ADAPTIVE MESH GENERATION METHODS
FOR PATH CONSTRAINED OPTIMAL CONTROL

TONI LASSILA

Abstract. The numerical solution of an optimal control problem with path constraints is studied in this work. Using an approximate solution of a NLP problem, several methods for identifying the boundary arcs on the optimal trajectory are obtained. By incorporating knowledge of the boundary arcs into the mesh selection, a more accurate solution can be obtained with fewer mesh points. Higher order discretization methods can then be used to improve accuracy further without excessive penalty from increasing the dimension of the problem.

1. Introduction

1.1. The path constrained optimal control problem. Given a dynamic system with state \( x(t) : \mathbb{R} \rightarrow \mathbb{R}^n \), the path constrained optimal control problem is to find a piecewise continuous input control \( u(t) : \mathbb{R} \rightarrow \mathbb{R}^m \) which solves the problem

\[
\begin{align*}
\min_u J(u) &= \Psi(x(t_f), t_f) \\
\text{on a fixed time interval } [t_0, t_f] \text{ subject to }
\end{align*}
\]

\[
\begin{align*}
\dot{x}(t) &= f(x(t), u(t), t), & \text{a.e.} \\
h(x(t)) &\leq 0, & \forall t \in [t_0, t_f] \\
B_0 x(t_0) + B_f x(t_f) &= \beta,
\end{align*}
\]

where \( f, h, \) and \( \Psi \) are sufficiently smooth functions of all the variables, and the boundary condition matrices \( B_0 \) and \( B_f \) are such that the matrix \([B_0, B_f]\) has rank \( n \). This is called the Mayer form of an optimal control problem and can be converted to equivalent Bolza or Lagrange forms.

1.2. Numerical solution of optimal control problems. In practice, the path constraint makes an analytic solution of the necessary conditions for an extremal difficult. Several numerical methods have been developed to solve optimal control problems approximately.

1.2.1. Direct methods. Perhaps the most popular class of methods are so called direct methods, where either the input control function or both the input control and state functions are discretized, and problem (1.1) is written as a finite-dimensional non-linear programming problem. The method used in this work will be the direct collocation method, in which both the state and control functions are discretized at the same mesh points. The state and input control functions are approximated with continuous piecewise polynomial functions. The approximate state function must fulfill the differential equation at a chosen finite set of so called collocation points. The remaining free parameters are nonlinear programming (NLP) variables.

An alternative direct method is based on the multiple-shooting method for solving numerical boundary value problems. For details about these methods and others, good sources include [3], [4], [6], [7], and [11].
1.2.2. Indirect methods. Consider first problem (1.1) without the path contraints. Using calculus of variations, it can be shown (see [9] for an introduction) that the necessary first-order conditions for an optimal trajectory $x^*$ and an optimal control $u^*$ are
\begin{equation}
\frac{\partial H}{\partial x} = -\dot{p}(t), \quad \frac{\partial H}{\partial p} = \dot{x}(t), \quad \frac{\partial H}{\partial u} = 0
\end{equation}
where the Hamiltonian of the problem is defined as the function
\begin{equation}
H(x(t), u(t), p(t), t) := \Psi(x(t_f), t_f) - t_0 + p^T(t) [ f(x(t), u(t), t) - \dot{x}(t) ]
\end{equation}
and the $p^*(t)$ on the optimal trajectory is called the costate trajectory, a function of bounded variation which vanishes only at isolated points and is piecewise absolutely continuous. This results in a two-point boundary value problem which have been extensively studied in literature (see for example [1] for numerical solution methods).

Path contraints can be introduced into the problem with help of penalty functions. Consider the case $h : \mathbb{R}^n \to \mathbb{R}$. Let $\lambda : \mathbb{R}^n \to \mathbb{R}^+$ be a continuously differentiable function such that
\begin{equation}
\lambda(x) := \begin{cases} 
\text{large,} & h(x) > 0 \\
\text{small,} & h(x) \leq 0
\end{cases}
\end{equation}
Define the augmented Hamiltonian as
\begin{equation}
H_a(x(t), u(t), p(t), t) := H(x(t), u(t), p(t), t) + M\lambda(x(t)), \quad M > 0.
\end{equation}
By writing again the first-order necessary conditions and solving this unconstrained boundary value problem with increasing values of $M$ we can get approximations for the solution of the constrained problem. For various reasons such an approach does not seem to be very popular at the moment, and direct methods are favored in practice. The most difficulties arise from constructing an initial guess for the costate variables $p(t)$, which do not usually have any physical significance to the problem.

1.3. Boundary arcs and choice of discretization mesh. Consider a scalar path constraint $h(x(t)) \leq 0$. The points $t \in \mathcal{B}(x^*)$ where $h(x^*(t)) = 0$ on the optimal trajectory $x^*$ are called boundary points. An interval $[t_{\text{entry}}, t_{\text{exit}}] \subset \mathcal{B}(x^*)$ is called a boundary arc if it is a closed component of $\mathcal{B}(x^*)$ with positive measure. The interval end points are called entry and exit points respectively. The set of all entry and exit points is called the set of junction points.

At the junction points we can only guarantee a certain level of smoothness for the optimal trajectory, for example that the $q$ first derivatives of $x(t)$ are continuous. We might even have $q = 0$ (the solution is merely continuous) for minimum time bang-bang problems where the optimal control input is discontinuous. It therefore makes sense to require that discretization mesh points lie exactly on the junction points in order to guarantee that the local truncation error for the chosen finite difference scheme is bounded from above.
1.4. Organisation of the work. This work is organized as follows. First, the implementation of the direct collocation method is briefly explained. Then, some results from literature for boundary arcs on the optimal trajectory are presented. Using these results several methods for identifying the junction points given an approximate numerical solution are presented. Then, adaptive mesh selection theory for boundary value ordinary differential equations (ODEs) from [1] is presented as applicable to numerical optimal control problems. Finally, the methods are combined to give an adaptive mesh selection algorithm for use with direct collocation. The methods are tested with two simple optimal control problems.

