FPGA-MPC

Model Predictive Control

Direct Transcription

$$\min_{x_0, u_0 \ldots x_{N-1}, u_{N-1}, x_N} l_f(x_N) + \sum_{k=0}^{N-1} l(x_k, u_k)$$

Minimize a separable cost across time

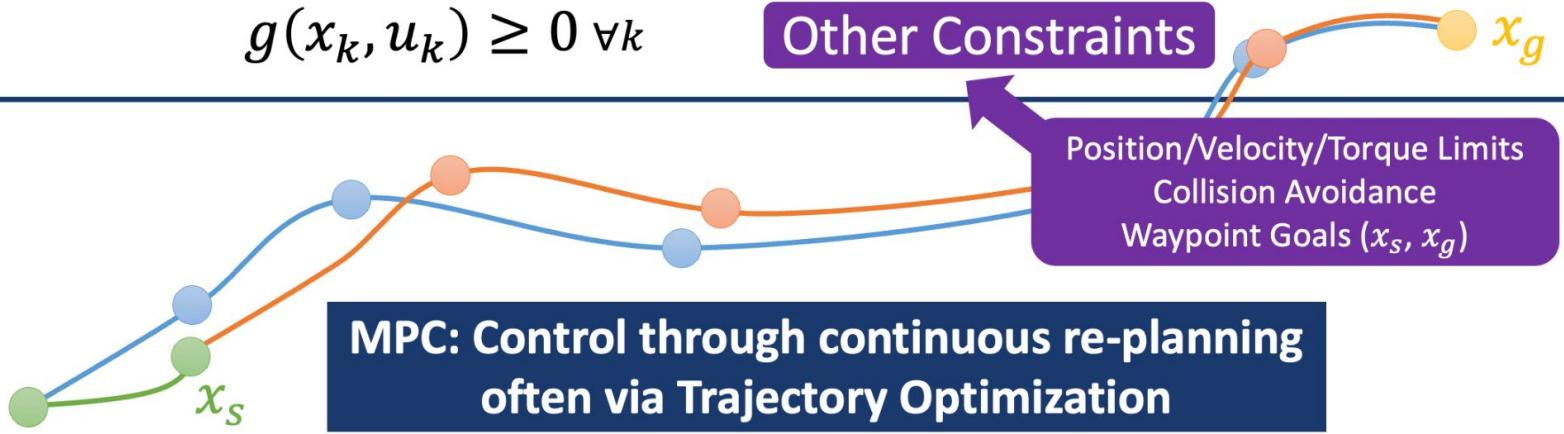
subject to: $f(x_k, u_k) = x_{k+1} \ \forall k \in [0, N)$

Obey Physics

$g(x_k, u_k) \geq 0 \ \forall k$

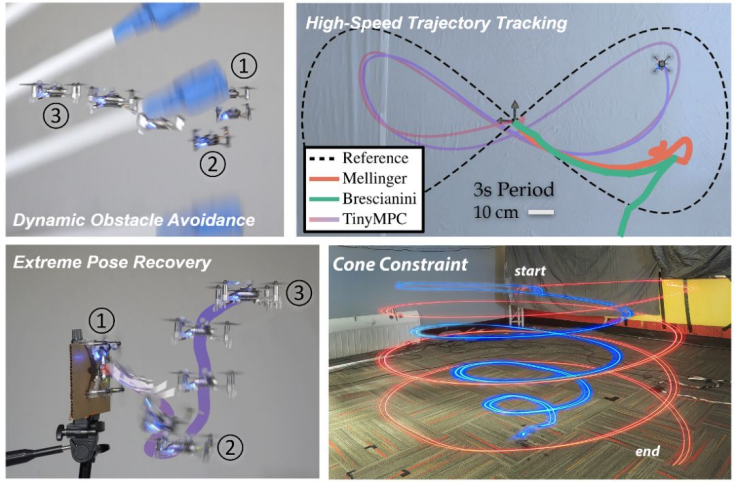
Other Constraints

Position/Velocity/Torque Limits
Collision Avoidance
Waypoint Goals (x_s, x_g)



MPC: Control through continuous re-planning often via Trajectory Optimization

TinyMPC: Model-Predictive Control on Resource-Constrained Microcontrollers



STM32F405
168 Mhz MCU

196kB RAM
1MB Flash

Khai Nguyen[1]*, Sam Schoedel[2]*, Anoushka Alavilli[3]*, Brian Plancher[4], Zachary Manchester[2]
Sam Schoedel[2]*, Khai Nguyen[1]*, Elakhya Nedumaran[3], Brian Plancher[4], Zachary Manchester[2]
1: Mechanical Engineering, 2: Robotics Institute, 3: Electrical & Computer Engineering, Carnegie Mellon University
4: Barnard College, Columbia University *These authors contributed equally to this work.

