Optimizing Placement in Mixed-Size Circuits

Arunachalam Ramesh
EP22B004
B.Tech, Engineering Physics, IIT Madras

Indian Institute of Technology, Madras

December 2,2024

Pre-Introduction

- Mixed Size Placement is a critical step in the physical design flow of integrated circuits.
- The solution must satisfy density constraints, minimize wire length, and ensure legal positions.
- Traditional stochastic approaches like simulated annealing fail due to complexity and poor convergence, often getting stuck in local minima.
- Analytical approaches generally use pseudo energy or potential functions.

Introduction and Objective

- The problem is analyzed through the perspective of a physics-based approach.
- The cells in the system are modeled as particles that exhibit attractive forces with connected components, while repelling each other at very close proximities to avoid overlap. This behavior is analogous to Lennard-Jones potential used to simulated classical particles.
- The objective is to address the question: "Are there any special repulsive potentials that help in faster relaxation to desired states without overlap?"

Theoretical Background

- **Gradient Descent:** Minimizes a function iteratively using local gradients.
- **Natural Gradient:** Adapts step size dynamically by accounting for the geometry of the energy landscape/cost function completely.
- **Manhattan Distance:** Suitable for grid-based placements using linear paths.
- **Log-Sum Exponent (LSE):** Smooth approximation for HPWL (Half-Perimeter Wire Length).
- **Green's Function:** Solution for a dynamic system with an impulse input.
- **Automatic Differentiation:** Calculates Jacobians and Hessians for differential simulations.

Description about the project

So the pathway of approach starts with two ways:

- To try and test our Natural Gradient, we take a network problem in hand, in which our goal is to set all nodes of that network into motion via a natural gradient, like what happens in a over-damped spring mass system.
- Our final solution is going to be a repulsive potential, and a differential equation referring to the dynamics of motion in it.
- To obtain the above mentioned things, we are trying to use a method called **Differential Simulations**.
- To work on differential simulations we take a simple one particle spring motion, where we find the right input for desired trajectory, which is called **Forward Simulations** and then implement a differential simulation on that system, so that we get familiarized with it and hence we go ahead and implement it in the density constraint equation we want.

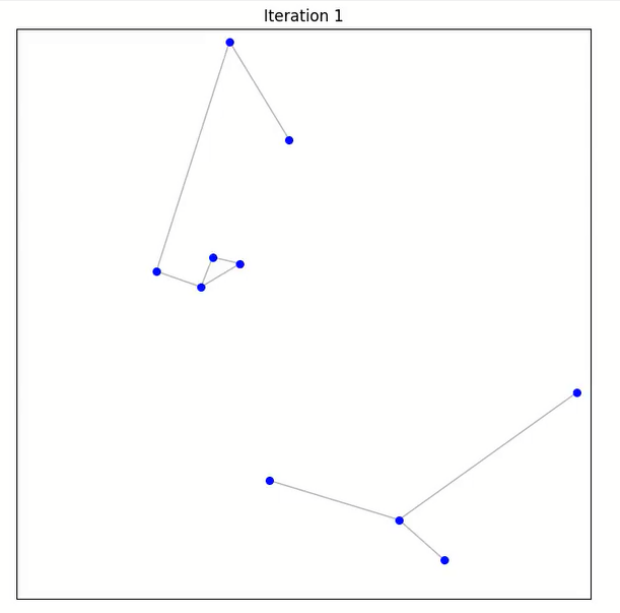
Graph Networks as Spring-Mass Systems

- Network formation method.
- Static Gradient Descent and its output.
- Natural Gradient Descent - What, Why, and How?
- **Note: In all cases we will be using a very highly damped case and for negligible masses.**