2. Direct Collocation Method

The idea of the direct collocation method in its simplest form is to choose a mesh of points
\begin{equation}
\pi: \ t_0 = t_1 < t_2 < \ldots < t_N = t_f.
\end{equation}

For every interval \([t_i, t_{i+1}]\) choose a set of collocation points
\begin{equation}
0 \leq \rho_1 < \rho_2 < \ldots < \rho_p \leq 1
\end{equation}

where the collocation points on each subinterval \([t_i, t_{i+1}]\) are then
\begin{equation}
\tau_{ij} = t_i + h_i \rho_j, \quad j = 1, \ldots, p
\end{equation}

where \(h_i := t_{i+1} - t_i\) is the length of the subinterval.

The basic strategy of collocation is to approximate the solution of problem (1.1) using a continuous piecewise polynomial approximation \(x_\pi\) for the state \(x\), and a piecewise linear approximation \(u_\pi\) for the input control \(u\). The state equation must be fulfilled by our approximate solution \(x_\pi\), so we get a set of conditions for all \(i = 1, \ldots, N, j = 1, \ldots, p\):

1. \(\dot{x}_\pi(\tau_{ij}) = f(x_\pi(\tau_{ij}), u_\pi(\tau_{ij}), \tau_{ij})\) (first derivatives at collocation points);
2. \(h(\pi_i(t_i)) \leq 0\) (path constraint at mesh points);
3. Boundary conditions.

The simplest choice is to let the collocation points be the subinterval endpoints
\begin{equation}
\rho_1 = 0, \quad \rho_2 = 1.
\end{equation}

This is called a (third order) Lobatto scheme. The values of \(x_\pi\) and \(u_\pi\) at the meshpoints, \(x_\pi(t_i)\) and \(u_\pi(t_i)\), are taken to be free and act as the optimization variables. With this choice of collocation points and given the values of \(x_\pi(t_i)\) and \(u_\pi(t_i)\), \(x_\pi\) is the unique interpolating piecewise cubic polynomial
\begin{equation}
x_\pi(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3, \quad t \in [t_i, t_{i+1}],
\end{equation}

where, denoting by \(f_i\) the value of \(f(x_\pi(t_i), u(t_i), t_i)\), the coefficients are [12]
\begin{align}
c_0 & = x_\pi(t_i); \\
c_1 & = h_i f_i; \\
c_2 & = -3x_\pi(t_i) - 2h_i f_i + 3x_\pi(t_{i+1}) - h_i f_{i+1}; \\
c_3 & = 2x_\pi(t_i) + h_i f_i - 2x_\pi(t_{i+1}) + h_i f_{i+1}.
\end{align}

Problem (1.1) is then approximated with a NLP problem with nonlinear constraints represented by the collocation conditions on each subinterval, plus the boundary conditions and path constraints.
For notational purposes we compile the discretized state and control input into one vector:

\[(2.10)\]
\[y^i = [x^i_1 \ldots x^i_n u^i_1 \ldots u^i_m]^T\]

\[(2.11)\]
\[y^T = [y^1]^T \ldots [y^N]^T].\]

2.1. Linear problems. If the system equation is of linear form

\[(2.12)\]
\[\dot{y} = A(t)y + q(t)\]

with \(m\) boundary conditions

\[(2.13)\]
\[B_0y^1 + B_f y^N = \beta,\]

then using the trapezoid rule at each interval \([t_i, t_{i+1}]\) we have [1]:

\[(2.14)\]
\[S_i = -h_i^{-1}I - \frac{1}{2}A(t_i), \quad R_i = h_i^{-1}I - \frac{1}{2}A(t_{i+1})\]

\[(2.15)\]
\[q_i = \frac{1}{2}[q(t_{i+1}) + q(t_i)],\]

from which the conditions at the collocation points can be combined together with the boundary conditions:

\[
\begin{bmatrix}
S_1 & R_1 & & & \\
& S_2 & R_2 & & \\
& & \ddots & \ddots & \\
& & & S_{N-1} & R_{N-1} \\
B_0 & & & & B_f
\end{bmatrix}
\begin{bmatrix}
y^1 \\
y^2 \\
\vdots \\
y^{N-1} \\
y^N
\end{bmatrix}
= \begin{bmatrix}
q^1 \\
q^2 \\
\vdots \\
q^{N-1} \\
\beta
\end{bmatrix}.
\]

We denote this linear equation as \(Ty = q\). Note that the matrix \(T\) is of size \([nN + m] \times nN\) and very sparse. Now we can write the optimization problem with a linear cost function, linear constraints, and a possibly nonlinear inequality constraint:

\[(2.16)\]
\[\min_{y} J(y) = \Psi(y^N)\]

subject to

\[
\begin{cases}
Ty = q \\
h(x_i) \leq 0, \quad i = 1, \ldots, N.
\end{cases}
\]

2.2. Nonlinear problems. For nonlinear systems the solution procedure is more general. The nonlinear system equation is

\[(2.17)\]
\[\dot{y} = f(y, t)\]

with \(m\) nonlinear boundary conditions

\[(2.18)\]
\[g(y^1, y^N) = 0\]

and \(k\) inequality constraints at the mesh points

\[(2.19)\]
\[h(x_i) \leq 0, \quad i = 1, \ldots, N.\]

