Optimal Control and Applications

Session 1
2nd AstroNet-II Summer School

Victor M. Becerra
School of Systems Engineering
University of Reading

5-6 June 2013
Contents of Session 1

- Introduction to optimal control
- A brief on constrained nonlinear optimisation
- Introduction to calculus of variations
- The optimal control problem
- Optimality conditions
- Pontryagin’s minimum principle
- Equality path constraints
- Singular arcs
- Inequality path constraints
Contents of Session 2

- Direct collocation methods for optimal control
- Local discretisations
- Global (pseudospectral) discretisations
- Practical aspects
  - Multi-phase problems
  - Efficient sparse differentiation
  - Scaling
  - Estimating the discretisation accuracy
  - Mesh refinement
Introduction to optimal control

For the purposes of these lectures, a dynamical system can be described by a system of ordinary differential equations, as follows:

\[ \dot{x} = f[x(t), u(t), t] \]

where \( x : [t_0, t_f] :\rightarrow R^n_x \) is the state, \( u(t) : [t_0, t_f] :\rightarrow R^{nu} \) is the control, \( t \) is time, and \( f : R^n_x \times R^{nu} \times R :\rightarrow R^n_x \). Suppose that we are interested in an interval of time such that \( t \in [t_0, t_f] \).
We are also interested in a measure of the performance of the system along a trajectory.

We are going to quantify the performance of the system by means of a functional $J$, this is a rule of correspondence that assigns a value to a function or set of functions.

The functional of interest depends on the state history $x(t)$ and the control history $u(t)$ over the period of interest $t \in [t_0, t_f]$, the initial time $t_0$ and the final time $t_f$.

$$J = J[x(\cdot), u(\cdot), t_0, t_f]$$
Introduction to optimal control

The performance index $J$ might include, for example, measures of:

- control effort
- fuel consumption
- energy expenditure
- time taken to reach a target
What is optimal control?

Optimal control is the process of finding control and state histories for a dynamic system over a period of time to \textit{minimise} or \textit{maximise} a performance index.
Introduction to optimal control

- Optimal control theory has been developed over a number of centuries.
- It is closely related to the theory of calculus of variations which started in the late 1600’s.
- Also related with developments in dynamic programming and nonlinear programming after the second world war.
- The digital computer has enabled many complex optimal control problems to be solved.
- Optimal control is a fundamental tool in space mission design, as well as in many other fields.
Introduction to optimal control

- As a form of motivation, consider the design of a **zero propellant manoeuvre** for the international space station by means of control moment gyroscopes (CMGs).

- The example work was reported by Bhatt (2007) and Bedrossian and co-workers (2009).

- The original 90 and 180 degree manoeuvres were computed using a pseudospectral method.
Introduction to optimal control

- Savings for NASA of around US$1.5m in propellant costs.
- The case described below corresponds with a $90^\circ$ maneuver lasting 7200 seconds and using 3 CMG’s.
Introduction to optimal control

The problem is formulated as follows. Find \( q_c(t) = [q_{c,1}(t) \ q_{c,2}(t) \ q_{c,3}(t) \ q_{c,4}]^T, \ t \in [t_0, t_f] \) and the scalar parameter \( \gamma \) to minimise,

\[
J = 0.1\gamma + \int_{t_0}^{t_f} \|u(t)\|^2 dt
\]

subject to the dynamical equations:

\[
\begin{align*}
\dot{q}(t) &= \frac{1}{2} T(q)(\omega(t) - \omega_o(q)) \\
\dot{\omega}(t) &= J^{-1}(\tau_d(q) - \omega(t) \times (J\omega(t)) - u(t)) \\
\dot{h}(t) &= u(t) - \omega(t) \times h(t)
\end{align*}
\]

where \( J \) is a \( 3 \times 3 \) inertia matrix, \( q = [q_1, q_2, q_3, q_4]^T \) is the quaternion vector, \( \omega \) is the spacecraft angular rate relative to an inertial reference frame and expressed in the body frame, \( h \) is the momentum, \( t_0 = 0 \) s, \( t_f = 7200 \) s
Introduction to optimal control

The path constraints:

\[
\|q(t)\|^2 = 1 \\
\|q_c(t)\|^2 = 1 \\
\|h(t)\|^2 \leq \gamma \\
\|\dot{h}(t)\|^2 = \dot{h}_{\text{max}}^2
\]

The parameter bounds

\[
0 \leq \gamma \leq h_{\text{max}}^2
\]

And the boundary conditions:

\[
q(t_0) = \bar{q}_0 \quad \omega(t_0) = \omega_o(\bar{q}_0) \quad h(t_0) = \bar{h}_0 \\
q(t_f) = \bar{q}_f \quad \omega(t_f) = \omega_o(\bar{q}_f) \quad h(t_f) = \bar{h}_f
\]
**Introduction to optimal control**

$T(q)$ is given by:

$$
T(q) = \begin{bmatrix}
-q_2 & -q_3 & -q_4 \\
q_1 & -q_4 & q_3 \\
q_4 & q_1 & -q_2 \\
-q_3 & q_2 & q_1 \\
\end{bmatrix}
$$
**Introduction to optimal control**

\( u \) is the control force, which is given by:

\[
    u(t) = J \left( K_P \tilde{\epsilon}(q, q_c) + K_D \tilde{\omega}(\omega, q_c) \right)
\]

where

\[
    \tilde{\epsilon}(q, q_c) = 2T(q_c)^T q
\]

\[
    \tilde{\omega}(\omega, \omega_c) = \omega - \omega_c
\]
Introduction to optimal control

\( \omega_o \) is given by:

\[
\omega_o(q) = nC_2(q)
\]

where \( n \) is the orbital rotation rate, \( C_j \) is the \( j \) column of the rotation matrix:

\[
C(q) = \begin{bmatrix}
1 - 2(q_3^2 + q_4^2) & 2(q_2q_3 + q_1q_4) & 2(q_2q_4 - q_1q_3) \\
2(q_2q_3 - q_1q_4) & 1 - 2(q_2^2 + q_4^2) & 2(q_3q_4 + q_1q_2) \\
2(q_2q_4 + q_1q_3) & 2(q_3q_4 - q_1q_2) & 1 - 2(q_2^2 + q_3^2)
\end{bmatrix}
\]

\( \tau_d \) is the disturbance torque, which in this case only incorporates the gravity gradient torque \( \tau_{gg} \), and the aerodynamic torque \( \tau_a \):

\[
\tau_d = \tau_{gg} + \tau_a = 3n^2C_3(q) \times (JC_3(q)) + \tau_a
\]
Introduction to optimal control

Video animation of the zero propellant maneouvre using real telemetry.
http://www.youtube.com/watch?v=MIp27Ea9_2I
Introduction to optimal control

Numerical results obtained by the author with PSOPT using a pseudospectral discretisation method with 60 grid points, neglecting the aerodynamic torque.
Introduction to optimal control

Ingredients for solving realistic optimal control problems

- Methods for systems of nonlinear algebraic equations
- Methods for solving ODE's
- Methods for constrained nonlinear optimisation

Direct methods

Indirect methods
Introduction to optimal control

- Indirect methods
  - Optimality conditions of continuous problems are derived
  - Attempt to numerically satisfy the optimality conditions
  - Requires numerical root finding and ODE methods.

- Direct methods
  - Continuous optimal control problem is discretised
  - Need to solve a nonlinear programming problem

Later in these lectures, we will concentrate on some direct methods.
A brief on constrained nonlinear optimisation

Consider a twice continuously differentiable scalar function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Then $y^*$ is called a **strong minimiser** of $f$ if there exists $\delta \in \mathbb{R}$ such that

$$f(y^*) < f(y^* + \Delta y)$$

for all $0 < ||\Delta y|| \leq \delta$.

Similarly $y^*$ is called a **weak minimiser** if it is not strong and there is $\delta \in \mathbb{R}$ such that $f(y^*) \leq f(y^* + \Delta y)$ for all $0 < ||\Delta y|| \leq \delta$. The minimum is said to be **local** if $\delta < \infty$ and **global** if $\delta = \infty$. 
A brief on constrained nonlinear optimisation

The gradient of $f(y)$ with respect to $y$ is a column vector constructed as follows:

$$\nabla_y f(y) = \frac{\partial f}{\partial y}^T = f_y^T = \begin{bmatrix} \frac{\partial f}{\partial y_1} \\ \vdots \\ \frac{\partial f}{\partial y_n} \end{bmatrix}$$
A brief on constrained nonlinear optimisation

The second derivative of $f(y)$ with respect to $y$ is an $n_y \times n_y$ symmetric matrix known as the **Hessian** and is expressed as follows:

$$\nabla^2_y f(y) = f_{yy} = \begin{bmatrix}
\frac{\partial f^2}{\partial y_1^2} & \cdots & \frac{\partial f^2}{\partial y_1 \partial y_{n_y}} \\
\vdots & \ddots & \vdots \\
\frac{\partial f^2}{\partial y_{n_y} \partial y_1} & \cdots & \frac{\partial f^2}{\partial y_{n_y}^2}
\end{bmatrix}$$
A brief on constrained nonlinear optimisation

Consider the second order Taylor expansion of $f$ about $y^*$:

$$f(y^* + \Delta y) \approx f(y^*) + f_y(y^*)\Delta y + \frac{1}{2} \Delta y^T f_{yy}(y^*) \Delta y$$

A basic result in unconstrained nonlinear programming is the following set of conditions for a local minimum.

- **Necessary conditions:**
  - $f_y(y^*) = 0$, i.e. $y^*$ must be a stationary point of $f$.
  - $f_{yy}(y^*) \geq 0$, i.e. the Hessian matrix is positive semi-definite.