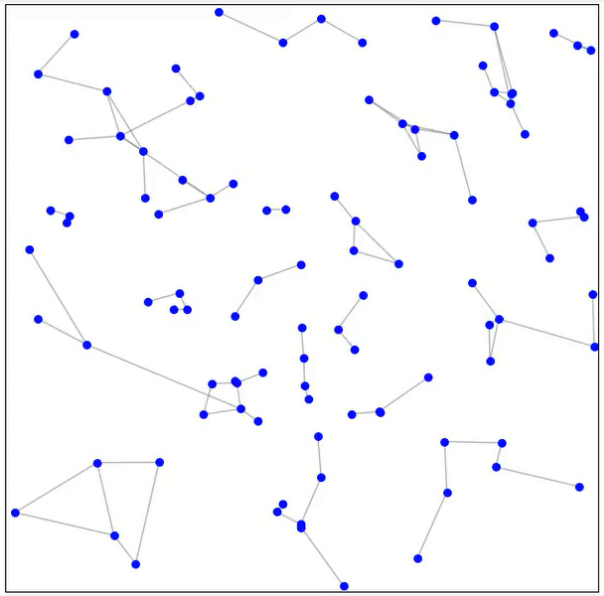
Network Formation

- The network formation algorithm initially selects m nodes from the nearest m^2 elements within a set of n randomly generated nodes.
- It enforces an upper bound on the number of nodes per net by removing distant nodes when the net's content exceeds this limit.
- A lower bound is applied to ensure no node remains isolated.
- While these constraints are implemented, the process may not always achieve 100% accuracy due to factors like triangulation and net formation order.
- Nevertheless, this approach generates a power-law structure for the graph.

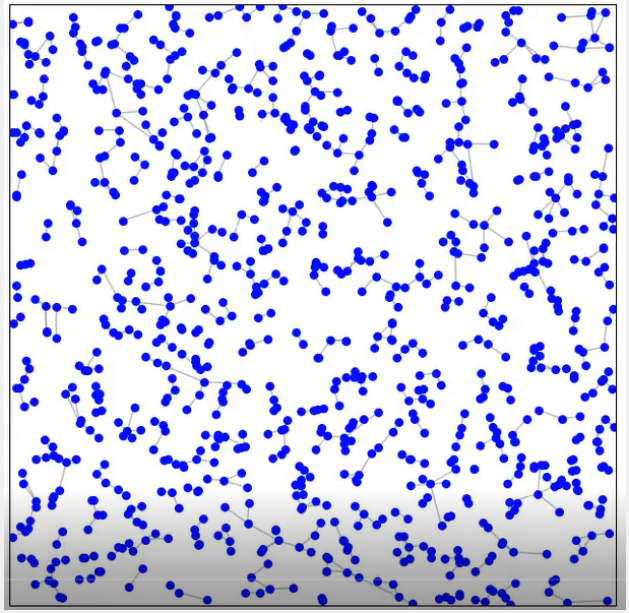
Images of Formed Network



(a) 10 nodes



(b) 100 nodes



(c) 1000 nodes

Parameters used: m = 2, lower bound being 2 and upper bound being 6

Static Gradients

- It is known as static Gradients since the step size is fixed, and the direction it goes is normal to the trajectory.
- Motion governed by:

$$\mathcal{V}(\mathbf{x}_i) = \sum_{i,j} \frac{1}{2} \mathbf{x}_{ij}^{\mathsf{T}} \mathbb{K} \mathbf{x}_{ij}, \quad \mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i$$

- Dynamics include:

$$\dot{\mathbf{x}}_i = \dot{\mathbf{x}}_i(1 - \mathrm{d}t) - \nabla_{\mathbf{x}_i} \mathcal{V}(\mathbf{x}_i)\mathrm{d}t/\eta,$$
$$\mathbf{x}_i = \mathbf{x}_i + \dot{\mathbf{x}}_i \mathrm{d}t$$

Force Balance for Overdamped Motion

Using Newton's Second Law:

$$m\frac{d\mathbf{v}_i}{dt} = \mathbf{F}_{\text{spring}} + \mathbf{F}_{\text{damping}} + \mathbf{F}_{\text{ext}}$$

For an overdamped system ($m \to 0$):

$$0 = -k(\mathbf{x}_i - \mathbf{x}_{i,\text{initial}}) - \gamma\mathbf{v}_i - \nabla_{\mathbf{x}_i} L$$

Rearrange to find the velocity:

$$\mathbf{v}_i = -\frac{1}{\gamma}\left[k(\mathbf{x}_i - \mathbf{x}_{i,\text{initial}}) + \nabla_{\mathbf{x}_i} L\right]$$

Forces contributing to motion:

- **Spring force:** $-k(\mathbf{x}_i - \mathbf{x}_{i,\text{initial}})$
- **Damping force:** $-\gamma\mathbf{v}_i$
- **External force:** $-\nabla_{\mathbf{x}_i} L$

