Global Control Architecture

2 Controller

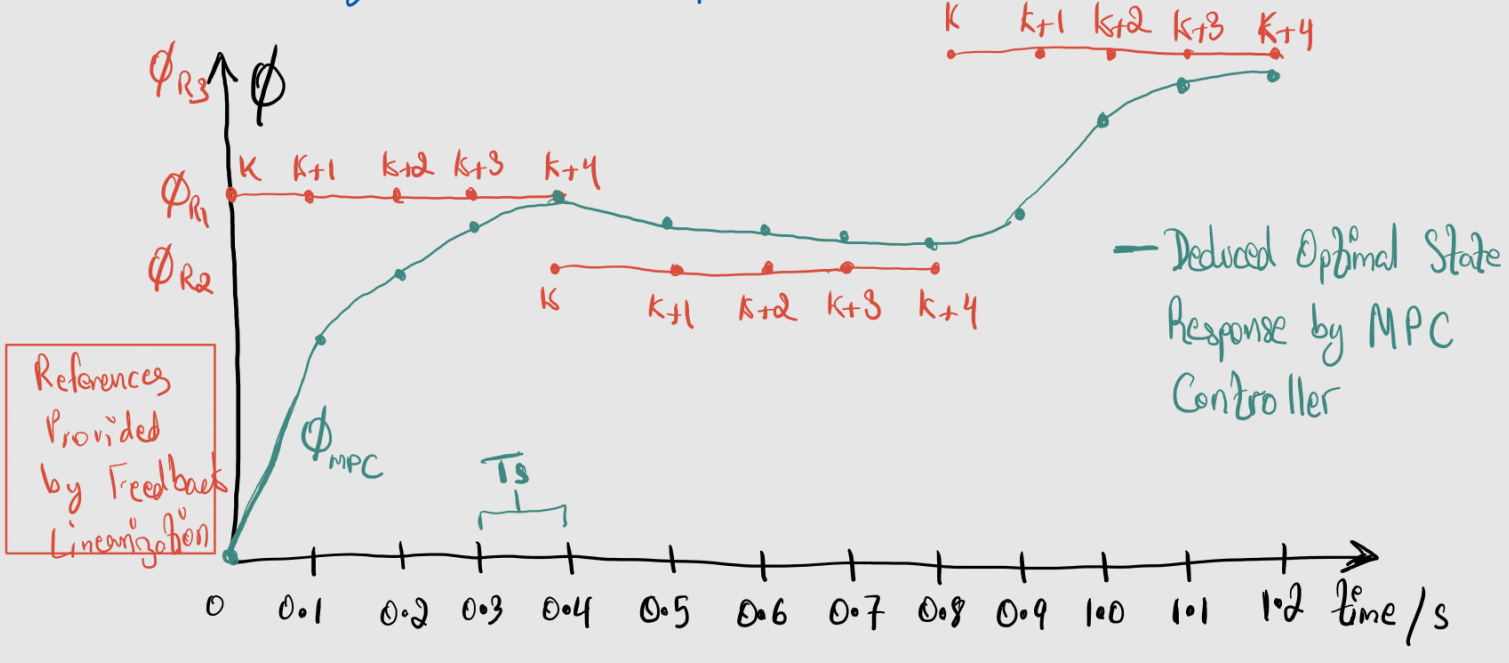
Feedback Linearization
- ⊕ Outer loop
- ⊕ Position Controller
- ⊕ Creates reference for MPC to follow
- ⊕ Runs slower than the MPC controller

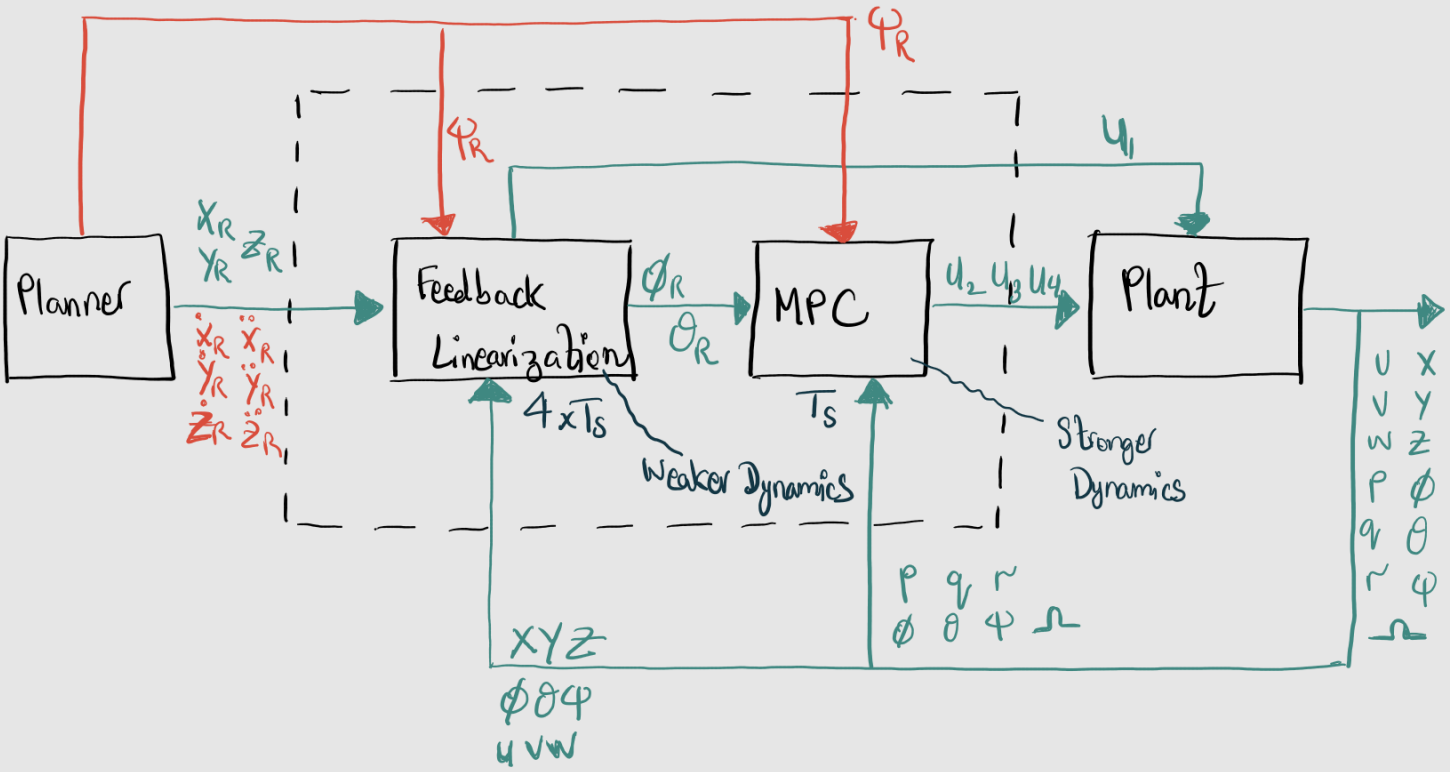
Model Predictive Control
- ⊕ Inner Loop
- ⊕ Attitude Controller
- ⊕ MPC needs time to approach provided reference by Feedback Linearization
- ⊕ Runs faster

Weaker Dynamics — 1:4 frequency ratio in this course — Stronger Dynamics

Let $T_s = 0.1s$

* Using ϕ as an example





Why Two Cascaded Controllers?

- In the derived EOMs, the linear velocity derivative equations are much more non-linear as compared to the angular velocity time derivatives.

