The Direct Transcription Method For Optimal Control

Part 2: Optimal Control

John T. Betts *

*Partner, Applied Mathematical Analysis, LLC

Fundamental Principle of Transcription Methods

- *Transcription Method*

  1. Transcribe a dynamic system into a problem with a finite set of variables
  2. Solve the finite dimensional problem using a parameter optimization method (i.e. the nonlinear programming subproblem)
  3. Assess accuracy of finite dimensional problem, and if necessary repeat transcription and optimization steps

- Purpose: Identify the NLP
  - variables
  - constraints
  - objective function

  for common applications.

Dynamic Systems

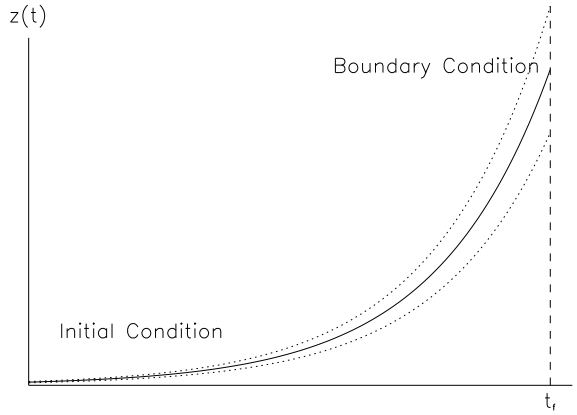
- Dynamics described for $t_0 \leq t \leq t_f$ by a system of n ordinary differential equations:

$$\dot{\mathbf{z}} = \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_n \end{pmatrix} = \begin{pmatrix} f_1[z_1(t), \ldots, z_n(t), t] \\ f_2[z_1(t), \ldots, z_n(t), t] \\ \vdots \\ f_n[z_1(t), \ldots, z_n(t), t] \end{pmatrix} = \mathbf{f}(\mathbf{z}, t)$$

- *Initial Value Problem:* Given the initial value for the dependent variables $\mathbf{z}(t_0)$ determine the values at some other point t_f.

- *Boundary Value Problem:* Determine the dependent variables such that they have specified values at two or more points, say t_0 and t_f.

Example–One Variable



Dynamics: $\frac{dz}{dt} = z$

Analytic solution: $z = z(t_0)e^t$

Problem: Find $z(0) \equiv z_0$ such that $z(t_f) \equiv z_f = b$.

Transcription formulation:

- One Variable; $x \equiv z_0$
- One Constraint; $c(x) = z_f = z_0 e^{t_f} = x e^{t_f} = b$

Shooting Method

- Iterative method
  1. Guess initial conditions $\mathbf{x} = \mathbf{z}(t_0)$;
  2. Propagate differential equations from t_0 to t_f, i.e. “shoot”
  3. Evaluate error in boundary conditions $\mathbf{c}(\mathbf{x}) = \mathbf{z}(t_f) - \mathbf{b}$.
  4. Use NLP to adjust variables $\mathbf{x}$ to satisfy constraints $\mathbf{c}(\mathbf{x}) = \mathbf{0}$, i.e. repeat steps 1-3.
- Advantage — small number of variables
- Disadvantage — small change in initial condition can produce very large change in final conditions (“the tail wagging the dog”).

Example–One Variable

- Reduce sensitivity by breaking problem into two shorter steps, i.e. don't shoot as far

  1. First step: $t_0 \leq t \leq t_1$
  2. Second step: $t_1 \leq t \leq t_f$

  where $t_1 = \frac{1}{2}(t_f + t_0)$

- Guess value of z at midpoint to start second step

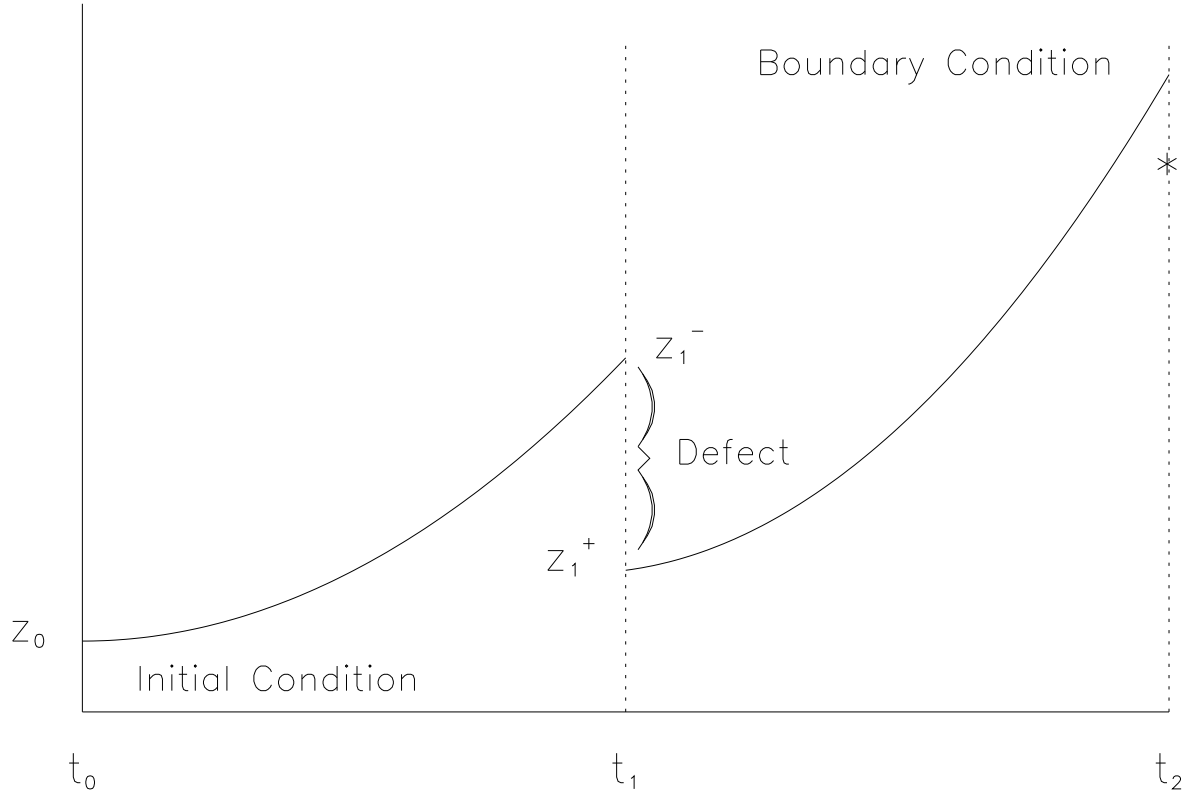
- Add a constraint to force continuity between steps, i.e.

