Andrew Cunningham

Introduction

Problem Statement

Approach

Incremental Cost Function

Dynamic Programming

Results

Citations

Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators

Andrew Cunningham

Rensselaer Polytechnic Institute

cunnia5@rpi.edu

October 7, 2016

Overview

1 Introduction

2 Problem Statement

3 Approach
- Incremental Cost Function
- Dynamic Programming

4 Results

5 Citations

Introduction

Andrew Cunningham

Introduction

Problem Statement

Approach

Incremental Cost Function

Dynamic Programming

Results

Citations

Motivation

Robot manipulators are widely used in industry and optimal operation of these devices can reduce monetary costs for manufactured goods.

Challenges

Optimal control of robot manipulators is challenging because they have coupled, nonlinear dynamics and have torque constraints on their joints.

Background

Andrew Cunningham

Introduction

Problem Statement

Approach

Incremental Cost Function

Dynamic Programming

Results

Citations

The problem of robot control is typically broken down into two problems

1. Path trajectory planning: The process by which a trajectory profile is generated. The profile takes a spatial path as input and outputs a time indexed plan of the robot's position, velocity and joint torques.
2. Path tracking: Control loop used to follow the plane generated in the path trajectory planner

Informal Problem Statement

Andrew Cunningham

Introduction
Problem Statement
Approach
Incremental Cost Function
Dynamic Programming
Results
Citations

- We want to find the control signals that will drive a robot from a start position to an end position
- Suppose the path has already been parameterized to avoid collisions

$$q^i = f^i(\lambda) = a_4\lambda^3 + a_3\lambda^2 + a_2\lambda + a_1 \tag{1}$$

- Our robot has constraints on joint torques and forces must follow joint dynamics
- We have some cost defined for the set of controls we select

Informal Problem Statement

Informal Problem Statement

What control signals will direct a robot to follow a pre-specified curve in joint space with minimum cost while satisfying constraints on joint torques, robot dynamics and boundary conditions?

Formal Problem Statement

$$\begin{cases} find: \\ \min_{u_i} \int_0^{\lambda_{max}} L(\lambda, \theta, u_i) d\lambda \\ subject\ \ to \\ u_i = J_{ij}\ddot{q}^j + C_{ijk}q^j\dot{q}^k + R_{ij}\dot{q}^j + G_i \quad \text{(Robot Dynamics)} \\ \forall_i : \ \ u_{i\ min} < u_i < u_{i\ max} \quad \text{(Realizable Torques)} \end{cases} \tag{2}$$

Symbol	Meaning
L	Cost Function
u_i	Torque for joint i
J_{ij}	Mass Inertia Matrix
R_{ij}	Viscous Friction Matrix
C_{ijk}	Centrifugal and Coriolis coefficients
G_i	Gravitational Loading vector
$E(q, \dot{q})$	Realizable Torques

Table: Meaning of symbols in Equation 2

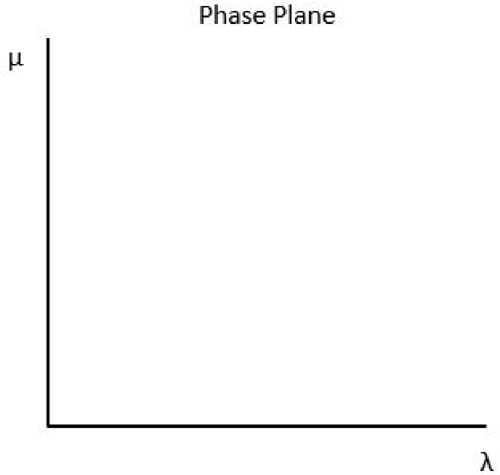
Approach

- Dynamic Programming can be used to find the joint torques
- Due to the fact that the path is parameterized, there will be only two state variables (λ and $\dot{\lambda}$) regardless of the number of joints so the **curse of dimensionality is avoided**