The Alternating Direction Method of Multipliers (ADMM)

$$\min_{y} f(y)$$

subject to: $y \in \Omega$ **Must be a Convex Set!**

$$\min_{y} f(y) + I_{\Omega}(z)$$

subject to: $y = z$

Indicator Function

$$I_{\Omega}(z) = \begin{cases} 0 & \text{if } z \in \Omega \\ \infty & \text{otherwise} \end{cases}$$

Augmented Lagrangian

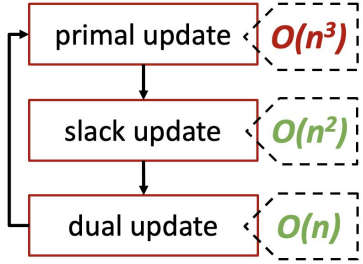
$$\mathcal{L}_A(y, z, \lambda) = \underbrace{f(y) + I_{\Omega}(z)}_{\text{Cost}} + \underbrace{\lambda^T(y - z) + \frac{\rho}{2}\|y - z\|_2^2}_{\text{Penalty}}$$

ADMM Algorithm

$$y^+ = \arg\min_{y} \mathcal{L}_A(y, z, \lambda)$$

$$z^+ = \arg\min_{z} \mathcal{L}_A(y^+, z, \lambda)$$

$$\lambda^+ = \lambda + \rho(y^+ - z^+)$$



Limiting factor $O(n^3)$!

TinyMPC: compressed ADMM for MPC on microcontrollers

$$\min_{X,U} \quad x_N^T Q_N x_N + \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k$$

subject to: $Ax_k + Bu_k = x_{k+1} \ \forall k \in [0, N)$

Primal Update

Key Idea

To **save memory and compute** what if we used the **infinite horizon solution** around a **single linearization**?

