The Direct Collocation Method for Optimal Control

Gilbert Gede

May 26, 2011

Outline

What is This?

A method to solve optimal control problems.

Which Optimal Control Problems?

Usually, trajectory optimization, parameter optization, or a combination thereof.

Why This Method?

From what I understand, the current optimization softwares are better as you add constraints, even if the dimensionality increases.

Why This Method?

From what I understand, the current optimization softwares are better as you add constraints, even if the dimensionality increases.

This method does that, in addition to better relating the states to the augmented cost function.

History

From what I can tell, Hargrave's 1987 paper in J.G.C.D. seems to be the first major publication of this method.

History

From what I can tell, Hargrave's 1987 paper in J.G.C.D. seems to be the first major publication of this method.

I think use of the collocation method for optimal control goes back to the 1970's, and for general use to the 1960's.

ODE's

The collocation method is a way of solving ODE's numerically.

ODE's

The collocation method is a way of solving ODE's numerically.

This is actually an implicit Runge-Kutta method.

ODE's

The RK4 method is:

$$\dot{y} = f(t, y) y_{n+1} = \frac{1}{6} h(k_1 + 2k_2 + 2k_3 + k_4)$$

Where h is the timestep, and each k is a slope, evaluated at multiple times and values of y. (partial steps)

This allows forward integration to calculate the state at each timestep.

ODE's

With the collocation method the states are defined as functions of states and derivatives (they are implicit).

Polynomial Representation

The states are approximated as polynomials between two boundaries. Cubic polynomials seem to be most commonly used.

$$x = C_0 + C_1 s + C_2 s^2 + C_3 S^3$$

where s is a point a general time interval between 0 and 1.

Polynomial Representation

With the prior knowledge of x (an x_0 an x_1) and $f(x)$ (derivatives at those points), at s = 0 and s = 1, and differentiation of the polynomial, we can calculate the coefficients.

Polynomial Representation

We can make things simpler by examing midpoint of this interval. (s = 0.5)
This simplifies to:

$$x_c = \frac{1}{2}(x_1 + x_2) + \frac{\Delta t}{8}(f_1 - f_2)\dot{x_c} = -\frac{3}{2\Delta t}(x_1 - x_2) - \frac{1}{4}(f_1 + f_2)$$

where f is the derivative of x, evaluated at x_1 and x_2.

Polynomial Representation

Now we have an expression for the states and their derivatives at the interval midpoint, as represented by a cubic polynomial.

Polynomial Representation

Now we have an expression for the states and their derivatives at the interval midpoint, as represented by a cubic polynomial.

There is some error in this representation, and reducing this error is how we solve for the correct states.

Polynomial Defects

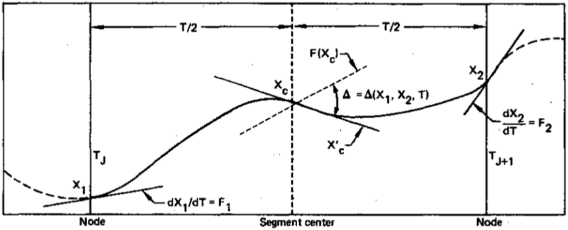
We now are going to add equality constraints to the optimization problem: the error in the polynomial representation of the states.

This error is described by "defects":

$$\Delta = f(x_c) - \dot{x}_c$$

Polynomial Defects

This defect is the difference between the derivative evaluated at the approximated midpoint state, and the differentiation of the approxiation.



Hargraves1987

NonLinear Programming

Nonlinear programming (NLP) is optimization of nonlinear objective and constraint functions.

NonLinear Programming

Nonlinear programming (NLP) is optimization of nonlinear objective and constraint functions.

This is typically done by linearizing the functions at a point, then taking steps in the appropriate directions.

NonLinear Programming

NLP is the most general case of local optimization.

NonLinear Programming

NLP is the most general case of local optimization.

All functions can be nonlinear, and the constraints can be inequality and/or inequality constraints, allowing bounds to be placed on variables.

Using NLP

Now it is clear what the previously defined defects will be used for: inputs into the NLP problem as equality constraints.

Using NLP

Now it is clear what the previously defined defects will be used for: inputs into the NLP problem as equality constraints.

These will be driven to 0 and this must be maintained, as the solver is free to explore different areas in the input vector space in an attempt to minimize the objective function.

Using NLP

I won't discuss too much mroe detail about using NLP for these problems; it is simply the solver which you give your objective and constraint functions to and it returns an optimal result (hopefully).