$$z(t_1^-) \equiv z_1^- = z_1^+ \equiv z(t_1^+)$$

- Transcription formulation:
  - Two Variables; $\mathbf{x} \equiv [z_0, z_1^+]$
  - Two Constraints:

$$\begin{bmatrix} c_1(\mathbf{x}) \\ c_2(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} z_1^- - z_1^+ \\ z_f - b \end{bmatrix} = \begin{bmatrix} x_1 e^{(t_1 - t_0)} - x_2 \\ x_2 e^{(t_f - t_1)} - b \end{bmatrix}.$$

Example–(Cont.)



Multiple Shooting Method

- Break interval into N pieces (*segments*)

  $$t_0 \leq t_1 \leq \ldots \leq t_N = t_f$$

  and denote $\mathbf{z}(t_k) \equiv \mathbf{z}_k$

- Iterative method

  1. Guess initial values for $\mathbf{z}$ at all segments; $\mathbf{x} = [\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_N]$;
  2. Propagate differential equations from t_k to t_{k+1}, for all N segments
  3. Evaluate error across all segment boundaries and at final point

     $$\mathbf{c}(\mathbf{x}) = \left[\zeta_1, \zeta_2, \ldots, \zeta_{N-1}, \Psi_f\right]^\top$$

     where

     $$\zeta_k = \mathbf{z}_k^- - \mathbf{z}_k^+$$

     and $\Psi_f \equiv \mathbf{z}_f - \mathbf{b}$.
  4. Use NLP to adjust variables $\mathbf{x}$ to satisfy constraints $\mathbf{c}(\mathbf{x}) = \mathbf{0}$, i.e. repeat steps 1-3.

- Advantage — nonlinearity is not propagated, sensitivities localized

- Disadvantage — large number of variables and constraints

Propagation–One Step Methods

- Shooting and multiple shooting require *propagation*—given the value $\mathbf{z}(t)$, find the value $\mathbf{z}(t+h)$.

- Define $\bar{t} = t + h$ where *integration stepsize* h; Notation:

$$\mathbf{z} \equiv \mathbf{z}(t) \qquad \bar{\mathbf{z}} \equiv \mathbf{z}(t+h) = \mathbf{z}(\bar{t}\,)$$

$$\mathbf{f} \equiv \mathbf{f}[\mathbf{z}(t), t] \qquad \bar{\mathbf{f}} \equiv \mathbf{f}[\mathbf{z}(t+h), t+h] = \mathbf{f}[\bar{\mathbf{z}}, \bar{t}\,]$$

- Propagation over single step:

$$\begin{aligned} \bar{\mathbf{z}} &= \mathbf{z} + \int_t^{\bar{t}} \dot{\mathbf{z}} dt \\ &= \mathbf{z} + \int_t^{\bar{t}} \mathbf{f}(\mathbf{z}, t) dt \end{aligned}$$

- *One step methods* approximate integral using quadrature formula which require evaluation of integrand inside the interval.

k-stage Runge-Kutta Scheme

- Subdivide the integration step into k subintervals

$$t_j = t + h\rho_j$$

with

$$0 \leq \rho_1 \leq \rho_2 \leq \ldots \leq \rho_k \leq 1$$

for $1 \leq j \leq k$.

- Quadrature formula for $\int_t^{\bar{t}} \mathbf{f}\, dt$ gives:

$$\bar{\mathbf{z}} = \mathbf{z} + h \sum_{j=1}^{k} \beta_j \mathbf{f}_j$$

where $\mathbf{f}_j \equiv \mathbf{f}(t_j, \mathbf{z}_j)$ and a quadrature formula for $\int_t^{t_j} \mathbf{f}\, dt$ defines the intermediate points

$$\mathbf{z}_j = \mathbf{z} + h \sum_{\ell=1}^{k} \alpha_{j\ell} \mathbf{f}_\ell$$

for $1 \leq j \leq k$.

k-stage Runge-Kutta Scheme (cont)

- Family of integration schemes defined by coefficients, i.e. *Butcher Array*

$$\begin{array}{c|ccc} \rho_1 & \alpha_{11} & \dots & \alpha_{1k} \\ \vdots & \vdots & & \vdots \\ \rho_k & \alpha_{k1} & \dots & \alpha_{kk} \\ \hline & \beta_1 & \dots & \beta_k \end{array}$$

- *Explicit Methods:*
  - $\rho_1 = 0$ and $\alpha_{jl} = 0$ for $j \leq l$ (α-block is lower triangular)
  - $\bar{\mathbf{z}}$ can be computed explicitly from values at t and intermediate points.
- *Implicit Methods:*
  - $\bar{\mathbf{z}}$ cannot be computed explicitly from values at t and intermediate points.
  - $\bar{\mathbf{z}}$ and $\mathbf{z}$ appear implicitly in nonlinear quadrature formula
  - Computation of $\bar{\mathbf{z}}$ requires
    - an initial guess (the *predictor*)
    - an iterative solution (the *corrector*)
    - to drive the discretization "defect" to zero.
- Boundary Value problem is inherently implicit

Eulers Method

- Explicit, $k = 1$

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

- Common representation:

$$\bar{\mathbf{z}} = \mathbf{z} + h\mathbf{f}$$

Classical Runge-Kutta Method

- Explicit, $k = 4$

0	0	0	0	0
$1/2$	$1/2$	0	0	0
$1/2$	0	$1/2$	0	0
1	0	0	1	0
	$1/6$	$1/3$	$1/3$	$1/6$

- Common representation:

$$
\begin{aligned}
\mathbf{k}_1 &= h\mathbf{f} \\
\mathbf{k}_2 &= h\mathbf{f}\left(\mathbf{z}+\frac{1}{2}\mathbf{k}_1, t+\frac{h}{2}\right) \\
\mathbf{k}_3 &= h\mathbf{f}\left(\mathbf{z}+\frac{1}{2}\mathbf{k}_2, t+\frac{h}{2}\right) \\
\mathbf{k}_4 &= h\mathbf{f}(\mathbf{z}+\mathbf{k}_3, \bar{t}\,) \\
\bar{\mathbf{z}} &= \mathbf{z}+\frac{h}{6}(\mathbf{k}_1+2\mathbf{k}_2+2\mathbf{k}_3+\mathbf{k}_4)
\end{aligned}
$$