Normal O(n³) Backwards Pass

$$K_k = (R + B^T P_{k+1} B)^{-1} (B^T P_{k+1} A)$$

$$d_k = (R + B^T P_{k+1} B)^{-1} (B^T p_{k+1} + r_k)$$

$$P_k = Q + K_k^T R K_k + (A - BK_k)^T P_{k+1} (A - BK_k)$$

$$p_k = q_k + (A - BK_k)^T (p_{k+1} - P_{k+1} B d_k) + K_k^T (R d_k - r_k)$$

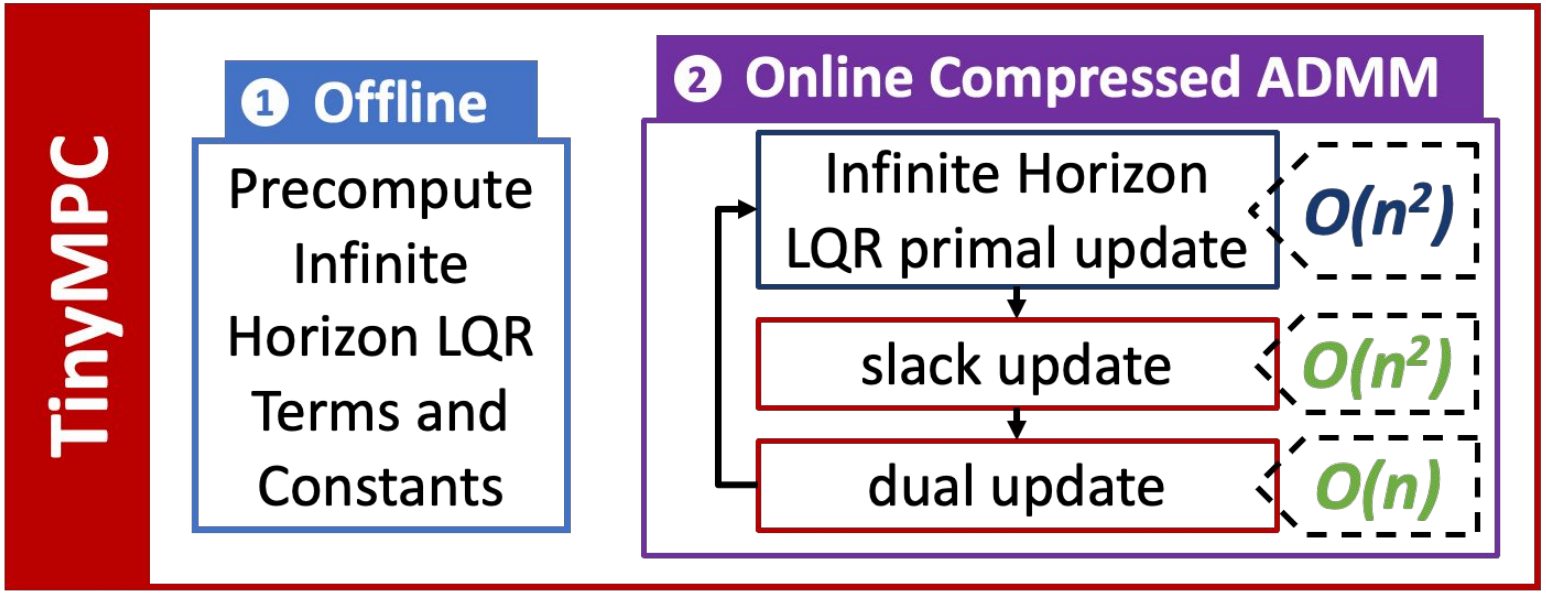
Feedback gain

$$\cancel{K_k = (R + B^T P_{k+1} B)^{-1} (B^T P_{k+1} A)} \rightarrow K_\infty$$

$$\cancel{d_k = (R + B^T P_{k+1} B)^{-1} (B^T p_{k+1} + r_k)} \rightarrow C_1 (B^T p_{k+1} + r_k)$$

$$\cancel{P_k = Q + K_k^T R K_k + (A - BK_k)^T P_{k+1} (A - BK_k)} \rightarrow P_\infty$$

$$\cancel{p_k = q_k + (A - BK_k)^T (p_{k+1} - P_{k+1} B d_k) + K_k^T (R d_k - r_k)} \rightarrow q_k + C_2 p_{k+1} - K_\infty r_k$$