Set 1

$$\dot{u} = vr - wq + g\sin\theta$$
$$\dot{v} = wp - ur - g\sin\phi\cos\theta$$
$$\dot{w} = uq - vp - g\cos\phi\cos\theta + U_1/m$$

Set 2

$$\dot{p} = \frac{(I_{yy} - I_{zz})}{I_{xx}} qr - \frac{J_{TP}}{I_{xx}} q\Omega + \frac{U_2}{I_{xx}}$$
$$\dot{q} = \frac{(I_{zz} - I_{xx})}{I_{yy}} pr + \frac{J_{TP}}{I_{yy}} p\Omega + \frac{U_3}{I_{yy}}$$
$$\dot{r} = \frac{(I_{xx} - I_{yy})}{I_{zz}} pq + \frac{U_4}{I_{zz}}$$

- Coming up with a single scheme to deal with both set of equations isn't possible without sacrificing accuracy.

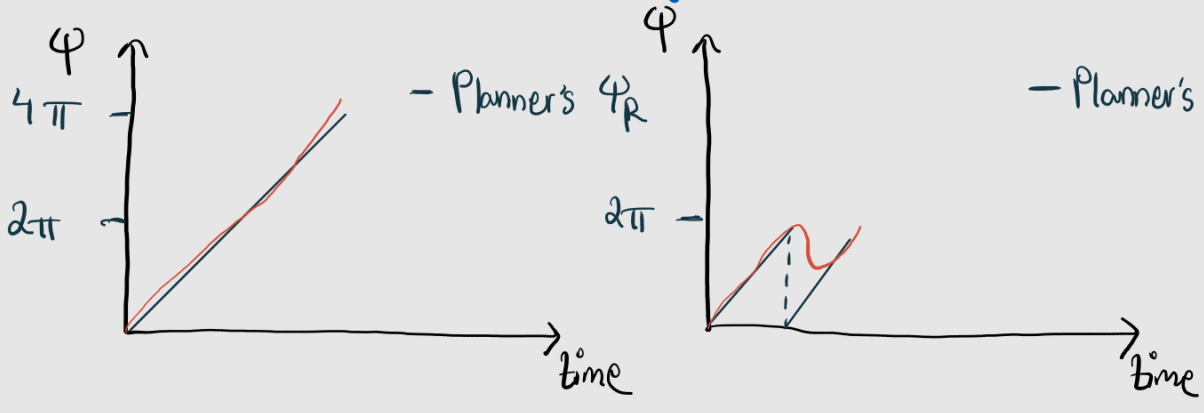
- Thus, we bifurcate the problem into an individual position and attitude problem.

- Based on equation format and associated inputs and outputs, we select the best controller for both position and attitude control.

The Planner

- Note that Ψ_R is the only angle we generate from the planner. This is because Ψ is the only angle that can be inferred from trajectory equations. Furthermore, Ψ is the only angle in the intrinsic formulation that coincides with one of XYZ axis (Z axis in case of Ψ)

- When generating Ψ_R values in code, the angle is treated as continuous to avoid big jumps in numbers. Sudden jumps can create potential problems. This is better represented by the following graphs:



Continuous Treatment of Ψ_R

Non-Continuous Treatment of Ψ_R

- For our application we use smooth and well defined trajectories that can be represented analytically.

$$X_R = f_1$$
$$Y_R = f_2$$
$$Z_R = f_3$$

- These f_1, f_2, f_3 upon taking derivatives will give the expressions of X_R, Y_R, Z_R & $\dot{X}_R, \dot{Y}_R, \dot{Z}_R$.

alongside ψ_R

- These ↑ all will be computed every $4T_s$ intervals to generate reference commands to feed to the controller.

Model Predictive Controller

- MPC will have dynamic horizon this time around because the outerloop only updates the reference values after a set number of inner loop iterations.

Model Predictive Control Attitude Controller

- MPC is a discrete controller.

Equation of Motions
⇓
State Space Equations
⇓
Continuous LTI System
⇓
Discrete LTI System

For an LTI-MPC Controller

Assumption in Transfer Matrix

- Performance vs Ease of modelling tradeoff.
- We simplify the equations with the hope that it will have little effect on its performance but it will be easier to design around and also be computationally friendly.

- We assume zero angle pitch and roll within the transfer matrix to allow us to conveniently connect body states with attitude rate of change. Choosing the Rxyz convention helps here because the transfer matrix doesn't contain ψ.
- ψ can have very large value so a zero assumption for it isn't rational.

$$\phi = 0 \text{ rad}, \quad \theta = 0 \text{ rad}$$

$$T = \begin{bmatrix} 1 & \sin\phi \tan\theta & \cos\phi \tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi / \cos\theta & \cos\phi / \cos\theta \end{bmatrix}$$

$$\longrightarrow T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Taking this assumptions boils the transfer matrix down to an identity matrix.

= This isn't a good assumption for a high speed agile controller because we will often have large angles. But this is a reasonable assumption for a first controller design.

Now,

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \underbrace{I}_{\text{Identity Matrix}} \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \qquad \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \underbrace{I}_{\text{Identity Matrix}} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}$$

$$\dot{\phi} = p \qquad \ddot{\phi} = \dot{p}$$
$$\dot{\theta} = q \qquad \ddot{\theta} = \dot{q}$$
$$\dot{\psi} = r \qquad \ddot{\psi} = \dot{r}$$

- Actual "plant" will not make this assumption. So we can measure the performance of controller with this simplification.

Attitude EOM Simplification

$$\dot{p} = \frac{(I_{yy} - I_{zz})}{I_{xx}} qr - \frac{J_{TP}}{I_{xx}} q\Omega + \frac{U_2}{I_{xx}}$$

$$\dot{q} = \frac{(I_{zz} - I_{xx})}{I_{yy}} pr + \frac{J_{TP}}{I_{yy}} p\Omega + \frac{U_3}{I_{yy}}$$

$$\dot{r} = \frac{(I_{xx} - I_{yy})}{I_{zz}} pq + \frac{U_4}{I_{zz}}$$

After Assumption

$$\ddot{\phi} = \frac{(I_{yy} - I_{zz})}{I_{xx}} \dot{\theta}\dot{\psi} - \frac{J_{TP}}{I_{xx}} \dot{\theta}\Omega + \frac{U_2}{I_{xx}}$$

$$\ddot{\theta} = \frac{(I_{zz} - I_{xx})}{I_{yy}} \dot{\phi}\dot{\psi} + \frac{J_{TP}}{I_{yy}} \dot{\phi}\Omega + \frac{U_3}{I_{yy}}$$

$$\ddot{\psi} = \frac{(I_{xx} - I_{yy})}{I_{zz}} \dot{\phi}\dot{\theta} + \frac{U_4}{I_{zz}}$$

Putting EOMs in linear format

- We need both $\ddot{\phi}$ and $\dot{\phi}$ as outputs. $\ddot{\phi}$ will be needed to iterate next value of $\dot{\phi}$.
- $\dot{\phi}$ is needed to find ϕ by integration and $\ddot{\phi}$ by putting it into the state space equations.
- Ω is part of the 'A' matrix but we haven't classified it as a state even though Ω keeps on changing. This is because we don't need to classify it as a state to be able to control the system.
- While constructing the matrices on next page it has been taken care that $\dot{\psi}$ don't appear inside the 'A' matrix. In $\ddot{\psi}$ eqn $\frac{(I_{xx} - I_{yy})}{I_{zz}} \dot{\phi}\dot{\theta}$ is halved and included as 2 entries.
- Reason for this will become clear when we study LPV.

$$\begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{-J_{TP}}{I_{xx}}\Omega & 0 & \frac{(I_{yy}-I_{zz})}{I_{xx}}\dot{\theta} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & J_{TP}/I_{yy}\,\Omega & 0 & 0 & 0 & \frac{(I_{zz}-I_{xx})}{I_{yy}}\dot{\phi} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{(I_{xx}-I_{yy})}{I_{zz}}\frac{\dot{\theta}}{2} & 0 & \frac{(I_{xx}-I_{yy})}{I_{zz}}\frac{\dot{\phi}}{2} & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & 0 \\ 1/I_{xx} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1/I_{yy} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1/I_{zz} \end{bmatrix} \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

It is clear that this matrix is non-linear in nature. This is not an LTI and not an LTV system.