The collocation conditions are written as the residual error of a given difference approximation for the derivative \(\dot{x}\) on each interval \([t_i, t_{i+1}]\):

\[(2.20)\]
\[\mathcal{N}(y^i) = x^{i+1} - x^i - h_i \sum_{j=1}^q \beta_j f_{ij} = 0, \quad i = 1, \ldots, N - 1\]

where

\[(2.21)\]
\[f_{ij} = f(y_i + h_i \sum_{l=1}^q \alpha_{jl} f_{il}, t_i + h_i \rho_j),\]
where \( \alpha, \rho, \) and \( \beta \) are the weights. If \( \rho_j = 0 \) and \( \alpha_{jl} = 0 \) when \( j \leq l \), then the method is explicit, otherwise implicit. When the method is implicit, the local unknowns \( f_{ij} \) in the collocation equations require either:

1. The solution of nonlinear equation (2.21) to determine the values \( f_{ij} \) at every evaluation of the collocation constraints (see [4], p.69)
2. Introduction of the local unknowns \( f_{ij} \) as extra variables into the optimization problem (1.1).

Both methods increase the computational effort of the optimization iteration. However, if there is only one collocation point on the interval \( (0, 1) \) it’s possible to write an explicit formula for the method. This allows the third order implicit Lobatto method:

\[
x_{i+1} - x_i = \frac{h_i}{6} \left[ f(y_i, t_i) + 4\tilde{f} + f(y_{i+1}, t_{i+1}) \right]
\]

where

\[
\tilde{f} := f\left(\frac{1}{2}(y_i + y_{i+1}) + \frac{h_i}{8}(f(y_i, t_i) - f(y_{i+1}, t_{i+1})), t + h_i/2\right).
\]

The NLP problem is then:

\[
\min_u \Psi(y^N)
\]

subject to

\[
\begin{cases}
\mathcal{N}(y^i) = 0, & i = 1, \ldots, N \\
B_0 y_1 + B_f y^N = 0 \\
h(x^i) \leq 0
\end{cases}
\]

where the boundary conditions have been linearized:

\[
B_0 := \frac{\partial g(y^1, y^N)}{\partial y^1}, \quad B_f := \frac{\partial g(y^1, y^N)}{\partial y^N}.
\]

Efficient solvers for such problems have been developed using the method of sequential quadratic programming (SQP), and will not be elaborated further in this work. See [2] for an introduction to SQP.

3. Boundary arcs

To begin, a few results from [8] and [10]. Consider problem (1.1) with a scalar input control \( u(t) \). Denote again by \( U \) the set of feasible input controls.

**Theorem 3.1** (Jacobson-Lele-Speyer). Assume first the technical conditions that follow:

1. \( u \in U, \quad U := \{u(t) : u(t) \text{ is piecewise continuous and bounded}\};
2. \( h(x(t)) \) is \((p + 1)\)-times continuously differentiable with respect to \( x \);
3. Problem (1.1) has an optimal solution with finite cost;
4. Along a boundary arc \((h = 0)\) the control that maintains \( \frac{d}{dt}h(x) = 0 \) is \( p \)-times continuously differentiable with respect to time;
5. Along the optimal solution \( \frac{d}{dt} \left( \frac{d}{dt}h(x) \right) \neq 0, \quad \forall t \in [t_0, t_f] \).

Assume further that problem (1.1) has an optimal trajectory with a single boundary arc \([t_{entry}, t_{exit}]\). For the adjoined cost functional

\[
J = \Psi(x(t_f), t_f) + \int_{t_0}^{t_f} p^T(t)(f - \dot{x})dt + \int_{t_0}^{t_f} h(x)d\eta(t)
\]

where \( \eta(t) \) is a function of bounded variation, at the junction points we have the conditions:

\[
p(t_{entry}^-) - p(t_{entry}^+) = \eta(t_{entry}^+) \nabla_x h(t_{entry});
\]
\( p(t^-_{\text{exit}}) - p(t^+_{\text{exit}}) = \eta(t^-_{\text{exit}}) \nabla_x h(t_{\text{exit}}), \)

where \( p(t^+) \) and \( p(t^-) \) mean the right- and left-sided limits of \( p(t) \).

**Theorem 3.2** (Maurer). Let \( u^* \in \mathcal{U} \) be the global optimal control for the problem fulfilling the technical assumptions of Theorem 3.1 with the Hamiltonian

\[
H(x, u, p, t) := \frac{\Psi(x(t_f), t_f)}{t_f - t_0} + p^T f(x, u, t)
\]

and the Lagrangian

\[
L(u, \eta, t) := H(x, u, p, t) + \eta^T h(x)
\]

where \( \eta(t) \) is the Lagrange multiplier related to the inequality constraint \( h(x(t)) \leq 0 \) and \( p(t) \) is the costate. If:

1. The Hamiltonian is uniformly strongly convex, i.e.:
   \[
   \exists \gamma > 0 : H_{uu}(u(t), x(t), p(t)) \geq \gamma, \quad \forall u \in \mathcal{U}, \quad \forall t \in [t_0, t_f];
   \]
2. The constraint \( h(x) \) is of order \( q \), i.e. we find the smallest \( q \geq 1 \) s.t.:
   \[
   \frac{\partial}{\partial u} \frac{dn}{dt} h(x) \neq 0, \quad \forall t \in [t_0, t_f], \quad \forall u \in \mathcal{U};
   \]
3. The set of junction points is finite and \( h(x^*(t_f)) < 0 \),

then it holds that:

1. \( u^* \) is everywhere continuous.
2. \( \eta \) and \( u^* \) are \( C^1 \) smooth outside the junction points. Also, \( u^* \) has \( q - 2 \) continuous time derivatives at the junction points.
3. \( \eta \) vanishes outside the boundary arcs.