Simple Example

This is example is from vonStryk1993, which was from Bryson before that.

We have a double integrator with specified boundary conditions.

Simple Example's Dynamics

State space equations and boundaries:

$$\begin{array}{lll} \dot{x} = v & x(0) = 0 & x(1) = 0 \\ \dot{v} = u & v(0) = 1 & v(1) = -1 \\ \dot{w} = \frac{u^2}{2} & w(0) = 0 & \end{array}$$

Simple Example Cost and Constraint

Cost function:

$$min \quad w(1)$$

Additional constraints:

$$l - x(t) \geq 0$$

What is This System?

This system is analogous to a mass with velocity in one direction.

What is This System?

This system is analogous to a mass with velocity in one direction.

We want to switch the sign of that velocity within the time interval, and end at the starting position.

What is This System?

This system is analogous to a mass with velocity in one direction.

We want to switch the sign of that velocity within the time interval, and end at the starting position.

The mass isn't allowed to move further than l away from the starting position in the positive direction.

What is This System?

This system is analogous to a mass with velocity in one direction.

We want to switch the sign of that velocity within the time interval, and end at the starting position.

The mass isn't allowed to move further than l away from the starting position in the positive direction.

We want to minizie the integral of force over this time interval.

MATLAB Code

Objective Function:

```
function f= objfun(u)
    f = u(end);
```

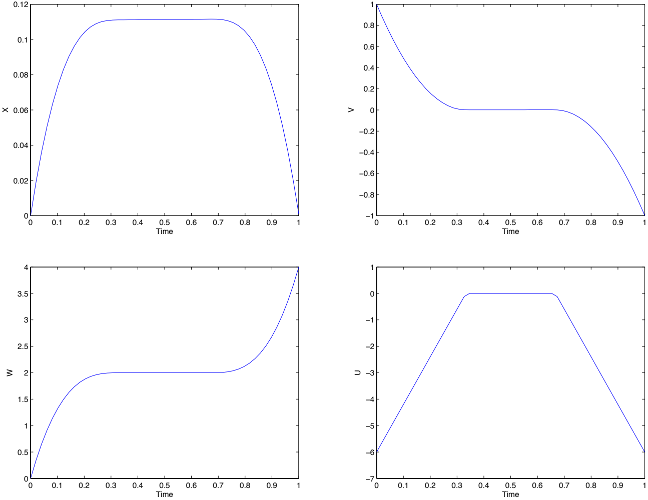
State Derivative Function:

```
function Y = innerFunc(T,X)
    U = interp1q(t',u',T');
    Y=[X(:,2), U, U.^2/2];
```

Constraints Function:

```
x = u(N+1:end);
u = u(1:N);
x = reshape(x,N,ST);

t = linspace(0,1,N);
dt = t(2) - t(1);
tc = linspace (t(1)+dt/2,t(end)-dt/2,N-1);

xdot = innerFunc(t,x);

xll = x(1:end-1,:);
xrr = x(2:end,:);
xdotll = xdot(1:end-1,:);
xdotrr = xdot(2:end,:);

xc = .5*(xll+xrr)+ dt/8*(xdotll-xdotrr);
xdotc = innerFunc(tc,xc);

ceq = (xdotc+3/2/dt*(xll-xrr)+1/4*(xdotll+xdotrr));
ceq = reshape(ceq,1,N*ST-ST);
ceq = [ceq x(1,:)-[0,1,0] x(end,1) x(end,2)+1];
c = (x(:,1)'-l);
```

Results



Conclusion

The Direct Collocation Method has considerable advantages over simpler “shooting methods” due to the additionaly constraints (even when adding more variables.

Conclusion

The Direct Collocation Method has considerable advantages over simpler “shooting methods” due to the additionaly constraints (even when adding more variables.

While it has shortcomings, it is very good at problems which do not encounter a lot of inequality constraints.

References

Hargraves, C. R., & Paris, S. W. (1987). Direct trajectory optimization using nonlinear programming and collocation. Journal of Guidance, Control, and Dynamics, 10(4), 338-342. doi: 10.2514/3.20223.

Von Stryk, O. (1993). Numerical solution of optimal control problems by direct collocation. International Series of Numerical Mathematics, 111(4), 1-13.

"Runge-Kutta Methods" Wikipedia, the Free Encyclopedia. 25 May 2011. `http://en.wikipedia.org/wiki/Runge-Kutta_methods`.