(matrices depend on time but not on states themselves)

$$\dot{\vec{x}} = A\vec{x} + B\vec{u}$$
$$\vec{y} = C\vec{x} + D\vec{u}$$

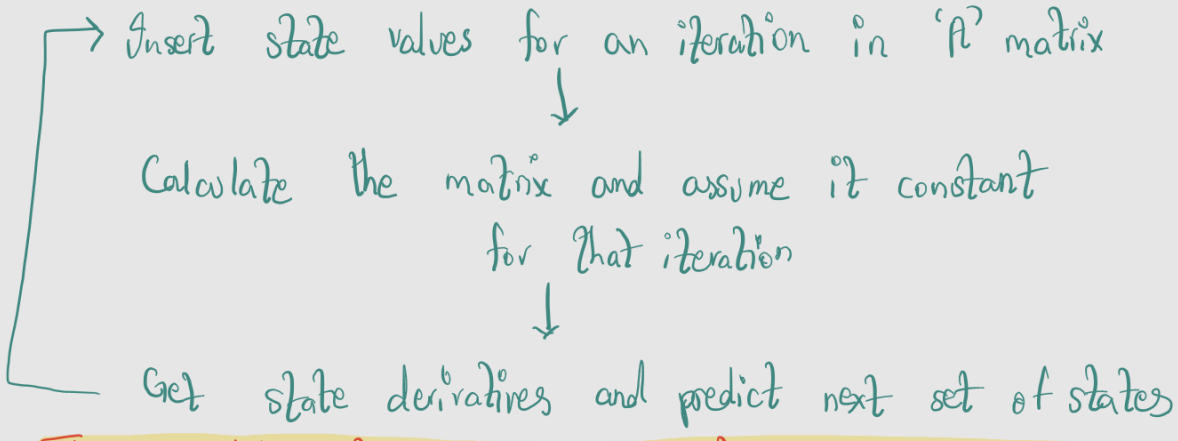
For our case

$$C = [1\ 0\ 1\ 0\ 1\ 0]$$

Linear Parameter Varying Technique

- May not give optimal solution as compared to implementation of Non-Linear techniques but it can yield quiet good results still while saving a lot of computational power that would otherwise be needed for Non-Linear techniques.
- Thus, implementing high frequency Non-linear strategies on Jetson Nano later on may not be a very good idea from computational point of view.

- The LPV technique is part of Robust Control. It is a highly researched topic.
- The idea behind it is:



- The variables that we have to vary need to stay in some kind of range and they can't have crazy values. If the values vary across a wide spectrum than the controller won't be able to control the drone.

  - Thus, we have not kept $\dot{\psi}$ inside the 'A' matrix while formulating it as ψ is the angle from which we can expect the broadest range of values and rates of change.
  - Furthermore, for the $\dot{\psi}$ equation we incorporated both $\dot{\theta}$ & $\dot{\phi}$ in the 'A' to keep it representative and more adaptive to changes in pitch and roll. Our fidelity this way is increased.
  - For very agile applications like flips etc, the values of $\dot{\theta}$ and $\dot{\phi}$ might exceed acceptable range too. So keep this at back of the mind when adapting the controller for increased agility.

Important Note:

+ The transfer matrix simplification was performed only to simplify the EOMs. We won't equate euler rates of change with body frame angular velocities.

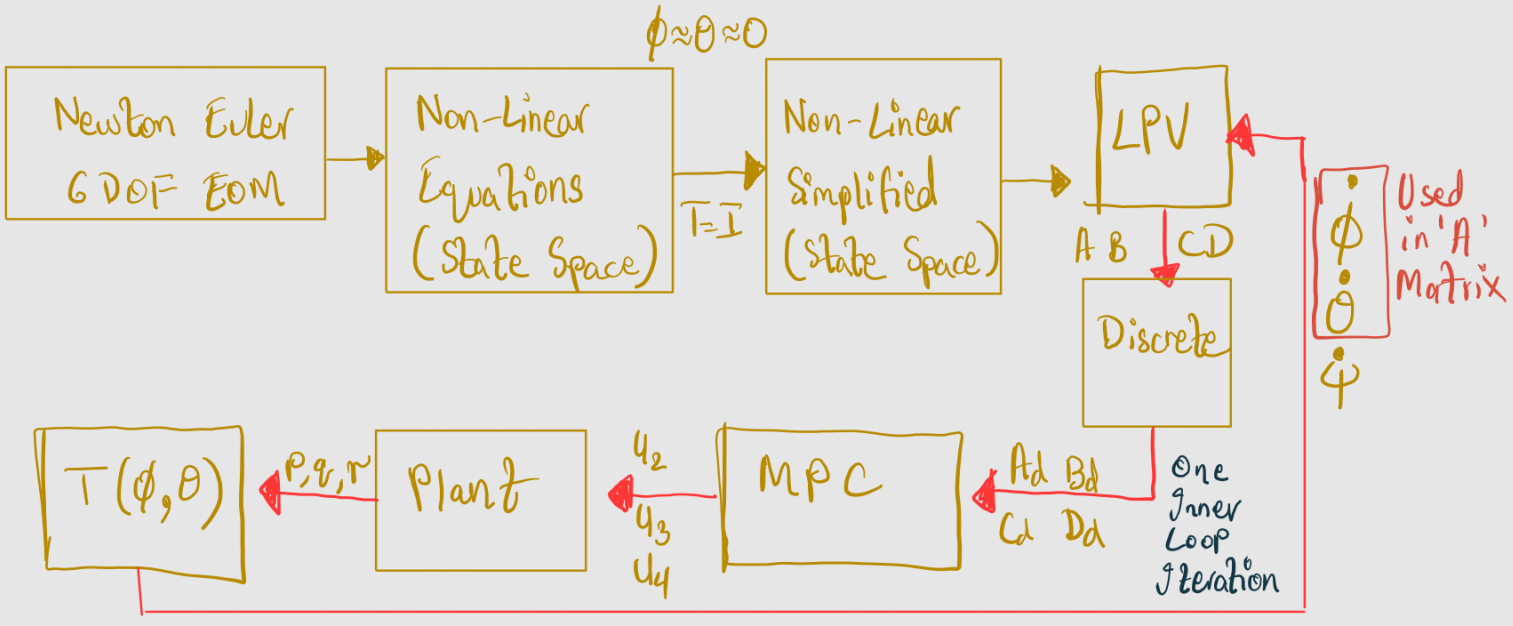
+ While updating the $\dot{\theta}$ & $\dot{\phi}$ values for the LPV 'A' matrix during each inner loop iteration, the transfer matrix will be applied in original form to the last iteration's p, q & r.

