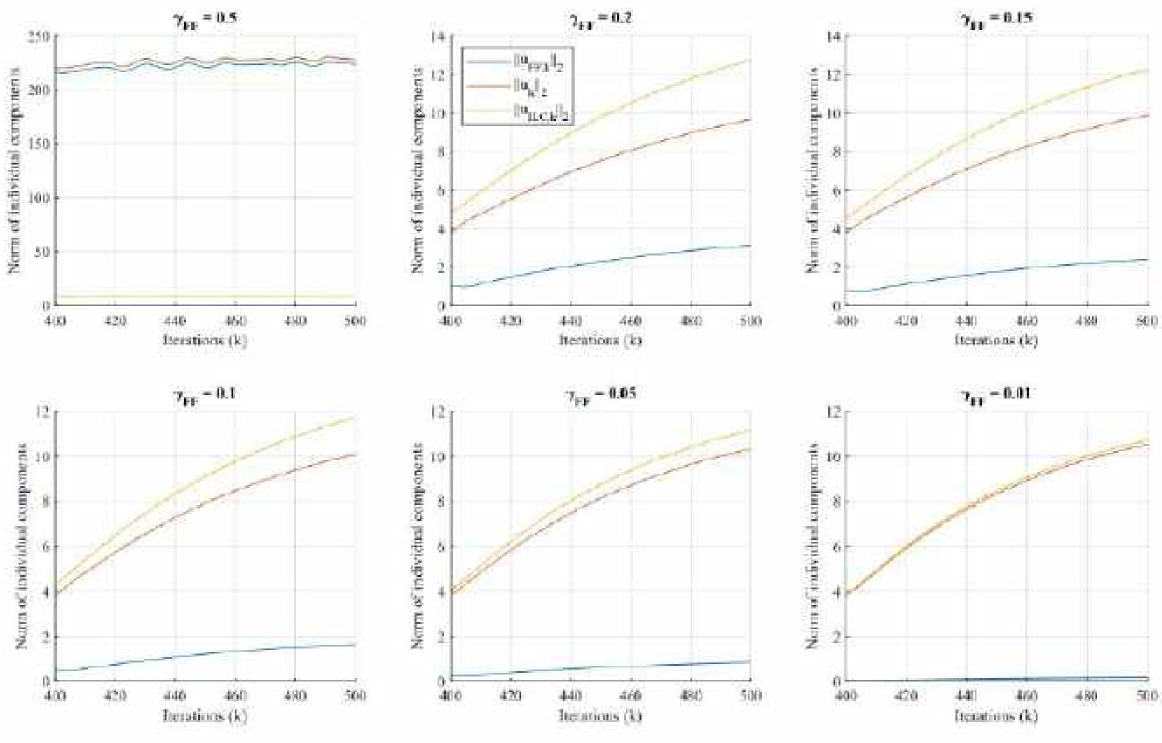




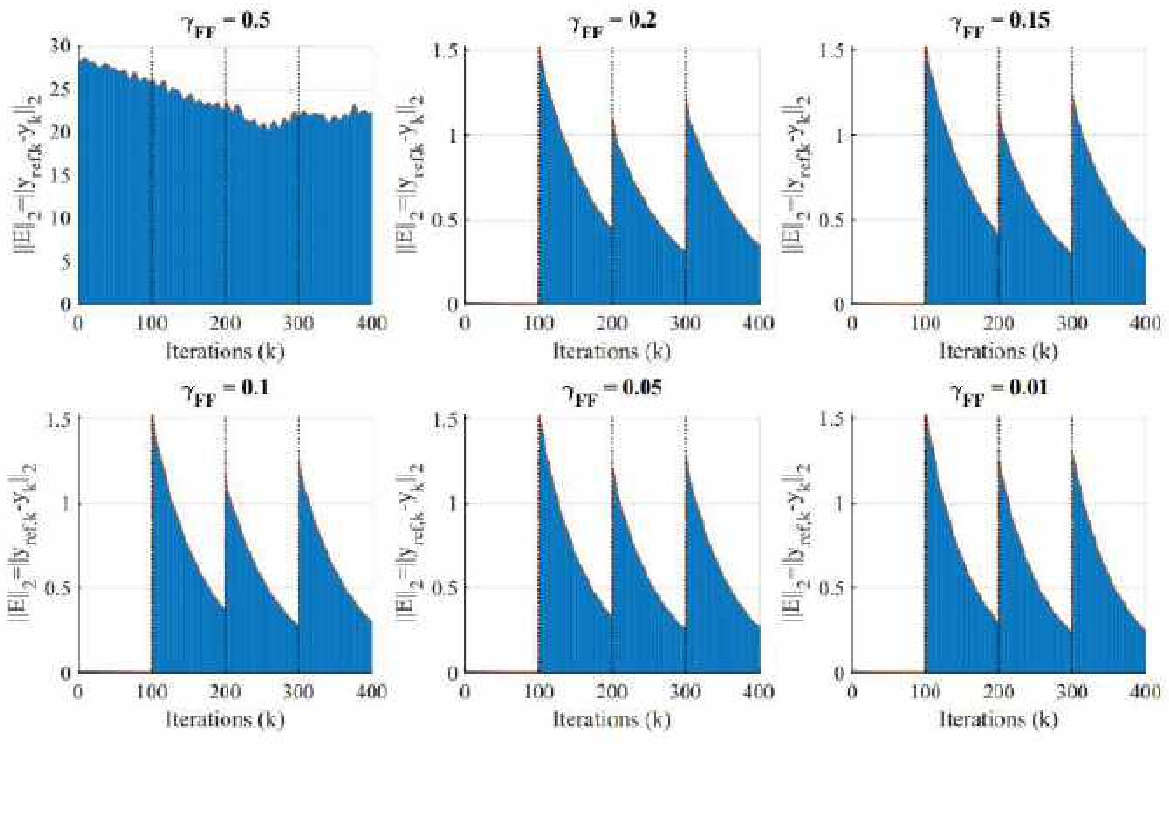


5. Changing γ_{FF}:



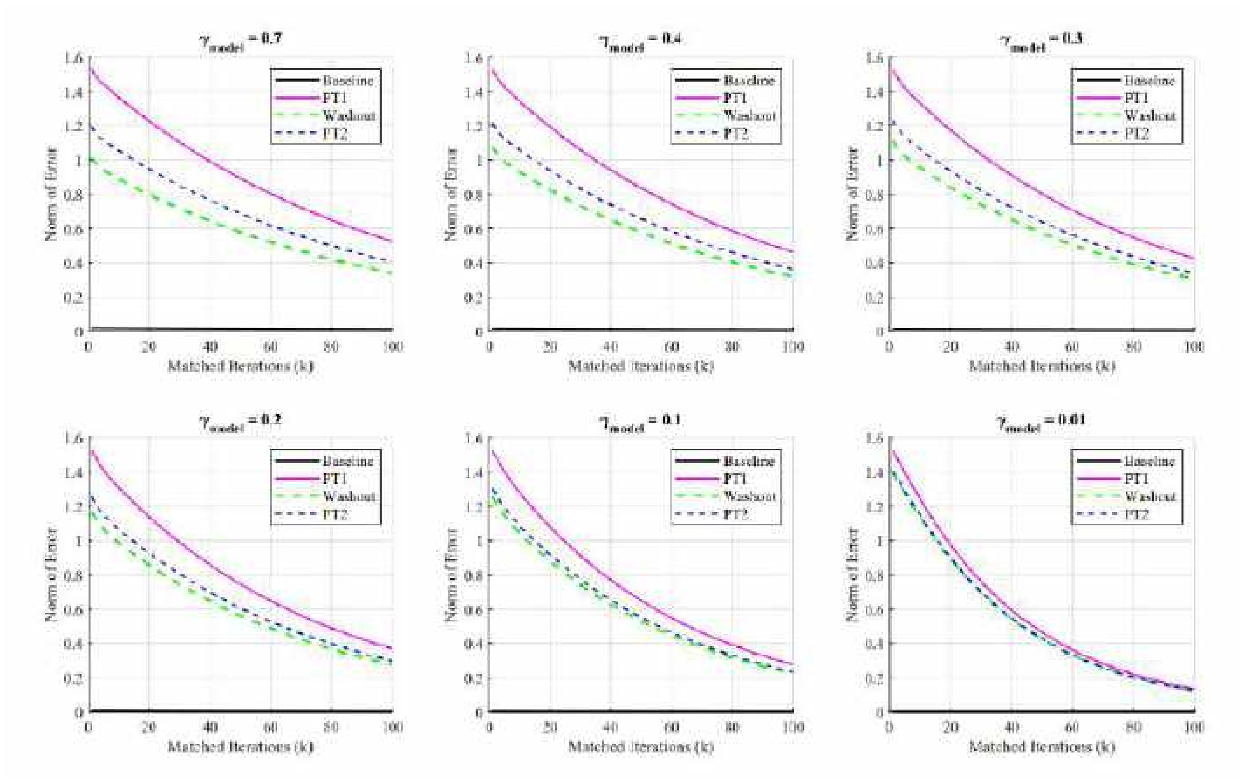


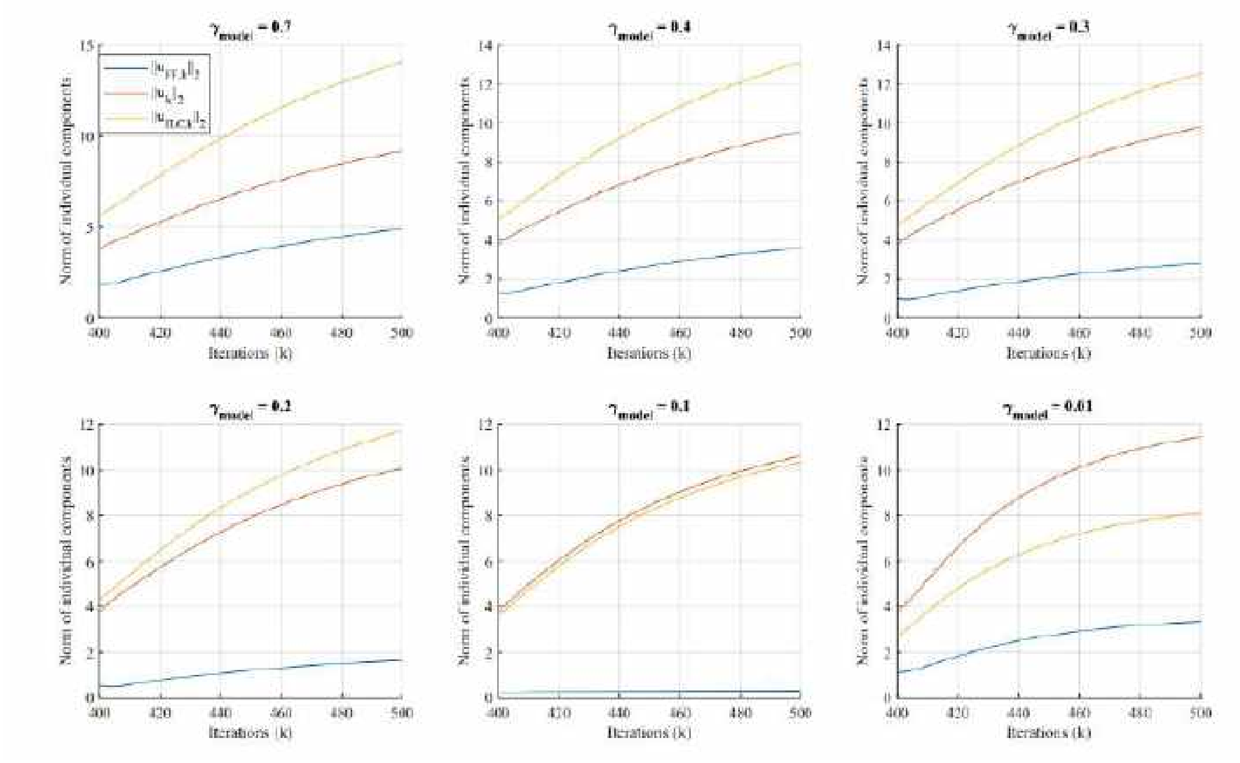
γ_{FF} = 0.5
γ_{FF} = 0.2
γ_{FF} = 0.15
γ_{FF} = 0.1
γ_{FF} = 0.05
γ_{FF} = 0.01
$||u_{FF,k}||_2$
$||u_k||_2$
$||u_{ILC,k}||_2$
Norm of individual components
Iterations (k)