- **Sufficient conditions:**
  - $f_y(y^*) = 0$, i.e. $y^*$ must be a stationary point of $f$.
  - $f_{yy}(y^*) > 0$, i.e. the Hessian matrix is positive definite.
A brief on constrained nonlinear optimisation

For example, the parabola \( f(x) = x_1^2 + x_2^2 \) has a global minimum at \( x^* = [0, 0]^T \), where the gradient \( f_x(x^*) = [0, 0] \) and the Hessian is positive definite:

\[
    f_{xx}(x^*) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}
\]

Surface plot of \( f(x,y)=x_1^2 + x_2^2 \)
A brief on constrained nonlinear optimisation

Constrained nonlinear optimisation involves finding a decision vector $y \in \mathbb{R}^n$ to minimise:

$$f(y)$$

subject to

$$h_l \leq h(y) \leq h_u,$$

$$y_l \leq y \leq y_u,$$

where $f : \mathbb{R}^n \twoheadrightarrow \mathbb{R}$ is a twice continuously differentiable scalar function, and $h : \mathbb{R}^n \twoheadrightarrow \mathbb{R}^n$ is a vector function with twice continuously differentiable elements $h_i$. Problems such as the above are often called **nonlinear programming** (NLP) problems.
A brief on constrained nonlinear optimisation

A particular case that is simple to analyse occurs when the nonlinear programming problem involves only equality constraints, such that the problem can be expressed as follows. Choose $y$ to minimise

$$f(y)$$

subject to

$$c(y) = 0,$$

where $c : \mathbb{R}^{n_y} \mapsto \mathbb{R}^{n_c}$ is a vector function with twice continuously differentiable elements $c_i$, and $n_c < n_y$. 
A brief on constrained nonlinear optimisation

Introduce the Lagrangian function:

\[ \mathcal{L}[\mathbf{y}, \lambda] = f(\mathbf{y}) + \lambda^T \mathbf{c}(\mathbf{y}) \]  

(1)

where \( \mathcal{L} : \mathbb{R}^{n_y} \times \mathbb{R}^{nc} \rightarrow \mathbb{R} \) is a scalar function and \( \lambda \in \mathbb{R}^{nc} \) is a Lagrange multiplier.
A brief on constrained nonlinear optimisation

The first derivative of $c(y)$ with respect to $y$ is an $n_c \times n_y$ matrix known as the **Jacobian** and is expressed as follows:

$$
\frac{\partial c(y)}{\partial y} = c_y(y) = \begin{bmatrix}
\frac{\partial c_1}{\partial y_1} & \cdots & \frac{\partial c_1}{\partial y_{n_y}} \\
\vdots & \ddots & \vdots \\
\frac{\partial c_{n_c}}{\partial y_1} & \cdots & \frac{\partial c_{n_c}}{\partial y_{n_y}}
\end{bmatrix}
$$
The first order necessary conditions for a point \((y^*, \lambda)\) to be a local minimiser are as follows:

\[
\begin{align*}
\mathcal{L}_y[y^*, \lambda]^T &= f_y(y^*)^T + [c_y(y^*)]^T \lambda = 0 \\
\mathcal{L}_\lambda[y^*, \lambda]^T &= c(y^*) = 0
\end{align*}
\]  

Note that equation (2) is a system of \(n_y + n_c\) equations with \(n_y + n_c\) unknowns.
A brief on constrained nonlinear optimisation

For a scalar function $f(x)$ and a single equality constraint $h(x)$, the figure illustrates the collinearity between the gradient of $f$ and the gradient of $h$ at a stationary point, i.e. $\nabla_x f(x^*) = -\lambda \nabla_x c(x^*)$. 
A brief on constrained nonlinear optimisation

For example, consider the minimisation of the function $f(x) = x_1^2 + x_2^2$ subject to the linear equality constraint $c(x) = 2x_1 + x_2 + 4 = 0$.

The first order necessary conditions lead to a system of three linear equations whose solution is the minimiser $x^*$. 
A brief on constrained nonlinear optimisation

In fairly simple cases, the necessary conditions may help find the solution. In general, however,

- There is no analytic solution
- It is necessary to solve the problem numerically
- Numerical methods approach the solution iteratively.
A brief on constrained nonlinear optimisation

If the nonlinear programming problem also includes a set of inequality constraints, then the resulting problem is as follows. Choose $y$ to minimise

$$f(y)$$

subject to

$$c(y) = 0,$$
$$q(y) \leq 0$$

where $q : \mathbb{R}^{ny} \mapsto \mathbb{R}^{nq}$ is a twice continuously differentiable vector function.
A brief on constrained nonlinear optimisation

The Lagrangian function is redefined to include the inequality constraints function:

$$\mathcal{L}[\mathbf{y}, \lambda, \mu] = f(\mathbf{y}) + \lambda^T \mathbf{c}(\mathbf{y}) + \mu^T \mathbf{q}(\mathbf{y})$$

where $\mu \in \mathcal{R}^{n_q}$ is a vector of Karush-Kuhn-Tucker multipliers.
Definition

Active and inactive constraints An inequality constraint $q_i(y)$ is said to be active at $y$ if $q_i(y) = 0$. It is inactive at $y$ if $q_i(y) < 0$.

Definition

Regular point Let $y^*$ satisfy $c(y^*) = 0$ and $q(y^*) \leq 0$, and let $I_c(y^*)$ be the index set of active inequality constraints. Then we say that $y^*$ is a regular point if the gradient vectors:

$$\nabla_y c_i(y^*), \nabla_y q_j(y^*), 1 \leq i \leq n_c, j \in I_c(y^*)$$

are linearly independent.
A brief on constrained nonlinear optimisation

In this case, the first order necessary conditions for a local minimiser are the well known Karush-Kuhn-Tucker (KKT) conditions, which are stated as follows. If $y^*$ is a regular point and a local minimiser, then:

$$
\mu \geq 0 \\
\mathcal{L}_y[y^*, \lambda, \mu] = f_y(y^*) + [c_y(y^*)]^T \lambda + [q_y(y^*)]^T \mu = 0 \\
\mathcal{L}_\lambda[y^*, \lambda] = c(y^*) = 0 \\
\mu^T q(y^*) = 0
$$

(4)
A brief on constrained nonlinear optimisation

- Methods for the solution of nonlinear programming problems are well established.
- Well known methods include Sequential Quadratic Programming (SQP) and Interior Point methods.
- A quadratic programming problem is a special type of nonlinear programming problem where the constraints are linear and the objective is a quadratic function.
A brief on constrained nonlinear optimisation

- **SQP methods** involve the solution of a sequence of quadratic programming sub-problems, which give, at each iteration, a search direction which is used to find the next iterate through a line search on the basis of a merit (penalty) function.

- **Interior point methods** involve the iterative enforcement of the KKT conditions doing a line search at each major iteration on the basis of a barrier function to find the next iterate.
A brief on constrained nonlinear optimisation

- Current NLP implementations incorporate developments in numerical linear algebra that exploit sparsity in matrices.
- It is common to numerically solve NLP problems involving variables and constraints in the order of hundreds of thousands.
- Well known large scale NLP solvers methods include SNOPT (SQP) and IPOPT (Interior Point).
A quick introduction to calculus of variations

**Definition (Functional)**

A functional $J$ is a rule of correspondence that assigns to each function $x$ a unique real number.

For example, Suppose that $x$ is a continuous function of $t$ defined in the interval $[t_0, t_f]$ and

$$J(x) = \int_{t_0}^{t_f} x(t) \, dt$$

is the real number assigned by the functional $J$ (in this case, $J$ is the area under the $x(t)$ curve).
A quick introduction to calculus of variations

The **increment** of a functional $J$ is defined as follows:

$$\Delta J = J(x + \delta x) - J(x)$$ \hspace{1cm} (5)

where $\delta x$ is called the *variation* of $x$
A quick introduction to calculus of variations

The variation of a functional $J$ is defined as follows:

$$\delta J = \lim_{||\delta x|| \to 0} \Delta J = \lim_{||\delta x|| \to 0} J(x + \delta x) - J(x)$$

Example: Find the variation $\delta J$ of the following functional:

$$J(x) = \int_0^1 x^2(t) dt$$

Solution:

$$\Delta J = J(x + \delta x) - J(x) = \ldots$$

$$\delta J = \int_0^1 (2x\delta x) dt$$
A quick introduction to calculus of variations

In general, the variation of a differentiable functional of the form:

$$J = \int_{t_0}^{t_f} L(x) dt$$

can be calculated as follows for fixed $t_0$ and $t_f$:

$$\delta J = \int_{t_0}^{t_f} \left\{ \frac{\partial L(x)}{\partial x} \right\} \delta x \ dt$$

If $t_0$ and $t_f$ are also subject to small variations $\delta t_0$ and $\delta t_f$, then the variation of $J$ can be calculated as follows:

$$\delta J = L(x(t_f)) \delta t_f - L(x(t_0)) \delta t_0 + \int_{t_0}^{t_f} \left\{ \frac{\partial L(x)}{\partial x} \right\} \delta x \ dt$$
A quick introduction to calculus of variations

If \( x \) is a vector function of dimension \( n \), then we have, for fixed \( t_0 \) and \( t_f \):

\[
\delta J = \int_{t_0}^{t_f} \left\{ \left[ \frac{\partial L(x)}{\partial x} \right] \delta x \right\} dt = \int_{t_0}^{t_f} \left\{ \sum_{i=1}^{n} \left[ \frac{\partial L(x)}{\partial x_i} \right] \delta x_i \right\} dt
\]
A quick introduction to calculus of variations

A functional $J$ has a local minimum at $x^*$ if for all functions $x$ in the vicinity of $x^*$ we have:

$$\Delta J = J(x) - J(x^*) \geq 0$$

For a local maximum, the condition is:

$$\Delta J = J(x) - J(x^*) \leq 0$$
A quick introduction to calculus of variations