For matrix 'A'

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T(\phi,\theta) \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

This ensures greater accuracy

For Greater Clarification:



Discretizing The LPV system

Continous LTI (A: system/state Matrix; C: Output Matrix; B: Input Matrix)

$$\dot{\vec{x}}(t) = A\vec{x}(t) + B\vec{u}(t)$$
$$\vec{y}(t) = C\vec{x}(t) + D\vec{u}(t)$$

Discrete LTI

$$\vec{x}_{k+1} = A_d\vec{x}_k + B_d u_k$$
$$\vec{y}_k = C_d\vec{x}_k + D_d u_k$$

derivation: (Forward Euler Method)

$$\dot{x}(k) = \frac{x_{k+1} - x_k}{T_s}$$ → Sample Time

$$\dot{\vec{x}}(k) = A\vec{x}(k) + B\vec{u}(k)$$
$$\frac{\vec{x}_{k+1} - \vec{x}_k}{T_s} = A\vec{x}_k + B\vec{u}_k$$
$$\vec{x}_{k+1} = \vec{x}_k + AT_s\vec{x}_k + BT_s\vec{u}_k$$
$$\vec{x}_{k+1} = \underbrace{(I + AT_s)}_{A_d}\vec{x}_k + \underbrace{BT_s}_{B_d}\vec{u}_k$$

$$\boxed{\vec{x}_{k+1} = A_d\vec{x}_k + B_d\vec{u}_k}$$

$$\vec{y}(k) = C\vec{x}(k) + D\vec{u}(k)$$
$$\vec{y}_k = \underbrace{C}_{C_d}\vec{x}_k + \underbrace{D}_{D_d}\vec{u}_k$$

$$\boxed{\vec{y}_k = C_d\vec{x}_k + D_d\vec{u}_k}$$

Predicting Future States For MPC controller

- The instructor talks about two approaches: Simplified & Non-Simplified
- Simplified approach was used in course code and so I will only make notes of that due to time shortage.

- Non-simplified approach can be reviewed from Lecture 154 to 156.

When to Use Non-Simplified Form?

(1) Long Horizon Periods
(2) When greater Fidelity is required
(3) Instructor recommended the non-simplified approach for crazy acrobatics / maneuvers.

- Perhaps incorporating this approach for Final Presentation could be a good way to show progress in increasing the fidelity of the controller.

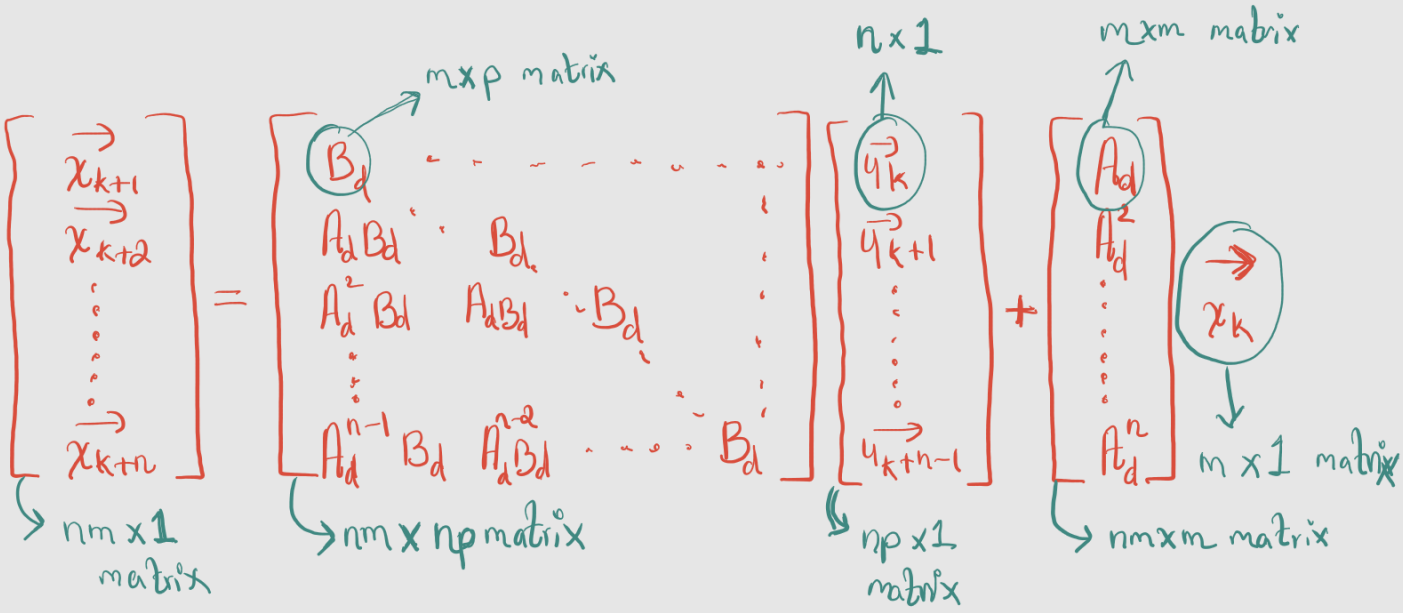
Simplified Future State Prediction Formulation

- In the simplified approach we keep the A_d matrix constant across the horizon for one inner loop iteration.
- So we get the same shorthand formula that we derived earlier in the first course of this series.
- The shorthand is included below for reference.

$$\vec{x}_{k+n} = A_d^n \vec{x}_k + \begin{bmatrix} A_d^{n-1}B_d & A_d^{n-2}B_d & \cdots & A_dB_d & B_d \end{bmatrix} \begin{bmatrix} \vec{u}_k \\ \vdots \\ \vec{u}_{k+n-1} \end{bmatrix}$$

"n is an integer"

Matrix Multiplication (Not Element wise!)



Let $n \rightarrow$ Horizon Period Steps
$m \rightarrow$ # of states
$p \rightarrow$ # of Control Inputs

For our case

$n=4$, $m=6$, $p=3$, Let $k=0$

$$\begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vec{x}_3 \\ \vec{x}_4 \end{bmatrix} = \begin{bmatrix} B_d & 0 & 0 & 0 \\ A_dB_d & B_d & 0 & 0 \\ A_d^2B_d & A_dB_d & B_d & 0 \\ A_d^3B_d & A_d^2B_d & A_dB_d & B_d \end{bmatrix} \begin{bmatrix} \vec{u}_0 \\ \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \end{bmatrix} + \begin{bmatrix} A_d \\ A_d^2 \\ A_d^3 \\ A_d^4 \end{bmatrix} \vec{x}_0$$

Control input at time 'k' affects states at time 'k+1'

- [ ] Later, make notes for the non-simplified future prediction too.