Trapezoidal Method

- Implicit, $k = 2$

0	0	0
1	$1/2$	$1/2$
	$1/2$	$1/2$

- Common representation:

$$\bar{\mathbf{z}} = \mathbf{z} + \frac{h}{2}\left(\mathbf{f} + \bar{\mathbf{f}}\right)$$

Hermite-Simpson Method

- Implicit, $k = 3$

0	0	0	0
$1/2$	$5/24$	$1/3$	$-1/24$
1	$1/6$	$2/3$	$1/6$
	$1/6$	$2/3$	$1/6$

- Common representation:

$$\begin{aligned}
\widetilde{\mathbf{z}} &= \frac{1}{2}(\mathbf{z}+\overline{\mathbf{z}}) + \frac{h}{8}(\mathbf{f}-\bar{\mathbf{f}}) \\
\widetilde{\mathbf{f}} &= \mathbf{f}\left(\widetilde{\mathbf{z}}, t+\frac{h}{2}\right) \\
\overline{\mathbf{z}} &= \mathbf{z}+\frac{h}{6}\left(\mathbf{f}+4\widetilde{\mathbf{f}}+\bar{\mathbf{f}}\right)
\end{aligned}$$

Boundary Value Example Problem

- Golf Ball Dynamics*:

$$\ddot{\mathbf{z}} = -g\mathbf{k} + gn_3\mathbf{n} - \mu_k gn_3 \frac{\dot{\mathbf{z}}}{\|\dot{\mathbf{z}}\|}$$

where geometry of the green is

$$z_3 = S(z_1, z_2) = \frac{(z_1 - 10)^2}{125} + \frac{(z_2 - 5)^2}{125} - 1$$

with

$$\mathbf{n} = \frac{\mathbf{N}}{\|\mathbf{N}\|} \qquad\qquad \mathbf{N} = \left(-\frac{\partial S}{\partial z_1}, -\frac{\partial S}{\partial z_2}, 1\right)$$

- The Putting Problem:
  - Choose (4) *Variables* $\dot{\mathbf{z}}(0)$ and t_f
  - Subject to (4) *Constraints*

$$\begin{aligned} \mathbf{z}(t_f) &= \mathbf{z}_H \\ \|\dot{\mathbf{z}}\| &= 0 \end{aligned}$$

given $\mathbf{z}(0) = (0,0,0)$ and $\mathbf{z}_H = (20,0,0)$.

*A Motivational Example for the Numerical Solution of Two-Point Boundary-Value Problems, S. Alessandrini, SIAM Review, Vol 37, No. 3., Sept. 1995.

Boundary Value Example Problem (cont)

- *Ill-posed Problem!* At desired boundary $\|\dot{\mathbf{z}}\| = 0$ the friction force term $\mu_k g n_3 \frac{\dot{\mathbf{z}}}{\|\dot{\mathbf{z}}\|}$ is not defined!

- Author suggests "realistic" problem if $\|\dot{\mathbf{z}}\| \leq s_T$
  - What is s_T?
  - Solution is not unique! How does iteration terminate?

- **The Real Putting Problem:** Two regions
  - On the green: $\|\mathbf{z}(t) - \mathbf{z}_H\| \geq R_H$ (dynamics as above)
  - In the hole: $\|\mathbf{z}(t) - \mathbf{z}_H\| \leq R_H$ where dynamics have

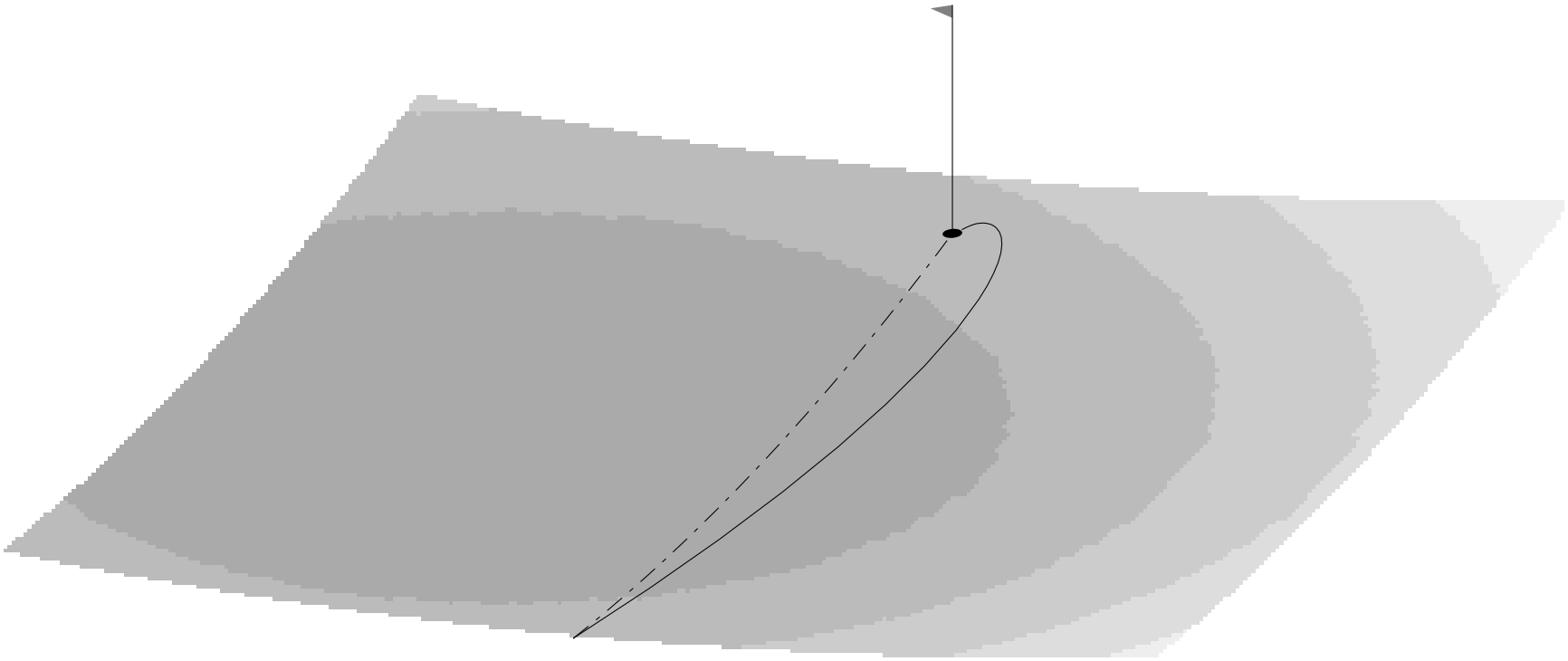
$$g n_3 \mathbf{n} - \mu_k g n_3 \frac{\dot{\mathbf{z}}}{\|\dot{\mathbf{z}}\|} = 0$$

