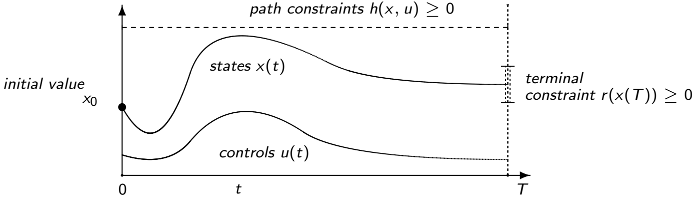
Direct Methods

Moritz Diehl

Overview

- Direct Single Shooting
- Direct Collocation
- Direct Multiple Shooting
- Structure Exploitation by Condensing
- Structure Exploitation by Riccati Recursion

Simplified Optimal Control Problem in ODE



$$\underset{x(\cdot),\, u(\cdot)}{\text{minimize}} \quad \int_0^T L(x(t), u(t))\, dt \; + \; E\,(x(T))$$

subject to

$$\begin{aligned}
x(0) - x_0 &= 0, & & \text{(fixed initial value)} \\
\dot{x}(t) - f(x(t), u(t)) &= 0, & t \in [0, T], & \text{(ODE model)} \\
h(x(t), u(t)) &\geq 0, & t \in [0, T], & \text{(path constraints)} \\
r\,(x(T)) &\geq 0 & & \text{(terminal constraints).}
\end{aligned}$$

Recall: Optimal Control Family Tree

Hamilton-Jacobi-Bellman Equation:
Tabulation in State Space

Indirect Methods, Pontryagin:
Solve Boundary Value Problem

Direct Methods:
Transform into Nonlinear Program (NLP)

Single Shooting:
Only discretized controls in NLP (sequential)

Collocation:
Discretized controls and states in NLP (simultaneous)

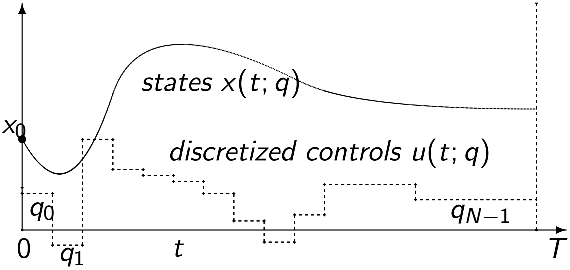
Multiple Shooting:
Controls and node start values in NLP (simultaneous/hybrid)

Direct Methods

- "First discretize, then optimize"
- Transcribe infinite problem into finite dimensional, **Nonlinear Programming Problem (NLP)**, and solve NLP.
- Pros and Cons:
  - + Can use state-of-the-art methods for NLP solution.
  - + Can treat inequality constraints and multipoint constraints much easier.
  - - Obtains only suboptimal/approximate solution.
- Nowadays most commonly used methods due to their easy applicability and robustness.

Direct Single Shooting [Hicks, Ray 1971; Sargent, Sullivan 1977]

Discretize controls $u(t)$ on fixed grid $0 = t_0 < t_1 < \ldots < t_N = T$, regard states $x(t)$ on $[0, T]$ as dependent variables.



Use numerical integration to obtain state as function $x(t; q)$ of finitely many control parameters $q = (q_0, q_1, \ldots, q_{N-1})$

NLP in Direct Single Shooting

After control discretization and numerical ODE solution, obtain NLP:

$$\underset{q}{\text{minimize}} \quad \int_0^T L(x(t;q), u(t;q))\,dt + E\left(x(T;q)\right)$$

subject to

$$h(x(t_i;q), u(t_i;q)) \geq 0, \quad i = 0, \ldots, N, \qquad \textit{(discretized path constraints)}$$

$$r\left(x(T;q)\right) \geq 0. \qquad \textit{(terminal constraints)}$$

Solve with finite dimensional optimization solver, e.g. Sequential Quadratic Programming (SQP).