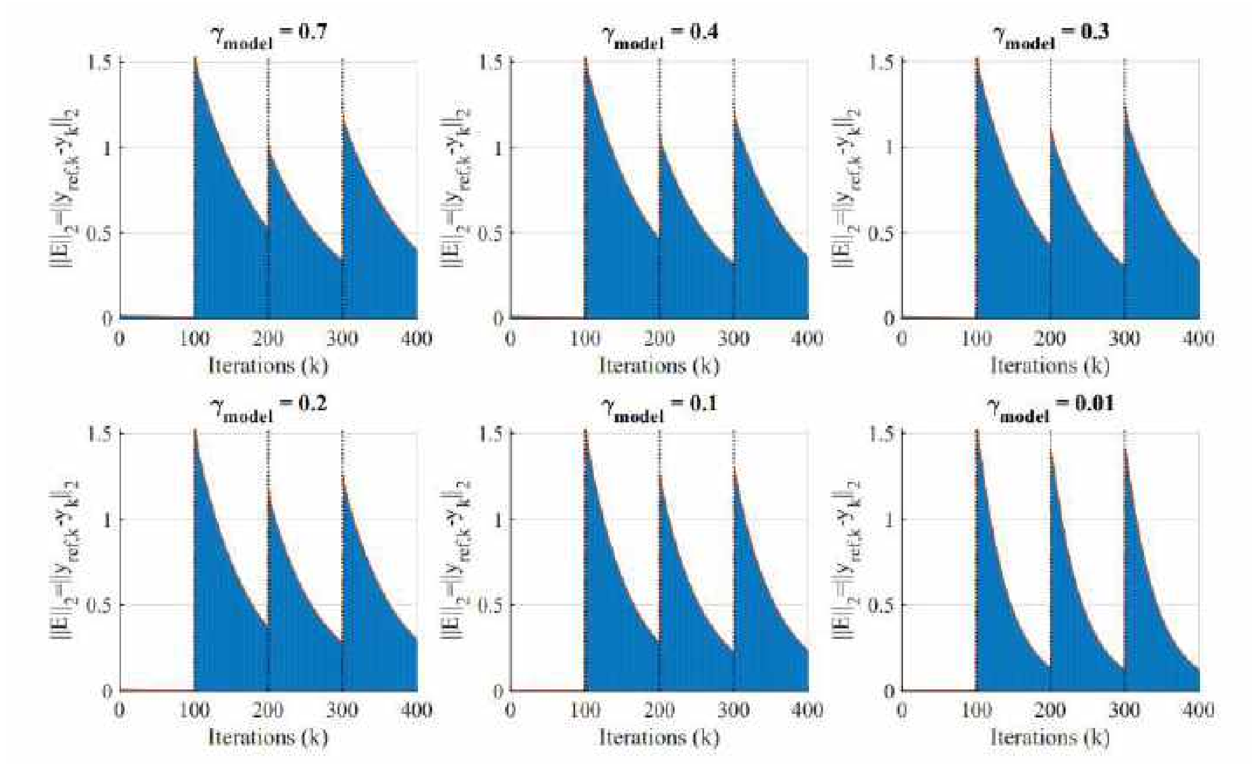


γ_{FF} = 0.5
γ_{FF} = 0.2
γ_{FF} = 0.15
γ_{FF} = 0.1
γ_{FF} = 0.05
γ_{FF} = 0.01
$||E||_2 = ||y_{ref,k} - y_k||_2$
Iterations (k)