  - Choose (5) *Variables* $\dot{\mathbf{z}}(0)$, t_f and t_1
  - Subject to (2) *Constraints*

$$\begin{aligned} \|\mathbf{z}(t_1) - \mathbf{z}_H\| &= R_H \\ \mathbf{z}(t_f) - \mathbf{z}_H &\leq R_H \end{aligned}$$

  - Minimize final horizontal velocity $\dot{z}_1^2 + \dot{z}_2^2$

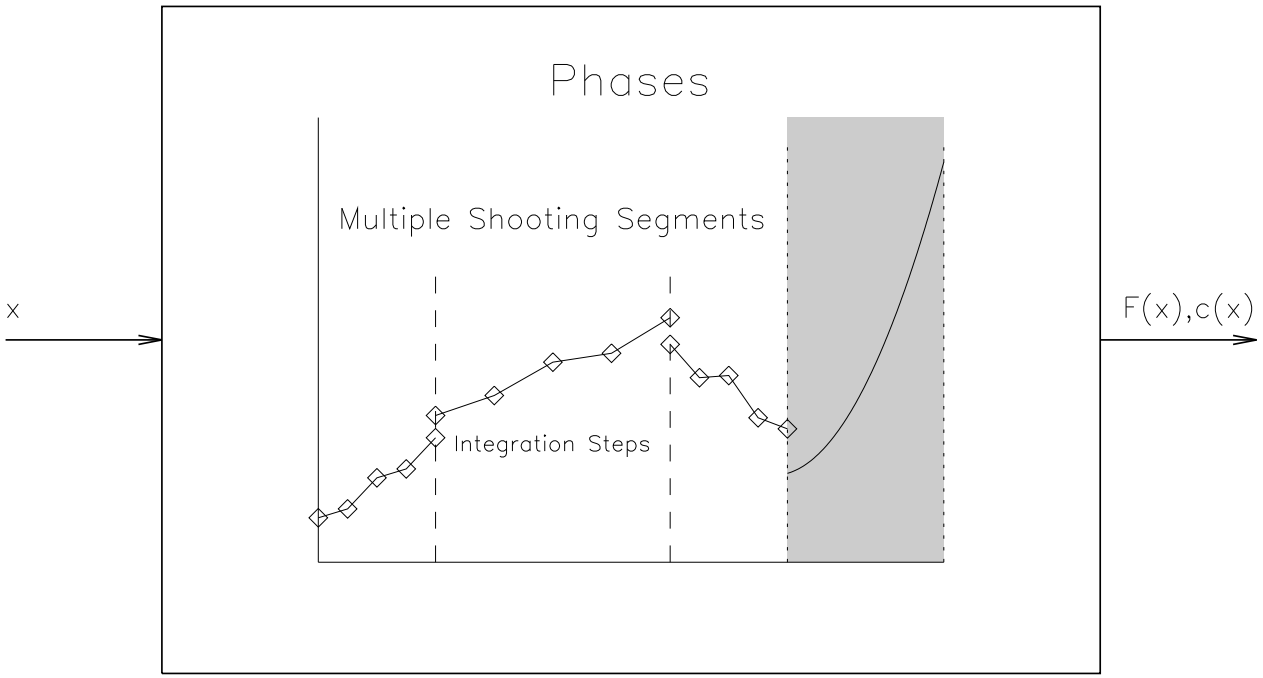
Putting Problem



Dynamic Modeling Concepts

- A trajectory is made up of a collection of *phases*
- The set of differential equations cannot change during a phase; conversely different sets of differential equations can be used in different phases.
- Phases are *linked* together by *linkage conditions* to construct complete problem description
- Phases terminate at *events*.
- Multiple shooting segments may (or may not) be treated as phases.
- Spectrum of possiblities:
  - (One multiple shooting segment) and (Multiple number of integration steps) $\Rightarrow$ Shooting method
  - (Limited number of multiple shooting segments, e.g. 5) and (Multiple number of integration steps per segment)
  - (Number of multiple shooting segments) $=$ (Number of integration steps)

Function Generator



NLP Considerations

- **Goal:** Construct a function generator, i.e. select a transcription method, which is noise free.
- Consistency vs. Accuracy
  - A *consistent* function generator executes the same sequence of arithmetic operations for all values of $\mathbf{x}$.
  - An *accurate* function generator computes accurate approximations to the dynamics $\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, t)$.
- Adaptive quadrature (e.g. variable order and/or stepsize) is "noise" to the NLP—-it is not consistent!
- A fixed number of steps with an explicit integration method is consistent, but not accurate.
- **Achieving the Goal:**
  - Select transcription with one integration step per multiple shooting segment $\Longrightarrow$ *consistent*
  - Treat accuracy outside the NLP

Discretize Then Optimize

Optimal Control Problem

- Find the n_u-dimensional control vector $\mathbf{u}(t)$ to minimize the performance index

$$\phi[\mathbf{y}(t_f), t_f]$$

evaluated at the final time t_f and satisfy the state equations

$$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}(t), \mathbf{u}(t), t]$$

where the n_y dimension state vector $\mathbf{y}$ may have some specified initial and terminal values.