**Important result**

If $x^*$ is a local minimiser or maximiser, then the variation of $J$ is zero:

\[ \delta J[x^*, \delta x] = 0 \]
A quick introduction to calculus of variations

A basic problem in calculus of variations is the minimisation or maximisation of the following integral with fixed $t_0$, $t_f$, $x(t_0)$, $x(t_f)$:

$$J = \int_{t_0}^{t_f} L[x(t), \dot{x}(t), t] \, dt$$

for a given function $L$. To solve this problem, we need to compute the variation of $J$, for a variation in $x$. This results in:

$$\delta J = \int_{t_0}^{t_f} \left[ \frac{\partial L}{\partial x} - \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{x}} \right) \right] \delta x \, dt$$
A quick introduction to calculus of variations

Setting $\delta J = 0$ results in the Euler-Lagrange equation:

$$\frac{\partial L}{\partial x} - \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{x}} \right) = 0$$

This equation needs to be satisfied by the function $x^*$ which
minimises or maximises $J$. This type of solution is called an
*extremal* solution.
Example

Find an extremal for the following functional with $x(0) = 0$, $x(\pi/2) = 1$.

$$J(x) = \int_0^{\pi/2} [\dot{x}^2(t) - x^2(t)] \, dt$$

Solution: the Euler-Lagrange equation gives $-2x - 2\ddot{x} = 0$ such that $\ddot{x} + x = 0$

The solution of this second order ODE is $x(t) = c_1 \cos t + c_2 \sin t$. Using the boundary conditions gives $c_0 = 0$ and $c_1 = 1$, such that the extremal is:

$$x^*(t) = \sin t$$
A general optimal control problem

Optimal control problem

Determine the state $x(t)$, the control $u(t)$, as well as $t_0$ and $t_f$, to minimise

$$J = \Phi[x(t_0), t_0, x(t_f), t_f] + \int_{t_0}^{t_f} L[x(t), u(t), t] dt$$

subject to

- Differential constraints:
  $$\dot{x} = f[x, u, t]$$
- Algebraic path constraints
  $$h_L \leq h[x(t), u(t), t] \leq h_U$$
- Boundary conditions
  $$\psi[x(t_0), t_0, x(t_f), t_f] = 0$$
A general optimal control problem

Important features include:

- It is a functional minimisation problem, as \( J \) is a functional.
- Hence we need to use calculus of variations to derive its optimality conditions.
- All optimal control problems have constraints.
A general optimal control problem

There are various classes of optimal control problems:

- Free and fixed time problems
- Free and fixed terminal/initial states
- Path constrained and unconstrained problems
- ...

Optimal Control and Applications
V. M. Becerra - Session 1 - 2nd AstroNet-II Summer School
A general optimal control problem

For the sake of simplicity, the first order optimality conditions for the general optimal control problem will be given ignoring the path constraints.

The derivation involves constructing an augmented performance index:

\[ J_a = \Phi + \nu^T \psi + \int_{t_0}^{t_f} \left[ L + \lambda^T (f - \dot{x}) \right] dt \]

where \( \nu \) and \( \lambda(t) \) are Lagrange multipliers for the boundary conditions and the differential constraints, respectively. \( \lambda(t) \) is also known as the co-state vector. The first variation of \( J_a \), \( \delta J_a \), needs to be calculated. We assume that \( u(t), x(t), x(t_0), x(t_f), t_0, t_f, \lambda(t), \lambda(t_0), \lambda(t_f) \), and \( \nu \) are free to vary.
A general optimal control problem

Define the Hamiltonian function as follows:

\[ H[x(t), u(t), \lambda(t), t] = L[x(t), u(t), t] + \lambda(t)^T f[x(t), u(t), t] \]

After some work, the variation \( \delta J_a \) can be expressed as follows:

\[ \delta J_a = \delta \Phi + \delta \nu^T \psi + \nu^T \delta \psi + \delta \int_{t_0}^{t_f} \left[ H - \lambda^T \dot{x} \right] dt \]

where:

\[ \delta \Phi = \frac{\partial \Phi}{\partial x(t_0)} \delta x_0 + \frac{\partial \Phi}{\partial t_0} \delta t_0 + \frac{\partial \Phi}{\partial x(t_f)} \delta x_f + \frac{\partial \Phi}{\partial t_f} \delta t_f \]

\[ \delta \psi = \frac{\partial \psi}{\partial x(t_0)} \delta x_0 + \frac{\partial \psi}{\partial t_0} \delta t_0 + \frac{\partial \psi}{\partial x(t_f)} \delta x_f + \frac{\partial \psi}{\partial t_f} \delta t_f \]

\[ \delta x_0 = \delta x(t_0) + \dot{x}(t_0) \delta t_0, \quad \delta x_f = \delta x(t_f) + \dot{x}(t_f) \delta t_f \]
A general optimal control problem

and

\[ \delta \int_{t_0}^{t_f} [H - \lambda \dot{x}] dt = \lambda^T(t_0) \delta x_0 - \lambda^T(t_f) \delta x_f \]

\[ + H(t_f) \delta t_f - H(t_0) \delta t_0 \]

\[ + \int_{t_0}^{t_f} \left[ \left( \frac{\partial H}{\partial x} + \dot{\lambda}^T \right) \delta x + \left( \frac{\partial H}{\partial \lambda} + \dot{x}^T \right) \delta \lambda + \frac{\partial H}{\partial u} \delta u \right] dt \]
A general optimal control problem

Assuming that initial and terminal times are free, and that the control is unconstrained, setting $\delta J_a = 0$ results in the following first order necessary optimality conditions.

\[
\dot{x} = \left[ \frac{\partial H}{\partial \lambda} \right]^T = f[x(t), u(t), t]
\]

(state dynamics)

\[
\dot{\lambda} = - \left[ \frac{\partial H}{\partial x} \right]^T
\]

(costate dynamics)

\[
\frac{\partial H}{\partial u} = 0
\]

(stationarity condition)
A general optimal control problem

In addition we have the following conditions at the boundaries:

\[ \psi[x(t_0), t_0, x(t_f), t_f] = 0 \]

\[
\begin{bmatrix}
\lambda(t_0) + \left[ \frac{\partial \Phi}{\partial x(t_0)} \right]^T + \left[ \frac{\partial \psi}{\partial x(t_0)} \right]^T \nu \\
\end{bmatrix}^T \delta x_0 + \left[ -H(t_0) + \frac{\partial \Phi}{\partial t_0} + \nu^T \frac{\partial \psi}{\partial t_0} \right] \delta t_0 = 0
\]

\[
\begin{bmatrix}
-\lambda(t_f) + \left[ \frac{\partial \Phi}{\partial x(t_f)} \right]^T + \left[ \frac{\partial \psi}{\partial x(t_f)} \right]^T \nu \\
\end{bmatrix}^T \delta x_f + \left[ H(t_f) + \frac{\partial \Phi}{\partial t_f} + \nu^T \frac{\partial \psi}{\partial t_f} \right] \delta t_f = 0
\]

Note that \( \delta x_0 \) depends on \( \delta t_0 \) so in the second equation above the terms that multiply the variations of \( \delta x_0 \) and \( \delta t_0 \) cannot be made zero independently. Something similar to this happens in the third equation.
A general optimal control problem

- The conditions in boxes are *necessary conditions*, so solutions $(x^*(t), \lambda^*(t), u^*(t), \nu^*)$ that satisfy them are *extremal solutions*.
- They are only candidate solutions.
- Each extremal solution can be a minimum, a maximum or a saddle.
A general optimal control problem

- Note the dynamics of the state - co-state system:

\[ \dot{x} = \nabla_\lambda H \]
\[ \dot{\lambda} = -\nabla_x H \]

- This is a Hamiltonian system.

- Boundary conditions are at the beginning and at the end of the interval, so this is a two point boundary value problem.

- If \( H \) is not an explicit function of time, \( H = \text{constant} \).

- If the Hamiltonian is not an explicit function of time and \( t_f \) is free, then \( H(t) = 0 \).
Example: The brachistochrone problem

This problem was solved by Bernoulli in 1696. A mass $m$ moves in a constant force field of magnitude $g$ starting at rest at the origin $(x, y) = (x_0, y_0)$ at time 0. It is desired to find a path of minimum time to a specified final point $(x_1, y_1)$.
Example: The brachistochrone problem

The equations of motion are:

\[
\begin{align*}
\dot{x} &= V \cos \theta \\
\dot{y} &= V \sin \theta 
\end{align*}
\]

where \( V(t) = \sqrt{2gy} \). The performance index (we are after minimum time), can be expressed as follows:

\[
J = \int_{0}^{t_f} 1 \, dt = t_f
\]

and the boundary conditions on the states are

\((x(0), y(0)) = (x_0, y_0), (x(t_f), y(t_f)) = (x_1, y_1)\).
Example: The brachistochrone problem

In this case, the Hamiltonian function is:

\[ H = 1 + \lambda_x(t) V(t) \cos \theta(t) + \lambda_y(t) V(t) \sin \theta(t) \]

The costate equations are:

\[ \dot{\lambda}_x = 0 \]

\[ \dot{\lambda}_y = -\frac{g}{V} (\lambda_x \cos \theta + \lambda_y \sin \theta) \]

and the stationarity condition is:

\[ 0 = -\lambda_x V \sin \theta + \lambda_y V \cos \theta \implies \tan \theta = \frac{\lambda_y}{\lambda_x} \]
Example: The brachistochrone problem