Solution by Standard SQP

Summarize problem as

$$\min_{q} F(q) \quad \text{s.t.} \quad H(q) \geq 0.$$

Solve e.g. by Sequential Quadratic Programming (SQP), starting with guess q^0 for controls. $k := 0$

1. Evaluate $F(q^k), H(q^k)$ by ODE solution, and derivatives!
2. Compute correction Δq^k by solution of QP:

$$\min_{\Delta q} \nabla F(q_k)^T \Delta q + \frac{1}{2} \Delta q^T A^k \Delta q \quad \text{s.t.} \quad H(q^k) + \nabla H(q^k)^T \Delta q \geq 0.$$

3. Perform step $q^{k+1} = q^k + \alpha_k \Delta q^k$ with step length α_k determined by line search.

Hessian in Quadratic Subproblem

Matrix A^k in QP

$$\min_{\Delta q} \nabla F(q_k)^T \Delta q + \frac{1}{2} \Delta q^T A^k \Delta q \quad \text{s.t.} \quad H(q^k) + \nabla H(q^k)^T \Delta q \geq 0.$$

is called the Hessian matrix. Several variants exist:

- exact Hessian: $A^k = \nabla_q^2 \mathcal{L}(q, \mu)$ with μ the constraint multipliers. Delivers fast quadratic local convergence.
- Update Hessian using consecutive Lagrange gradients, e.g. by BFGS formula: superlinear
- In case of least squares objective $F(q) = \frac{1}{2} \|R(q)\|_2^2$ can also use Gauss-Newton Hessian (good linear convergence).

$$A^k = \left(\frac{\partial R}{\partial q}(q^k) \right)^T \frac{\partial R}{\partial q}(q^k)$$

Direct Single Shooting

- **Sequential** simulation and optimization.
- Pros and Cons
  - $+$ Can use state-of-the-art ODE/DAE solvers.
  - $+$ Few degrees of freedom even for large ODE/DAE systems.
  - $+$ Active set changes easily treated.
  - $+$ Need only initial guess for controls q.
  - \- Cannot use knowledge of x in initialization (e.g. in tracking problems).
  - \- ODE solution $x(t; q)$ can depend very nonlinearly on q.
  - \- Unstable systems difficult to treat.
- Often used in engineering applications e.g. in packages gOPT (PSE), DYOS (Marquardt), ...

Direct Collocation (Sketch) [Tsang et al. 1975]

- Discretize controls and states on **fine** grid with node values $s_i \approx x(t_i)$.
- Replace infinite ODE

$$0 = \dot{x}(t) - f(x(t), u(t)), \quad t \in [0, T]$$

by finitely many equality constraints

$$\begin{array}{rrcl} & c_i(q_i, s_i, s_{i+1}) & = & 0, \quad i = 0, \dots, N-1, \\ \text{e.g.} & c_i(q_i, s_i, s_{i+1}) & := & \frac{s_{i+1}-s_i}{t_{i+1}-t_i} - f\left(\frac{s_i+s_{i+1}}{2}, q_i\right) \end{array}$$

- Approximate also integrals, e.g.

$$\int_{t_i}^{t_{i+1}} L(x(t), u(t))dt \approx l_i(q_i, s_i, s_{i+1}) := L\left(\frac{s_i + s_{i+1}}{2}, q_i\right)(t_{i+1}-t_i)$$

NLP in Direct Collocation

After discretization obtain large scale, but sparse NLP:

$$\underset{s,\,q}{\text{minimize}} \quad \sum_{i=0}^{N-1} l_i(q_i, s_i, s_{i+1}) \; + \; E\,(s_N)$$

subject to

$$\begin{array}{rcll}
s_0 - x_0 & = & 0, & & \text{(fixed initial value)} \\
c_i(q_i, s_i, s_{i+1}) & = & 0, & i = 0, \ldots, N-1, & \text{(discretized ODE model)} \\
h(s_i, q_i) & \geq & 0, & i = 0, \ldots, N, & \text{(discretized path constrain} \\
r\,(s_N) & \geq & 0. & & \text{(terminal constraints)}
\end{array}$$

Solve e.g. with SQP method for sparse problems, or interior point methods (IPM).

What is a sparse NLP?

General NLP:

$$\min_{w} F(w) \quad \text{s.t.} \quad \begin{cases} G(w) & = & 0, \\ H(w) & \geq & 0. \end{cases}$$

is called sparse if the Jacobians (derivative matrices)

$$\nabla_w G^T = \frac{\partial G}{\partial w} = \left(\frac{\partial G}{\partial w_j} \right)_{ij} \quad \text{and} \quad \nabla_w H^T$$

contain many zero elements.