- *Autonomous Form* when $\mathbf{f}[\mathbf{y}(t), \mathbf{u}(t), t] = \mathbf{f}[\mathbf{y}(t), \mathbf{u}(t)]$

- Transcribe to finite dimensional NLP
  - NLP optimization variables

$$\mathbf{x} = (\mathbf{u}_1, \mathbf{y}_2, \mathbf{u}_2, \ldots, \mathbf{y}_M, \mathbf{u}_M)$$

  - NLP constraints (Eulers Method)

$$c_k(\mathbf{x}) = \mathbf{y}_{k+1} - \mathbf{y}_k - h\mathbf{f}(\mathbf{y}_k, \mathbf{u}_k)$$

    for $k = 1, \ldots, M-1$, with $\mathbf{y}_0$ and $\mathbf{u}_0$ given, and $h = t_f/M$
  - NLP objective function – minimize

$$\phi(\mathbf{x}) = \phi(\mathbf{y}_M)$$

Necessary Conditions for Discrete Problem

- Define the *Lagrangian*

$$\begin{aligned} L(\mathbf{x},\lambda) &= \phi(\mathbf{x}) - \lambda^{\mathsf{T}}\mathbf{c}(\mathbf{x}) \\ &= \phi(\mathbf{y}_M) - \sum_{k=1}^{M-1} \lambda_k^{\top}\left[\mathbf{y}_{k+1} - \mathbf{y}_k - h\mathbf{f}(\mathbf{y}_k,\mathbf{u}_k)\right] \end{aligned}$$

- Necessary Conditions

$$\frac{\partial L}{\partial \lambda_k} = \mathbf{y}_{k+1} - \mathbf{y}_k - h\mathbf{f}(\mathbf{y}_k,\mathbf{u}_k) = 0$$

$$\frac{\partial L}{\partial \mathbf{u}_k} = h\lambda_k^{\top}\frac{\partial \mathbf{f}}{\partial \mathbf{u}_k} = 0$$

$$\frac{\partial L}{\partial \mathbf{y}_k} = (\lambda_k - \lambda_{k-1}) + h\lambda_k^{\top}\frac{\partial \mathbf{f}}{\partial \mathbf{y}_k} = 0$$

$$\frac{\partial L}{\partial \mathbf{y}_M} = -\lambda_M + \frac{\partial \phi}{\partial \mathbf{y}_M} = 0$$

Necessary Conditions for Limiting Form of Discrete Problem

- Let $M \to \infty$, $h \to 0$
- *State Equations*

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, \mathbf{u})$$

- *Pontryagin Maximum Principle*

$$\frac{\partial H}{\partial \mathbf{u}} = \boldsymbol{\lambda}^{\top} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} = 0$$

- *Adjoint Equations*

$$\dot{\boldsymbol{\lambda}} = -\boldsymbol{\lambda}^{\top} \frac{\partial \mathbf{f}}{\partial \mathbf{y}}$$

- *Transversality Condition*

$$\boldsymbol{\lambda}(t_f) - \frac{\partial \phi}{\partial \mathbf{y}} \Big|_{t=t_f} = 0$$

Direct vs Indirect Methods

Indirect Construct and solve adjoint equations, Maximum principle, and boundary conditions.

- Maximum Principle is
  - an algebraic side condition, (possibly implicit)
  - (state + adjoint + maximum principle) $\Rightarrow$ differential-algebraic system (DAE)
  - an NLP at each time step $\Rightarrow$ choose controls $\mathbf{u}(t_k)$ to maximize the *Hamiltonian* $H = \lambda^{\mathsf{T}}\mathbf{f}$
- Solution via shooting, multiple shooting, or discretization
- *Pros:* Accurate solution for "special cases", e.g. singular arcs
- *Cons:* Not general (requires derivation and implementation of adjoint equations); Not robust

Direct Directly optimize the objective without formation of the necessary conditions (adjoints, transversality, etc.)

- Control Parameterization
- State and Control Parameterization
- *Pros:* Very robust and general
- *Cons:* Special treatment for "special cases"

Relationship Between Continuous and Discrete Problem

Discrete		Continuous	
$(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M)$	NLP Variables	$\mathbf{y}(t)$	State Variables
$(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M)$	NLP Variables	$\mathbf{u}(t)$	Control Variables
$\zeta_k = 0$	Defect Constraints	$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, \mathbf{u}, t)$	State Equations
$(\lambda_1, \lambda_2, \ldots, \lambda_M)$	Lagrange Multipliers	$\lambda(t)$	Adjoint Variables

General Formulation

- Formulate problem as collection of N *phases*, where

$$t_0^{(k)} \leq t \leq t_f^{(k)}.$$

- *Dynamic variables*

$$\mathbf{z}^{(k)} = \begin{bmatrix} \mathbf{y}^{(k)}(t) \\ \mathbf{u}^{(k)}(t) \end{bmatrix}$$

made of *state variables* $\mathbf{y}^{(k)}(t)$ and *control variables* $\mathbf{u}^{(k)}(t)$.

- *Parameters* $\mathbf{p}^{(k)}$ independent of t.

- *State equations*

$$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t]$$

- *Boundary conditions*

$$\begin{aligned} \boldsymbol{\psi}_{0\ell} &\leq \boldsymbol{\psi}[\mathbf{y}(t_0), \mathbf{u}(t_0), \mathbf{p}, t_0] \leq \boldsymbol{\psi}_{0u}, \\ \boldsymbol{\psi}_{f\ell} &\leq \boldsymbol{\psi}[\mathbf{y}(t_f), \mathbf{u}(t_f), \mathbf{p}, t_f] \leq \boldsymbol{\psi}_{fu}, \end{aligned}$$

General Formulation (Cont.)