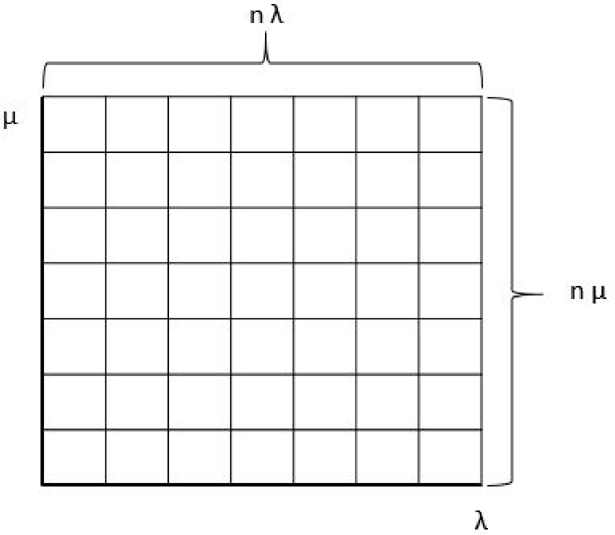
Approach - Steps to Apply Dynamic Programming

1. Divide phase plane into a grid
2. Rework cost function to be a function of phase coordinates
3. Apply dynamic programming

Approach - Phase plane to grid



Approach - Phase plane to grid



Approach - Phase plane cost function

Andrew Cunningham
Introduction
Problem Statement
Approach
Incremental Cost Function
Dynamic Programming
Results
Citations

Given the system dynamics and the parameterization of a path:

$$u_i = J_{ij}\ddot{q}^j + C_{ijk}q^j\dot{q}^k + R_{ij}\dot{q}^j + G_i \quad \text{(Dynamics)} \tag{3}$$

$$q^i = f^i(\lambda) \quad \text{(Paramaterized Path)} \tag{4}$$

We can substitute equation (4) into equation (3) to find joint torques as a function of path parameters to get:

$$u_i = J_{ij}\frac{df^j}{d\lambda}\dot{\mu} + (J_{ij}\frac{d^2f^j}{d\lambda^2} + C_{ijk}\frac{df^j}{d\lambda}\frac{df^k}{d\lambda})\mu^2 + R_{ij}\frac{df^j}{d\lambda}\mu + G_i \tag{5}$$

Approach - Phase plane cost function

For simplicity rewrite:

$$u_i = J_{ij}\frac{df^j}{d\lambda}\dot{\mu} + (J_{ij}\frac{d^2 f^j}{d\lambda^2} + C_{ijk}\frac{df^j}{d\lambda}\frac{df^k}{d\lambda})\mu^2 + R_{ij}\frac{df^j}{d\lambda}\mu + G_i \quad (5)$$

into

$$u_i = M_i\dot{\mu} + Q_i\mu^2 + R_i\mu + G_i \quad (6)$$

Approach - Phase plane cost function

Equations in 6 can then be made into a single equation by projecting u_i onto the velocity vector $\frac{df^i}{d\lambda}$

$$U = u_i \frac{df^i}{d\lambda} = M\dot{\mu} + Q\mu^2 + R\mu + G$$
$$where\ M = M_i \frac{df^i}{d\lambda}, Q = Q_i \frac{df^i}{d\lambda}, G = G_i \frac{df^i}{d\lambda} \tag{7}$$

Approach - Phase plane cost function

If Equation 7 is then divided by μ we can get a dynamics expression that is not directly dependent on time:

$$M\frac{d\mu}{d\lambda} + Q\mu + R + \frac{1}{\mu}(G - U) = 0 \tag{8}$$

This is significant because the dynamic programming algorithm can use λ,μ as the state variables for dynamic programming

Approach - Phase plane cost function

A solution that satisfies our previously developed dynamics

$$U = u_i \frac{df^i}{d\lambda} = M\dot{\mu} + Q\mu^2 + R\mu + G\frac{df^i}{d\lambda} \tag{7}$$

at boundary conditions $\mu(\lambda_k) = \mu_0, \ \mu(\lambda_{k+1}) = \mu_1$ can take the form of:

$$u_i = Q_i\mu^2 + R_i\mu + V_i \tag{9}$$

Approach - Phase plane cost function

Andrew Cunningham

Introduction

Problem Statement

Approach

Incremental Cost Function

Dynamic Programming

Results

Citations

Our newly developed u

$$u_i = Q_i \mu^2 + R_i \mu + V_i \tag{9}$$

can be plugged into

$$M \frac{d\mu}{d\lambda} + Q\mu + R + \frac{1}{\mu}(G - U) = 0 \tag{8}$$

to generate:

$$\frac{d\mu}{d\lambda} = -\frac{1}{\mu} \frac{(S - V)}{M} \tag{10}$$

Approach - Phase plane cost function

$$\frac{d\mu}{d\lambda} = -\frac{1}{\mu}\frac{(S-V)}{M} \tag{10}$$

can be solved to get

$$\lambda = K - \frac{M}{2(S-V)}\mu^2 \tag{11}$$

We solve for the constants of integration K and V to meet boundary conditions $(\mu(\lambda_k) = \mu_0,\ \mu(\lambda_k + 1) = \mu_1)$ to get

$$\lambda = \frac{\lambda_k(\mu_1^2 - \mu^2) + \lambda_{k+1}(\mu^2 - \mu_0^2)}{\mu_1^2 - \mu_0^2} \tag{12}$$

Approach - Phase plane cost function

Andrew Cunningham

Introduction

Problem Statement

Approach

Incremental Cost Function

Dynamic Programming

Results

Citations

Finally, we can solve for μ

Fuction for μ in terms of λ

$$\mu = \sqrt{\frac{(\lambda_{k+1} - \lambda){\mu_0}^2 + (\lambda_k - \lambda){\mu_1}^2}{\lambda_{k+1} - \lambda_k}} \tag{13}$$

Fuction for u_i in terms of λ

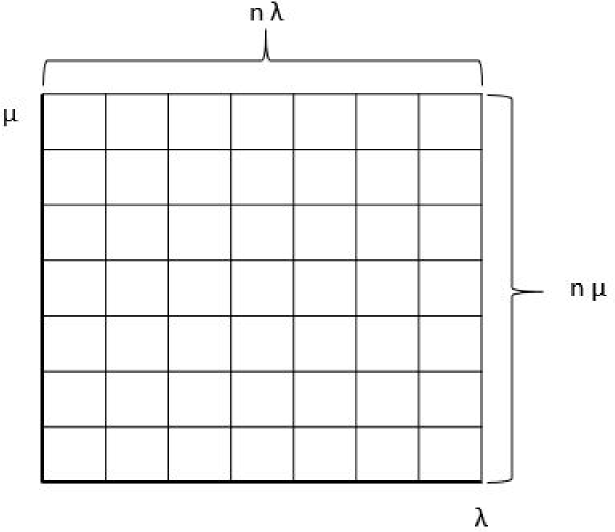
$$u_i = Q_i\mu^2 + R_i\mu + S_i + M_i\frac{\mu_1^2 - \mu_0^2}{2(\lambda_{k+1} - \lambda_k)} \tag{14}$$

Approach - Phase plane to grid

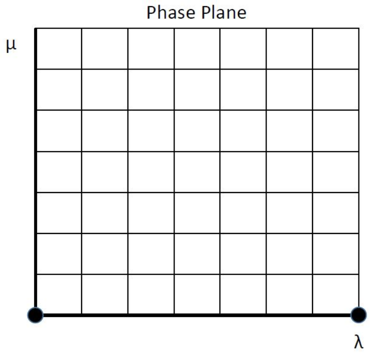
We have μ and u_i solved for! So we can now find the incremental cost:

$$C_{inc} = \int_{\lambda_k}^{\lambda_{k+1}} L(\lambda, \mu, u_i) d\lambda \tag{15}$$

Approach - Dynamic Programming



Approach - Dynamic Programming



Andrew Cunningham
Introduction
Problem Statement
Approach
Incremental Cost Function
Dynamic Programming
Results
Citations