Discretization of Velocity Equation

Starting from the overdamped velocity equation:

$$\mathbf{v}_i = -\frac{1}{\gamma}\left[k(\mathbf{x}_i - \mathbf{x}_{i,\text{initial}}) + \nabla_{\mathbf{x}_i} L\right]$$

Using forward Euler discretization:

$$\frac{\mathbf{v}_i(t+\Delta t) - \mathbf{v}_i(t)}{\Delta t} = -\frac{k}{\gamma}(\mathbf{x}_i(t) - \mathbf{x}_{i,\text{initial}}) - \frac{\gamma}{\gamma}\mathbf{v}_i(t) - \frac{1}{\gamma}\nabla_{\mathbf{x}_i} L$$

Rearrange to find:

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \Delta t \cdot \left[-\frac{k}{\gamma}(\mathbf{x}_i - \mathbf{x}_{i,\text{initial}}) - \mathbf{v}_i(t) - \frac{1}{\gamma}\nabla_{\mathbf{x}_i} L\right]$$

Key Updates:

- **Velocity:** Combines spring, damping, and external forces.
- **Position (implicit in discretization):**

$$\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \mathbf{v}_i(t) \cdot \Delta t$$

Natural Gradient with Mahalanobis Metric

- It has adaptive step length due to the fact that it has knowledge about the entire trajectory due to factor of covariance of the potential gradient being used.
- **Matrix A:** Ensures positive semi-definiteness via $A = B^T B$.
- **Loss Function:**

$$V_k = \sum_{ij} x_{ij}^T A x_{ij}, \quad x_{ij} = x_j - x_i$$

- **Covariance Regularization:**

$$C_{\text{mahab}}(x_i) = C(x_i) + \lambda I$$

- **Natural Gradient:**

$$M^{-1}(x_i) = (C_{\text{mahab}}^{-1}(x_i))^T \nabla_{x_i} V(x_i)$$

Results and Observation

- **Convergence using Static Gradient**
  - 1000 nodes, $>$ 10000 iterations are required.
- **Convergence using Mahalanobis Gradient:**
  - 1000 nodes, 170-180 iterations with $dt \in [0.033, 0.034], \eta \in [0.99, 0.995]$.
  - 1000 nodes, $\eta \approx 1$, $dt = 0.03$, only 95 iterations.
  - $\lambda = 0.01$ used for Regularization is the ideal value.
  - More the number of nodes, less the distance between nodes, faster the convergence.
- **Dynamic Stability:** Low damping ($\eta \ll 0.5$) causes erratic motion since all the approximations have been done for over damped case.

Manhattan Distance and Log-Sum Exponent

- **Manhattan Distance:**
  - In our actual usage, we might need to deal with only restricted movements along certain directions.
  - Hence we try to implement Manhattan based coordinate system and set them into motion
- **Log-Sum Exponent:**
  - Here the maximum length in a net is taken to calculate the potential, via taking log of sum of all the of all the distances.
  - This however slows down the motion as it reduces the values of gradient,i.e., it has a nearly tangential hyperbolic kind of structure, making it slower to reach the optimal point.
  - Nearly 370-380 iterations for suitably adjusted parameters for a system with 1000 nodes.
  - Values of the parameters: $\gamma = 0.1, \mathrm{d}t = 0.04$

Note: The potential energy was the only thing which was computed by LSE method. Networking was done by the previous method only.

Adding Constraints and observing the dynamics

- **Process of adding the constraints** : In every net a random point was chosen, and it was fixed to the position it was present initially.
- **Observation and Inferences from the dynamics**:
  - Every net had its convergence point as the point which was fixed in a particular net.
  - The convergence even slower due to the fact that distance between nodes is not being changed by both but only one of the nodes, and hence the convergence eventually ends up taking higher number of iterations from there on.
  - The most optimized convergence took 750-950 iterations when the potential used was based on LSE.
  - Speed of convergence tends to be on the positive side when its quadratic version of potential is used.

Dynamics of a Spring-Mass System

- The motion of a spring-mass system is governed by:

$$m\ddot{x} + \eta\dot{x} + kx = f$$

  where:
  - m: Mass
  - η: Damping constant
  - k: Spring constant
  - f: External forcing
- The solution depends on initial conditions and the driving force.