- *Algebraic path constraints*

$$\mathbf{g}_\ell \leq \mathbf{g}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t] \leq \mathbf{g}_u,$$

- *Variable bounds*

$$\begin{aligned} \mathbf{y}_\ell &\leq \mathbf{y}(t) \leq \mathbf{y}_u, \\ \mathbf{u}_\ell &\leq \mathbf{u}(t) \leq \mathbf{u}_u, \\ \mathbf{p}_\ell &\leq \mathbf{p} \leq \mathbf{p}_u. \end{aligned}$$

- *Optimal Control Problem:* Find dynamic variables $\mathbf{z}$ and parameters $\mathbf{p}$ subject to constraints and bounds, which minimize

$$J = \phi \left[\mathbf{y}(t_0^{(1)}), t_0^{(1)}, \mathbf{y}(t_f^{(1)}), \mathbf{p}^{(1)}, t_f^{(1)}, \ldots, \mathbf{y}(t_0^{(N)}), t_0^{(N)}, \mathbf{y}(t_f^{(N)}), \mathbf{p}^{(N)}, t_f^{(N)} \right].$$

Direct Transcription Formulation

- Discretization: M grid points; stepsize $h_k \equiv t_{k+1} - t_k$

$$\begin{aligned} \mathbf{y}_k &\equiv \mathbf{y}(t_k) & \qquad \bar{\mathbf{u}}_{k+1} &\equiv \mathbf{u}[(t_{k+1}+t_k)/2] \\ \mathbf{u}_k &\equiv \mathbf{u}(t_k) & \qquad \mathbf{f}_k &\equiv \mathbf{f}[\mathbf{y}(t_k), \mathbf{u}(t_k), \mathbf{p}, t_k] \end{aligned}$$

- Trapezoidal
  - NLP variables

$$\mathbf{x} = [\mathbf{y}_1, \mathbf{u}_1, \mathbf{y}_2, \mathbf{u}_2, \ldots, \mathbf{y}_M, \mathbf{u}_M, \mathbf{p}, t_1, t_M]^\top$$

  - Defect constraints for $k = 1, \ldots, M-1$

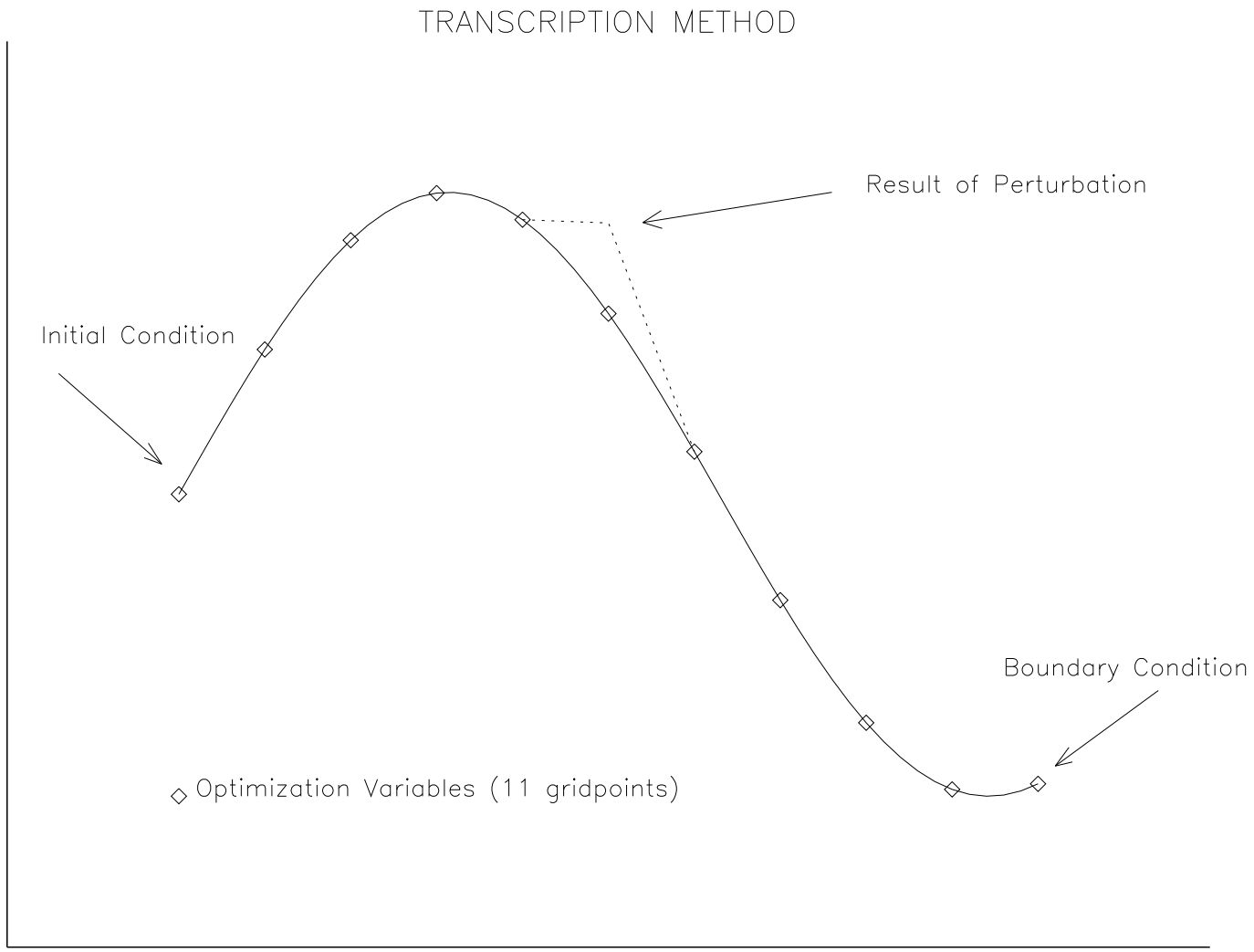
$$\zeta_k = \mathbf{y}_{k+1} - \mathbf{y}_k - \frac{h_k}{2}[\mathbf{f}_{k+1} + \mathbf{f}_k] = 0$$

- Hermite-Simpson (Compressed)
  - NLP variables

$$\mathbf{x} = [\mathbf{y}_1, \mathbf{u}_1, \bar{\mathbf{u}}_2, \mathbf{y}_2, \mathbf{u}_2, \bar{\mathbf{u}}_3, \ldots, \mathbf{y}_M, \mathbf{u}_M, \mathbf{p}, t_1, t_M]^\top$$

  - Defect constraints for $k = 1, \ldots, M-1$

$$\zeta_k = \mathbf{y}_{k+1} - \mathbf{y}_k - \frac{h_k}{6}\left[\mathbf{f}_{k+1} + 4\bar{\mathbf{f}}_{k+1} + \mathbf{f}_k\right] = 0$$


TRANSCRIPTION METHOD
Result of Perturbation
Initial Condition
Boundary Condition
Optimization Variables (11 gridpoints)

NLP Considerations — Sparsity

- Changing a variable at gridpoint only alters nearby constraints $\Longrightarrow$ the derivatives of many of the constraints with respect to many of the variables are zero.