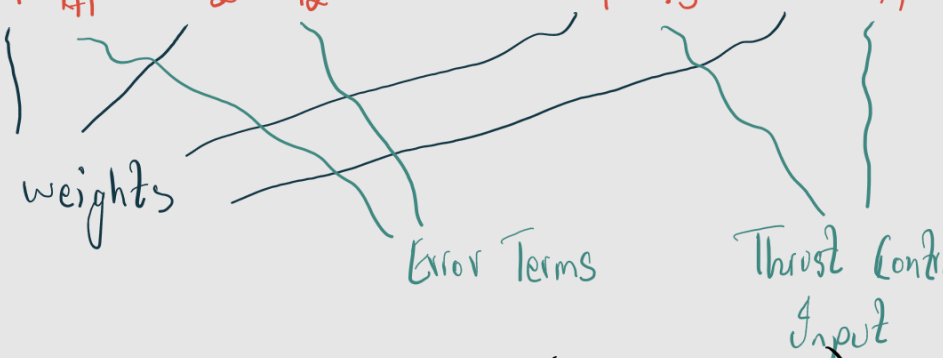
Cost Function Derivation

Benefits of a Quadratic Cost Function

- It ensures that the minima calculated is the absolute/global minimum and not a local minima.
- Recall that in the Andrew Ng's Machine Learning course, we used to have the problem of local minima. That was because we had features of higher orders in the cost function to better fit the data. [Concept of overfitting & underfitting]

Cost Function Example Using Rocket Example in Course 01

$$J = w_1 e_{k+1}^2 + w_2 e_{k+2}^2 + \dots + w_9 T_{k+3}^2 + w_{10} T_{k+4}^2$$



Matrix Representation of Cost Function (Rocket Example)

$$J = \frac{1}{2} \vec{e}_{k+N}^T S \vec{e}_{k+N} + \frac{1}{2} \sum_{i=0}^{N-1} \left[\vec{e}_{k+i}^T Q \vec{e}_{k+i} + \vec{u}_{k+i}^T R \vec{u}_{k+i} \right]$$

- Added for later algebraic simplification
- Special weight assignment for the final state error across the horizon period
- Weight Matrix for intermediate state errors
- Weight Matrix for control input magnitude minimization

Error Matrix in our case

$$\vec{e} = \begin{bmatrix} e_\phi \\ e_\theta \\ e_\psi \end{bmatrix} = \begin{bmatrix} \phi_R - \phi \\ \theta_R - \theta \\ \psi_R - \psi \end{bmatrix}$$

Weight Matrices in our case

$$Q = \begin{bmatrix} Q_1 & 0 & 0 \\ 0 & Q_2 & 0 \\ 0 & 0 & Q_3 \end{bmatrix} \qquad S = \begin{bmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \\ 0 & 0 & S_3 \end{bmatrix}$$

$$R = \begin{bmatrix} R_1 & 0 & 0 \\ 0 & R_2 & 0 \\ 0 & 0 & R_3 \end{bmatrix}$$

Cost Function Goal

- Minimize error over the horizon
- Minimize changes in control inputs to allow us to follow the trajectory as smoothly as possible.
  - Don't equate agility with smoothlessness
  - We may set small weight values in matrix "R" to allow for agility but we will still include $\vec{\Delta u}$ as a minimization goal to assure maximum possible smooth trajectory tracking.

Minimization Terms: $\vec{e}$ u $\vec{\Delta u}$

- The cost function in Matrix form for the Rocket example will thus be reformulated.

State Augmentation

- We have $\Delta\vec{u}$ as a minimization term

$$\Delta\vec{u}_k = \vec{u}_k - \boxed{u_{k-1}}$$

This needs to be added to $\vec{x}_k$ so that $\Delta\vec{u}_k$ may be computed later for the plant

Re-writing the Discrete State Space

$$\vec{x}_{k+1} = A_d\vec{x}_k + B_d\vec{u}_k$$
$$\vec{x}_{k+1} = A_d\vec{x}_k + B_d(\vec{u}_{k-1} + \Delta\vec{u}_k)$$
$$\vec{x}_{k+1} = A_d\vec{x}_k + B_d\vec{u}_{k-1} + B_d\Delta\vec{u}_k$$
$$\vec{u}_k = \vec{u}_{k-1} + \Delta\vec{u}_k$$

$$\underbrace{\begin{bmatrix}\vec{x}_{k+1}\\ \vec{u}_k\end{bmatrix}}_{\tilde{x}_{k+1}} = \underbrace{\begin{bmatrix}A_d & B_d\\ 0_{3\times6} & I_{3\times3}\end{bmatrix}}_{\tilde{A}_d}\underbrace{\begin{bmatrix}\vec{x}_k\\ \vec{u}_{k-1}\end{bmatrix}}_{\tilde{x}_k} + \underbrace{\begin{bmatrix}B_d\\ I_{3\times3}\end{bmatrix}}_{\tilde{B}_d}\Delta\vec{u}_k$$

(6×1, 6×6, 6×3, 6×1, 3×1, 3×1)

$$\vec{y}_k = C_d\vec{x}_k + D_d\vec{u}_k$$

$D_d \to 0$ in our case

$$\vec{y}_k = C_d\vec{x}_k$$

$$\underbrace{\begin{bmatrix}\vec{y}_k\\ 0_{3\times1}\end{bmatrix}}_{\tilde{y}_k} = \underbrace{\begin{bmatrix}C_d & 0_{3\times3}\end{bmatrix}}_{\tilde{C}_d}\underbrace{\begin{bmatrix}\vec{x}_k\\ \vec{u}_{k-1}\end{bmatrix}}_{\tilde{x}_k}$$

(3×1, 3×1, 3×6, 6×1, 3×1)

Matrix Dimensions

$$\tilde{A}_d = 9\times9 \;,\; \tilde{B}_d = 9\times3 \;,\; \tilde{C}_d = 3\times9$$

Selecting First Control output Vector From MPC

- MPC provides us with control output vectors equal to the horizon period.

$$\vec{u}_k = \vec{u}_{k-1} + \begin{bmatrix} I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \underbrace{\begin{bmatrix} \Delta\vec{u}_k \\ \Delta\vec{u}_{k+1} \\ \Delta\vec{u}_{k+2} \\ \Delta\vec{u}_{k+3} \end{bmatrix}}_{\Delta\vec{U}_g}$$

Global

Re-Formulating the Cost Function & Simplification

$$J = \frac{1}{2}\vec{e}_{k+N}^{\,T} S \vec{e}_{k+N} + \frac{1}{2}\sum_{i=0}^{N-1}\left[\vec{e}_{k+i}^{\,T} Q \vec{e}_{k+i} + \Delta\vec{u}_{k+i}^{\,T} R \Delta\vec{u}_{k+i}\right]$$

$$\vec{e}_{k+N} = \vec{r}_{k+N} - \tilde{C}_d \vec{x}_{k+N}$$

Reference values vector

$$\vec{e}_{k+i} = \vec{r}_{k+i} - \tilde{C}_d \vec{x}_{k+i}$$

$$J = \frac{1}{2}\left(\vec{r}_{k+N} - \tilde{C}_d \vec{\tilde{x}}_{k+N}\right)^T S \left(\vec{r}_{k+N} - \tilde{C}_d \vec{\tilde{x}}_{k+N}\right) +$$

$$\frac{1}{2}\sum_{i=0}^{N-1}\left[\left(\vec{r}_{k+i} - \tilde{C}_d \vec{\tilde{x}}_{k+i}\right)^T Q \left(\vec{r}_{k+i} - \tilde{C}_d \vec{\tilde{x}}_{k+i}\right) + \vec{\Delta u}_{k+i}^T R \, \vec{\Delta u}_{k+i}\right]$$

Step 1 $\left(\vec{r}_{k+N} - \tilde{C}_d \vec{\tilde{x}}_{k+N}\right)^T = \vec{r}_{k+N}^T - \left(\tilde{C}_d \vec{\tilde{x}}_{k+N}\right)^T$

$$= \vec{r}_{k+N}^T - \vec{\tilde{x}}_{k+N}^T \tilde{C}_d^T$$

Step 2 $\frac{1}{2}\left(\vec{r}_{k+N} - \tilde{C}_d \vec{\tilde{x}}_{k+N}\right)^T S \left(\vec{r}_{k+N} - \tilde{C}_d \vec{\tilde{x}}_{k+N}\right)$

$$= \frac{1}{2}\left(\vec{r}_{k+N}^T - \vec{\tilde{x}}_{k+N}^T \tilde{C}_d^T\right) S \left(\vec{r}_{k+N} - \tilde{C}_d \vec{\tilde{x}}_{k+N}\right)$$

$$= \frac{1}{2}\vec{r}_{k+N}^T S \vec{r}_{k+N} - \frac{1}{2}\vec{\tilde{x}}_{k+N}^T \tilde{C}_d^T S \vec{r}_{k+N} - \frac{1}{2}\vec{r}_{k+N}^T S \tilde{C}_d \vec{\tilde{x}}_{k+N}$$

$$+ \frac{1}{2}\vec{\tilde{x}}_{k+N}^T \tilde{C}_d^T S \tilde{C}_d \vec{\tilde{x}}_{k+N}$$