We need to satisfy the boundary conditions:

$$\psi = \begin{bmatrix} x(0) - x_0 \\ y(0) - y_0 \\ x(t_f) - x_1 \\ y(t_f) - y_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Also since the Hamiltonian is not an explicit function of time and \( t_f \) is free, we have that \( H(t) = 0 \). Using this, together with the stationarity condition we can write:

$$\lambda_x = -\frac{\cos \theta}{V}$$

$$\lambda_y = -\frac{\sin \theta}{V}$$
Example: The brachistochrone problem

After some work with these equations, we can write:

\[
\begin{align*}
  y &= \frac{y_1}{\cos^2 \theta(t_f)} \cos^2 \theta \\
  x &= x_1 + \frac{y_1}{2 \cos^2 \theta(t_f)} \left[2(\theta(t_f) - \theta) + \sin 2\theta(t_f) - \sin 2\theta\right] \\
  t_f - t &= \sqrt{\frac{2y_1}{g} \frac{\theta - \theta(t_f)}{\cos \theta(t_f)}}
\end{align*}
\]

Given the current position \((x, y)\) these three equations can be used to solve for \(\theta(t)\) and \(\theta(t_f)\) and the time to go \(t_f - t\).
Example: The brachistochrone problem

Using the above results, it is possible to find the states:

\[ y(t) = a(1 - \cos \phi(t)) \]
\[ x(t) = a(\phi - \sin \phi(t)) \]

where

\[ \phi = \pi - 2\theta \]
\[ a = \frac{y_1}{1 - \cos \phi(t_f)} \]

The solution describes a cycloid in the \( x - y \) plane that passes through \((x_1, y_1)\).
Example: The brachistochrone problem

The figures illustrates numerical results with $(x(0), y(0)) = (0, 0)$, $(x(t_f), y(t_f)) = (2, 2)$ and $g = 9.8$. 
Pontryagin’s minimum principle

Assume that the optimal control is $u^*$, and that the states and co-states are fixed at their optimal trajectories $x^*(t), \lambda^*(t)$. we have that for all $u$ close to $u^*$:

$$J(u^*) \leq J(u) \implies J(u) - J(u^*) \geq 0$$

Then we have that $J(u) = J(u^*) + \delta J[u^*, \delta u]$. 
Pontryagin’s minimum principle

From the variation in the augmented functional $\delta J_a$, we have in this case:

$$\delta J[u^*, \delta u] = \int_{t_0}^{t_f} \left[ \frac{\partial H}{\partial u} \right] \delta u dt$$
Pontryagin’s minimum principle

But, using a Taylor series we know that:

\[ H[x^*, u^* + \delta u, \lambda^*] = H[x^*, u^*, \lambda^*] + \left[ \frac{\partial H}{\partial u} \right]_{u^*} \delta u + \ldots \]

So that, to first order:

\[ \delta J[u^*, \delta u] = \int_{t_0}^{t_f} (H[x^*, u^* + \delta u, \lambda^*] - H[x^*, u^*, \lambda^*]) \, dt \geq 0 \]
Pontryagin’s minimum principle

This leads to:

\[ H[x^*, u^* + \delta u, \lambda^*] - H[x^*, u^*, \lambda^*] \geq 0 \]

for all admissible variations in \( u \). This in turn leads to the minimum principle:

\[ u^* = \arg\min_{u \in U} H[x^*, u, \lambda^*] \]

where \( U \) is the admissible control set.
Pontryagin’s minimum principle

Assuming that the admissible control set is closed then, if the control lies in the interior of the set, we have that the minimum principle implies that:

$$\frac{\partial H}{\partial u} = 0$$

Otherwise, if the optimal control is at the boundary of the admissible control set, the minimum principle requires that the optimal control must be chosen to minimise the Hamiltonian:

$$u^* = \arg \min_{u \in U} H[x^*, u, \lambda^*]$$
Example: Minimum time control with constrained input

Consider the following optimal control problem. Chose $t_f$ and $u(t) \in \mathcal{U} = [-1, 1]$, $t \in [t_0, t_f]$ to minimise:

$$J = t_f = \int_{0}^{t_f} 1dt$$

subject to

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = u$$

with the boundary conditions:

$$x_1(0) = x_{10}, \quad x_1(t_f) = 0$$
$$x_2(0) = x_{20}, \quad x_2(t_f) = 0$$
Example: Minimum time control with constrained input

In this case, the Hamiltonian is:

$$H = 1 + \lambda_1 x_2 + \lambda_2 u$$

Calculating $\frac{\partial H}{\partial u}$ gives no information on the control, so we need to apply the minimum principle:

$$u^* = \arg\min_{u \in [-1,1]} H[x^*, u, \lambda^*] = \begin{cases} 
+1 & \text{if } \lambda_2 < 0 \\
\text{undefined} & \text{if } \lambda_2 = 0 \\
-1 & \text{if } \lambda_2 > 0 
\end{cases}$$

This form of control is called **bang-bang**. The control switches value when the co-state $\lambda_2$ changes sign. The actual form of the solution depends on the values of the initial states.
Example: Minimum time control with constrained input

The costate equations are:

\[
\dot{\lambda}_1 = 0 \implies \lambda_1(t) = \lambda_{10} = \text{constant}
\]
\[
\dot{\lambda}_2 = -\lambda_1 \implies \lambda_2(t) = \lambda_{10}t + \lambda_{20}
\]

There are four cases:

1. \(\lambda_2\) is always positive.
2. \(\lambda_2\) is always negative.
3. \(\lambda_2\) switches from positive to negative.
4. \(\lambda_2\) switches from negative to positive.

So there will be at most one switch in the solution. *The minimum principle helps us to understand the nature of the solution.*
Equality path constraints

- Let’s assume that we impose equality path constraints of the form:

\[ c[x(t), u(t), t] = 0 \]

- This type of constraint may be part of the model of the system.

- The treatment of path constraints depends on the Jacobian matrix of the constraint function.
Equality path constraints

If the matrix $\frac{\partial c}{\partial u}$ is full rank, then the system of differential and algebraic equations formed by the state dynamics and the path constraints is a DAE of index 1. In this case, the Hamiltonian can be replaced by:

$$H = L + \lambda^T f + \mu^T c$$

which results in new optimality conditions.
Equality path constraints

- If the matrix $\frac{\partial c}{\partial u}$ is rank deficient, the index of the DAE is higher than one.
- This may require repeated differentiations of $c$ with respect to $u$ until one of the derivatives of $c$ is of full rank, a procedure called *rank reduction*.
- This may be difficult to perform and may be prone to numerical errors.
Singular arcs

- In some optimal control problems, extremal arcs satisfying the stationarity condition $\partial H / \partial u = 0$ occur where the Hessian matrix $H_{uu}$ is singular.
- These are called singular arcs.
- Additional tests are required to verify if a singular arc is optimizing.
Singular arcs

- A particular case of practical relevance occurs when the Hamiltonian function is linear in at least one of the control variables.
- In such cases, the control is not determined in terms of the state and co-state by the stationarity condition $\partial H / \partial u = 0$.
- Instead, the control is determined by repeatedly differentiating $\partial H / \partial u$ with respect to time until it is possible to obtain the control by setting the $k$-th derivative to zero.
Singular arcs

- In the case of a single control $u$, once the control is obtained by setting the $k$-th time derivative of $\partial H/\partial u$ to zero, then additional conditions known as the generalized Legendre-Clebsch conditions must be checked:

$$(-1)^k \frac{\partial}{\partial u} \left\{ \frac{d^{(2k)}}{dt^{(2k)}} \frac{\partial H}{\partial u} \right\} \geq 0, \quad k = 0, 1, 2, \ldots$$

- If the above conditions are satisfied, then the singular arc will be optimal.

- The presence of singular arcs may cause difficulties to computational optimal control methods to find accurate solutions if the appropriate conditions are not enforced a priori.
Inequality path constraints

- These are constraints of the form:
  \[ h_L \leq h[x(t), u(t), t] \leq h_U \]

  with \( h_{L,i} \neq h_{U,i} \).

- Inequality constraints may be active (\( h \) at one of the limits) or inactive (\( h \) within the limits) at any one time.

- As a result the time domain \( t \in [t_0, t_f] \) may be partitioned into constrained and unconstrained arcs.

- Different optimality conditions apply during constrained and unconstrained arcs.
Inequality path constraints

Three major complications:

1. The number of constrained arcs is not necessarily known a priori.
2. The location of the junction points between constrained and unconstrained arcs is not known a priori.
3. At the junction points it is possible that both the controls $u$ and the costates $\lambda$ are discontinuous.
4. Additional boundary conditions need to be satisfied at the junction points ($\implies$ multipoint boundary value problem).
Summary

- Optimal control theory for continuous time systems is based on calculus of variations.
- Using this approach, we formulate the first order optimality conditions of the problem.
- This leads to a Hamiltonian boundary value problem.
Summary

- Even if we use numerical methods, a sound understanding of control theory is important.
  - It provides a systematic way of formulating the problem.
  - It gives us ways of analysing the numerical solution.
  - It gives us conditions that we can incorporate into the formulation of the numerical problem.
Further reading

Optimal Control and Applications

Session 2
2nd AstroNet-II Summer School

Victor M. Becerra
School of Systems Engineering
University of Reading

5-6 June 2013
Contents of Session 2

- Direct collocation methods for optimal control
- Local discretisations
- Global (pseudospectral) discretisations
- Practical aspects
  - Multi-phase problems
  - Efficient sparse differentiation
  - Scaling
  - Estimating the discretisation accuracy
  - Mesh refinement
This session introduces a family of methods for the numerical solution of optimal control problems known as **direct collocation methods**.

These methods involve the **discretisation** of the ODSs of the problem.

The discretised ODEs becomes a set of **algebraic equality constraints**.

The problem time interval is **divided into segments** and the decision variables include the values of controls and state variables at the grid points.
Direct Collocation methods for optimal control

- ODE discretisation can be done, for example, using trapezoidal, Hermite-Simpson, or pseudospectral approximations.
- This involves defining a grid of points covering the time interval \([t_0, t_f]\), \(t_0 = t_1 < t_2 \cdots < t_N = t_f\).
The nonlinear optimization problems that arise from direct collocation methods may be very large.