- The matrix of partial derivatives, i.e. the Jacobian matrix is sparse.

- The Jacobian matrix is defined as

$$\mathbf{G}_{ij} = \frac{\text{Change in Defect Constraint on Segment i}}{\text{Change in Optimization Variable at Grid Point j}}$$

  with m rows (the total number of defect constraints) and n columns (the total number of optimization variables)

- Reduced Sensitivity in Boundary Value Problem $\Longleftrightarrow$ Matrix Sparsity

- *Sparse Differences* used to compute $\mathbf{G}$ and $\mathbf{H}_L$
  - Number of index sets $\gamma \approx$ number of dynamic variables $\mathbf{z}$; not number of grid points M
  - $\mathbf{G}$ and $\mathbf{H}_L$ computed with $\gamma(\gamma+3)/2$ perturbations of function generator.

Standard Approach

- Example Problem: (4 states, 1 control, $M = 10$)

$$\begin{aligned} \dot{y}_1 &= y_3 \\ \dot{y}_2 &= y_4 \\ \dot{y}_3 &= a\cos u \\ \dot{y}_4 &= a\sin u \end{aligned}$$

- NLP variables: $\mathbf{x} = (t_f, \mathbf{y}_1, \mathbf{u}_1, \ldots, \mathbf{y}_M, \mathbf{u}_M)$
- Approximate ODE

$$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}, \mathbf{u}, t]$$

  with NLP constraints:

$$\mathbf{0} = \mathbf{y}_{k+1} - \mathbf{y}_k - \frac{h_k}{2}[\mathbf{f}_{k+1} + \mathbf{f}_k] \doteq \mathbf{c}(\mathbf{x})$$

  for $k = 1, \ldots, M-1$ (Trapezoidal Method).
- NLP Jacobian:

$$\mathbf{G} \doteq \frac{\partial \mathbf{c}}{\partial \mathbf{x}}$$

- Standard Approach: $\mathbf{c} = \mathbf{q}$ and $\mathbf{G} = \mathbf{D}$
  - Index sets: $\gamma = 2 * (n_y + n_u) + 1 = 11$
  - Variables: $n = M(n_y + n_u) + 1 = 51$
  - Constraints: $m = (M-1)n_y = 36$

Discretization Separability

- Group terms by grid point

$$\mathbf{0} = \mathbf{y}_{k+1} - \mathbf{y}_k - \frac{h_k}{2}\left[\mathbf{f}_{k+1} + \mathbf{f}_k\right] = \left[\mathbf{y}_{k+1} - \frac{h_k}{2}\mathbf{f}_{k+1}\right] + \left[-\mathbf{y}_k - \frac{h_k}{2}\mathbf{f}_k\right]$$

- NLP constraints: $\mathbf{c}(\mathbf{x}) = \mathbf{B}\mathbf{q}(\mathbf{x})$ where

$$\mathbf{B} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & & & & \\ & & \mathbf{I} & \mathbf{I} & & \\ & & & & \ddots & \\ & & & & \mathbf{I} & \mathbf{I} \end{bmatrix} \qquad \mathbf{q}(\mathbf{x}) = \begin{bmatrix} -\mathbf{y}_1 - \frac{h_1}{2}\mathbf{f}_1 \\ \mathbf{y}_2 - \frac{h_1}{2}\mathbf{f}_2 \\ \vdots \\ -\mathbf{y}_{M-1} - \frac{h_{M-1}}{2}\mathbf{f}_{M-1} \\ \mathbf{y}_M - \frac{h_{M-1}}{2}\mathbf{f}_M \end{bmatrix}$$

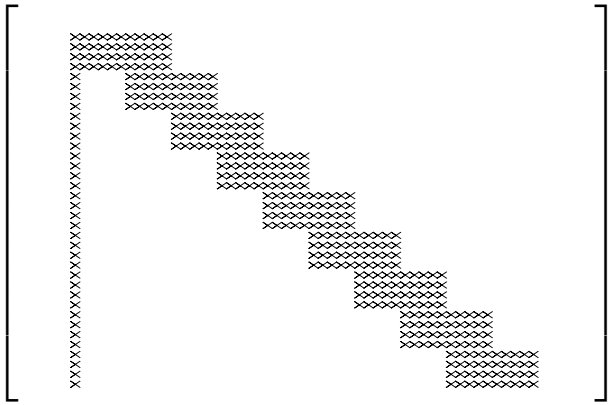
- Construct sparse difference estimates for the matrix

$$\mathbf{D} \doteq \frac{\partial \mathbf{q}}{\partial \mathbf{x}}$$

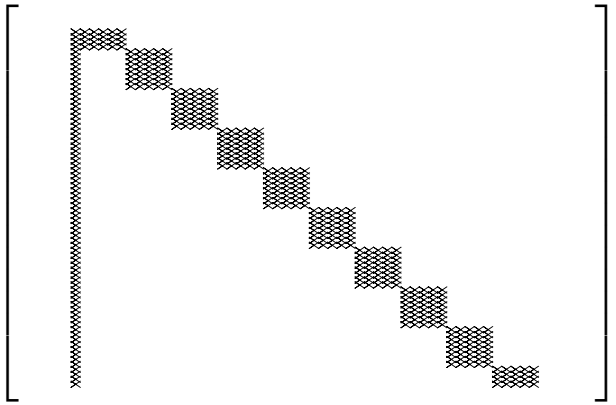
- NLP Jacobian:

$$\mathbf{G} \doteq \frac{\partial \mathbf{c}}{\partial \mathbf{x}} = \mathbf{B}\mathbf{D}$$

Discretization Separability

$\mathbf{G} =$ 

$\gamma = 11$

$\mathbf{D} =$ 

$\gamma = 6$

Right Hand Side Sparsity

- Group terms by grid point and isolate linear terms

$$\begin{aligned}\mathbf{0} &= \mathbf{y}_{k+1} - \mathbf{y}_k - \frac{h_k}{2}\left[\mathbf{f}_{k+1} + \mathbf{f}_k\right] \\ &= \left[\mathbf{y}_{k+1} - \mathbf{y}_k\right] - \frac{1}{2}\tau_k\left[t_f\mathbf{f}_{k+1}\right] - \frac{1}{2}\tau_k\left[t_f\mathbf{f}_k\right]\end{aligned}$$

  where $h_k = \tau_k(t_f - t_0) = \tau_k t_f$ with $0 < \tau_k < 1$

- NLP constraints: $\mathbf{c}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{q}(\mathbf{x})$ where