Note that requirement 1 is very strong and often does not hold in practical problems. Especially minimum-time problems with bang-bang type solutions have optimal trajectories which are not everywhere differentiable and optimal control inputs which contain jumps. On the other hand, we can apply regularization to the optimality measure that attempts to fulfill this condition. For example, consider the controlwise linear optimal control problem \( \dot{x} = f(x) + bu \) with the performance measure

\[
J(u) = \frac{1}{2} \int_{t_0}^{t_f} x u^2 \, dt
\]

and the state constraint \( |x(t)| \leq 1 \). The regularized performance measure

\[
\tilde{J}(u) = \frac{1}{2} \int_{t_0}^{t_f} \left\{ x u^2 + (1 + \varepsilon) u^2 \right\} dt
\]

fulfills the condition \( H_{uu} = x + (1 + \varepsilon) \geq \varepsilon \); 0.

When the NLP problem (2.24) is solved using SQP, in addition to the solution, estimates of the Lagrange multipliers \( \eta_i \) related to the path inequality constraint at each mesh point are obtained. Is there a connection between the discrete Lagrange multipliers \( \eta_i \) and the piecewise smooth Lagrange multiplier function \( \eta \) seen in (3.1)?

In [12] the problem (1.1) is discretized using a uniform mesh and the Lobatto-3 rule. Among the first order necessary conditions (1.2) for the problem are

\[
\dot{p}(t) = -p(t)^T \frac{\partial f(x(t), u(t), t)}{\partial x} - \eta(t)^T \frac{\partial h(x(t))}{\partial x}.
\]
The first order necessary Karush-Kuhn-Tucker conditions for the discrete NLP problem include

\[
\frac{3}{2} \lambda_i = -\frac{3}{2} \lambda_i \frac{\partial f(x(t_i), u(t_i), t_i)}{\partial x} - \eta_i \frac{\partial h(x(t_i))}{\partial x}
\]

for all \( i = 1, \ldots, N \). This demonstrates that \( \eta(t_i) \) and \( \eta_i \) differ only by a constant factor dependant on the discretization method.

**Proposition 3.1.** For a uniform mesh \( \pi \) with \( N \) points, the Lobatto-3 collocation solution of problem (2.24) has pointwise Lagrange multipliers \( \lambda_i \) and \( \eta_i \) for the collocation and state inequality constraint \( h(x(t)) \leq 0 \) respectively such that:

\[
p(t_i) = \frac{3}{2} c(N) \lambda_i, \quad \eta(t_i) = c(N) \eta_i,
\]

where \( p(t) \) is the costate and \( \eta(t) \) is a function of bounded variation.

### 3.1. Sub-arc optimization estimator

For notational simplicity, assume that problem (1.1) has been converted to Lagrange form

\[
\min_u J(u) = \int_{t_0}^{t_f} g(x(t), u(t), t) dt
\]

and that we are working with a scalar path constraint \( h(x(t)) \leq 0 \). Assume that the problem has been solved approximately and the solution \( y^*_{\pi} \) has been obtained. Then looking at the sub-problem defined only on the interval \([t_i, t_{i+1}]\) containing exactly one junction point, say \( t_{\text{entry}}\):

\[
\min_u \tilde{J}(u) = \int_{t_i}^{t_{i+1}} g(x(t), u(t), t) dt
\]

subject to

\[
\begin{align*}
\dot{x}(t) &= f(x(t), u(t), t) \\
x(t_i) &= x^*_\pi^i \\
x(t_{i+1}) &= x^*_\pi^{i+1} \\
h(x(t)) &\leq 0, \quad h(x(t_{i+1})) = 0
\end{align*}
\]

where \( x_\pi \) is defined in (2.5), we can hope that if \( x_\pi \) is close to the optimal trajectory \( x^* \), then so are \( x^{\pi}_i \) and \( x^{\pi+1}_i \) and the solution to our sub-problem will also be close to the optimal trajectory on \([t_i, t_{i+1}]\).

Choose a point \( \tau_k \in (t_i, t_{i+1}) \) and discretize the problem (3.8) using the three-point mesh \([t_i, \tau_k, t_{i+1}]\). Solving the problem will produce a sub-arc \( x^{\star}_{\text{sub}} \) for the optimal trajectory.

Denote the value of the constraint at \( \tau_k \), \( h(x^{\star}_{\text{sub}}(\tau_k)) =: \xi_k \leq 0 \). We attempt to find the entry point \( t_{\text{entry}} \) using a combination of the secant and bisection methods applied to \( h \):

1. Let \( \alpha_1 = t_i, \beta_1 = t_{i+1} \) and choose \( \tau_1 = \frac{\alpha_1 + \beta_1}{2} \) and set \( \xi_0 := h(x^*_i) \).

2. For each \( k \): Compute \( \xi_k(\tau_k) \) by solving the local NLP problem on the 3-point mesh. If \( \xi_k = 0 \), i.e. the constraint becomes active then let:

\[
\alpha_{k+1} := \alpha_k, \quad \beta_{k+1} := \tau_k, \quad \tau_{k+1} := \frac{\alpha_{k+1} + \beta_{k+1}}{2}.
\]

If \( \xi_k < 0 \) then let:

\[
\alpha_{k+1} := \tau_k, \quad \beta_{k+1} := \beta_k, \quad \tau_{k+1} := \tau_k + \xi_k \frac{\alpha_{k} - \beta_{k}}{\xi_k - \xi_{k-1}},
\]
provided that $\tau_{k+1} \in [\alpha_{k+1}, \beta_{k+1}]$. Otherwise fall back to the bisection step:

$$\alpha_{k+1} := \tau_k, \quad \beta_{k+1} := \beta_k, \quad \tau_{k+1} := \frac{\alpha_{k+1} + \beta_{k+1}}{2}.$$

(3) Increment $k$ until the stopping criteria:

$$0 < |\alpha_k - \beta_k| < \varepsilon$$

is fulfilled for some $\varepsilon > 0$, or until sufficiently many steps have been taken.