Formal Solution with Green's Function

- Assuming $x(t) \to x^*$, the Green's function for the system is:

$$g(t) = e^{-\eta t/2m} \frac{e^{dt/2m} - e^{-dt/2m}}{d},$$

where:

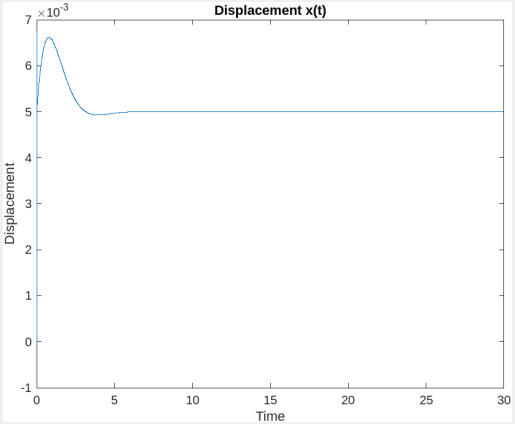
$$d = \sqrt{\eta^2 - 4km}.$$

- The general solution is given by:

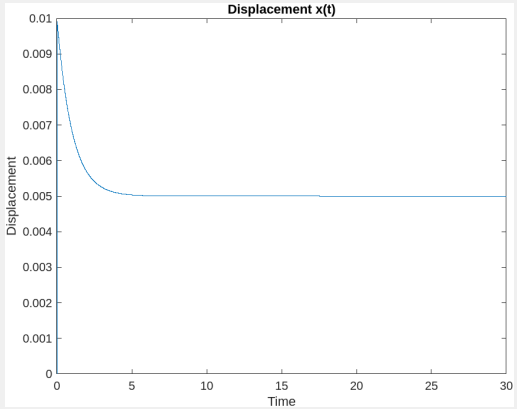
$$x(t) = \int_0^t g(t - t')q(t')\, dt',$$

with $q(t')$ representing the external force.

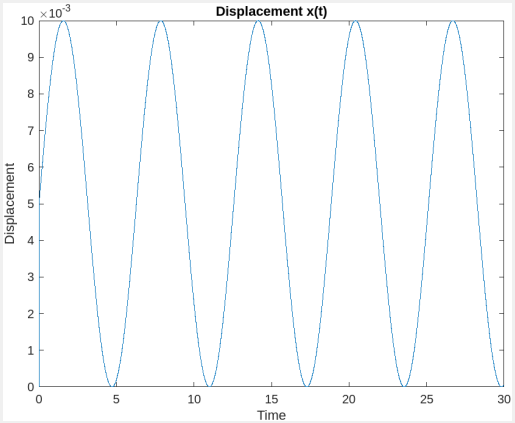
Solutions for Different Driving Forces



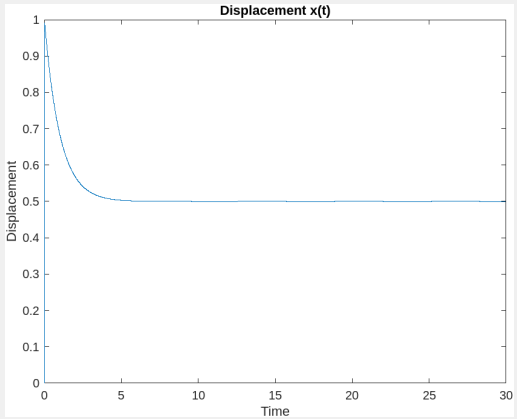
(a) $q(t) = e^{-t}sin(t) + 1$



(b) $q(t) = 100e^{-100t} + sin(t)/20 + 100$



(c) $q(t) = sin(t) + 1$



(d) $q(t) = e^{-t} + 1$

Control Input derived using PD Control

- The objective is to drive the system to a target displacement x_0, minimizing the projected error at some T=20:

$$\lim_{t \to \infty} \left(x^*(T) - x_0 \right) = 0.$$

- Control force is defined as:

$$q(t) = -K_p x - K_d \dot{x},$$

where K_p and K_d are optimized gains.