For instance, Betts (2003) describes a spacecraft low thrust trajectory design with 211,031 variables and 146,285 constraints.
Direct Collocation methods for optimal control

- It is interesting that such large NLP problems may be easier to solve than boundary-value problems.
- The reason for the relative ease of computation of direct collocation methods is that the associated NLP problem is often quite sparse, and efficient software exist for its solution.
Direct Collocation methods for optimal control

- With direct collocation methods there is no need to explicitly derive the necessary conditions of the continuous problem.
- For a given problem, the region of attraction of the solution is usually larger with direct methods than with indirect methods.
- Moreover, direct collocation methods do not require an a priori specification of the sequence of constrained arcs in problems with inequality path constraints.
Direct Collocation methods for optimal control

- Then the range of problems that can be solved with direct methods is larger than the range of problems that can be solved via indirect methods.
- Direct methods can be used to find a starting guess for an indirect method to improve the accuracy of the solution.
Consider the following optimal control problem:

**Problem 1.** Find all the free functions and variables (the controls $u(t)$, $t \in [t_0, t_f]$, the states $x(t)$, $t \in [t_0, t_f]$, the parameters vector $p$, and the times $t_0$, and $t_f$), to minimise the following performance index,

$$J = \varphi[x(t_0), t_0, x(t_f), t_f] + \int_{t_0}^{t_f} L[x(t), u(t), t] dt$$

subject to:

$$\dot{x}(t) = f[x(t), u(t), p, t], \ t \in [t_0, t_f],$$
Direct Collocation methods for optimal control

Boundary conditions (often called event constraints):

$$
\psi_L \leq \psi[x(t_0), u(t_0), x(t_f), u(t_f), p, t_0, t_f] \leq \psi_U.
$$

Inequality path constraints:

$$
h_L \leq h[x(t), u(t), p, t] \leq h_U, \ t \in [t_0, t_f].
$$
Direct Collocation methods for optimal control

Bound constraints on controls, states and static parameters:

\[ u_L \leq u(t) \leq u_U, \quad t \in [t_0, t_f], \]
\[ x_L \leq x(t) \leq x_U, \quad t \in [t_0, t_f], \]
\[ p_L \leq p \leq p_U. \]

The following constraints allow for cases where the initial and/or final times are not fixed but are to be chosen as part of the solution to the problem:

\[ t_0 \leq t_0 \leq \bar{t}_0, \]
\[ \underline{t}_f \leq t_f \leq \bar{t}_f, \]
\[ t_f - t_0 \geq 0. \]
Direct Collocation methods for optimal control

Consider the continuous domain \([t_0, t_f] \subset \mathcal{R}\), and break this domain using intermediate nodes as follows:

\[ t_0 < t_1 \leq \cdots < t_N = t_f \]

such that the domain has been divided into the \(N\) segments \([t_k, t_{k+1}], j = 0, \ldots, N - 1\).

The figure illustrates the idea that the control \(u\) and state variables \(x\) are discretized over the resulting grid.
Direct Collocation methods for optimal control

By using direct collocation methods, it is possible to discretise Problem 1 to obtain a nonlinear programming problem.

Problem 2. Choose $y \in \mathcal{R}^{ny}$ to minimise:

$$F(y)$$

subject to:

$$\begin{bmatrix} 0 \\ H_L \\ \psi_L \end{bmatrix} \leq \begin{bmatrix} Z(y) \\ H(y) \\ \psi(y) \end{bmatrix} \leq \begin{bmatrix} 0 \\ H_U \\ \psi_U \end{bmatrix}$$

$$y_L \leq y \leq y_U$$
Direct Collocation methods for optimal control

where

- $F$ is simply a mapping resulting from the evaluation of the performance index,
- $Z$ is the mapping that arises from the discretisation of the differential constraints over the gridpoints,
- $H$ arises from the evaluation of the path constraints at each of the grid points, with associated bounds $H_L, H_U$,
- $\Psi$ is simply a mapping resulting from the evaluation of the event constraints.
The form of Problem 2 is the same regardless of the method used to discretize the dynamics.

What may change depending on the discretisation method used is the decision vector $y$ and the differential defect constraints.
Local discretisations

Suppose that the solution of the ODE is approximated by a polynomial \( \tilde{x}(t) \) of degree \( M \) over each interval \( t \in [t_k, t_{k+1}] \), \( k = 0, \ldots, N - 1 \):

\[
\tilde{x}(t) = a_0^{(k)} + a_1^{(k)}(t - t_k) + \cdots + a_M^{(k)}(t - t_k)^M
\]

where the polynomial coefficients \( \{a_0^{(k)}, a_1^{(k)}, \ldots, a_M^{(k)}\} \) are chosen such that the approximation matches the function at the beginning and at the end of the interval:

\[
\tilde{x}(t_k) = x(t_k) \\
\tilde{x}(t_{k+1}) = x(t_{k+1})
\]
Local discretisations

and the time derivative of the approximation matches the time derivative of the function at $t_k$ and $t_{k+1}$:

$$\frac{d\tilde{x}(t_k)}{dt} = f[x(t_k), u(t_k), p, t_k]$$

$$\frac{d\tilde{x}(t_{k+1})}{dt} = f[x(t_{k+1}), u(t_{k+1}), p, t_{k+1}]$$

The last two equations are called *collocation conditions*. 
Local discretisations

The figure illustrates the collocation concept. The solution of the ODE is approximated by a polynomial of degree $M$ over each interval, such that the approximation matches at the beginning and the end of the interval.
Local discretisations

Many of the local discretisations used in practice for solving optimal control problems belong to the class of so-called classical Runge-Kutta methods. A general $K$-stage Runge-Kutta method discretises the differential equation as follows:

$$
x_{k+1} = x_k + h_k \sum_{i=1}^{K} b_i f_{ki}
$$

where

$$
x_{ki} = x_k + h_k \sum_{l=1}^{K} a_{il} f_{kl}
$$

$$
f_{ki} = f[x_{ki}, u_{ki}, p, t_{ki}]
$$

where $u_{ki} = u(t_{ki})$, $t_{ki} = t_k + h_k \rho_i$, and $\rho_i = (0, 1]$. 
Trapezoidal method

The trapezoidal method is based on a quadratic interpolation polynomial:

\[ \ddot{x}(t) = a_{0}^{(k)} + a_{1}^{(k)}(t - t_{k}) + a_{2}^{(k)}(t - t_{k})^{2} \]

Let: \( f_{k} \equiv f[x(t_{k}), u(t_{k}), p, t_{k}] \) and \( f_{k+1} \equiv f[x(t_{k+1}), u(t_{k+1}), p, t_{k+1}] \)

The compressed form of the trapezoidal method is as follows:

\[ \zeta(t_{k}) = x(t_{k+1}) - x(t_{k}) - \frac{h_{k}}{2}(f_{k} + f_{k+1}) = 0 \]

where \( h_{k} = t_{k+1} - t_{k} \), and \( \zeta(t_{k}) = 0 \) is the differential defect constraint at node \( t_{k} \) associated with the trapezoidal method. Allowing \( k = 0, \ldots, N - 1 \), this discretisation generates \( Nn_{x} \) differential defect constraints.
Trapezoidal method

Note that the trapezoidal method is a 2-stage Runge-Kutta method with $a_{11} = a_{12} = 0$, $a_{21} = a_{22} = 1/2$, $b_1 = b_2 = 1/2$, $\rho_1 = 0$, $\rho_2 = 1$. 
Trapezoidal method

Using direct collocation with the compressed trapezoidal discretisation, the decision vector for single phase problems is given by

\[
\mathbf{y} = \begin{bmatrix}
\text{vec}(\mathbf{U}) \\
\text{vec}(\mathbf{X}) \\
\mathbf{p} \\
t_0 \\
t_f
\end{bmatrix}
\]

where \( \text{vec}(\mathbf{U}) \) represents the vectorisation of matrix \( \mathbf{U} \), that is an operator that stacks all columns of the matrix, one below the other, in a single column vector,
Trapezoidal method

The expressions for matrices $U$ and $X$ are as follows:

$$U = [u(t_0) \ u(t_1) \ldots u(t_{N-1})] \in \mathcal{R}^{n_u \times N}$$
$$X = [x(t_0) \ x(t_1) \ldots x(t_N)] \in \mathcal{R}^{n_x \times N+1}$$
The Hermite-Simpson method is based on a cubic interpolating polynomial:

\[ \tilde{x}(t) = a_0^{(k)} + a_1^{(k)}(t - t_k) + a_2^{(k)}(t - t_k)^2 + a_3^{(k)}(t - t_k)^3 \]

Let \( \bar{t}_k = (t_k + t_{k+1})/2 \), and \( \bar{u}(\bar{t}_k) = (u(t_k) + u(t_{k+1}))/2 \).
The compressed Hermite-Simpson discretisation is as follows:

\[ \ddot{x}(\bar{t}_k) = \frac{1}{2} [x(t_k) + x(t_{k+1})] + \frac{h_k}{8} [f_k - f_{k+1}] \]

\[ \bar{f}_k = f[\ddot{x}(\bar{t}_k), \bar{u}(\bar{t}_k), p, \bar{t}_k] \]

\[ \zeta(t_k) = x(t_{k+1}) - x(t_k) - \frac{h_k}{6} \left[ f_k + 4\bar{f}_k + f_{k+1} \right] = 0 \]

where \( \zeta(t_k) = 0 \) is the differential defect constraint at node \( t_k \) associated with the Hermite-Simpson method.
Hermite-Simpson method

Allowing $k = 0, \ldots, N - 1$, the compressed Hermite-Simpson discretisation generates $N n_x$ differential defect constraints. Using direct collocation with the compressed Hermite-Simpson method, the decision vector for single phase problems is given by

$$y = \begin{bmatrix} \text{vec}(U) \\ \text{vec}(\bar{U}) \\ \text{vec}(X) \\ p \\ t_0 \\ t_f \end{bmatrix}$$

where

$$\bar{U} = [u(\bar{t}_0) \ u(\bar{t}_1) \ \ldots \ u(\bar{t}_{N-1})] \in \mathcal{R}^{n_u \times N-1}$$
Hermite-Simpson method

Note that the Hermite-Simpson method is a 3-stage Runge-Kutta method with

\[ a_{11} = a_{12} = a_{13} = 0, \quad a_{21} = 5/24, \quad a_{22} = 1/3, \quad a_{23} = -1/24, \]
\[ a_{31} = 1/6, \quad a_{32} = 2/3, \quad a_{33} = 1/6, \quad b_1 = 1/6, \quad b_2 = 2/3, \quad b_3 = 1/6, \]
\[ \rho_1 = 0, \quad \rho_2 = 1/2, \quad \rho_3 = 1. \]
Local convergence

- The error in variable $z_k \equiv z(t_k)$ with respect to a solution $z^*(t)$ is of order $n$ if there exist $\bar{h}, c > 0$ such that

$$\max_k ||\tilde{z}_k - z^*(t_k)|| \leq ch^n, \forall h < \bar{h}$$

- It is well known that the convergence of the trapezoidal method for integrating differential equations is of order 2, while the convergence of the Hermite-Simpson method for integrating differential equations is of order 4.