6. Changing γ_{model}:

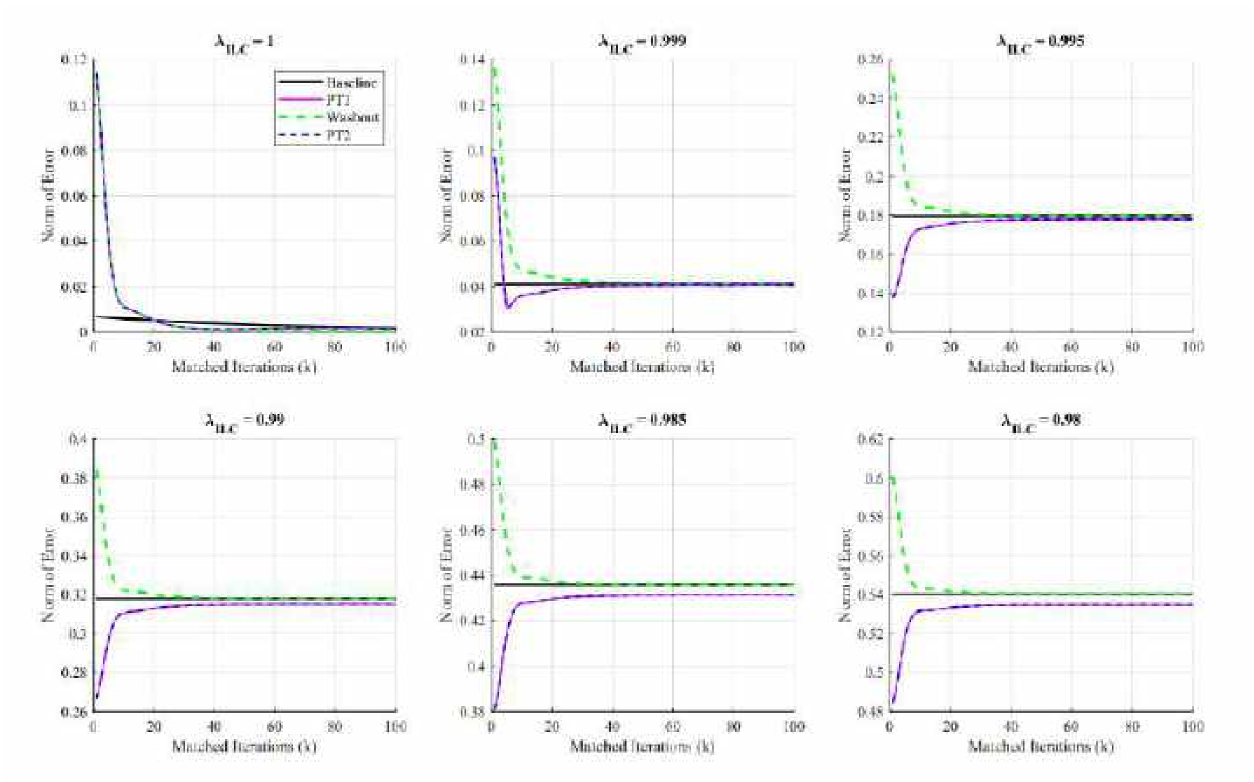


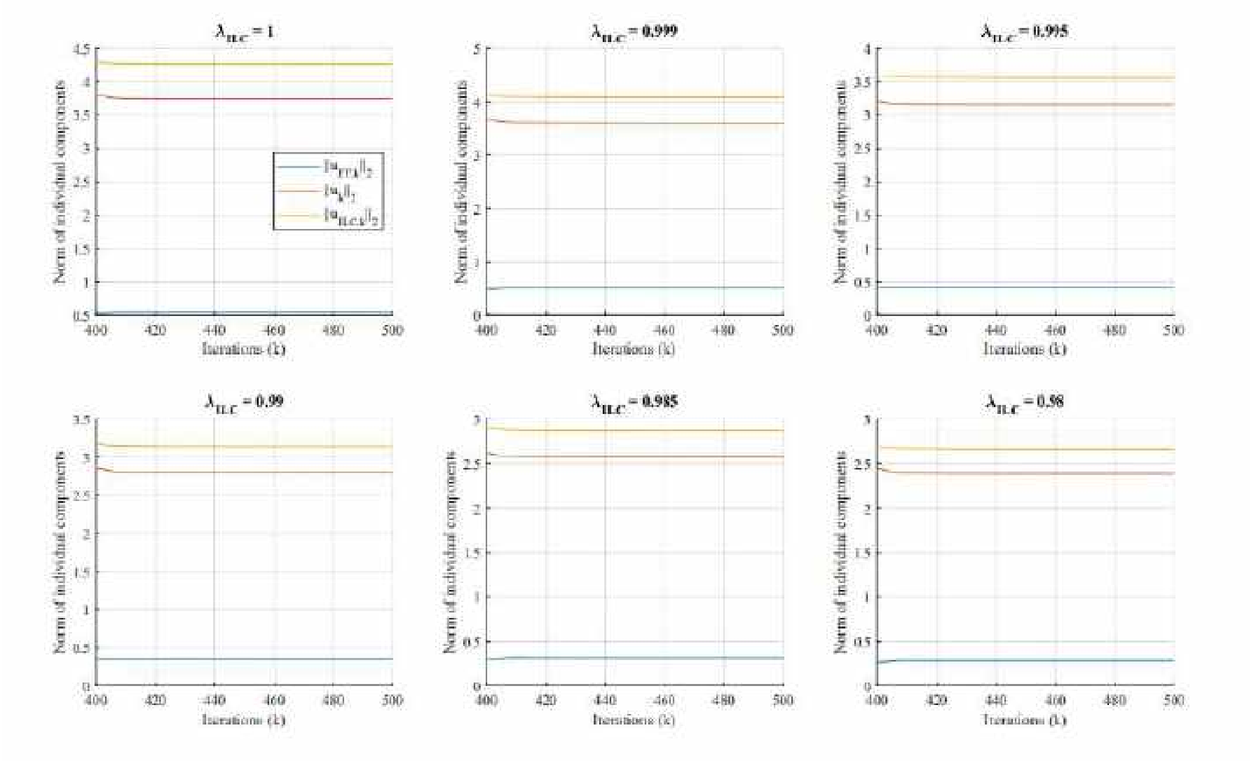


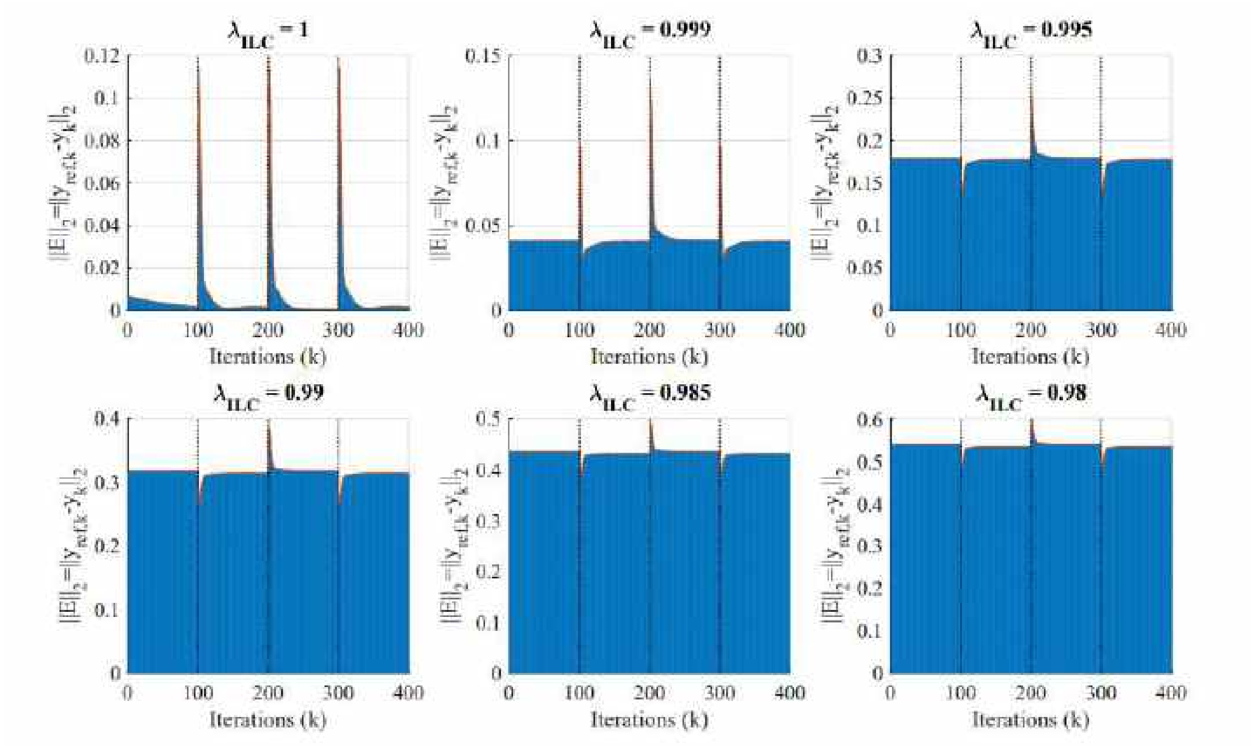


The third task was to establish a velocity based perturbation for the perturbation trials. Velocity based perturbations involve force fields that vary with varying velocity. Here again, similar simulations were done for different values of the parameters:

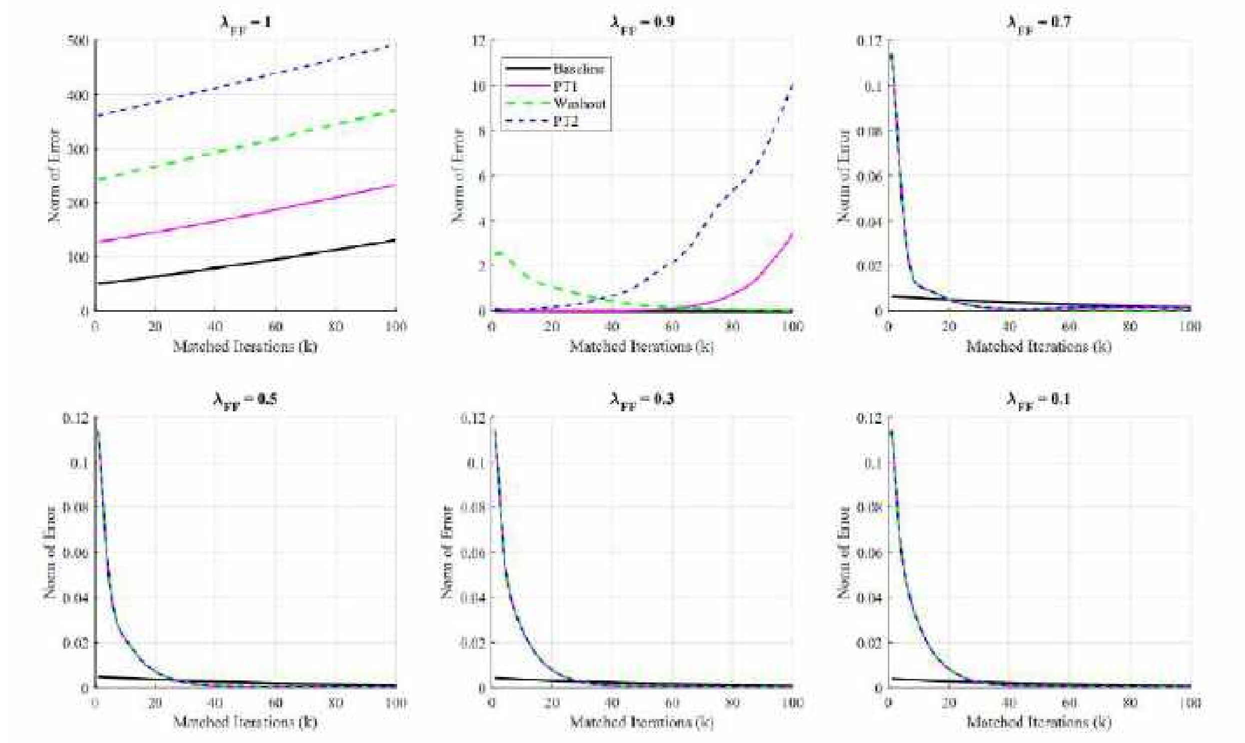
1. Changing λ_{ILC}

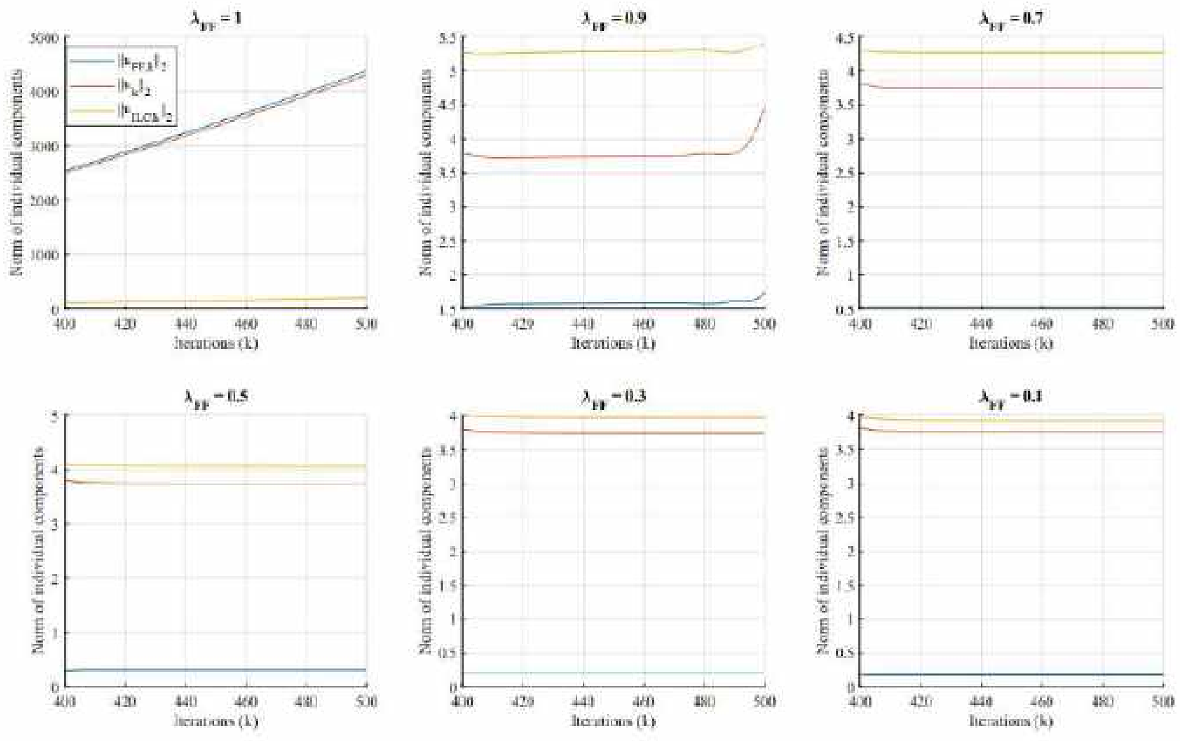




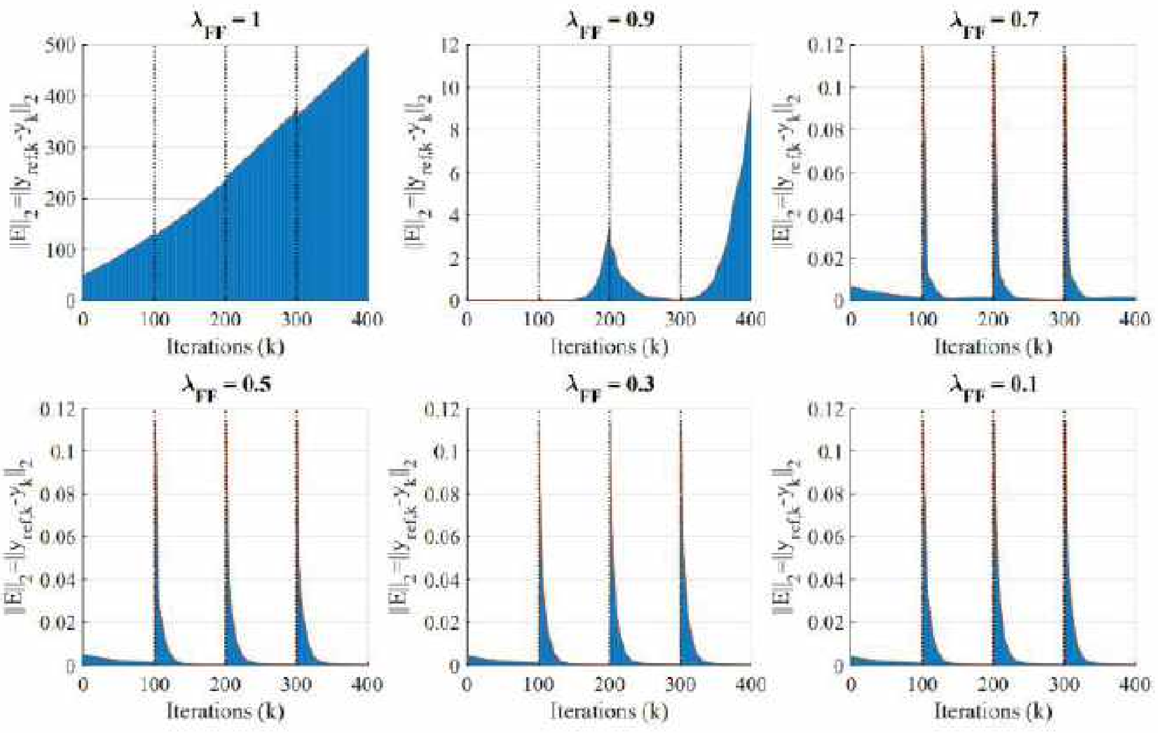


2. Changing λ_{FF}



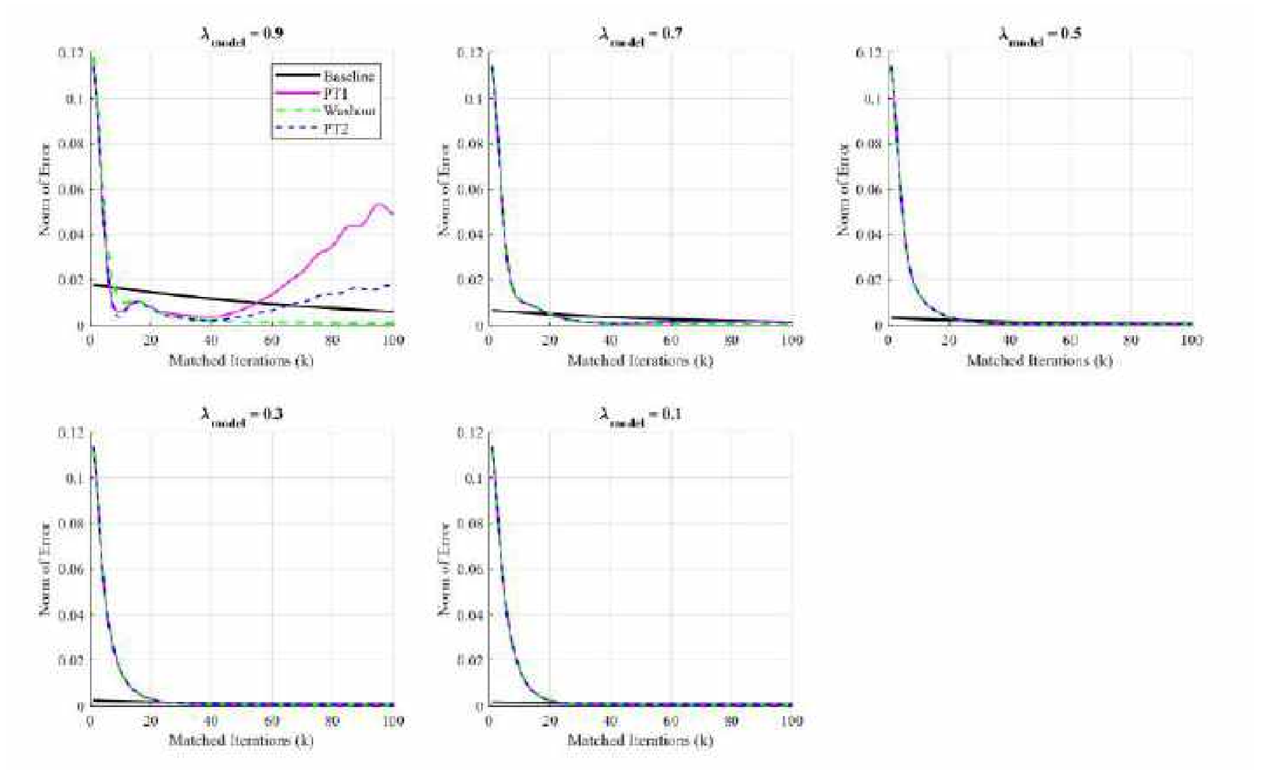


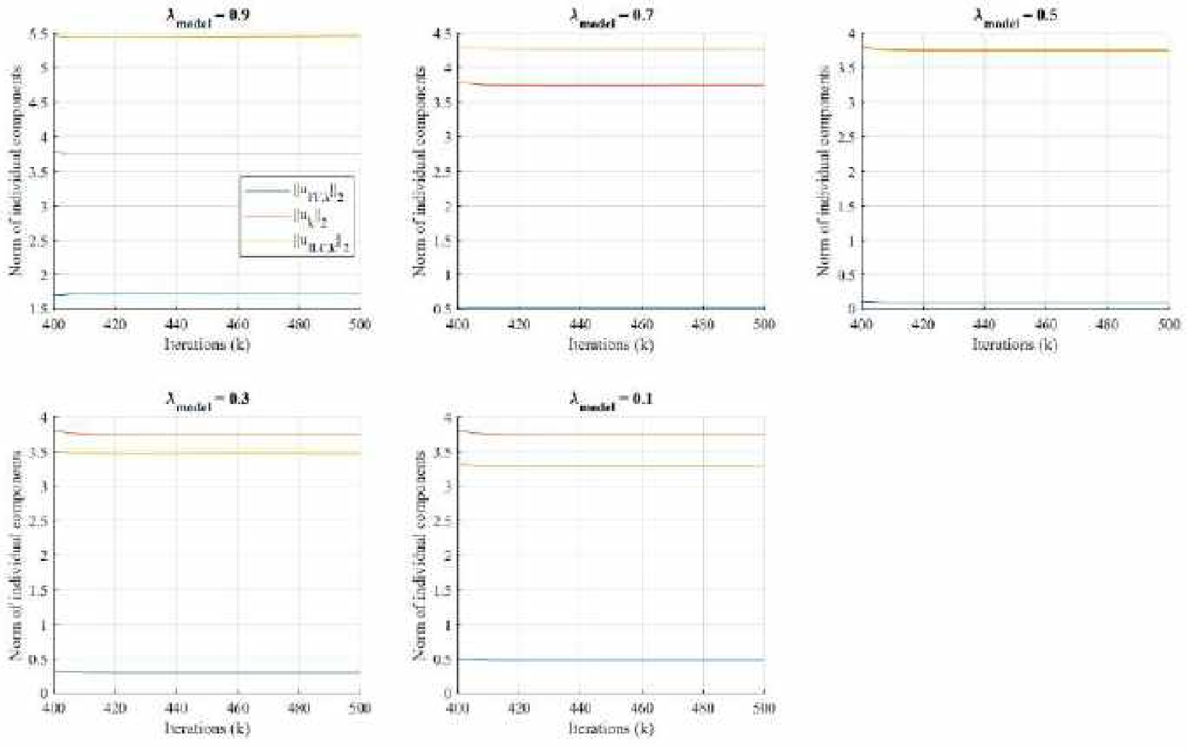
$\lambda_{FF} = 1$
$\lambda_{FF} = 0.9$
$\lambda_{FF} = 0.7$
$\lambda_{FF} = 0.5$
$\lambda_{FF} = 0.3$
$\lambda_{FF} = 0.1$
$\|u_{FF,k}\|_2$
$\|u_k\|_2$
$\|u_{ILC,k}\|_2$
Norm of individual components
Iterations (k)



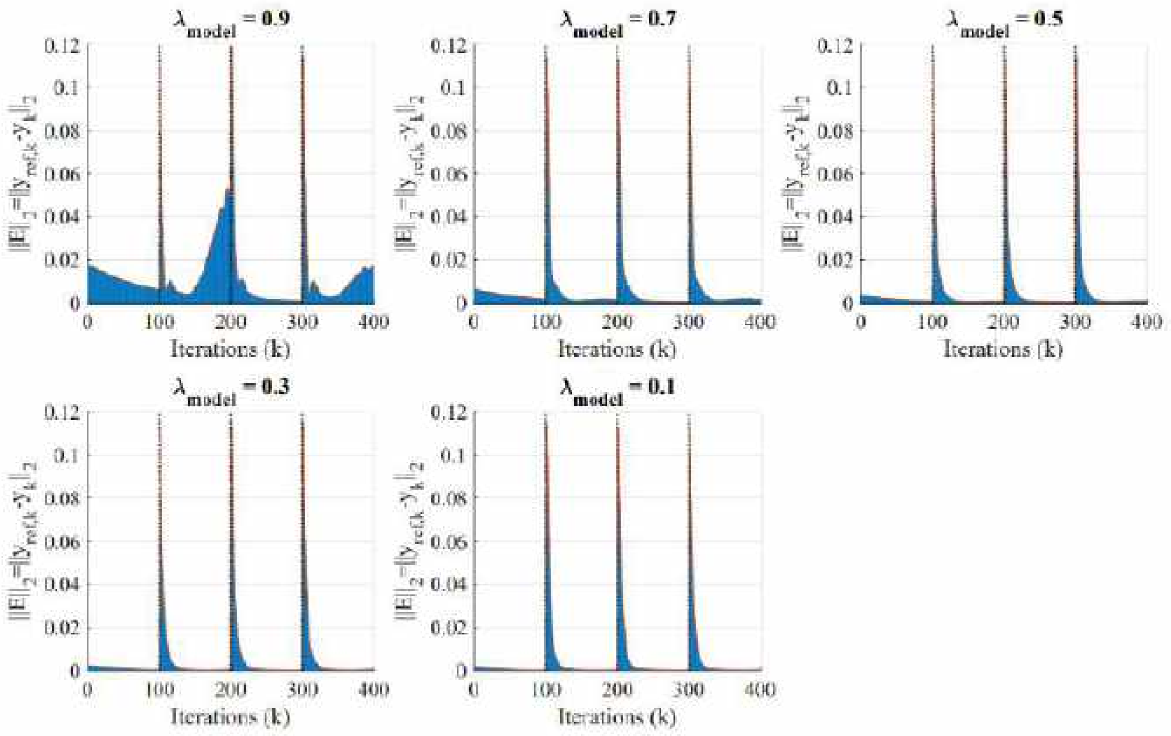
$\lambda_{FF} = 1$
$\lambda_{FF} = 0.9$
$\lambda_{FF} = 0.7$
$\lambda_{FF} = 0.5$
$\lambda_{FF} = 0.3$
$\lambda_{FF} = 0.1$
$\|E\|_2 = \|y_{ref,k} - y_k\|_2$
Iterations (k)