- Using Non-Linear Programming (NLP), the optimal gains were found:

$$K_p = 999.9997, \quad K_d = 999.995.$$

Simulation Derivative,Control Input and Trajectory

- Both $\mathbf{x}(t)$ and $q(t)$ are of the form

$$\mathbf{x}(t) = Ae^{-Dt}\sin(Bt + C) + E,$$

with variables being:

  - **For** $\mathbf{x}(t)$:
    $A = 11.55,\ B = 0.87,\ C = 1.047,\ D = 0.497,\ E = 9.99.$
  - **For** $q(t)$:
    $A = 11561,\ B = 0.87,\ C = 1.047,\ D = 0.497,\ E = -4990.$

- All the values obtained is for $m = 1000, \eta = 0.5, k = 1, x_0 = 10$

Simulation Derivative: The result which is nearly similar to one from sensitivity analysis - The amount of change in output with respect to change in input.

- Needed so that we can predict the change in loss function with respect to trajectory using this vector.
- $\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}q(t)} = a(t-d)^b + c$ where
  $a = 0.027, b = 1.362, c = 0.012, d = 2.27.$

Plots of Simulation Derivative,Control Input and Trajectory

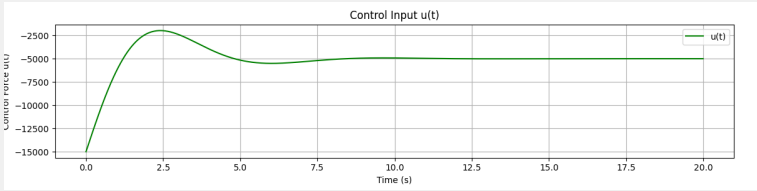


Figure: $q(t)$ for the given set of parameters (m, η, k).

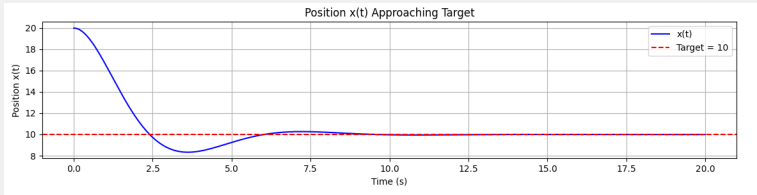


Figure: $x(t)$ for the given set of parameters (m, η, k).

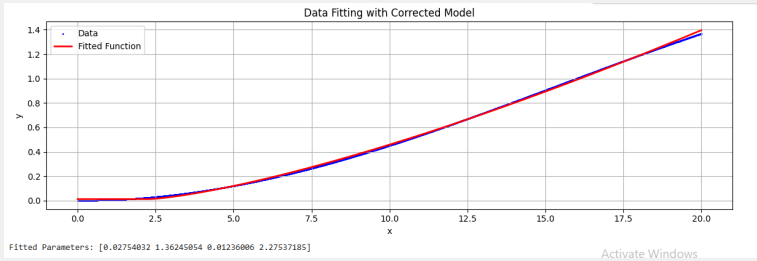


Figure: Plot for simulation derivative $\frac{dx}{dq}$.

Solution using Optimal Control

- **Cost Function**: Updated to include input along the path, with terminal cost add-up being difference from expected position.
- Using RK4 method to integrate the path from the current position and finding the prospective final position, and finding the ideal initial input value to start with to get the aspired trajectory goes key here.
- Hence multiple iterations are ran to find the right possible control input.

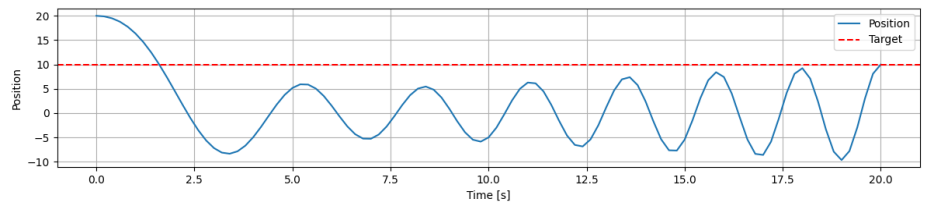


Figure: These can be possible trajectories which are not desired but satisfy the conditions. Hence multiple iterations must be ran.