- Under certain conditions, these properties more or less translate to the optimal control case, but detailed analysis is beyond the scope of these lectures.
Global discretisations: Pseudospectral methods

- Pseudospectral methods (PS) were originally developed for the solution of partial differential equations that arise in fluid mechanics.
- However, PS techniques have also been used as computational methods for solving optimal control problems.
- While Runge-Kutta methods approximate the derivatives of a function using local information, PS methods are, in contrast, global.
- PS methods are the combination of using global polynomials with orthogonally collocated points.
Global discretisations: Pseudospectral methods

- A function is approximated as a weighted sum of smooth basis functions, which are often chosen to be Legendre or Chebyshev polynomials.

- One of the main appeals of PS methods is their exponential (or spectral) rate of convergence, which is faster than any polynomial rate.
Another advantage is that with relatively coarse grids it is possible to achieve good accuracy.

Approximation theory shows that PS methods are well suited for approximating smooth functions, integrations, and differentiations, all of which are relevant to optimal control problems.
Global discretisations: Pseudospectral methods

PS methods have some disadvantages:

- The resulting NLP is not as sparse as NLP problems derived from Runge-Kutta discretizations.
- This results in less efficient (slower) NLP solutions and possibly difficulties in finding a solution.
- They may not work very well if the solutions of the optimal control problem is not sufficiently smooth, or if their representation require many grid points.
Approximating a continuous function using Legendre polynomials

Let $L_N(\tau)$ denote the Legendre polynomial of order $N$, which may be generated from:

$$L_N(\tau) = \frac{1}{2^N N!} \frac{d^N}{d\tau^N} (\tau^2 - 1)^N$$

Examples of Legendre polynomials are:

$$L_0(\tau) = 1$$
$$L_1(\tau) = \tau$$
$$L_2(\tau) = \frac{1}{2} (3\tau^2 - 1)$$
$$L_3(\tau) = \frac{1}{2} (5\tau^3 - 3\tau)$$
Approximating a continuous function using Legendre polynomials

The figure illustrates the Legendre polynomials $L_N(\tau)$ for $N = 0, \ldots, 3$. 

![Graph showing Legendre polynomials for N=0, 1, 2, and 3.]
Approximating a continuous function using Legendre polynomials

Let $\tau_k, \ k = 0, \ldots, N$ be the Lagrange-Gauss-Lobatto (LGL) nodes, which are defined as $\tau_0 = -1$, $\tau_N = 1$, and $\tau_k$, being the roots of $\dot{L}_N(\tau)$ in the interval $[-1, 1]$ for $k = 1, 2, \ldots, N - 1$.

There are no explicit formulas to compute the roots of $\dot{L}_N(\tau)$, but they can be computed using known numerical algorithms.
Approximating a continuous function using Legendre polynomials

For example, for $N = 20$, the LGL nodes $\tau_k$, $k = 0, \ldots, 12$ shown in the figure.
Approximating a continuous function using Legendre polynomials

Given any real-valued function \( f(\tau) : [-1, 1] \rightarrow \mathbb{R} \), it can be approximated by the Legendre pseudospectral method:

\[
f(\tau) \approx \tilde{f}(\tau) = \sum_{k=0}^{N} f(\tau_k) \phi_k(\tau)
\]

where the Lagrange interpolating polynomial \( \phi_k(\tau) \) is defined by:

\[
\phi_k(\tau) = \frac{1}{N(\tau_2 - 1)} L_N(\tau_k)(\tau_2 - 1) \dot{L}_N(\tau)
\]

It should be noted that \( \phi_k(\tau_j) = 1 \) if \( k = j \) and \( \phi_k(\tau_j) = 0 \), if \( k \neq j \), so that:

\[
\tilde{f}(\tau_k) = f(\tau_k)
\]
Approximating a continuous function using Legendre polynomials

- Regarding the accuracy and error estimates of the Legendre pseudospectral approximation, it is well known that for smooth functions $f(\tau)$, the rate of convergence of $\tilde{f}(\tau)$ to $f(\tau)$ is faster than any power of $1/N$.
- The convergence of the pseudospectral approximations has been analysed by Canuto et al. (2006).
Approximating a continuous function using Legendre polynomials

The figure shows the degree \( N \) interpolation of the function
\[
f(\tau) = \frac{1}{(1 + \tau + 15\tau^2)}
\]
in \((N+1)\) equispaced and LGL points for \( N = 20 \). With increasing \( N \), the errors increase exponentially in the equispaced case (this is known as the Runge phenomenon) whereas in the LGL case they decrease exponentially.
Differentiation with Legendre polynomials

The derivatives of $\tilde{f}(\tau)$ in terms of $f(\tau)$ at the LGL points $\tau_k$ can be obtained by differentiating Eqn. (1). The result can be expressed as follows:

$$\dot{f}(\tau_k) \approx \tilde{f}(\tau_k) = \sum_{i=0}^{N} D_{ki} f(\tau_i)$$

where

$$D_{ki} = \begin{cases} 
\frac{L_N(\tau_k)}{L_N(\tau_i)} \frac{1}{\tau_k - \tau_i} & \text{if } k \neq i \\
-N(N+1)/4 & \text{if } k = i = 0 \\
N(N+1)/4 & \text{if } k = i = N \\
0 & \text{otherwise} 
\end{cases}$$

(4)

which is known as the differentiation matrix.
Differentiation with Legendre polynomials

For example, this is the Legendre differentiation matrix for $N = 5$.

\[
D = \begin{bmatrix}
7.5000 & -10.1414 & 4.0362 & -2.2447 & 1.3499 & -0.5000 \\
1.7864 & 0 & -2.5234 & 1.1528 & -0.6535 & 0.2378 \\
-0.4850 & 1.7213 & 0 & -1.7530 & 0.7864 & -0.2697 \\
0.2697 & -0.7864 & 1.7530 & 0 & -1.7213 & 0.4850 \\
-0.2378 & 0.6535 & -1.1528 & 2.5234 & 0 & -1.7864 \\
0.5000 & -1.3499 & 2.2447 & -4.0362 & 10.1414 & -7.5000
\end{bmatrix}
\]
Differentiation with Legendre polynomials

Legendre differentiation of $f(t) = e^t \sin(5t)$ for $N = 10$ and $N = 20$. Note the vertical scales.
Numerical integration with Legendre polynomials

Note that if \( h(\tau) \) is a polynomial of degree \( \leq 2N - 1 \), its integral over \( \tau \in [-1, 1] \) can be exactly computed as follows:

\[
\int_{-1}^{1} h(\tau) d\tau = \sum_{k=0}^{N} h(\tau_k) w_k
\]

where \( \tau_k, k = 0, \ldots, N \) are the LGL nodes and the weights \( w_k \) are given by:

\[
w_k = \frac{2}{N(N+1)} \frac{1}{[L_N(\tau_k)]^2}, \quad k = 0, \ldots, N.
\]

If \( L(\tau) \) is a general smooth function, then for a suitable \( N \), its integral over \( \tau \in [-1, 1] \) can be approximated as follows:

\[
\int_{-1}^{1} L(\tau) d\tau \approx \sum_{k=0}^{N} L(\tau_k) w_k
\]
Discretising an optimal control problem

Note that PS methods require introducing the transformation:

\[
\tau \leftarrow \frac{2}{t_f - t_0} t - \frac{t_f + t_0}{t_f - t_0},
\]

so that it is possible to write Problem 1 using a new independent variable \(\tau\) in the interval \([-1, 1]\)
Discretising an optimal control problem

After applying the above transformation, Problem 1 can be expressed as follows.