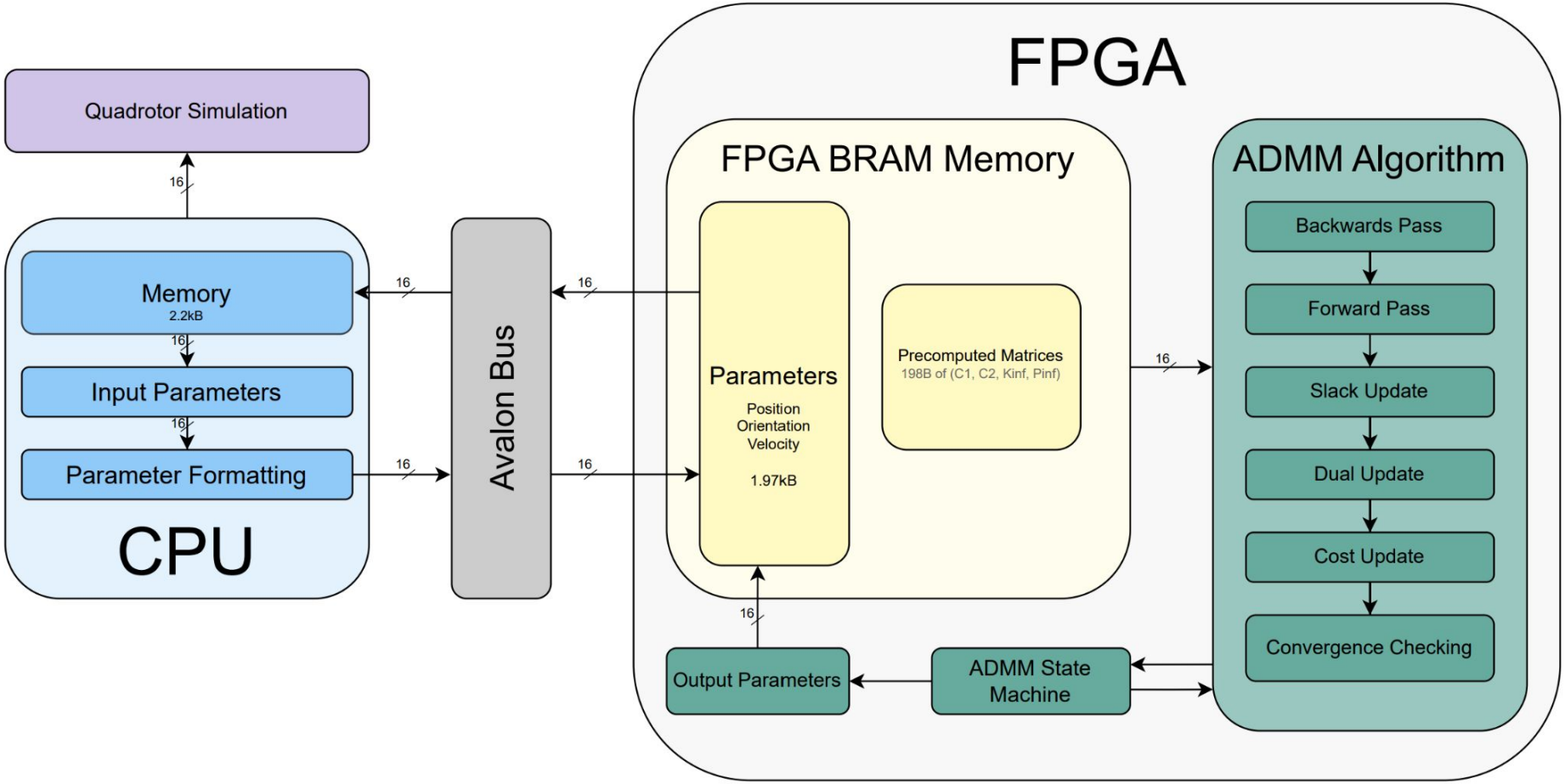
Offline Pre

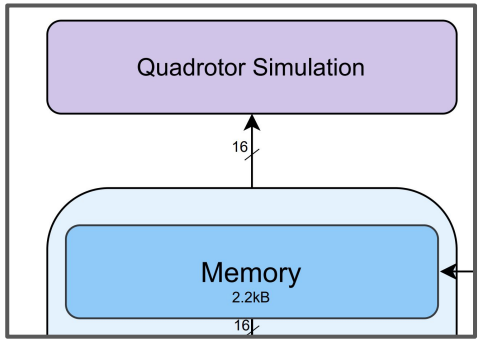
$$P_\infty, K_\infty \rightarrow \begin{cases} C_1 = (R + B^T P_\infty B)^{-1} \\ C_2 = (A - BK_\infty)^T \end{cases}$$

Online

$$d_k = C_1(B^T p_{k+1} + r_k)$$

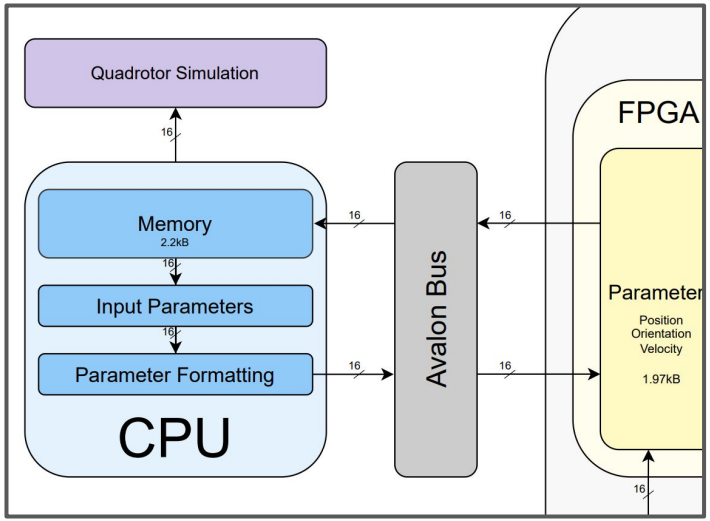
$$p_k = q_k + C_2 p_{k+1} - K_\infty r_k$$