The method is illustrated in Figure 3.1. A similar idea has been presented in [5].

### 3.2. Secant extrapolation estimator

If the strong convexity condition (1) from Theorem 3.2 holds and the constraint $h(x(t))$ is of order 1, we can assume that the Lagrange multiplier $\eta$ is everywhere continuous and differentiable on the boundary arc. Assume that there exists a boundary arc and a triplet of mesh point such that:

$$t_{\text{entry}} \in (t_{i-1}, t_i), \quad \eta_{i+1} > \eta_i > 0.$$

Attempt to extrapolate with a secant line drawn through $t_i$ and $t_{i+1}$. Denote by $h_i := t_{i+1} - t_i$ and choose the estimator:

$$\hat{t}_{\text{entry}} := t_i - h_i \frac{\eta(t_i)}{\eta(t_{i+1}) - \eta(t_i)}$$

provided that $t_{i-1} < \hat{t}_{\text{entry}} < t_i$. Similarly for the exit point:

$$\hat{t}_{\text{exit}} := t_i + h_{i-1} \frac{\eta(t_i)}{\eta(t_{i-1}) - \eta(t_i)}$$
provided that $t_i < \tilde{t}_{exit} < t_{i+1}$. This method is not very useful due to the fact that the Lagrange multiplier function $\eta(t)$ usually has jumps at the junction points, but it serves as an introduction to the next method.

3.3. Jump condition estimator. In case the Lagrange multiplier function $\eta(t)$ is not continuous at the junction points, we can use the jump conditions from Theorem 3.1. Using the costate at $p(t_{i-1})$ and $p(t_i)$ as estimators for the limits of the costate variables at $t_{entry}$:

$$p(t_{i-1}) \approx \lim_{\delta \to 0^-} p(t_{entry} + \delta) = p(t^-_{entry});$$

$$p(t_i) \approx \lim_{\delta \to 0^+} p(t_{entry} + \delta) = p(t^+_{entry}),$$

we have the approximate relation for the jump at $\eta(t_{entry})$:

$$\eta(t^+_{entry}) \nabla_x h(t_{entry}) \approx p(t_{i-1}) - p(t_i).$$

(3.12)

If $h$ is linear with respect to $x$, then $\nabla_x h$ is constant, and the equation is simple to solve for $\eta(t^+_{entry})$. Once the magnitude of the jump has been established, we proceed with a similar secant extrapolation method as in the previous case. Using Proposition 3.1 and the previous results, the equation for the entry point estimator takes the form:

$$\tilde{t}_{entry} = t_i + h_i \frac{\eta(t^+_{entry}) - \eta(t_i)}{\eta(t_{i+1}) - \eta(t_i)}$$

$$\approx t_i + h_i \frac{(\nabla_x h)^{-T}[p(t_{i-1}) - p(t_i)] - c(N)\eta_i}{c(N)[\eta_{i+1} - \eta_i]}$$

$$= t_i + h_i \frac{(3/2)c(N)(\nabla_x h)^{-T}[\lambda_{i-1} - \lambda_i] - c(N)\eta_i}{c(N)[\eta_{i+1} - \eta_i]}$$

$$= t_i + \frac{3}{2} h_i \frac{(\nabla_x h)^{-T}[\lambda_{i-1} - \lambda_i] - \eta_i}{\eta_{i+1} - \eta_i}$$

where $(\nabla_x h)^{-T} := [ (\nabla_x h)^{-1}_1 \ldots (\nabla_x h)^{-1}_m ]$. Similarly:

$$\tilde{t}_{exit} = t_i + h_i \frac{\eta(t^-_{exit}) - \eta(t_i)}{\eta(t_i) - \eta(t_{i-1})} \approx t_i + \frac{3}{2} h_i \frac{(\nabla_x h)^{-T}[\lambda_{i-1} - \lambda_i] - \eta_i}{[\eta_i - \eta_{i-1}]}.$$

The method is illustrated in Figure 3.2.

3.4. Local shooting estimator. Assume that at point $t_i$ on the optimal trajectory the state constraint is not yet active, but becomes active between $[t_i, t_{i+1}]$. Let $u_i = u(t_i)$ and integrate numerically the state equation:

$$\dot{x}(t) = f(x(t), u_i, t), \quad t \in [t_i, t_{i+1}]$$

(3.13)

with the initial value $x(t_i) = x^*(t_i)$ from the best numerical approximation available to obtain a local shooting solution $x_{LS}$. Starting from the point $t_i$, find the point $\tau$ where the state constraint $h$ becomes active, i.e. where:

$$h(x_{LS}(\tau), u_i) = 0.$$

(3.14)

This gives an estimate for the entry point $t_{entry}$. To find the exit point $t_{exit}$, look at the first mesh point where the state constraint has become inactive, and then integrate backwards in similar fashion.
3.5. **Boundary arc completion.** Assume that the optimal trajectory contains boundary arcs $I_1, \ldots, I_r$ that have been identified using some of the methods presented above. This information can be used to accelerate convergence of the optimization problem by reducing the set of feasible states. On each interval we have the additional condition:

\[(3.15) \quad h(x^*(t)) = 0, \quad \forall \, t \in I_j, j = 1, \ldots, r.\]