**Problem 3.** Find all the free functions and variables (the controls \( u(\tau), \tau \in [-1, 1] \), the states \( x(\tau), t \in [-1, 1] \), the parameters vector \( p \), and the times \( t_0 \), and \( t_f \)), to minimise the following performance index,

\[
J = \varphi[x(-1), t_0, x(1), t_f] + \int_{-1}^{1} L[x(\tau), u(\tau), t(\tau)] d\tau
\]

subject to:

\[
\dot{x}(\tau) = \frac{t_f - t_0}{2} f[x(\tau), u(\tau), p, t(\tau)], \tau \in [-1, 1],
\]
Discretising an optimal control problem

Boundary conditions:

$$\psi_L \leq \psi[x(-1), u(-1), x(1), u(1), p, t_0, t_f] \leq \psi_U.$$ 

Inequality path constraints:

$$h_L \leq h[x(\tau), u(\tau), p, t(\tau)] \leq h_U, \quad \tau \in [-1, 1].$$
Bound constraints on controls, states and static parameters:

\[
\begin{align*}
    u_L & \leq u(\tau) \leq u_U, \quad \tau \in [-1, 1], \\
    x_L & \leq x(\tau) \leq x_U, \quad \tau \in [-1, 1], \\
    p_L & \leq p \leq p_U.
\end{align*}
\]

And the following constraints:

\[
\begin{align*}
    t_0 & \leq t_0 \leq \bar{t}_0, \\
    t_f & \leq t_f \leq \bar{t}_f, \\
    t_f - t_0 & \geq 0.
\end{align*}
\]
Discretising an optimal control problem

In the Legendre pseudospectral approximation of Problem 3, the state function $x(\tau), \tau \in [-1, 1]$ is approximated by the $N$-order Lagrange polynomial $\tilde{x}(\tau)$ based on interpolation at the Legendre-Gauss-Lobatto (LGL) quadrature nodes, so that:

$$x(\tau) \approx \tilde{x}(\tau) = \sum_{k=0}^{N} x(\tau_k) \phi_k(\tau)$$

(8)
Moreover, the control $u(\tau)$, $\tau \in [-1, 1]$ is similarly approximated using an interpolating polynomial:

$$u(\tau) \approx \tilde{u}(\tau) = \sum_{k=0}^{N} u(\tau_k) \phi_k(\tau)$$

(9)

Note that, from (3), $\tilde{x}(\tau_k) = x(\tau_k)$ and $\tilde{u}(\tau_k) = u(\tau_k)$. 
Discretising an optimal control problem

The derivative of the state vector is approximated as follows:

$$\dot{x}(\tau_k) \approx \dot{x}(\tau_k) = \sum_{i=0}^{N} D_{ki} \ddot{x}(\tau_i), \; i = 0, \ldots, N$$  \hspace{1cm} (10)$$

where $D_{ki}$ is an element of $D$, which is the $(N + 1) \times (N + 1)$ differentiation matrix given by (4).
Define the following $n_u \times (N + 1)$ matrix to store the trajectories of the controls at the LGL nodes:

$$U = [u(\tau_0) \quad u(\tau_1) \quad \ldots \quad u(\tau_N)] \quad \text{(11)}$$
Discretising an optimal control problem

Define the following $n_x \times (N + 1)$ matrices to store, respectively, the trajectories of the states and their approximate derivatives at the LGL nodes:

\[
X = \begin{bmatrix} x(\tau_0) & x(\tau_1) & \cdots & x(\tau_N) \end{bmatrix} \quad (12)
\]

and

\[
\dot{X} = \begin{bmatrix} \dot{x}(\tau_0) & \dot{x}(\tau_1) & \cdots & \dot{x}(\tau_N) \end{bmatrix} \quad (13)
\]
Discretising an optimal control problem

Now, form the following $n_x \times (N + 1)$ matrix with the right hand side of the differential constraints evaluated at the LGL nodes:

$$ F = \frac{t_f - t_0}{2} \left[ f[\tilde{x}(\tau_0), \tilde{u}(\tau_0), p, \tau_0] \ldots f[\tilde{x}(\tau_N), \tilde{u}(\tau_N), p, \tau_N] \right] $$

(14)

Define the differential defects at the collocation points as the $n_x(N + 1)$ vector:

$$ \zeta = \text{vec} \left( \dot{X} - F \right) = \text{vec} \left( XD^T - F \right) $$
Discretising an optimal control problem

The decision vector $y$, which has dimension $n_y = (n_u(N + 1) + n_x(N + 1) + n_p + 2)$, is constructed as follows:

$$y = \begin{bmatrix} \text{vec}(U) \\ \text{vec}(X) \\ p \\ t_0 \\ t_f \end{bmatrix}$$ (15)
Multi-phase problems

- Some optimal control problems can be conveniently formulated as having a number of **phases**.

- Phases may be **inherent to the problem** (for example, a spacecraft drops a section and enters a new phase).

- Phases may also be **introduced by the analyst** to allow for peculiarities in the solution of the problem.

- Many real world problems need a multi-phase approach for their solution.
Multi-phase problems

The figure illustrates the notion of multi-phase problems.
Multi-phase problems

Typically, the formulation for multi-phase problems involve the following elements:

1. a performance index which is the **sum of individual performance indices** for each phase;
2. differential equations and path constraints for each phase.
3. phase linkage constraints which relate the variables at the boundaries between the phases;
4. event constraints which the initial and final states of each phase need to satisfy.
The numerical methods employed by NLP solvers are sensitive to the scaling of variables and constraints.

Scaling may change the convergence rate of an algorithm, the termination tests, and the numerical conditioning of systems of equations that need to be solved.

Modern optimal control codes often perform automatic scaling of variables and constraints, while still allowing the user to manually specify some or all of the scaling factors.
Scaling

- A scaled variable or function is equal to the original value times an scaling factor.
- Scaling factors for controls, states, static parameters, and time, are typically computed on the basis of known bounds for these variables, e.g. for state \( x_j(t) \)

\[
s_{x,j} = \frac{1}{\max(|x_{L,j}|, |x_{U,j}|)}
\]

- For finite bounds, the variables are scaled so that the scaled values are within a given range, for example \([-1, 1]\).
- The scaling factor of each differential defect constraint is often taken to be equal to the scaling factor of the corresponding state:

\[
s_{\zeta,j,k} = s_{x,j}
\]
Scaling

- Scaling factors for all other constraints are often computed as follows. The scaling factor for the \(i\)-th constraint of the NLP problem is the reciprocal of the 2-norm of the \(i\)-th row of the Jacobian of the constraints evaluated at the initial guess.

\[
s_{H,i} = \frac{1}{\| \text{row}(H_y, i) \|_2}
\]

- This is done so that the rows of the Jacobian of the constraints have a 2-norm equal to one.

- The scaling factor for the objective function can be computed as the reciprocal of the 2-norm of the gradient of the objective function evaluated at the initial guess.

\[
s_{F} = \frac{1}{\| F_y \|_2}
\]
Efficient sparse differentiation

- Modern optimal control software uses **sparse finite differences** or sparse **automatic differentiation** libraries to compute the derivatives needed by the NLP solver.
- Sparse finite differences exploit the structure of the Jacobian and Hessian matrices associated with an optimal control problem.
- The individual elements of these matrices can be numerically **computed very efficiently** by perturbing groups of variables, as opposed to perturbing single variables.
Efficient sparse differentiation

- **Automatic differentiation** is based on the application of the chain rule to obtain derivatives of a function given as a numerical computer program.

- Automatic differentiation exploits the fact that computer programs execute a sequence of elementary arithmetic operations or elementary functions.

- By applying the **chain rule of differentiation** repeatedly to these operations, derivatives of arbitrary order can be computed automatically and very accurately.

- Sparsity can also be exploited with automatic differentiation for increased efficiency.
Efficient sparse differentiation

- It is advisable to use, whenever possible, sparse automatic differentiation to compute the required derivatives, as they are more accurate and faster to compute than numerical derivatives.
- When computing numerical derivatives, the structure may be exploited further by separating constant and variable parts of the derivative matrices.
- The achievable sparsity is dependent on the discretisation method employed and on the problem itself.
It is important for optimal control codes implementing direct collocation methods to estimate the discretisation error. This information can be reported back to the user and employed as the basis of a mesh refinement scheme.
Measures of accuracy of the discretisation

Define the error in the differential equation as a function of time:

$$\epsilon(t) = \dot{\tilde{x}}(t) - f[\tilde{x}(t), \tilde{u}(t), p, t]$$

where $\tilde{x}$ is an interpolated value of the state vector given the grid point values of the state vector, $\dot{\tilde{x}}$ is an estimate of the derivative of the state vector given the state vector interpolant, and $\tilde{u}$ is an interpolated value of the control vector given the grid points values of the control vector.
Measures of accuracy of the discretisation

The type of interpolation used depends on the collocation method employed. The absolute local error corresponding to state $i$ on a particular interval $t \in [t_k, t_{k+1}]$, is defined as follows:

$$\eta_{i,k} = \int_{t_k}^{t_{k+1}} |\epsilon_i(t)| \, dt$$

where the integral is computed using an accurate quadrature method.
The local relative ODE error is defined as:

\[ \epsilon_k = \max_i \eta_{i,k} \]

\[ w_i = \max_{k=1}^N [ |\tilde{x}_{i,k}|, |\tilde{x}_{i,k}| ] \]

The error sequence \( \epsilon_k \), or a global measure of the size of the error (such as the maximum of the sequence \( \epsilon_k \)), can be analysed by the user to assess the quality of the discretisation. This information may also be useful to aid the mesh refinement process.
Mesh refinement

- The solution of optimal control problems may involve periods of time where the state and control variables are changing rapidly.
- It makes sense to use shorter discretisation intervals in such periods, while longer intervals are often sufficient when the variables are not changing much.
- This is the main idea behind automatic mesh refinement methods.
Mesh refinement

- Mesh refinement methods detect when a solution needs shorter intervals in regions of the domain by monitoring the accuracy of the discretisation.
- The solution grid is then refined aiming to improve the discretisation accuracy in regions of the domain that require it.
- The NLP problem is then solved again over the new mesh, using the previous solution as an initial guess (after interpolation), and the result is re-evaluated.
Mesh refinement

- Typically several mesh refinement iterations are performed until a given overall accuracy requirement is satisfied.
- With local discretisations, the mesh is typically refined by adding a limited number of nodes to intervals that require more accuracy, so effectively sub-dividing existing intervals. A well known local method was proposed by Betts.
- For pseudospectral methods, techniques have been proposed for segmenting the problem (adding phases) and automatically calculating the number of nodes in each segment.
Mesh refinement

The figure illustrates the mesh refinement process.
Example: vehicle launch problem