Quadrotor Simulation
16
Memory
2.2kB
16
Input Parameters
16
Parameter Formatting
CPU
16
16
Avalon Bus
16
16
FPGA
FPGA BRAM Memory
Parameters
Position
Orientation
Velocity
1.97kB
Precomputed Matrices
198B of (C1, C2, Kinf, Pinf)
16
ADMM Algorithm
Backwards Pass
Forward Pass
Slack Update
Dual Update
Cost Update
Convergence Checking
16
Output Parameters
ADMM State Machine



Quadrotor Input: u_1, u_2, u_3, u_4
Motor thrust commands, each between 0 and 1.
Each value is a fixed-point → 16 bits
Total = 16 * 4 = 64 bits.

Quadrotor Output: x, y, z, φ, θ, ψ, dx, dy, dz, dφ, dθ, dψ
Position, orientation, linear velocity, angular velocity
Each value is a fixed-point → 16 bits
Total = 16 * 12 = 192 bits



The hardware in FPGA part is built in Qsys. The data translate through Lightweight HPS-to-FPGA Bridge is converted into Avalon-MM master interface. So the IP PIO controller and HEX Controller works as the Avalon-MM slave in the system. They control the pins related to the LED and HEX to change the LED and HEX's state. This is similar to the system using NIOS II processor to control LED and HEX.

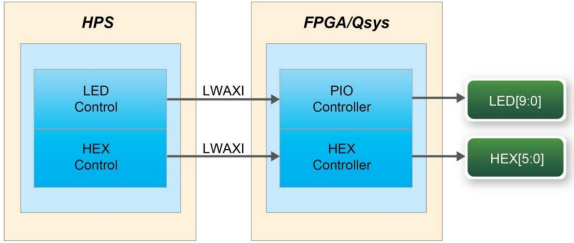
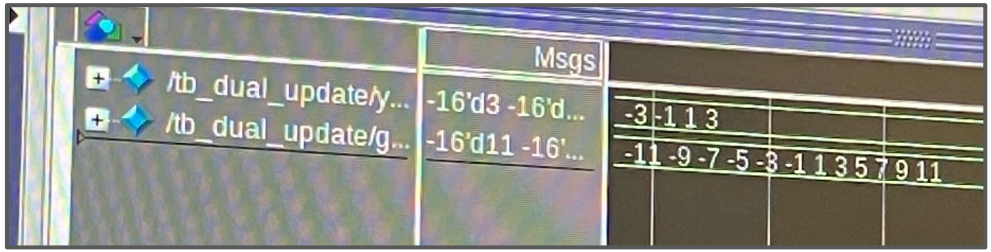


Figure 7-1 HPS Control FPGA LED and HEX

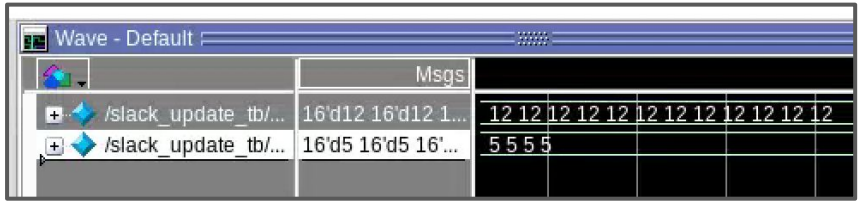
Individual Stage Outputs

Forwards Pass Output

Dual-Update Output



Slack-Update Output



As you can see slack-update is clipping properly where we purposely set the inputs to slack update to bc beyond the min and max values and we see correct clip.