Approach - Dynamic Programming

Andrew Cunningham

Introduction

Problem Statement

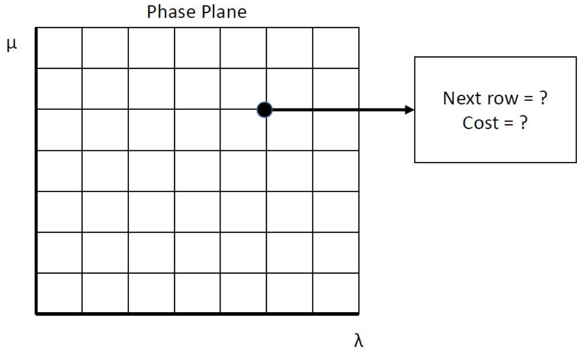
Approach

Incremental Cost Function

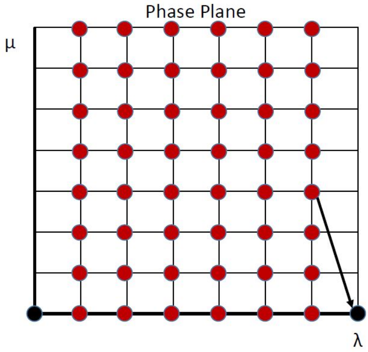
Dynamic Programming

Results

Citations



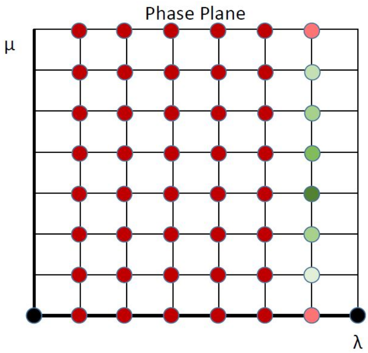
Approach - Dynamic Programming



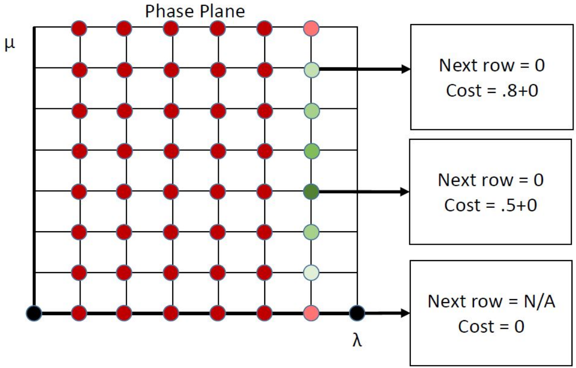
$$C = \int_{\lambda}^{\lambda_{\{k+1\}}} L\,(\lambda, \mu, u_i)\,d\lambda$$

Are joint torques within bounds?

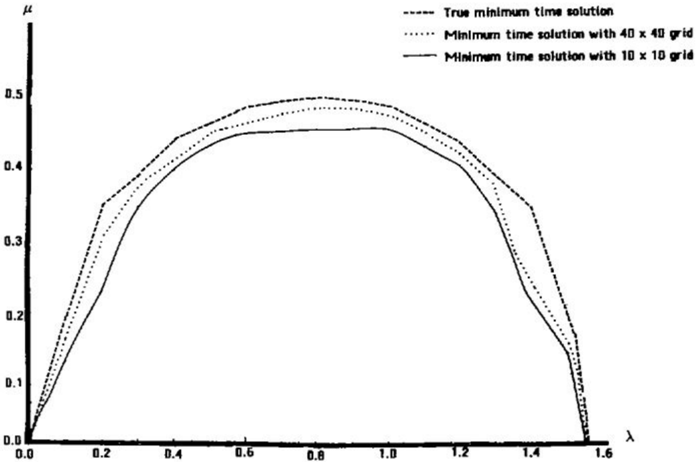
Approach - Dynamic Programming



Approach - Dynamic Programming



Paper's Results



Citations

Andrew Cunningham

Introduction

Problem Statement

Approach

Incremental Cost Function

Dynamic Programming

Results

Citations

1. Shin, Kang G., and Neil D. McKay. "A dynamic programming approach to trajectory planning of robotic manipulators." Automatic Control, IEEE Transactions on 31.6 (1986): 491-500.
2. Shin, Kang G., and Neil D. McKay. "Minimum-time control of robotic manipulators with geometric path constraints." Automatic Control, IEEE Transactions on 30.6 (1985): 531-541.