For all points $t_i$ that lie inside the boundary arcs we write a new constraint in (2.24):

\[h(x(t_i)) = 0, \quad \forall \, t_i : t_i \text{ is on a boundary arc}.\]

4. **Mesh Selection Issues**

4.1. **Error equidistribution.** From the theory of finite difference methods for the solution of boundary value ODEs, we know the importance of the selection of mesh points. The same principles apply to discretized optimal control problems. In particular, we attempt to measure the discretization error caused by replacing $x$ with an approximant $x_\pi$ using a *monitor function* $\varphi$ that measures the error on each interval $[t_i, t_{i+1}]$. We attempt to find a mesh $\pi$ such that the error on each interval is approximately the same. A mesh $\pi$ is called *asymptotically equidistributing* (abbreviated as eq.) if

\[(4.1) \quad \int_{t_i}^{t_{i+1}} \varphi(t, x(t)) dt = \lambda(1 + O(h_i)), \quad i = 1, \ldots, N - 1.\]

where the total error is

\[(4.2) \quad \|x - x_\pi\| = \int_{t_0}^{t_f} \varphi(t, x(t)) dt = N\lambda.\]

This definition is motivated as follows. Assume that we have an estimate for the collocation error at each interval given as a linear function of the interval length $h$:

\[(4.3) \quad \delta_i := \phi_i h_i, \quad i = 1, \ldots, N - 1.\]
where $\phi_i$ is independent of $h_i$. We then minimize the maximum error over all intervals as a function of the mesh points:

$$
\min_{\pi} \max_i |\delta_i|
$$

subject to

$$\sum_{i=1}^{N} h_i = t_f - t_0.
$$

The solution of this minimax problem is simple. By choosing

$$
\delta_i = (t_f - t_0) / \sum_{i=1}^{N} \phi_i^{-1} \text{ (constant)}
$$

for all $i$ we get an equidistributing mesh and this is the optimal choice.

4.1.1. Constructing an as.eg. mesh. The algorithm presented here is from [1]. Assume that a chosen difference method of order $s$ has local truncation error

$$
\epsilon_i(x) = h_i^s T(t_i) + O(h_i^r), \quad r > s.
$$

Then our monitor function $\phi(t,x(t))$ can be chosen to be

$$
\phi(t,x(t)) := |T(t)|^{1/s}
$$

where $T(t)$ contains the $(s+1)$st derivative of $x(t)$.

For example, using the third order Lobatto method (2.22) we choose a monitor function

$$
\phi(t,x(t)) := C |x^{(4)}(t)|^{1/3}.
$$

Since the actual solution $x(t)$ is not available, we replace it with the solution $x_{\pi}$, interpolated with cubic splines between the mesh points. The cubic splines will of course have

$$
x_{\pi}^{(4)}(t) \equiv 0,
$$

so instead we approximate linearly:

$$
x_{\pi}^{(3)}(t_i) = \nu(t_i), \quad \forall i = 1, \ldots, N
$$

where $\nu(t)$ continuous, piecewise linear on every subinterval $[t_i, t_{i+1}]$, and we get the piecewise constant monitor function

$$
\phi(t,x(t)) = |\dot{\nu}(t)|^{1/3}.
$$

Since $\phi(t,x(t))$ is piecewise constant, it is easy to find a mesh $\pi$ with $N$ points that fulfills (4.1). Another useful condition to impose on the mesh is that it is quasiquiform:

**Definition 4.1.** A mesh $\pi = \{t_1, \ldots, t_N\}$ is said to be quasiquiform if there exists a constant $K \geq 1$ such that:

$$
K^{-1} \left( \frac{t_f - t_0}{N} \right) \leq h_i \leq K \left( \frac{t_f - t_0}{N} \right), \quad i = 1, \ldots, N - 1.
$$

The optimal choice of $K$ depends on the problem in question.
4.2. Adaptive mesh algorithm. With all the pieces together, we can formulate an algorithm for constructing an adaptive mesh for the direct collocation method:

1. Form the NLP problem (2.24) using a uniform mesh.
2. Solve the NLP problem with a few iterations until sufficient (low) accuracy is obtained.
3. Find the junction points \( \tau_1, \ldots, \tau_l \) using one of the methods described.
4. Divide the interval \([t_0, t_f]\) into interior and boundary arcs:
   \[
   I_1 = [t_0, \tau_1], \quad I_2 = [\tau_1, \tau_2], \quad \ldots, \quad I_{k+1} = [\tau_k, t_f].
   \]
5. Compute an estimate of the local error \( \varepsilon_j \) incurred on each arc \( I_j \) using (4.11).
6. Decide the total number of subintervals to be used, \( N \). For each arc \( I_j \) the number of mesh points is:
   \[
   \#p_j = 2 + \lceil (N - k + 1) \cdot \varepsilon_j / \sum_{j=1}^{k+1} \varepsilon_j \rceil.
   \]
7. Form an as.eq. mesh for each arc \( I_j \) using the proportionally assigned number of mesh points \( \#p_j \). For each arc enforce the quasiuniformity condition:
   \[
   \exists K \geq 1 : K^{-1} (\frac{\tau_{i+1} - \tau_i}{\#p_j - 1}) \leq h_i \leq K (\frac{\tau_{i+1} - \tau_i}{\#p_j - 1}).
   \]
8. Form the new NLP problem on the adaptive mesh.
9. (optional) Add boundary arc completion conditions:
   \[
   h(x(t_i)) = 0, \quad \forall t_i : t_i \text{ is on a boundary arc}.
   \]
10. Solve the problem until convergence is achieved.