Primal Update

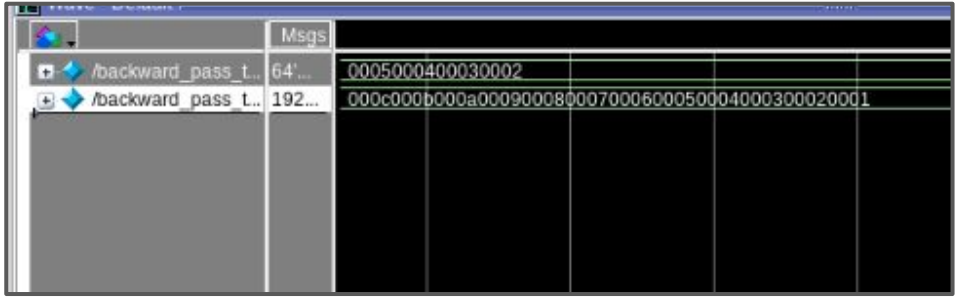
```
# addr | d_mem | p_mem | u_mem | x_out
# ------------------------------------
#  0   |     1 |     2 |
#  1   |     2 |     2 |
#      |       |       |     0 |
#      |       |       |    -1 |
#      |       |       |       | x[0]=    x
#      |       |       |       | x[1]=    0
#      |       |       |       | x[2]=    0
#      |       |       |       | x[3]=    x
# ** Note: $finish    : /homes/user/stud/spring25/rjh2173/work/tb_primal_update_alex3.sv(166)
#    Time: 245 ps  Iteration: 0  Instance: /tb_primal_update_alex3
# 1
# Break in Module tb_primal_update_alex3 at /homes/user/stud/spring25/rjh2173/work/tb_primal_update_alex3.sv li
ne 166

VSIM 345>
```

Backwards Pass

```
# --- d_k (feedforward) ---
# d_k[0] = 2
# d_k[1] = 3
# d_k[2] = 4
# d_k[3] = 5
# --- p_out (linear term) ---
# p_out[0] = 1
# p_out[1] = 2
# p_out[2] = 3
# p_out[3] = 4
# p_out[4] = 5
# p_out[5] = 6
# p_out[6] = 7
# p_out[7] = 8
# p_out[8] = 9
# p_out[9] = 10
# p_out[10] = 11
# p_out[11] = 12
# ** Note: $finish    : /homes/user/stud/spring25/rjh2173/work/backward_pass_tb.sv(113)
#    Time: 60 ns  Iteration: 0  Instance: /backward_pass_tb
# 1
# Break in Module backward_pass_tb at /homes/user/stud/spring25/rjh2173/work/backward_pass_tb.s
113
```