3. Changing λ_{model}



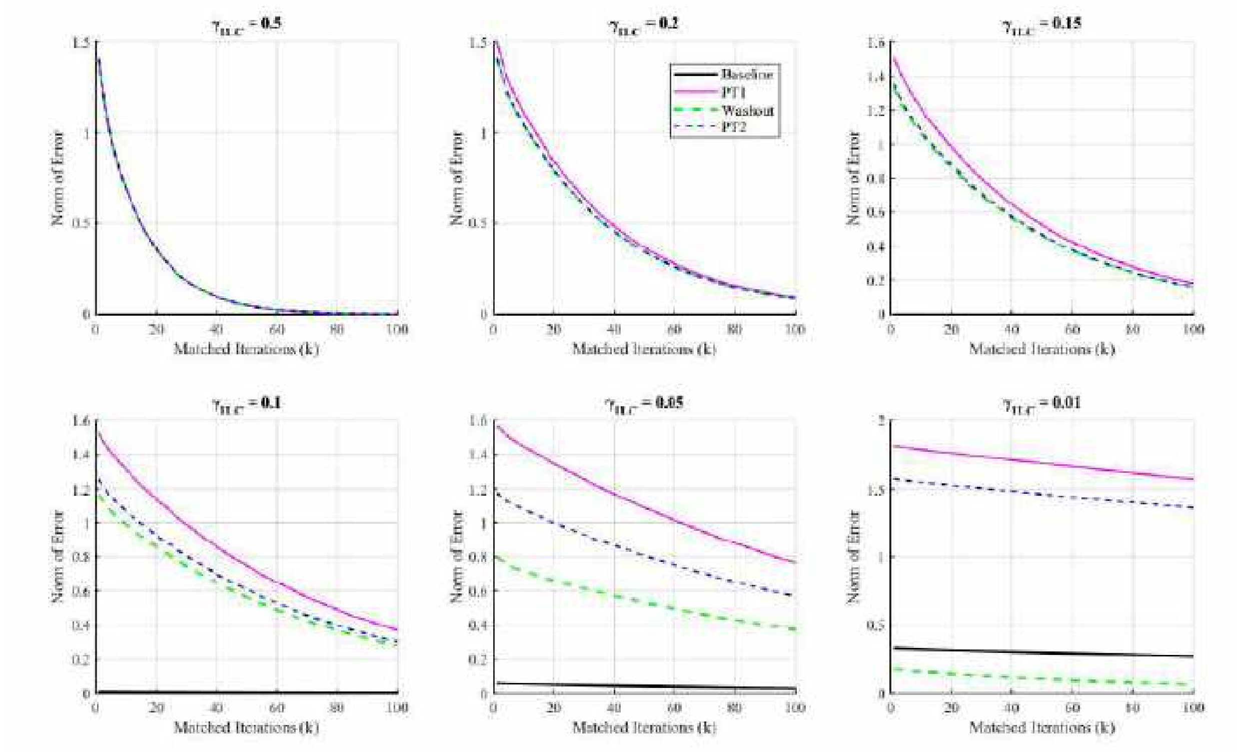


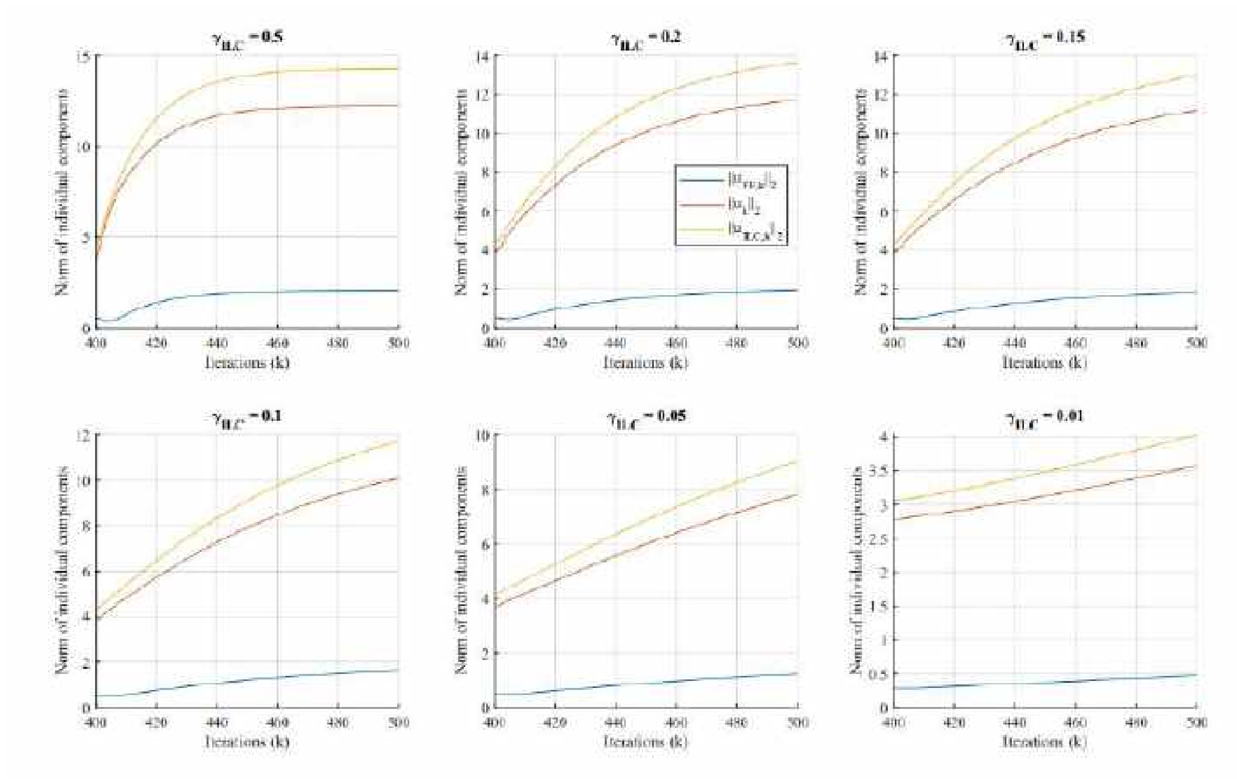
λ_{model} = 0.9
λ_{model} = 0.7
λ_{model} = 0.5
λ_{model} = 0.3
λ_{model} = 0.1
Norm of individual components
Iterations (k)
$\|u_{FF,k}\|_2$
$\|u_k\|_2$
$\|u_{ILC,k}\|_2$

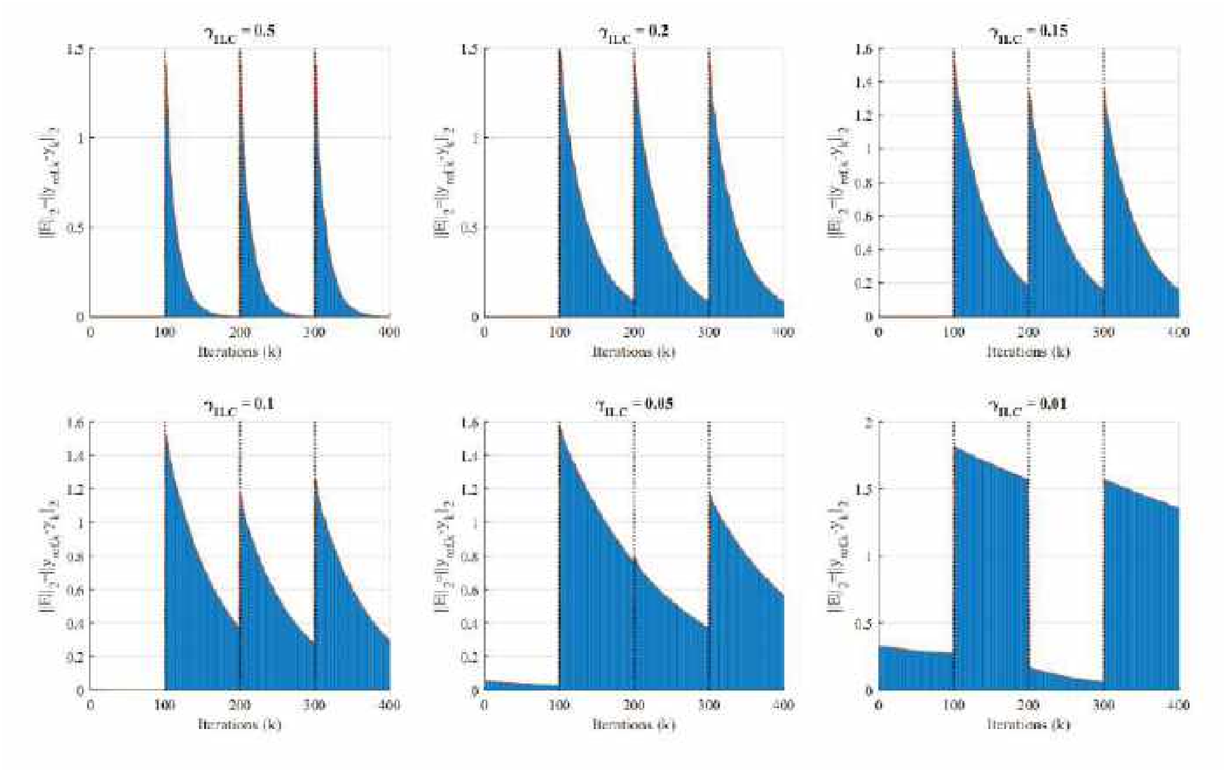


λ_{model} = 0.9
λ_{model} = 0.7
λ_{model} = 0.5
λ_{model} = 0.3
λ_{model} = 0.1
$\|E\|_2=\|y_{ref,k}-y_k\|_2$
Iterations (k)

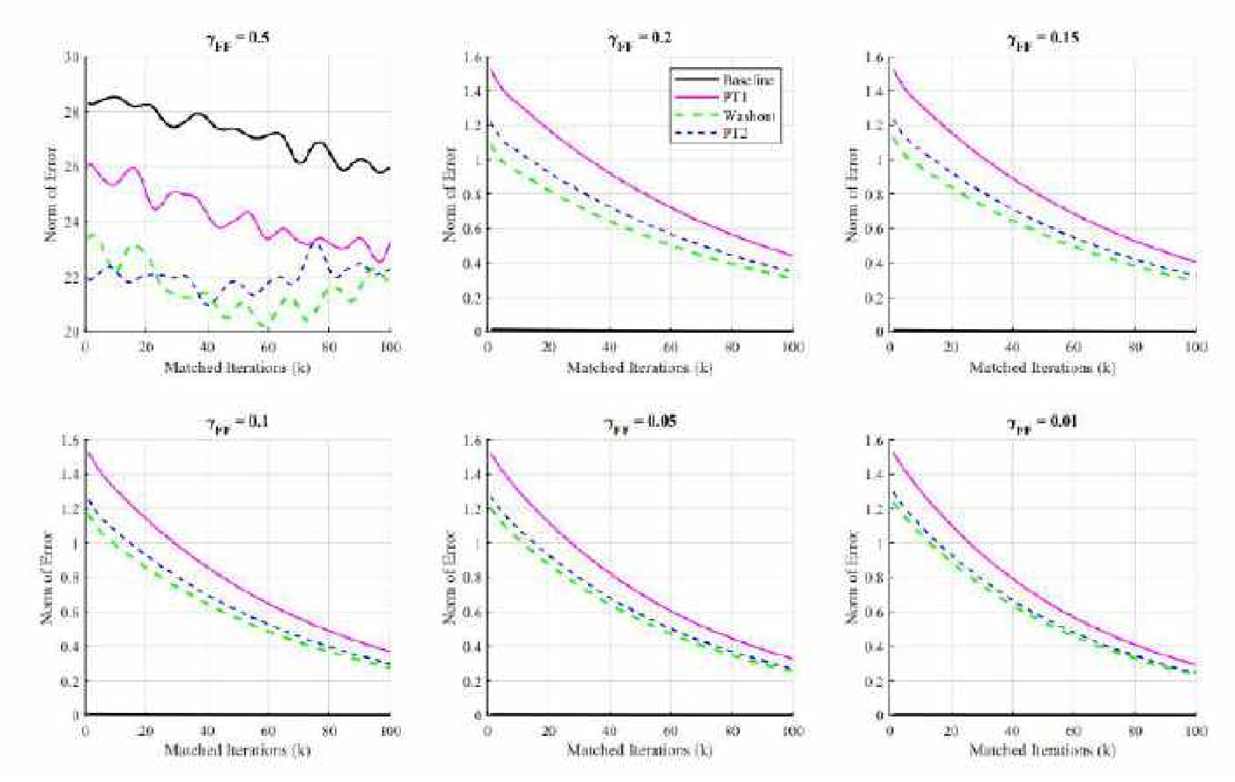
4. Changing γ_{ILC}:

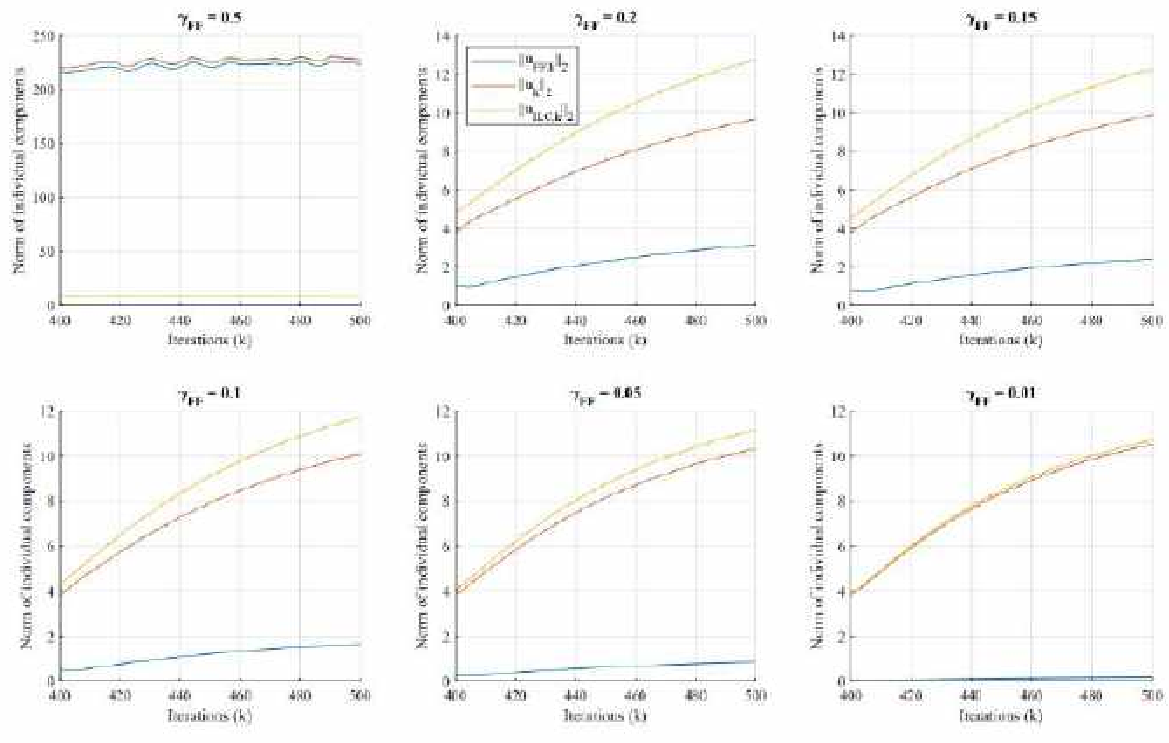


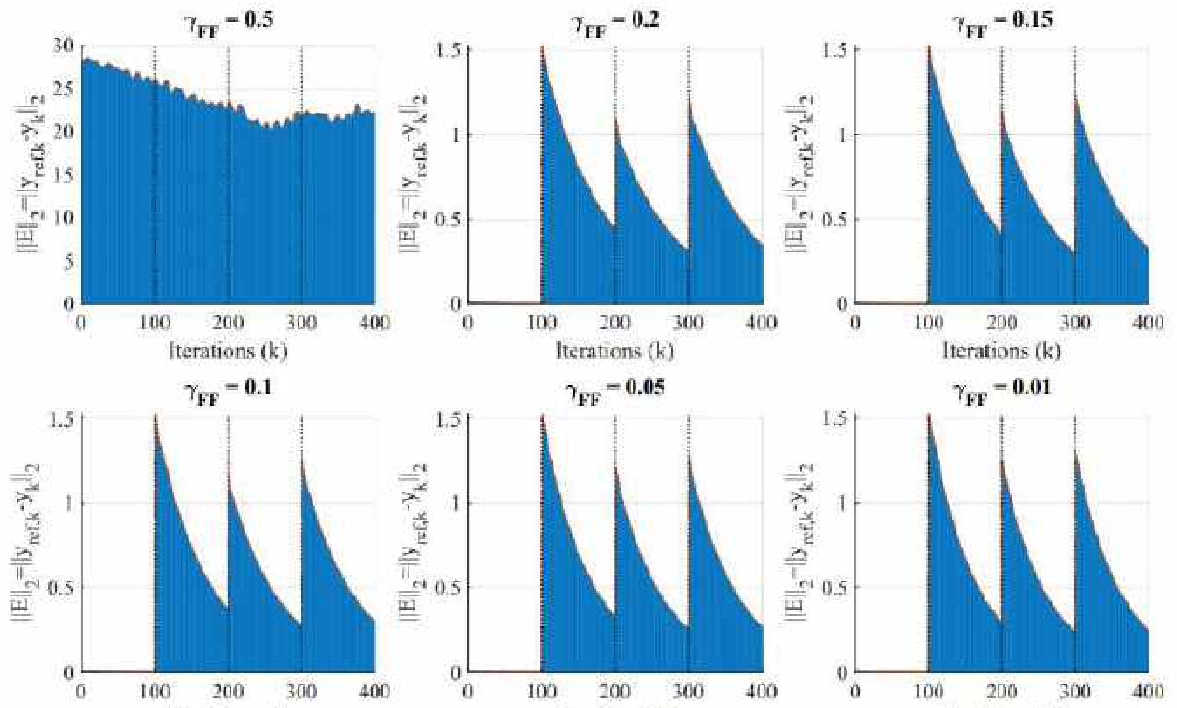




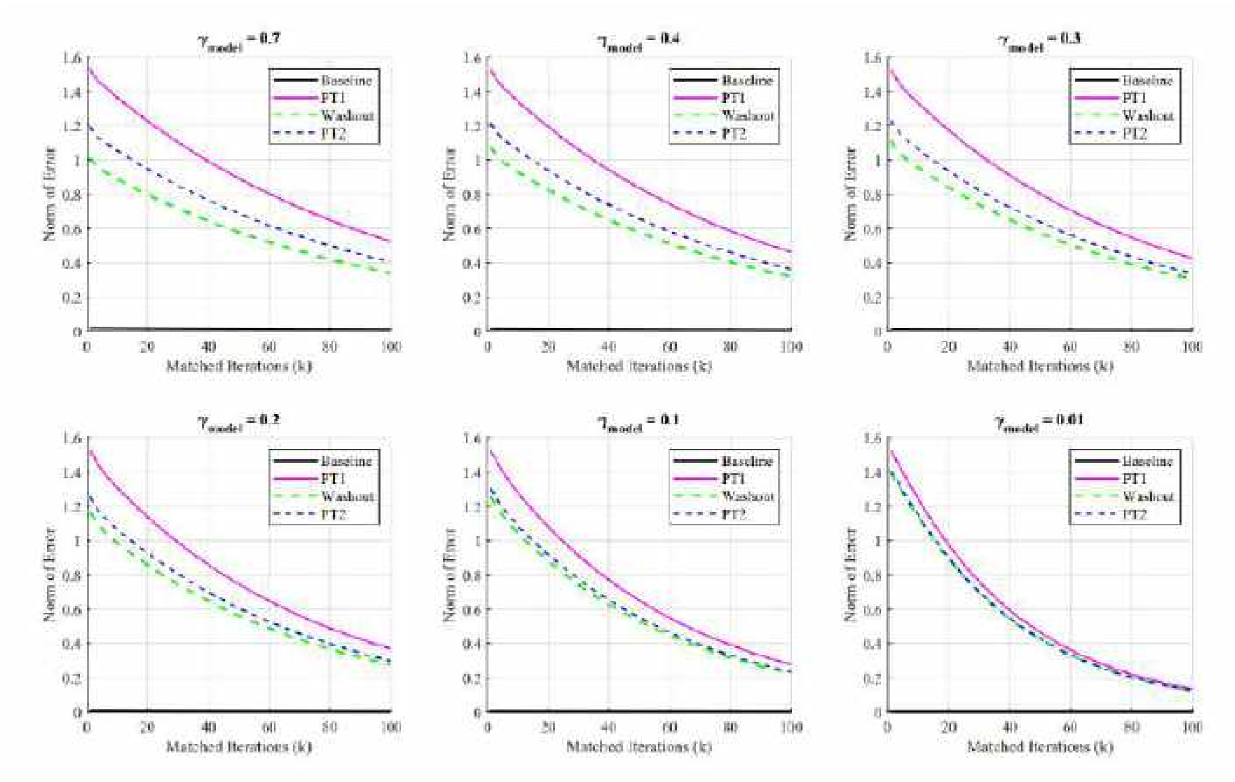
5. Changing γ_{FF}:

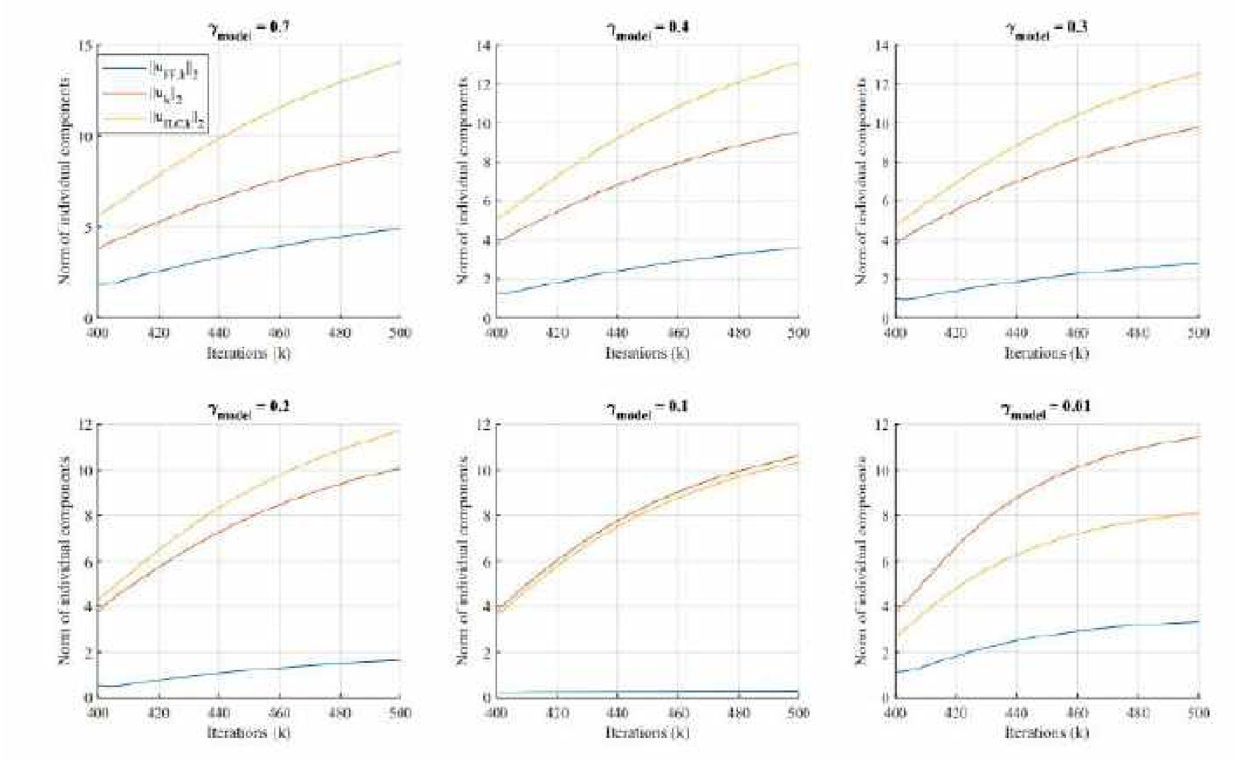


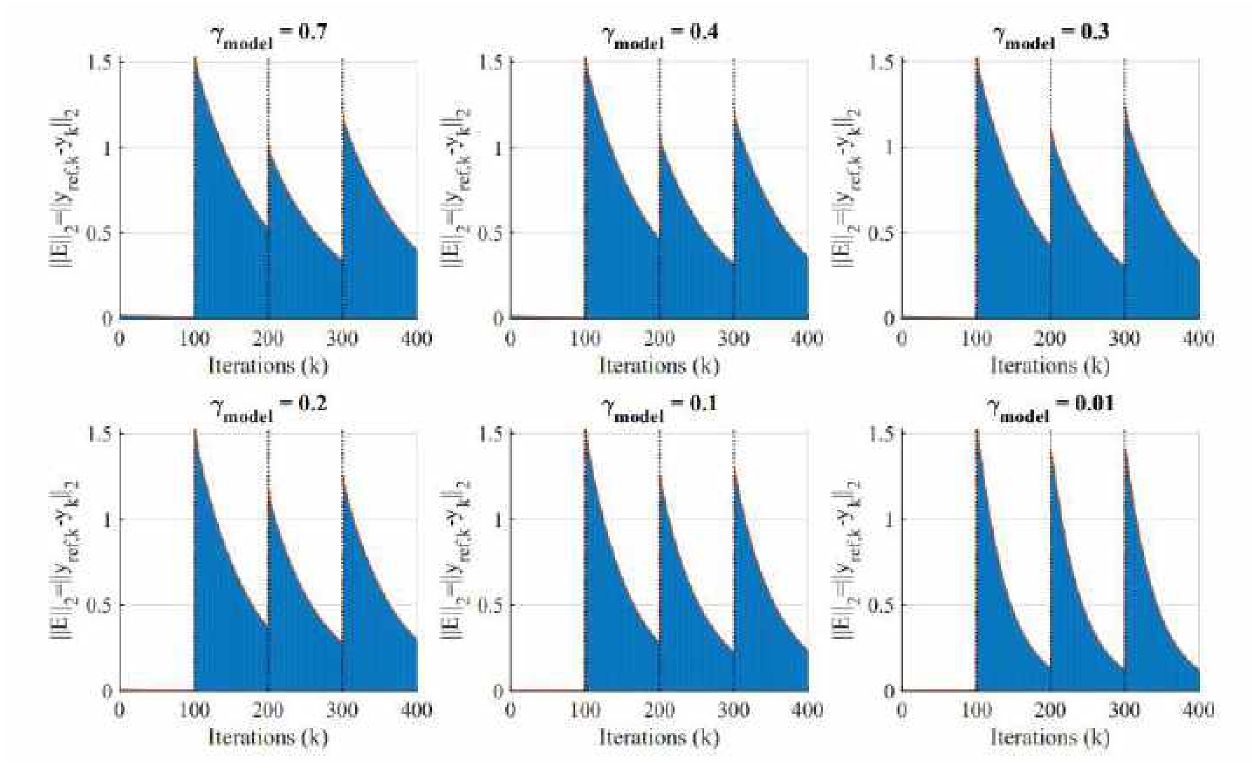

γ_{FF} = 0.5
γ_{FF} = 0.2
γ_{FF} = 0.15
γ_{FF} = 0.1
γ_{FF} = 0.05
γ_{FF} = 0.01
$\|u_{FF,k}\|_2$
$\|u_k\|_2$
$\|u_{ILC,k}\|_2$
Norm of individual components
Iterations (k)


γ_{FF} = 0.5
γ_{FF} = 0.2
γ_{FF} = 0.15
γ_{FF} = 0.1
γ_{FF} = 0.05
γ_{FF} = 0.01
$\|E\|_2=\|y_{ref,k}-y_k\|_2$
Iterations (k)

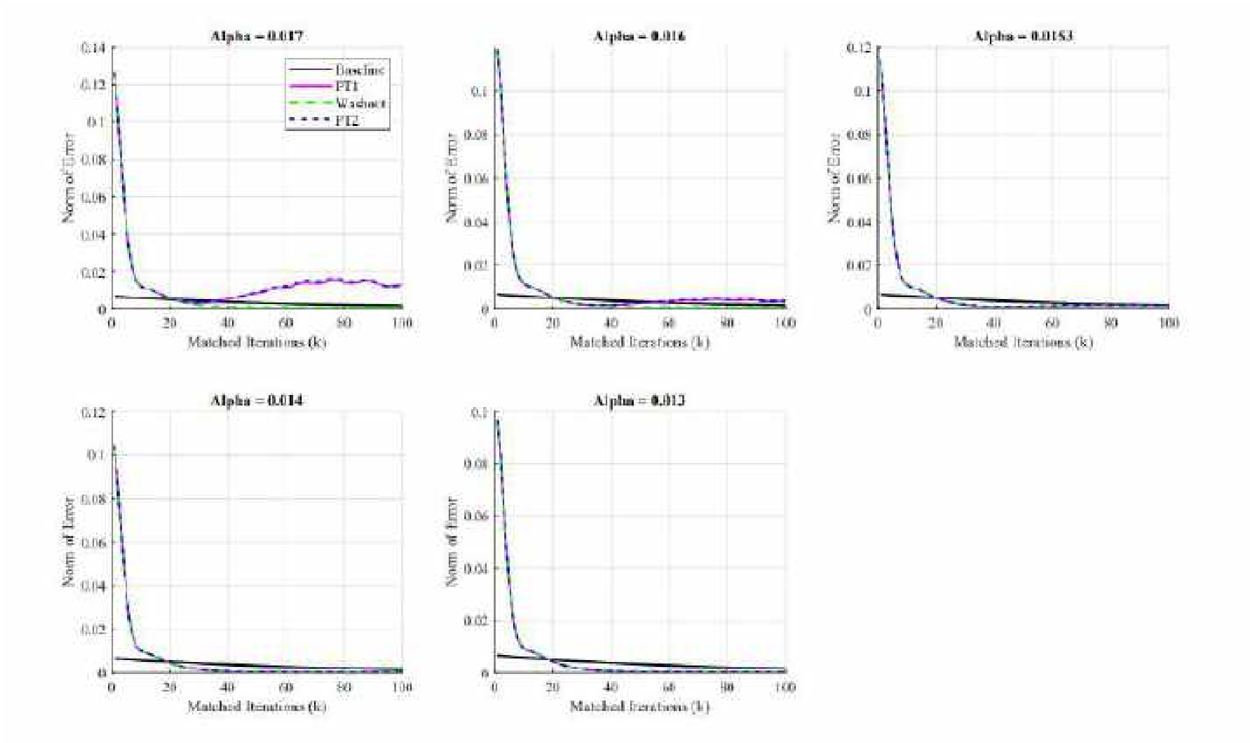
6. Changing γ_{model}:

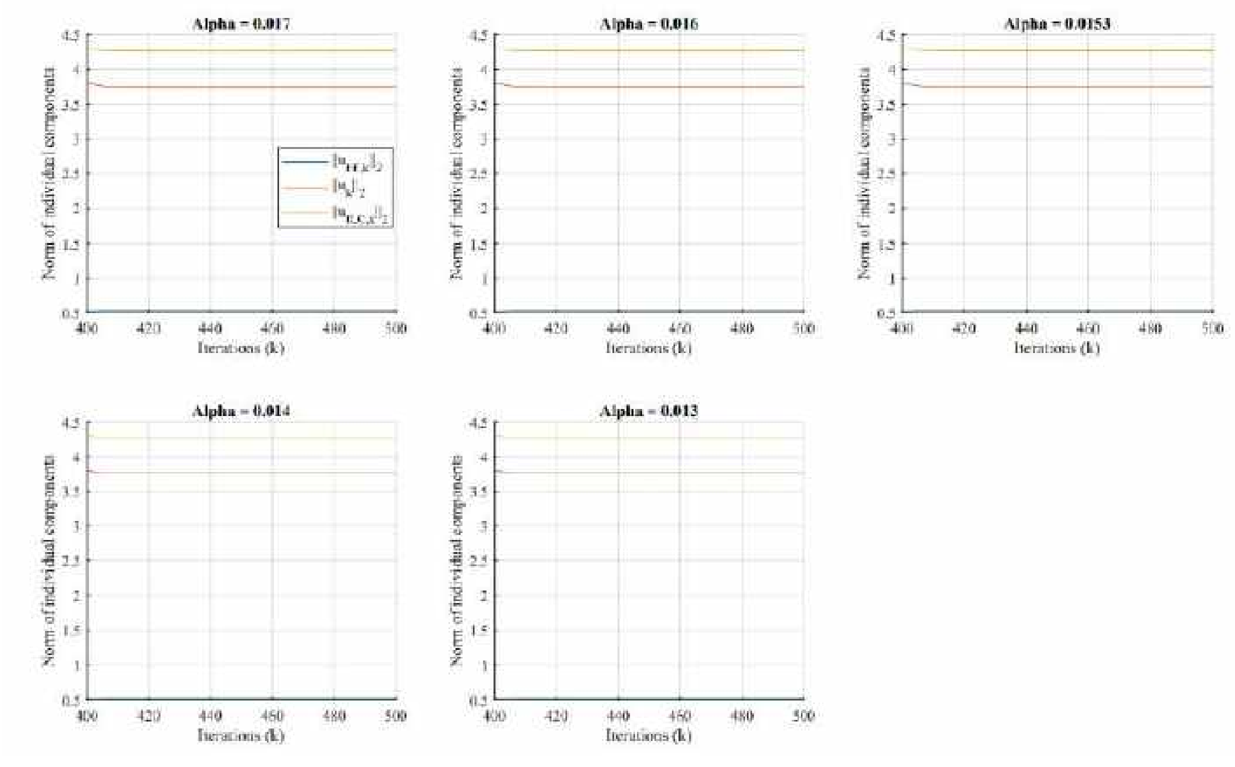


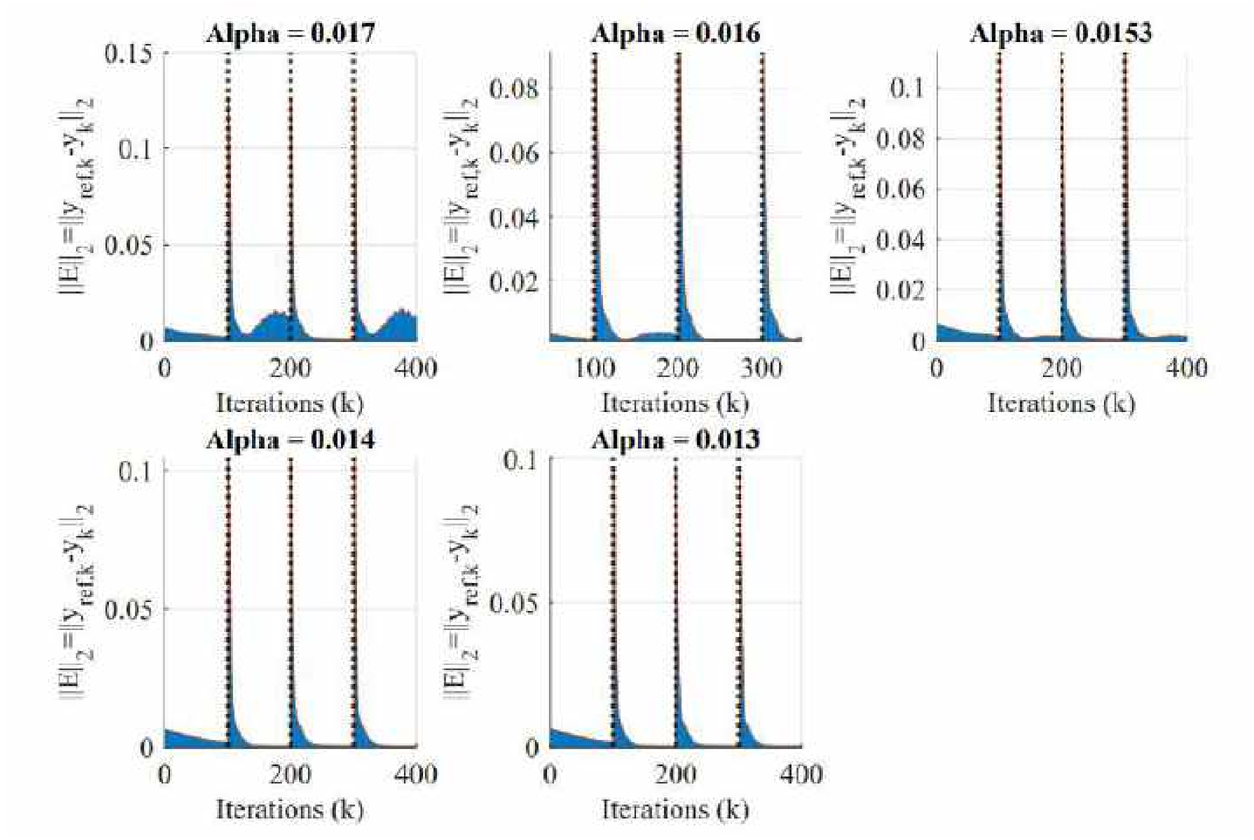




The fourth task was to analyze how sensitive the controller is to the strength of the velocity perturbations. For this, we vary the parameter alpha which represents the strength of the field. We got the following results from the simulations:


Alpha = 0.017
Alpha = 0.016
Alpha = 0.0153
Alpha = 0.014
Alpha = 0.013
Baseline
FT1
Washout
FT2
Norm of Error
Matched Iterations (k)


Alpha = 0.017
Alpha = 0.016
Alpha = 0.0153
Alpha = 0.014
Alpha = 0.013
$\|u_{FF,k}\|_2$
$\|u_k\|_2$
$\|u_{E,C,k}\|_2$
Norm of individual components
Iterations (k)



We observe that the controller is very highly sensitive to change in the perturbation strength, and only for a very very small value, it is able to show savings and retention.

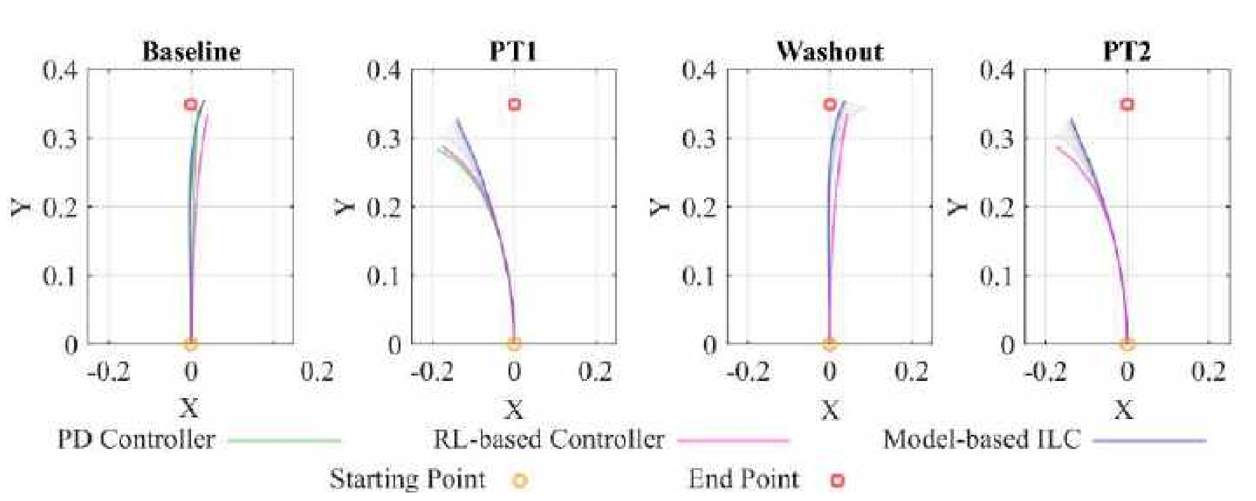
Reinforcement Learning Based Controller

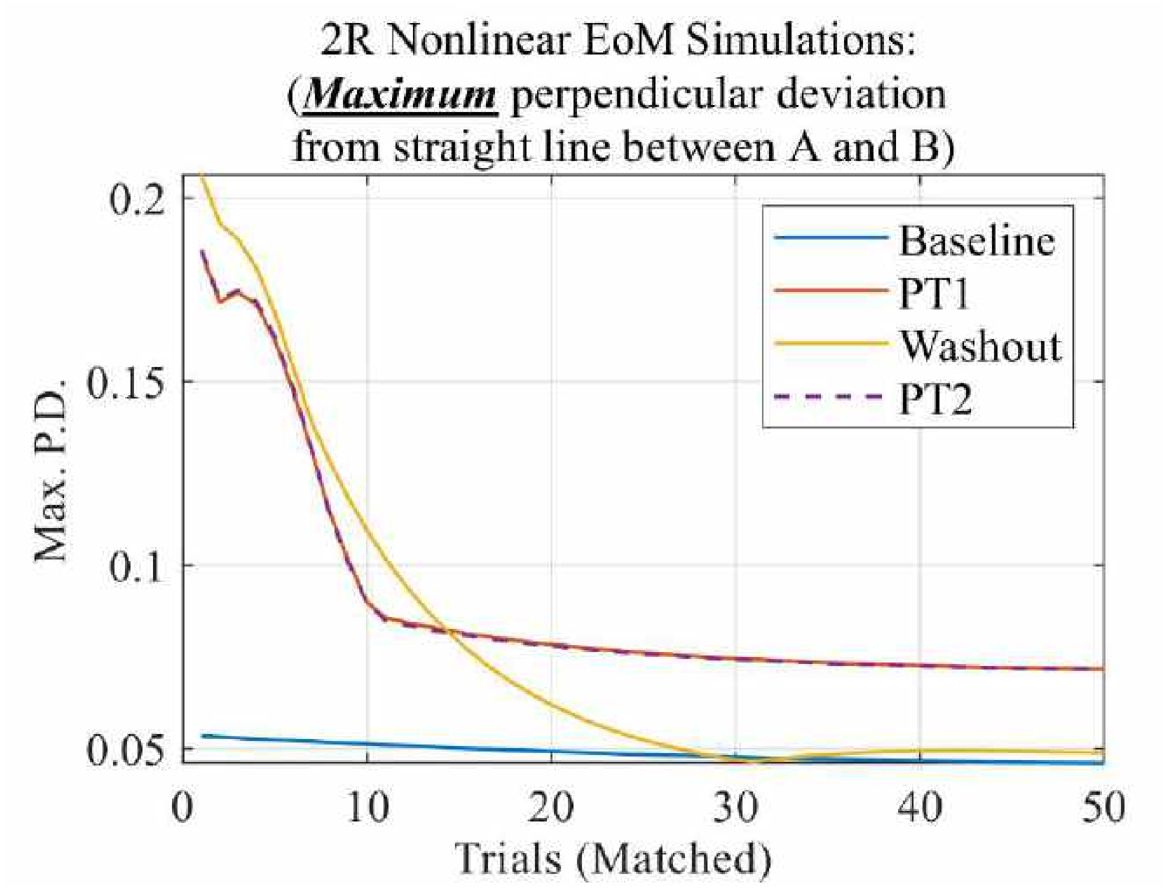
Reinforcement learning is based on taking suitable actions based on present state in order to maximize the reward obtained. It's a feedback based ML technique which helps an agent learn to behave in an environment by performing actions and seeing the results of the action.
In our case, the agent is the 2r robotic arm.

The method used in Reinforcement learning is DDPG- Deep Deterministic Policy Gradient, which has two networks named- the Actor and the Critic. The actor network chooses the action that the agent will take for a particular state and the Critic network judges the action taken and tells how good or bad the action is.

There were two steps associated with this. We first implemented the RL controller without any dynamics involved. It was implemented in python and the 2r robotic arm successfully learned the reaching task. The second step was to include the dynamics, i.e., include the dynamic equations, torques and forces at the joints etc. It was first implemented in python, using Tensorflow Keras framework. However, when dynamics was involved, the robotic arm wasn't able to learn the task using simple fixed neural networks for actor and critic networks. We tried implementing other kinds of neural networks such as CNN, RNN and LSTM, but there were a lot of implementation challenges due to lack of available documentation and resources.

After going through various tutorials and learning about RL implementations for different robots, we decided to implement the RL controller using MATLAB Simulink. We were successfully able to implement the RL controller using MATLAB Simulink and did a few simulations on it to analyze the performance of the RL controller. Below are the obtained results:





Conclusion

On comparing the two controllers we get the conclusion that the Model Based Iterative Learning Controller, is able to depict human behavior for a simple reaching task under fixed force perturbations. The entire behavior of the 2R robotic arm manipulator is very sensitive to all the parameters. On comparing the RL based controller with the MBIL controller, we realize that RL controller is unable to show features like savings and retention as is the MBIL controller

Experience

This SRIP project was a challenging and very interesting project. I got a basic understanding of robotics and how things work in this domain. I had to go through various tutorials and research papers at the start to be able to get a basic understanding of what I was going to work on. The first task was to get an understanding of the MBIL controller and understand how it works. Doing simulations on the MBIL controller gave a detailed understanding of it and its working. The second task of implementing the RL controller was the most difficult part of the project. The main reason for it was the lack of enough resources and documentation for implementing the desired controller. There weren't enough tutorials or projects related to this work, which could help implement the desired controller. We had to go through various tutorials and resources before we were able to implement the RL controller. We were still not able to implement it in python, and achieved our goal using MATLAB Simulink. This project gave an experience of what many researchers face. Getting stuck on something new, which very few or no one has done before, and fighting out your way someway or the other in order to get the desired result was really an enriching experience. I would like to thank Professor Harish P.M. and Sujay Kadam for this wonderful opportunity.

Reference

1. *Kadam, S. D., Jadav S. V., and Palanthandalam-Madapusi, H. J., 2022, "A Model-based Feedforward and Iterative Learning Controller With Human-like Learning Properties Exhibiting Human-motor-learning Features" (under review, IEEE Transactions on Cognitive and Developmental Systems, January 2022)*