'S' is diagonal so $S^T = S$ and thus the two terms are transpose of each other. Furthermore the dimension of final answer is a scalar and transpose of a scalar equals a scalar itself. Thus, both expressions are equal and may be combined.

$$= \frac{1}{2}\vec{r}_{k+N}^T S \vec{r}_{k+N} - \vec{r}_{k+N}^T S \tilde{C}_d \vec{\tilde{x}}_{k+N} + \frac{1}{2}\vec{\tilde{x}}_{k+N}^T \tilde{C}_d^T S \tilde{C}_d \vec{\tilde{x}}_{k+N}$$

Step 3 (Open up summation term just like in step 2)

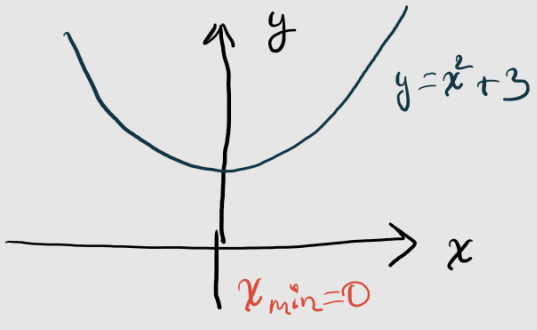
$$J = \underbrace{\frac{1}{2}\vec{r}_{k+N}^{T} S \vec{r}_{k+N}}_{\text{Constant}} - \vec{r}_{k+N}^{T} S \tilde{C}_d \vec{\tilde{x}}_{k+N} + \frac{1}{2}\vec{\tilde{x}}_{k+N}^{T} \tilde{C}_d^{T} S \tilde{C}_d \vec{\tilde{x}}_{k+N}$$

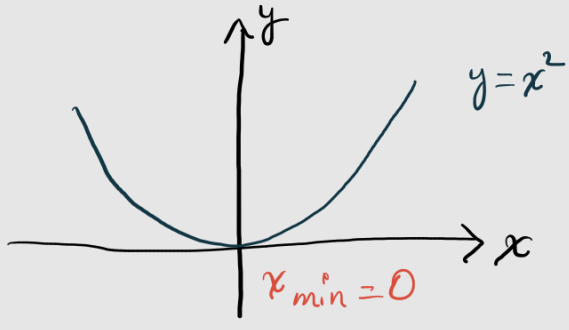
$$+ \frac{1}{2}\sum_{i=0}^{N-1}\Big[\underbrace{\vec{r}_{k+i}^{T} Q \vec{r}_{k+i}}_{\text{Constant}} - \underbrace{\vec{r}_{k+i}^{T} Q \tilde{C}_d \vec{\tilde{x}}_{k+i}}_{\text{Constant for } i=0} + \vec{\tilde{x}}_{k+i}^{T} \tilde{C}_d^{T} Q \tilde{C}_d \vec{\tilde{x}}_{k+i} + \vec{\Delta u}_{k+i}^{T} R \vec{\Delta u}_{k+i}\Big]$$

Note: $\vec{\Delta u}_k \neq$ a determined state. MPC decides $\vec{\Delta u}_k$ to send $\vec{u}_k$ to plant.

Step 4 (Eliminate Constants from Step 3)

- Constants have no influence on the minimum location. The minimum value of J will obviously change by excluding constants but its location w.r.t independant variables won't be effected.





Step 5 (Shorthand of writing summation)

- We ignore the constant terms and write the whole cost function in matrix form.

Let,

$$\vec{r}_G = \begin{bmatrix} \vec{r}_{k+1} \\ \vec{r}_{k+2} \\ \vec{r}_{k+3} \\ \vec{r}_{k+4} \end{bmatrix}, \quad \vec{\tilde{x}}_G = \begin{bmatrix} \vec{\tilde{x}}_{k+1} \\ \vec{\tilde{x}}_{k+2} \\ \vec{\tilde{x}}_{k+3} \\ \vec{\tilde{x}}_{k+4} \end{bmatrix}, \quad \vec{\Delta u}_G = \begin{bmatrix} \vec{\Delta u}_k \\ \vec{\Delta u}_{k+1} \\ \vec{\Delta u}_{k+2} \\ \vec{\Delta u}_{k+3} \end{bmatrix}$$

∴ N = 4 in our case ∴

$$J' = \frac{1}{2}\,\underset{1\times 36}{\vec{\tilde{x}}_G^{\,T}}\;\underbrace{\begin{bmatrix} \tilde{C}_d^T \tilde{Q} \tilde{C}_d & 0_{9\times 9} & 0_{9\times 9} & 0_{9\times 9} \\ 0_{9\times 9} & \tilde{C}_d^T \tilde{Q} \tilde{C}_d & 0_{9\times 9} & 0_{9\times 9} \\ 0_{9\times 9} & 0_{9\times 9} & \tilde{C}_d^T \tilde{Q} \tilde{C}_d & 0_{9\times 9} \\ 0_{9\times 9} & 0_{9\times 9} & 0_{9\times 9} & \tilde{C}_d^T S \tilde{C}_d \end{bmatrix}}_{\text{9x9}}\;\underbrace{\vec{\tilde{x}}_G}_{36\times 1} - \underset{1\times 12}{\vec{r}_G^{\,T}}\;\underbrace{\begin{bmatrix} \tilde{Q}\tilde{C}_d & 0_{3\times 9} & 0_{3\times 9} & 0_{3\times 9} \\ 0_{3\times 9} & \tilde{Q}\tilde{C}_d & 0_{3\times 9} & 0_{3\times 9} \\ 0_{3\times 9} & 0_{3\times 9} & \tilde{Q}\tilde{C}_d & 0_{3\times 9} \\ 0_{3\times 9} & 0_{3\times 9} & 0_{3\times 9} & S\tilde{C}_d \end{bmatrix}}_{\text{3x9}}\;\underbrace{\vec{\tilde{x}}_G}_{36\times 1}$$

$$= \underbrace{\tilde{\Theta}}_{\text{0}} + \frac{1}{2}\,\underset{1\times 12}{\vec{\Delta u}_G^{\,T}}\;\underbrace{\begin{bmatrix} R & 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} \\ 0_{3\times 3} & R & 0_{3\times 3} & 0_{3\times 3} \\ 0_{3\times 3} & 0_{3\times 3} & R & 0_{3\times 3} \\ 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} & R \end{bmatrix}}_{\tilde{R}}\;\underbrace{\vec{\Delta u}_G}_{12\times 1}$$

$$\begin{bmatrix} \tilde{Q} = 36\times 36 \text{ Matrix} \\ \tilde{T} = 12\times 36 \text{ Matrix} \\ \tilde{R} = 12\times 12 \text{ Matrix} \end{bmatrix}$$

$$\boxed{J' = \frac{1}{2}\vec{\tilde{x}}_G^{\,T}\tilde{Q}\vec{\tilde{x}}_G - \vec{r}_G^{\,T}\tilde{T}\vec{\tilde{x}}_G + \frac{1}{2}\vec{\Delta u}_G^{\,T}\tilde{R}\vec{\Delta u}_G}$$