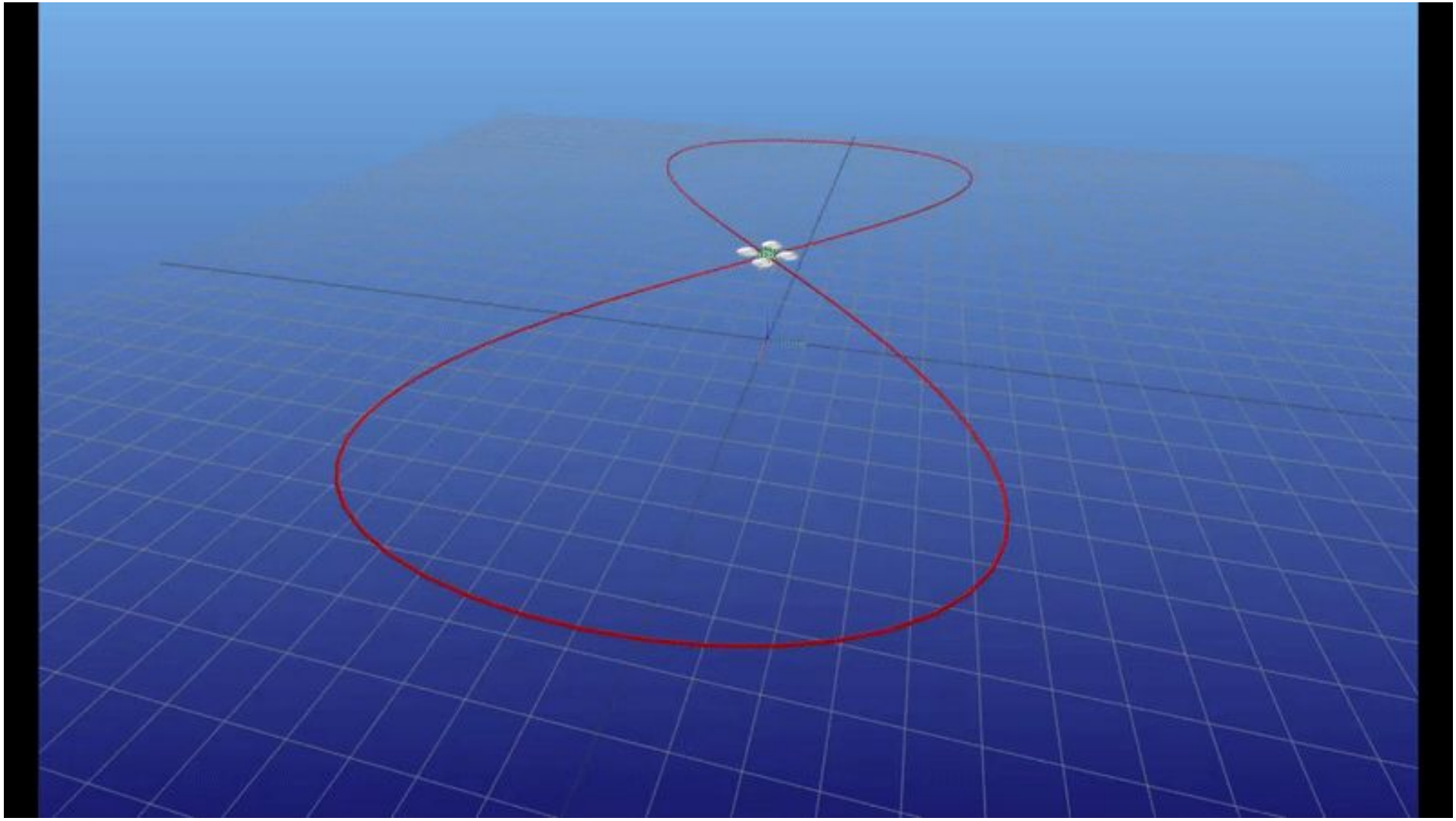
LL#55

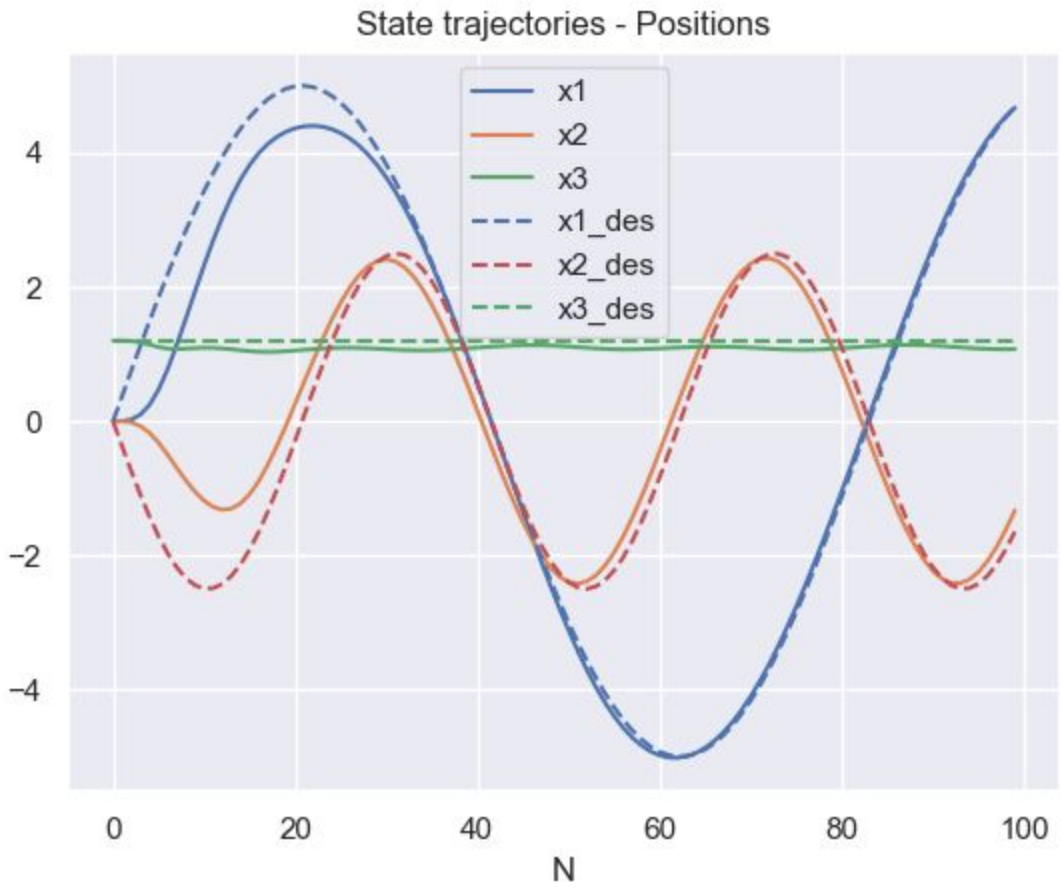


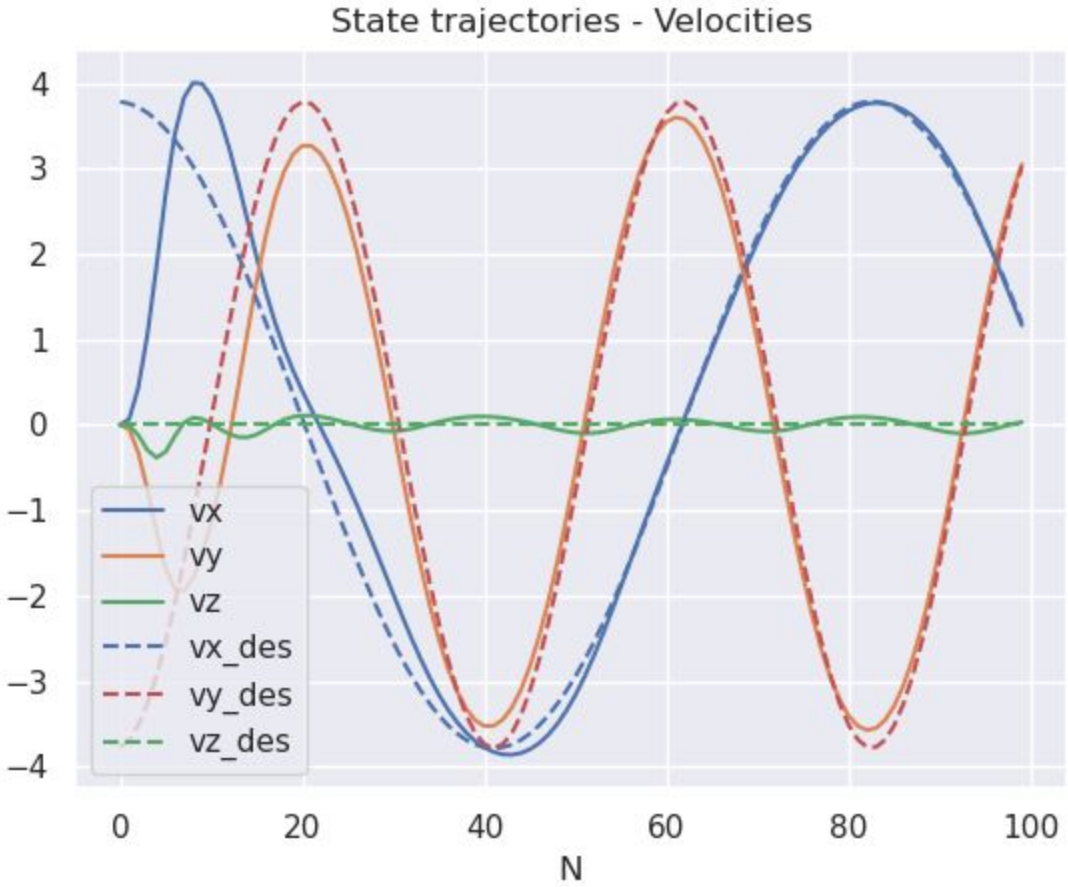
Residual Computation

```
# === Simulation Results ===
# dual_res    = 6  (expected 6)
# converged   = 0  (expected 0)
# done        = 1  (expected 1)
# ** Note: $finish    : /homes/user/stud/spring25/rjh2173/work/tb_residual_calculator_alex.sv(115)
#    Time: 285 ns  Iteration: 0  Instance: /tb_residual_calculator_alex
# 1
# Break in Module tb_residual_calculator_alex at /homes/user/stud/spring25/rjh2173/work/tb_residual_c
alculator_alex.sv line 115

VSIM 11>
```




State trajectories - Positions
x1
x2
x3
x1_des
x2_des
x3_des
4
2
0
−2
−4
0
20
40
60
80
100
N


State trajectories - Velocities
4
3
2
1
0
−1
−2
−3
−4
0
20
40
60
80
100
N
vx
vy
vz
vx_des
vy_des
vz_des