- This problem consists of the launch of a space vehicle (Benson, 2004).
- The flight of the vehicle can be divided into four phases, with dry masses ejected from the vehicle at the end of phases 1, 2 and 3.
- The final times of phases 1, 2 and 3 are fixed, while the final time of phase 4 is free.
- It is desired to find the control $u(t)$ and the final time to maximise the mass at the end of phase 4.
Example: vehicle launch problem

The dynamics are given by:

\[
\begin{align*}
\dot{\mathbf{r}} &= \mathbf{v} \\
\dot{\mathbf{v}} &= -\frac{\mu}{\|\mathbf{r}\|^3} \mathbf{r} + \frac{T}{m} \mathbf{u} + \frac{\mathbf{D}}{m} \\
\dot{m} &= -\frac{T}{g_0 I_{sp}}
\end{align*}
\]

where \( \mathbf{r}(t) = [x(t) \ y(t) \ z(t)]^T \) is the position, 
\( \mathbf{v} = [v_x(t) \ v_y(t) \ v_z(t)]^T \) is the Cartesian ECI velocity, \( T \) is the vacuum thrust, \( m \) is the mass, \( g_0 \) is the acceleration of gravity at sea level, \( I_{sp} \) is the specific impulse of the engine, 
\( \mathbf{u} = [u_x \ u_y \ u_z]^T \) is the thrust direction, and 
\( \mathbf{D} = [D_x \ D_y \ D_z]^T \) is the drag force.
Example: vehicle launch problem

The vehicle starts on the ground at rest relative to the Earth at time $t_0$, so that the initial conditions are

\[
\begin{align*}
\mathbf{r}(t_0) &= \mathbf{r}_0 = \begin{bmatrix} 5605.2 & 0 & 3043.4 \end{bmatrix}^T \text{ km} \\
\mathbf{v}(t_0) &= \mathbf{v}_0 = \begin{bmatrix} 0 & 0.4076 & 0 \end{bmatrix}^T \text{ km/s} \\
m(t_0) &= m_0 = 301454 \text{ kg}
\end{align*}
\]
The terminal constraints define the target transfer orbit, which is defined in orbital elements as

\[
\begin{align*}
a_f &= 24361.14 \text{ km}, \\
e_f &= 0.7308, \\
i_f &= 28.5 \text{ deg}, \\
\Omega_f &= 269.8 \text{ deg}, \\
\omega_f &= 130.5 \text{ deg}
\end{align*}
\]
Example: vehicle launch problem

The following linkage constraints force the position and velocity to be continuous and also model the discontinuity in the mass due to the ejections at the end of phases 1, 2 and 3:

\[ \mathbf{r}^{(p)}(t_f) - \mathbf{r}^{(p+1)}(t_0) = 0, \]
\[ \mathbf{v}^{(p)}(t_f) - \mathbf{v}^{(p+1)}(t_0) = 0, \quad (p = 1, \ldots, 3) \]
\[ m^{(p)}(t_f) - m^{(p)}_{\text{dry}} - m^{(p+1)}(t_0) = 0 \]

where the superscript \((p)\) represents the phase number. There is also a path constraint associated with this problem:

\[ ||\mathbf{u}||^2 = u_x^2 + u_y^2 + u_z^2 = 1 \]
Example: vehicle launch problem

- This problem has been solved using the author’s open source optimal control software PSOPT (see http://www.psopt.org).
- Parameter values can be found in the code.
- The problem was solved in eight mesh refinement iterations until the maximum relative ODE error $\epsilon_{\text{max}}$ was below $1 \times 10^{-5}$, starting from a coarse uniform grid and resulting in a finer non-uniform grid.
Example: vehicle launch problem

Figure: Altitude for the vehicle launch problem
Example: vehicle launch problem

Figure: Speed for the vehicle launch problem
Example: vehicle launch problem

Figure: Controls for the vehicle launch problem
Example: Two burn orbit transfer

- The goal of this problem is to compute a trajectory for an spacecraft to go from a standard space shuttle park orbit to a geosynchronous final orbit (Betts, 2010).
- The engines operate over two short periods during the mission.
- It is desired to compute the timing and duration of the burn periods, as well as the instantaneous direction of the thrust during these two periods.
- The objective is to maximise the final weight of the spacecraft.
- The mission then involves four phases: coast, burn, coast and burn.
Example: Two burn orbit transfer

The problem is formulated as follows. Find 
\[ u(t) = [\theta(t), \phi(t)]^T, \quad t \in [t_f^{(1)}, t_f^{(2)}] \text{ and } t \in [t_f^{(3)}, t_f^{(4)}], \]
and the instants \( t_f^{(1)}, t_f^{(2)}, t_f^{(3)}, t_f^{(4)} \) such that the following objective function is minimised:

\[ J = -w(t_f) \]

subject to the dynamic constraints for phases 1 and 3:

\[ \dot{y} = A(y)\Delta_g + b \]

the following dynamic constraints for phases 2 and 4:

\[ \dot{y} = A(y)\Delta + b \]

\[ \dot{w} = -T/I_{sp} \]
Example: Two burn orbit transfer

and the following linkages between phases

\[
y(t_f^{(1)}) = y(t_0^{(2)})
\]
\[
y(t_f^{(2)}) = y(t_0^{(3)})
\]
\[
y(t_f^{(3)}) = y(t_0^{(4)})
\]
\[
t_f^{(1)} = t_0^{(2)}
\]
\[
t_f^{(2)} = t_0^{(3)}
\]
\[
t_f^{(3)} = t_0^{(4)}
\]
\[
w(t_f^{(2)}) = w(t_0^{(4)})
\]

where \( y = [p, f, g, h, k, L, w]^T \) is the vector of modified equinoctial elements, \( w \) is the spacecraft weight, \( l_{sp} \) is the specific impulse of the engine, \( T \) is the maximum thrust.
Example: Two burn orbit transfer

Expressions for $A(y)$ and $b$ are given in (Betts, 2010). The disturbing acceleration is $\Delta = \Delta_g + \Delta_T$, where $\Delta_g$ is the gravitational disturbing acceleration due to the oblatness of Earth, and $\Delta_T$ is the thrust acceleration.

$$\Delta_T = Q_r Q_v \begin{bmatrix} T_a \cos \theta \cos \phi \\ T_a \cos \theta \sin \phi \\ T_a \sin \theta \end{bmatrix}$$

where $T_a(t) = g_0 T/w(t)$, $g_0$ is a constant, $\theta$ is the pitch angle and $\phi$ is the yaw angle of the thrust.
Example: Two burn orbit transfer

Matrix $Q_v$ is given by:

$$Q_v = \begin{bmatrix} \frac{v}{||v||}, \frac{v \times r}{||v \times r||}, \frac{v}{||v||} \times \frac{v \times r}{||v \times r||} \end{bmatrix}$$

Matrix $Q_r$ is given by:

$$Q_r = \begin{bmatrix} i_r & i_\theta & i_h \end{bmatrix} = \begin{bmatrix} \frac{r}{||r||} & \frac{(r \times v) \times r}{||r \times v|| ||r||} & \frac{(r \times v)}{||r \times v||} \end{bmatrix}$$

The results shown were obtained with PSOPT using local discretisations and 5 mesh refinement iterations until the maximum relative ODE error $\epsilon_{\text{max}}$ was below $1 \times 10^{-5}$. 
Example: Two burn orbit transfer
Example: Two burn orbit transfer
Example: Two burn orbit transfer

Mesh refinement statistics

<table>
<thead>
<tr>
<th>Iter</th>
<th>Method</th>
<th>Nodes</th>
<th>NV</th>
<th>NC</th>
<th>OE</th>
<th>CE</th>
<th>JE</th>
<th>HE</th>
<th>RHS</th>
<th>$\epsilon_{\text{max}}$</th>
<th>$\text{CPU}_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TRAPZ</td>
<td>40</td>
<td>308</td>
<td>298</td>
<td>637</td>
<td>636</td>
<td>20</td>
<td>0</td>
<td>48336</td>
<td>4.942e-02</td>
<td>2.400e-01</td>
</tr>
<tr>
<td>2</td>
<td>TRAPZ</td>
<td>56</td>
<td>428</td>
<td>402</td>
<td>25</td>
<td>26</td>
<td>23</td>
<td>0</td>
<td>2808</td>
<td>4.129e-03</td>
<td>3.600e-01</td>
</tr>
<tr>
<td>3</td>
<td>H-S</td>
<td>76</td>
<td>650</td>
<td>532</td>
<td>38</td>
<td>39</td>
<td>37</td>
<td>0</td>
<td>8580</td>
<td>1.568e-04</td>
<td>1.000e+00</td>
</tr>
<tr>
<td>4</td>
<td>H-S</td>
<td>104</td>
<td>888</td>
<td>714</td>
<td>46</td>
<td>47</td>
<td>41</td>
<td>0</td>
<td>14288</td>
<td>2.222e-05</td>
<td>1.590e+00</td>
</tr>
<tr>
<td>5</td>
<td>H-S</td>
<td>133</td>
<td>1132</td>
<td>902</td>
<td>66</td>
<td>67</td>
<td>57</td>
<td>0</td>
<td>26197</td>
<td>8.212e-06</td>
<td>2.880e+00</td>
</tr>
<tr>
<td>CPU$_b$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>812</td>
<td>815</td>
<td>178</td>
<td>0</td>
<td>100209</td>
<td>-</td>
<td>3.786e+01</td>
</tr>
</tbody>
</table>

Key: Iter=iteration number, NV=number of variables, NC=number of constraints, OE=objective evaluations, CE = constraint evaluations, JE = Jacobian evaluations, HE = Hessian evaluations, RHS = ODE right hand side evaluations, $\epsilon_{\text{max}}$ = maximum relative ODE error, $\text{CPU}_a$ = CPU time in seconds spent by NLP algorithm, $\text{CPU}_b$ = additional CPU time in seconds spent by PSOPT
Further reading