∴ Compact Cost Function

Accounting Augmentation in Future State Prediction Formula

Let k=0 & n=4

$$\begin{bmatrix} \vec{\tilde{x}}_1 \\ \vec{\tilde{x}}_2 \\ \vec{\tilde{x}}_3 \\ \vec{\tilde{x}}_4 \end{bmatrix} = \begin{bmatrix} \tilde{B}_d & 0 & 0 & 0 \\ \tilde{A}_d\tilde{B}_d & \tilde{B}_d & 0 & 0 \\ \tilde{A}_d^2\tilde{B}_d & \tilde{A}_d\tilde{B}_d & \tilde{B}_d & 0 \\ \tilde{A}_d^3\tilde{B}_d & \tilde{A}_d^2\tilde{B}_d & \tilde{A}_d\tilde{B}_d & \tilde{B}_d \end{bmatrix}\begin{bmatrix} \vec{\Delta u}_0 \\ \vec{\Delta u}_1 \\ \vec{\Delta u}_2 \\ \vec{\Delta u}_3 \end{bmatrix} + \begin{bmatrix} \tilde{A}_d \\ \tilde{A}_d^2 \\ \tilde{A}_d^3 \\ \tilde{A}_d^4 \end{bmatrix}\vec{\tilde{x}}_0$$

$\vec{\tilde{x}}_G$ = $\tilde{C}$ · $\vec{\Delta u}_G$ + $\hat{A}$

Further Cost Function Simplification

$$J' = \frac{1}{2}\vec{\tilde{x}}_G^T \bar{\bar{Q}}\, \vec{\tilde{x}}_G - \vec{r}_G^T \bar{\bar{T}}\, \vec{\tilde{x}}_G + \frac{1}{2}\Delta\vec{u}_G^T \bar{\bar{R}}\, \Delta\vec{u}_G$$

$$J' = \frac{1}{2}\left(\bar{\bar{C}}\Delta\vec{u}_G + \hat{\bar{A}}\vec{x}_k\right)^T \bar{\bar{Q}}\left(\bar{\bar{C}}\Delta\vec{u}_G + \hat{\bar{A}}\vec{x}_k\right) - \vec{r}_G^T \bar{\bar{T}}\left(\bar{\bar{C}}\Delta\vec{u}_G + \hat{\bar{A}}\vec{x}_k\right) + \frac{1}{2}\Delta\vec{u}_G^T \bar{\bar{R}}\,\Delta\vec{u}_G$$

≡ Lecture 166 contains some important information for the non-simplified state matrix update scheme

$$J' = \frac{1}{2}\left(\Delta\vec{u}_G^T \bar{\bar{C}}^T + \vec{x}_k^T \hat{\bar{A}}^T\right)\bar{\bar{Q}}\left(\bar{\bar{C}}\Delta\vec{u}_G + \hat{\bar{A}}\vec{x}_k\right) - \vec{r}_G^T \bar{\bar{T}}\left(\bar{\bar{C}}\Delta\vec{u}_G + \hat{\bar{A}}\vec{x}_k\right) + \frac{1}{2}\Delta\vec{u}_G^T \bar{\bar{R}}\,\Delta\vec{u}_G$$

$$J' = \frac{1}{2}\Delta\vec{u}_G^T \bar{\bar{C}}^T \bar{\bar{Q}}\,\bar{\bar{C}}\,\Delta\vec{u}_G + \left[\frac{1}{2}\Delta\vec{u}_G^T \bar{\bar{C}}^T \bar{\bar{Q}}\,\hat{\bar{A}}\vec{x}_k\right] + \left[\frac{1}{2}\vec{x}_k^T \hat{\bar{A}}^T \bar{\bar{Q}}\,\bar{\bar{C}}\Delta\vec{u}_G\right] + \left[\frac{1}{2}\vec{x}_k^T \hat{\bar{A}}^T \bar{\bar{Q}}\,\hat{\bar{A}}\vec{x}_k\right] - \vec{r}_G^T \bar{\bar{T}}\,\bar{\bar{C}}\,\Delta\vec{u}_G - \vec{r}_G^T \bar{\bar{T}}\,\hat{\bar{A}}\vec{x}_k + \frac{1}{2}\Delta\vec{u}_G^T \bar{\bar{R}}\,\Delta\vec{u}_G$$

Scalar qty Transpose of one another. Thus same thing

Constant

Constant

Ignoring Constants and Grouping

$$J'' = \frac{1}{2}\Delta\vec{u}_G^T\left(\bar{\bar{C}}^T \bar{\bar{Q}}\,\bar{\bar{C}} + \bar{\bar{R}}\right)\Delta\vec{u}_G + \begin{bmatrix}\vec{x}_k^T & \vec{r}_G^T\end{bmatrix}\begin{bmatrix}\hat{\bar{A}}^T \bar{\bar{Q}}\,\bar{\bar{C}} \\ -\bar{\bar{T}}\,\bar{\bar{C}}\end{bmatrix}\Delta\vec{u}_G$$

$\bar{\bar{H}}$ (12 x 12)

1 x 21

$\bar{\bar{F}}^T$ (21 x 12)

12 x 1

Final Cost Function

$$J'' = \frac{1}{2}\Delta\vec{u}_G^{\,T}\,\bar{\bar{H}}\,\Delta\vec{u}_G + \left[\vec{\tilde{x}}_k^{\,T} \;\; \vec{r}_G^{\,T}\right]\bar{\bar{F}}^T\,\Delta\vec{u}_G$$

- By doing all this manipulation the cost function now has only one unknown which is $\Delta\vec{u}_G$ that we are looking to compute to generate optimum control action for the plant.
- Minimum value of J'' will be when its first derivative is computed and put equal to zero.