In SQP or IPM methods, this makes subproblems much cheaper to build and to solve.

Direct Collocation

- **Simultaneous** simulation and optimization.
- Pros and Cons:
  - \+ Large scale, but very sparse NLP.
  - \+ Can use knowledge of x in initialization.
  - \+ Can treat unstable systems well.
  - \+ Robust handling of path and terminal constraints.
  - \- Adaptivity needs new grid, changes NLP dimensions.
- Successfully used for practical optimal control e.g. by Biegler and Wächter (IPOPT), Betts, Bock/Schulz (OCPRSQP), v. Stryk (DIRCOL), ...

Direct Multiple Shooting [Bock and Plitt, 1981]

- Discretize controls piecewise on a coarse grid

$$u(t) = q_i \quad \text{for} \quad t \in [t_i, t_{i+1}]$$

- Solve ODE on each interval $[t_i, t_{i+1}]$ numerically, starting with artificial initial value s_i:

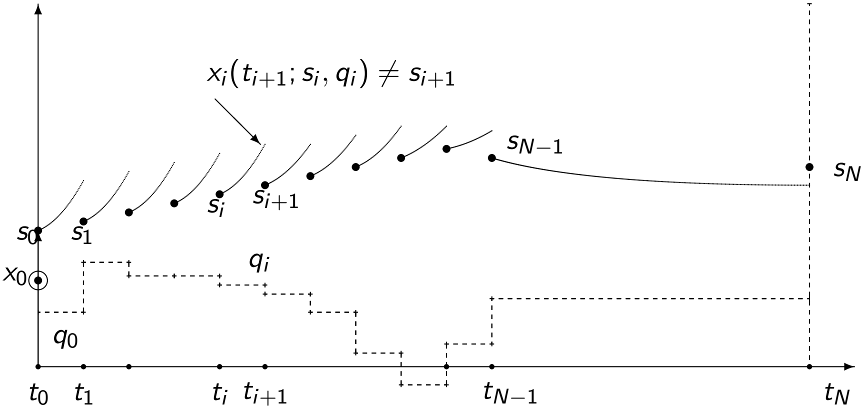
$$\begin{aligned} \dot{x}_i(t; s_i, q_i) &= f(x_i(t; s_i, q_i), q_i), \quad t \in [t_i, t_{i+1}], \\ x_i(t_i; s_i, q_i) &= s_i. \end{aligned}$$

  Obtain trajectory pieces $x_i(t; s_i, q_i)$.

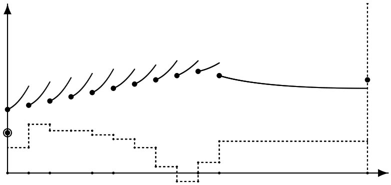
- Also numerically compute integrals

$$l_i(s_i, q_i) := \int_{t_i}^{t_{i+1}} L(x_i(t_i; s_i, q_i), q_i)dt$$

Sketch of Direct Multiple Shooting



NLP in Direct Multiple Shooting



$$\underset{s,q}{\text{minimize}} \quad \sum_{i=0}^{N-1} l_i(s_i, q_i) \; + \; E\left(s_N\right)$$

subject to

$$\begin{aligned}
s_0 - x_0 &= 0, && \text{(initial value)}\\
s_{i+1} - x_i(t_{i+1}; s_i, q_i) &= 0, \; i = 0, \ldots, N-1, && \text{(continuity)}\\
h(s_i, q_i) &\geq 0, \; i = 0, \ldots, N, && \text{(discretized path constraints}\\
r\left(s_N\right) &\geq 0. && \text{(terminal constraints)}
\end{aligned}$$

Structured NLP

- Summarize all variables as $w := (s_0, q_0, s_1, q_1, \ldots, s_N)$.
- Obtain structured NLP

$$\boxed{\min_{w} \; F(w) \quad \text{s.t.} \quad \begin{cases} G(w) &= 0 \\ H(w) &\geq 0. \end{cases}}$$

- Jacobian $\nabla G(w^k)^T$ contains dynamic model equations.
- Jacobians and Hessian of NLP are block sparse, can be exploited in numerical solution procedure.