The input and trajectory we finally obtained

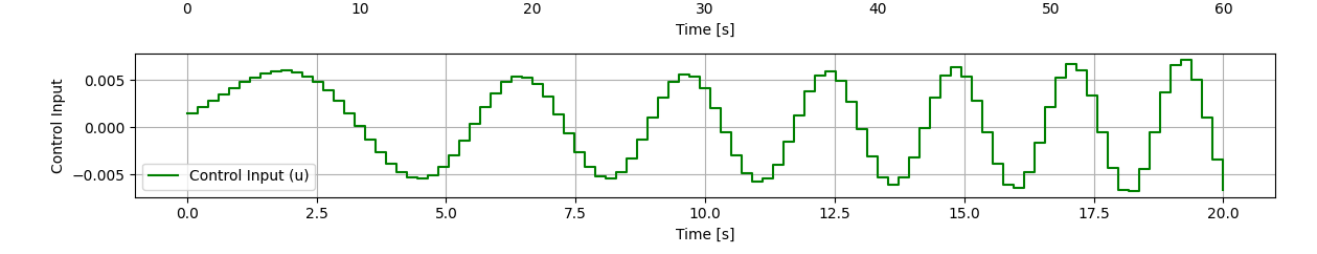


Figure: This is the input which should be given

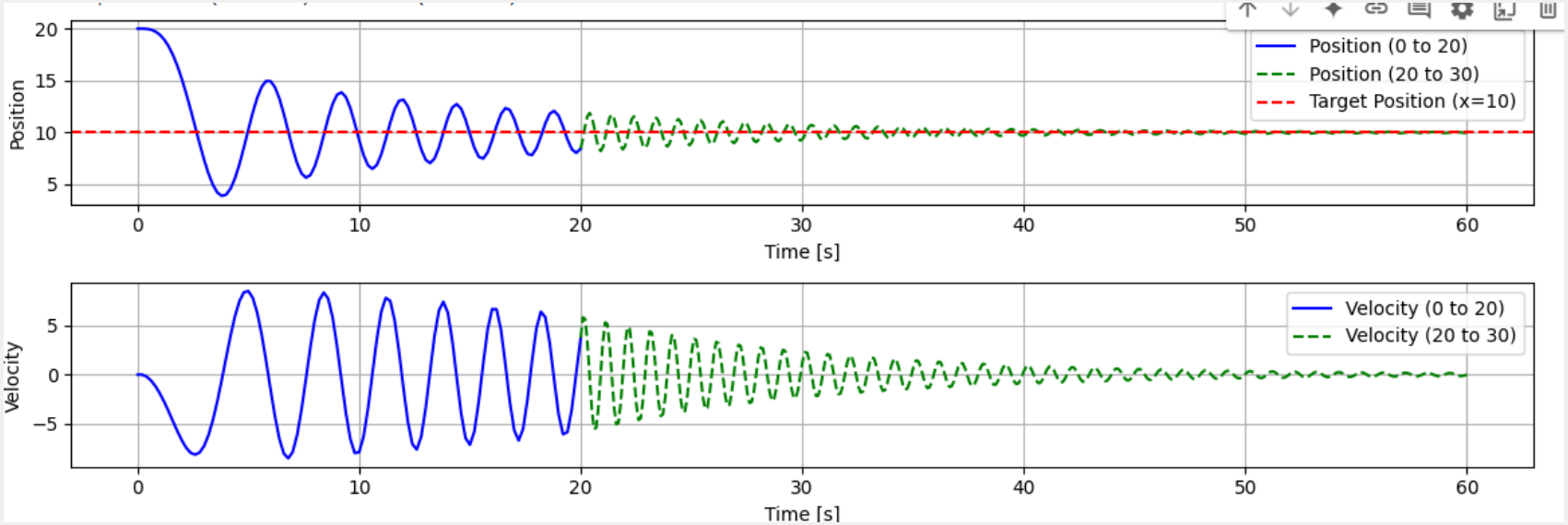


Figure: Top plot is trajectory, and bottom one is velocity. The blue line is obtained within the Control Horizon and dotted green line is post that.

Future steps to be taken

- Converting the positional constraints into density constraints.
- Implementing a forward simulation based ODE to get hold of differential simulations
- Finding a similar system ODE to optimize the density constraints and using differential simulations to actually implement it.
- Minimizing the cost function and optimizing the nodes' position using Mahalanobis Gradient implemented earlier