A reference implementation called MATCOL was written using the Matlab scientific computing environment and is available for download at
http://www.math.tkk.fi/~tlassila/
The comparison of methods presented in this work was done using MATCOL. The NLP solver used was Matlab’s fmincon.

5. Comparison of methods

Two different problems were used to test the methods presented. First, the linear optimal control problem studied by Bryson, Denham, and Dreyfus:

\[
(LP) \quad \min J(u) = \frac{1}{2} \int_0^1 u(t)^2 dt
\]
subject to
\[
\begin{align*}
   x'_1(t) &= x_2(t), & x_1(0) &= 0, & x_1(1) &= 0 \\
   x'_2(t) &= u(t), & x_2(0) &= 1, & x_2(1) &= -1 \\
   x_1(t) &\leq 1/9.
\end{align*}
\]
The state inequality constraint is of second order.

The other test problem was the Rayleigh optimal control problem:

\[
(NLP) \quad \min J(u) = \int_0^{2.5} u(t)^2 + x_1(t)^2 dt
\]
subject to
\[
\begin{align*}
   x'_1(t) &= x_2(t), & x_1(0) &= -5 \\
   x'_2(t) &= -x_1(t) + x_2(t)[c - \gamma x_2(t)^2] + ku(t), & x_2(0) &= -5 \\
   x_2(t) &\leq 3.5.
\end{align*}
\]
with $c = 1.4$, $\gamma = 0.14$, and $k = 4$. Here, the state inequality constraint is of first order. Note that both problems fulfill the assumptions of Theorem 3.2 and therefore the optimal control $u^*$ is everywhere continuous in both cases.

In both cases the state inequality constraint was chosen tight enough so that a proper boundary arc was formed on the optimal trajectory. The problems were first solved with an increasingly fine sequence of uniform meshes to confirm the convergence as well as obtain a reference solution $x_{ref}$ for the optimal trajectory.

5.1. Junction point identification methods. The junction point finding methods were tested first. Junction points for problem (NLP) were identified from the reference solution as $\{0.5051, 1.2374\}$. The three methods tested were:

(1) Local shooting estimator (LSE)
(2) Sub-arc optimization estimator (SAOE)
(3) Jump condition estimator (JCE)

Each method was initialized with an approximate solution obtained using a uniform mesh of $N$ point. Table 5.1 lists the absolute error

$$|\Delta t_{entry}| = |t_{entry,ref} - t_{entry,colloc}(N)|, \quad |\Delta t_{exit}| = |t_{exit,ref} - t_{exit,colloc}(N)|,$$

where $t_{entry,colloc}(N)$ and $t_{exit,colloc}(N)$ refer to the entry and exit points found using the particular junction point identification method using a mesh of $N$ sub-intervals. When noted as ###, the method failed to give a reasonable estimate.

| N   | LSE $|\Delta t_{entry}|$ | LSE $|\Delta t_{exit}|$ | SAE $|\Delta t_{entry}|$ | SAE $|\Delta t_{exit}|$ | JCE $|\Delta t_{entry}|$ | JCE $|\Delta t_{exit}|$ |
|-----|-------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|
| 10  | 0.0940                  | 0.2563                 | 0.0234                 |                        | 0.0497                 | ####                   |
| 15  | 0.0718                  | 0.0918                 | 0.0252                 | 0.0836                 | 0.0287                 | 0.0469                 |
| 20  | 0.0551                  | 0.1345                 | 0.0221                 | 0.0122                 | 0.0157                 | 0.0223                 |
| 25  | 0.0451                  | 0.0601                 | 0.0195                 | 0.0629                 | 0.0117                 | ####                   |
| 30  | 0.0384                  | 0.0938                 | 0.0174                 | 0.0240                 | 0.0100                 | ####                   |

5.2. Performance using the adaptive mesh. Two meshes were tested for both problems. First, a uniform mesh of $N$ points. Second, using the approximate solution obtained with the uniform mesh and the algorithm for adaptive mesh selection detailed earlier, a new mesh was formed that also had $N$ points. Junction points were identified with the sub-arc optimization method. Both meshes were compared for:

(1) Number of SQP iterations taken;
(2) Error against the reference solution, $||x_{π} - x_{ref}||_1$.

A stopping tolerance of $10^{-6}$ was used for both the target function as well as the optimization variables.

5.3. Indirect solution using a penalty function. To compare the results obtained using the direct collocation method, the problem (LP) was also solved indirectly by writing the necessary first-order optimality conditions obtained using
Table 5.2. Comparison of meshes for problem (LP)

<table>
<thead>
<tr>
<th>N</th>
<th>Uniform SQP iters</th>
<th>Uniform Error</th>
<th>Adaptive SQP iters</th>
<th>Adaptive Error</th>
<th>Adaptive (with BAC) SQP iters</th>
<th>Adaptive (with BAC) Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>0.30734</td>
<td>6</td>
<td>0.22851</td>
<td>3</td>
<td>0.19652</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>0.18025</td>
<td>19</td>
<td>0.12062</td>
<td>8</td>
<td>0.15437</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>0.11336</td>
<td>21</td>
<td>0.14585</td>
<td>12</td>
<td>0.12497</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
<td>0.082786</td>
<td>25</td>
<td>0.060324</td>
<td>20</td>
<td>0.069808</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>0.052504</td>
<td>31</td>
<td>0.057758</td>
<td>22</td>
<td>0.075685</td>
</tr>
</tbody>
</table>

Table 5.3. Comparison of meshes for problem (NLP)