QP = Discrete Time Problem

$$\min_{x,u} \sum_{i=0}^{N-1} \begin{bmatrix} 1 \\ \Delta s_i \\ \Delta q_i \end{bmatrix}^T \begin{bmatrix} 0 & q_i^T & s_i^T \\ q_i & Q_i & S_i^T \\ s_i & S_i & R_i \end{bmatrix} \begin{bmatrix} 1 \\ \Delta s_i \\ \Delta q_i \end{bmatrix} + \begin{bmatrix} 1 \\ \Delta s_N \end{bmatrix}^T \begin{bmatrix} 0 & p_N^T \\ p_N & P_N \end{bmatrix} \begin{bmatrix} 1 \\ \Delta s_N \end{bmatrix}$$

subject to

$$\begin{array}{rcllr} \Delta s_0 - x_0^{\mathrm{fix}} & = & 0, & & \text{(initial)} \\ \Delta s_{i+1} - A_i \Delta s_i - B_i \Delta q_i - c_i & = & 0, & i = 0, \ldots, N-1, & \text{(system)} \\ C_i \Delta s_i + D_i \Delta q_i - c_i & \leq & 0, & i = 0, \ldots, N-1, & \text{(path)} \\ C_N \Delta s_N - c_N & \leq & 0, & & \text{(terminal)} \end{array}$$

Interpretation of Continuity Conditions

- In direct multiple shooting, continuity conditions $s_{i+1} = x_i(t_{i+1}; s_i, q_i)$ represent discrete time dynamic system.
- *Linearized* reduced continuity conditions (used in *condensing* to eliminate $\Delta s_1, \ldots, \Delta s_N$) represent **linear discrete time system:**

$$\Delta s_{i+1} = (x_i(t_{i+1}; s_i, q_i) - s_{i+1}) + X_i \Delta s_i^x + Y_i \Delta q_i = 0,$$
$$i = 0, \ldots, N-1.$$

- If original system is linear, continuity is perfectly satisfied in all SQP iterations.
- Lagrange multipliers λ_i for the continuity conditions are approximation of **adjoint variables**. They indicate the costs of continuity.

Condensing Technique [Bock, Plitt, 1984]

As before in multiple shooting for BVPs, can use "condensing" of linear system equations

$$\begin{bmatrix} X_0 & Y_0 & -\mathbb{I} & & & & & \\ & & X_1 & Y_1 & -\mathbb{I} & & & \\ & & & & \ddots & & & \\ & & & & & \ddots & & \\ & & & & & X_{N-1} & Y_{N-1} & -\mathbb{I} \end{bmatrix} \begin{bmatrix} \Delta s_0 \\ \Delta q_0 \\ \Delta s_1 \\ \Delta q_1 \\ \Delta s_2 \\ \vdots \\ \Delta s_{N-1} \\ \Delta q_{N-1} \\ \Delta s_N \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix}$$

to eliminate $\Delta s_1, \dots, \Delta s_N$ from QP.
Results in *condensed QP* in variables Δs_0 and $\Delta q_0, \dots, \Delta q_N$ only.

Riccati Recursion

Alternative to condensing: can use Riccati recursion within QP solver addressing the full, uncondensed, but block sparse QP problem.

- Same algorithm as discrete time Riccati difference equation
- Linear effort in number N of shooting nodes, compared to $O(N^3)$ for condensed QP.
- Use Interiour Point Method to deal with inequalities, or Schur-Complement type reduction techniques.

Direct Multiple Shooting

- **Simultaneous** simulation and optimization.
- Pros and Cons
  - \+ uses **adaptive** ODE/DAE solvers
  - \+ but NLP has **fixed dimensions**
  - \+ can use knowledge of x in initialization (here bounds; more important in online context).
  - \+ can treat unstable systems well.
  - \+ robust handling of path and terminal constraints.
  - \+ easy to parallelize.
  - \- not as sparse as collocation.
- Used for practical optimal control e.g by Franke ("HQP"), Terwen (DaimlerChrysler); Santos and Biegler; Bock et al. ("MUSCOD-II")