Direct Collocation

Direct Collocation, Part 1

0.0/10.0 points (graded)

A popular and effective implementation of direct trajectory optimization is direct collocation. See the paper by Hargraves and Paris, linked in the Syllabus, for a reference. This approach defines the trajectory $x(t)$ as a *spline*, specified by its value at a series of knot points, and then enforces the constraint that the time derivative of this spline match the dynamics.

Let x_k and u_k be decision variables corresponding to knot points, where h is the time between knot points. Then, the spline of interest is defined as follows:

- For every k, $x(t)$ is defined to be a cubic polynomial for t in the interval t_k to t_{k+1}.
- $x(t_k) = x_k$ and $x(t_{k+1}) = x_{k+1}$
- $\dot{x}(t_k) = f(x_k, u_k)$ and $\dot{x}(t_{k+1}) = f(x_{k+1}, u_{k+1})$

Let $t_0 = 0$, $f_0 = f(x_0, u_0)$, and $f_1 = f(x_1, u_1)$. For t in the interval 0 to h, write the cubic polynomial for the spline $x_s(t)$. HINT: the cubic term is $\left(\frac{2}{h^3}(x_0 - x_1) + \frac{1}{h^2}(f_0 + f_1)\right)t^3$.

Now, we extract the state and its derivative at the midpoint of each spline. Continuing the example above, find $x_c = x_s(.5h)$ and $\dot{x}_c = \frac{dx_s}{dt}\Big|_{t=.5h}$, where "c" stands for the collocation point.

$x_c =$

$\dot{x}_c =$

Thus far, we have computed a number of expressions in terms of the decision variables, but we have not yet written the constraints for the optimization program. If we let assume a first-order hold on control input, $u_c = .5\,(u_0 + u_1)$, constrain the slope of the spline $\dot{x}_c$ to match the dynamics, evaluated at the collocation point:

$\dot{x}_c - f\,(x_c, u_c) = 0$
where we have one such constraint between every two knot points (so there is one fewer constraint than knot points).

Submit You have used 0 of 1 attempt

Direct Collocation, Part 2

0.0/10.0 points (graded)
To get some experience coding trajectory optimization algorithms, write the function that evaluates the constraint function and its gradient. We will use the pendulum as an example, with masses and gravity set to simplify the dynamics so $\ddot{\theta} + 10\sin\theta + b\dot{\theta} = u$.

- This example will use 10 knot points, with the vector of decision variables:

$$z = \begin{bmatrix} \theta_1 \\ \dot\theta_1 \\ \theta_2 \\ \dot\theta_2 \\ \vdots \\ \dot\theta_{10} \\ u_1 \\ \vdots \\ u_{10} \end{bmatrix}$$

- Evaluate the constraint function $g(z)$ and $dg(z) = \frac{\partial g}{\partial z}$, where g is a 18×1 vector and dg is a 18×30 matrix.
- Order $g(z)$ so that

$$g(z) = \begin{bmatrix} \dot x_c - f(x_c, u_c) = 0 \text{ between knot points 1 and 2} \\ \dot x_c - f(x_c, u_c) = 0 \text{ between knot points 2 and 3} \\ \vdots \\ \dot x_c - f(x_c, u_c) = 0 \text{ between knot points 9 and 10} \end{bmatrix}$$

  where x has the usual ordering $x = \begin{bmatrix} \theta \\ \dot\theta \end{bmatrix}$

- You may find it useful to numerically confirm your computation of the gradient, or even use numerical differencing to calculate it in your solution.

```
h = .1;
b = .1+rand;
% decision variables
z = 10*randn(30,1);

% x is 2x10, where x(:,k) is [theta_k;\dot \theta_k]
x = reshape(z(1:20),2,[]);

% u is 1x10, where u(k) is u_k
u = reshape(z(21:30),1,[])';

% your code here
g =
dg =
```

```
15
```

Unanswered

Run Code

Submit You have used 0 of 3 attempts