<table>
<thead>
<tr>
<th>N</th>
<th>Uniform SQP iters</th>
<th>Uniform Error</th>
<th>Adaptive (K = 1) SQP iters</th>
<th>Adaptive (K = 1) Error</th>
<th>Adaptive (BAC) SQP iters</th>
<th>Adaptive (BAC) Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>21</td>
<td>7.8743</td>
<td>16</td>
<td>6.5279</td>
<td>13</td>
<td>8.0868</td>
</tr>
<tr>
<td>15</td>
<td>33</td>
<td>4.1358</td>
<td>21</td>
<td>2.9521</td>
<td>25</td>
<td>3.6024</td>
</tr>
<tr>
<td>20</td>
<td>28</td>
<td>2.208</td>
<td>29</td>
<td>1.0459</td>
<td>32</td>
<td>2.049</td>
</tr>
<tr>
<td>25</td>
<td>27</td>
<td>1.1581</td>
<td>35</td>
<td>0.85025</td>
<td>30</td>
<td>1.9559</td>
</tr>
<tr>
<td>30</td>
<td>36</td>
<td>0.75308</td>
<td>30</td>
<td>1.0686</td>
<td>38</td>
<td>1.2303</td>
</tr>
</tbody>
</table>

![Figure 5.1. Uniform mesh division and absolute error of solution for problem (NLP) using N = 14](image)

The calculus of variations. The state inequality constraint was handled by augmenting the Hamiltonian with a penalty term:

\[
H(x_1, x_2, p_1, p_2, u) = \frac{1}{2} u^2 + p_1 x_2 + p_2 u + \lambda(x_1)
\]

where

\[
\lambda(x_1) := \left( \frac{x_1}{L} \right)^M
\]
with $M \in \mathbb{N}$ large, even. The necessary first-order optimality conditions were then:

\[
\begin{aligned}
\dot{u} + p_2 &= 0; \\
\dot{p}_1 + \frac{\partial \lambda}{\partial x_1} &= 0; \\
\dot{p}_2 + p_1 &= 0; \\
\dot{x}_1 - x_2 &= 0, \quad x_1(0) = 0, \quad x_1(1) = 0; \\
\dot{x}_2 + p_2 &= 0, \quad x_2(0) = 1, \quad x_2(1) = -1.
\end{aligned}
\]

This two-point boundary value problem was solved using a collocation solver. The solution accuracy compared to the reference solution was $||x_{bvp} - x_{ref}||_1 \approx 0.20206$. when using a penalty term $M = 150$. This accuracy compares with direct collocation using coarse meshes of only 10-15 mesh points, despite the fact that the collocation BVP solver (Matlab’s \textit{bvp4c}) used internally over 100 mesh points. Furthermore, the solution does not exhibit a proper boundary arc as can be seen from Figure 5.3.

6. Conclusions

Of the three junction point identification methods, the method based on the Lagrange multipliers of the NLP problem (JCE) exhibited the most accurate convergence. Unfortunately the method also suffers from instability in some cases. The reason for this instability was not clear. The method based on solving a three-point subproblem (SAOE) was almost as accurate but did not display any instability and was therefore chosen as the method of choice for testing the actual algorithms.

For real problems, correctly identifying the boundary arcs can be a difficult job, since in addition to proper boundary arcs it is possible that the optimal trajectory merely touches the boundary at a single touch point. In fact, such behaviour occurs every time the constraint is of odd order higher or equal to 3 ([8]). In such cases, identification of the touch points would require more robust methods.
Figure 5.3. Optimal trajectory for problem (LP) as given by the indirect collocation method. For comparison, the reference solution is drawn for the state component $x_1$ using a dotted line. The indirect collocation solution does not correctly follow the boundary arc $x(t) = 1/9$.

Problem (LP) was more suited to the uniform mesh. The linearity of the problem together with the uniform mesh causes the quasi-Newton SQP algorithm to converge with only a few steps. In contrast, the adaptive mesh takes many more steps to converge without boundary arc completion. If boundary arc completion is enabled, the final result is slightly less accurate but convergence is improved considerably. However, nothing suggests that an adaptive mesh is really necessary for simple linear path-constrained problems.

For problem (NLP) it can immediately be noted that more mesh points are needed to obtain a sensible solution. When $N \leq 25$, the solution with the adaptive mesh is superior to the one given by a uniform mesh. However, as the number of subintervals $N$ increases then the adaptive mesh starts to resemble locally the uniform mesh and any advantages disappear. Here the benefits of using boundary arc completion are negligible at best.

It should be noted that in the solution of problem (NLP), all meshes were uniform on each subinterval. While in theory better equidistributing meshes could be
achieved by increasing the quasiumiformity constant $K$, in practice this results in numerical instability that can destroy the theoretical benefits quite easily. This suggests that the choice of $K$ is dependant on the problem in hand and should be included as a parameter for the user.

While more accurate results were obtained using non-uniform adaptive meshes, the number of iterations required also grew. The introduction of efficient SQP based solvers like SNOPT\(^1\) makes such problems tractable, but for computational efficiency it would be better if the initial guess which produces the adaptive mesh would be cheaply computable. One idea is to use the indirect solution with a penalty function, as detailed in Section 5.3, but this approach does not allow for methods based on the Lagrange multipliers. There is also the added workload of deriving necessary conditions for the solution.

It should be noted that the adaptive mesh generation methods explained in this work do not necessarily require that the solution method used be specifically the direct collocation method, and further study of similar techniques using other numerical solution methods for optimal control problems is in order.

References


\(^1\)SNOPT is a FORTRAN Package for large-scale nonlinear programming, developed by Philip Gill (University of California, San Diego) and Walter Murray and Michael Saunders (Systems Optimization Laboratory, Stanford University, Stanford).