Finding $\Delta\vec{u}_G$

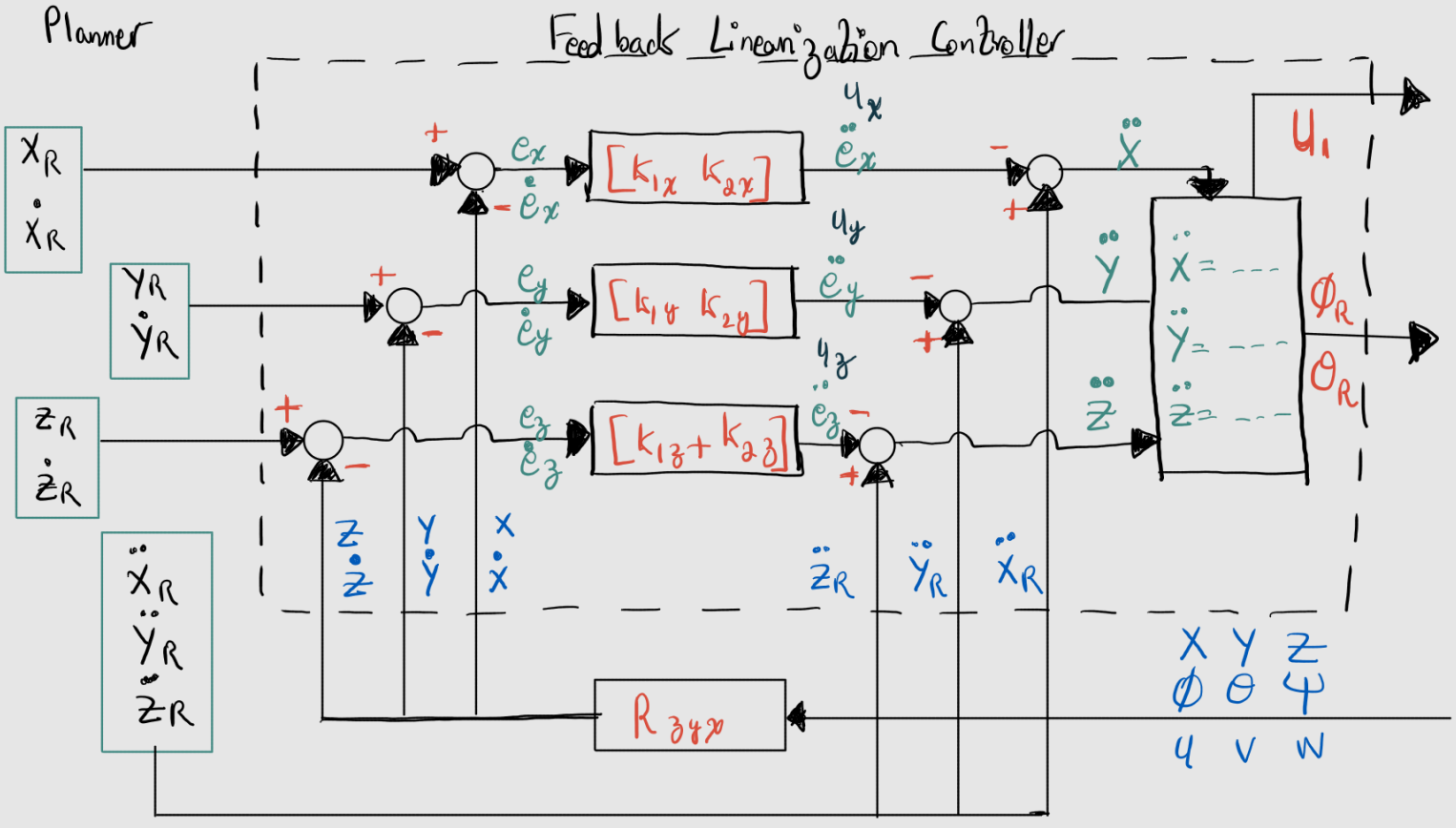
$$\Delta J'' = \bar{\bar{H}}\,\Delta\vec{u}_G + \bar{\bar{F}}\begin{bmatrix}\vec{\tilde{x}}_k \\ \vec{r}_G\end{bmatrix} = 0$$

$$\boxed{\Delta\vec{u}_G = -\bar{\bar{H}}^{-1}\,\bar{\bar{F}}\begin{bmatrix}\vec{\tilde{x}}_k \\ \vec{r}_G\end{bmatrix}}$$

∴ Final Product of the MPC Controller

≡ $\bar{\bar{H}}$ & $\bar{\bar{F}}^T$ change with each inner loop due to Linear Parameter Varying employed.

Feedback Linearization Controller



- Feedback linearization is a common non-linear control approach, whose main idea is to transform a non-linear system into a fully or partially decoupled linear one by means of state feedback and non-linear transformation, so that linear control strategies can be used.

- Also called **Exact Linearization**.

Equations Relevant for Position Control

$$\dot{u} = vr - wq + g\sin\theta$$
$$\dot{v} = wp - ur - g\sin\phi\cos\theta$$
$$\dot{w} = uq - vp - g\cos\phi\cos\theta + U_1/m$$

- We need expressions in inertial frame of reference so that we can track the trajectories generated by the planner.
- Converting the above 3 equations directly into inertial frame through the use of rotation matrix is very difficult.

- It is much easier to carry out the derivation using Newton-Euler 6DOF eqns but this time by representing forces directly in inertial frame as opposed to the body frame.

$$\vec{F}_{net} = m\vec{a}^E = m\ddot{\vec{\Gamma}}^E$$

$\vec{\Gamma}$ → Position Vector in Inertial Frame

$$\begin{bmatrix} \vec{\Gamma}^E = X\hat{i} + Y\hat{j} + Z\hat{k} \\ \dot{\vec{\Gamma}}^E = \dot{X}\hat{i} + \dot{Y}\hat{j} + \dot{Z}\hat{k} + X\dot{\hat{i}} + Y\dot{\hat{j}} + Z\dot{\hat{k}} \\ \dot{\vec{\Gamma}}^E = \dot{X}\hat{i} + \dot{Y}\hat{j} + \dot{Z}\hat{k} \\ \ddot{\vec{\Gamma}}^E = \ddot{X}\hat{i} + \ddot{Y}\hat{j} + \ddot{Z}\hat{k} \end{bmatrix}$$

$$m\begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}}_{\text{gravity}} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{\text{Gyroscopic Precession}} + \begin{bmatrix} R_{zyx} \underbrace{\begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix}}_{\text{Control Input}} \end{bmatrix}$$

Upon solution we get

Relevant EOMs in inertial frame. Degree of non-linearity is clearly evident

$$\ddot{x} = (\cos\phi\cos\theta\cos\psi + \sin\phi\sin\psi)\frac{u_1}{m}$$

$$\ddot{y} = (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)\frac{u_1}{m}$$

$$\ddot{z} = -g + \cos\phi\cos\theta\frac{u_1}{m}$$

- Degree of non-linearity in these equations justify the use of two separated cascaded controllers.
- Also note that the above representation is just a relationship of $\ddot{x}$, $\ddot{y}$ & $\ddot{z}$ with ϕ, θ, ψ & u_1. Note that the above isn't a state space formulation.

The Intermediate Control Inputs

- We have 3 'K' matrices in the schematic of this controller which get multiplied with error and error derivatives. In some essence the individual 'K' scalars are somewhat analogous to proportional gain. This analogy is true to some extent as we will later see similarities especially when using the root locus plots.
- Note that this is a mere analogy with the proportional controller and not some forced equivalence.
- The big question about the intermediate part is that how do we select the 'K' values? Let's dig deeper.

From the Controller Schematics

$$u_x = k_{1x} e_x + k_{2x} \dot{e}_x$$
$$u_y = k_{1y} e_y + k_{2y} \dot{e}_y$$
$$u_z = k_{1z} e_z + k_{2z} \dot{e}_z$$

Create an analogy b/w the two systems

Homogeneous LTI 2nd Order Differential Eqn

$$\ddot{y} + p\dot{y} + qy = 0$$
$$\ddot{y} = (-q)y + (-p)\dot{y}$$

Analogy results

$$u_x = \ddot{e}_x \;,\; u_y = \ddot{e}_y \;,\; u_z = \ddot{e}_z$$

Crux of the Philosophy

Although this is an analogy but it is a good one. Modelling the system this way will allow us to find $\ddot{e}$ values that will force $e \to 0$ when $t \to \infty$. The required $\ddot{e}$ for this will be used to find corresponding $\ddot{x}, \ddot{y}$ & $\ddot{z}$ that are best for the system in the long run. These $\ddot{x}, \ddot{y}$ & $\ddot{z}$ will be put into the derived Inertial Frame positional EOMs to find the required ϕ_R, θ_R and U_1 that are then fed downstream into the control architecture.

The Intermediate Control Law

$$\ddot{e}_x = k_{1x} e_x + k_{2x} \dot{e}_x$$
$$\ddot{e}_y = k_{1y} e_y + k_{2y} \dot{e}_y$$
$$\ddot{e}_z = k_{1y} e_z + k_{2y} \dot{e}_z$$

* 2^{nd} Order LTI System
* Error is the dependant variable
* Time is the independant variable

- This will now be leveraged to lead $e \to 0$ as $t \to \infty$.