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{I} & & \\ & & -\mathbf{I} & \mathbf{0} & \mathbf{I} & \\ \vdots & & & & & \ddots \end{bmatrix} \qquad \mathbf{B} = -\tfrac{1}{2}\begin{bmatrix} \tau_1\mathbf{I} & \tau_1\mathbf{I} & & & & \\ & & \tau_2\mathbf{I} & \tau_2\mathbf{I} & & \\ & & & & \ddots & \\ & & & & \tau_{M-1}\mathbf{I} & \tau_{M-1}\mathbf{I} \end{bmatrix}$$

  and

$$\mathbf{q} = \begin{bmatrix} t_f\mathbf{f}_1 \\ t_f\mathbf{f}_2 \\ \vdots \\ t_f\mathbf{f}_M \end{bmatrix}$$

Right Hand Side Sparsity (cont.)

- Right hand side sparsity template:

$$\left[\frac{\partial \mathbf{f}}{\partial \mathbf{y}} \middle| \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right] = \left[\begin{array}{cccc|c} . & . & x & . & . \\ . & . & . & x & . \\ . & . & . & . & x \\ . & . & . & . & x \end{array}\right]$$

- Construct right hand side template using *random* perturbations about *random* nominal points.

- Construct sparse difference estimates for the matrix

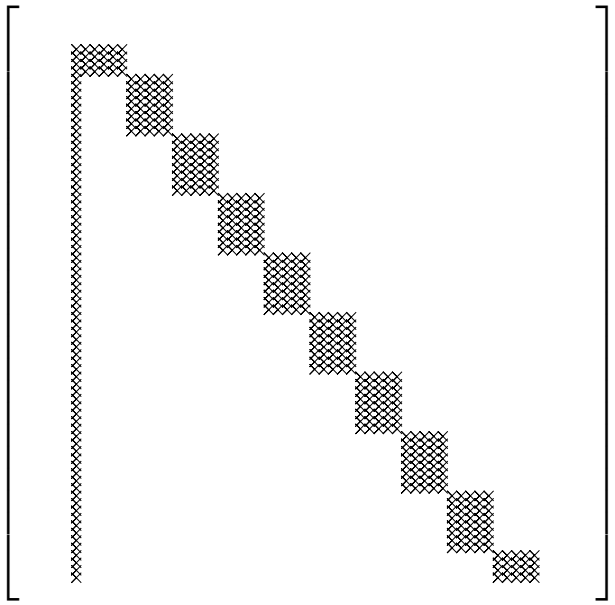
$$\mathbf{D} \doteq \frac{\partial \mathbf{q}}{\partial \mathbf{x}}$$

  using sparse right hand sides

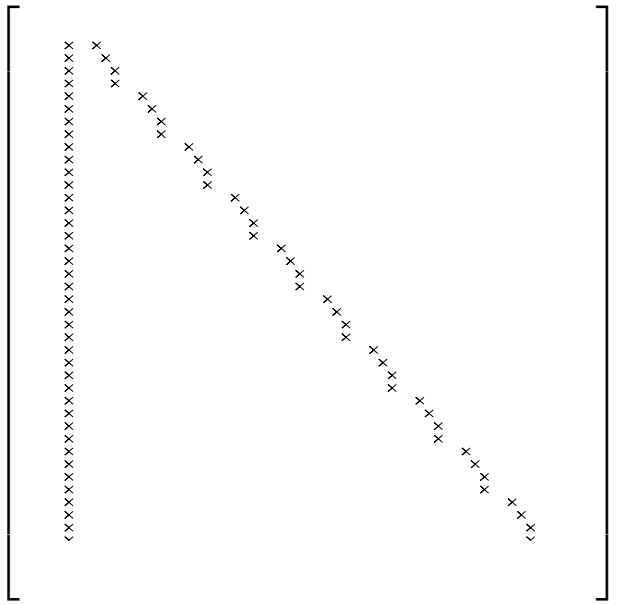
- NLP Jacobian:

$$\mathbf{G} = \mathbf{A} + \mathbf{B}\mathbf{D}$$

Right Hand Side Sparsity

$\mathbf{D} =$ 

$\gamma = 6$

$\mathbf{D} =$ 

$\gamma = 2$

Hermite-Simpson (Compressed)

- Hermite-Simpson defect constraints ($h_k = \tau_k t_f$)

$$\mathbf{0} = \mathbf{y}_{k+1} - \mathbf{y}_k - \frac{h_k}{6}\left[\mathbf{f}_{k+1} + 4\bar{\mathbf{f}}_{k+1} + \mathbf{f}_k\right]$$

  where

$$\begin{aligned} \bar{\mathbf{f}}_{k+1} &= \mathbf{f}\left[\bar{\mathbf{y}}_{k+1}, \bar{\mathbf{u}}_{k+1}, t_k + \frac{h_k}{2}\right] \\ \bar{\mathbf{y}}_{k+1} &= \frac{1}{2}(\mathbf{y}_{k+1} + \mathbf{y}_k) + \frac{h_k}{8}(\mathbf{f}_k - \mathbf{f}_{k+1}) \end{aligned}$$

- NLP variables: $\mathbf{x} = (t_f, \mathbf{y}_1, \mathbf{u}_1, \boxed{\bar{\mathbf{u}}_2}, \ldots, \boxed{\bar{\mathbf{u}}_M}, \mathbf{y}_M, \mathbf{u}_M)$

- NLP constraints:

$$\mathbf{c}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{q}(\mathbf{x})$$

  where

$$\mathbf{q} \doteq \begin{bmatrix} t_f\left(\mathbf{f}_2 + 4\bar{\mathbf{f}}_2 + \mathbf{f}_1\right) \\ t_f\left(\mathbf{f}_3 + 4\bar{\mathbf{f}}_3 + \mathbf{f}_2\right) \\ \vdots \\ t_f\left(\mathbf{f}_M + 4\bar{\mathbf{f}}_M + \mathbf{f}_{M-1}\right) \end{bmatrix}$$

Hermite-Simpson (Compressed) Sparsity

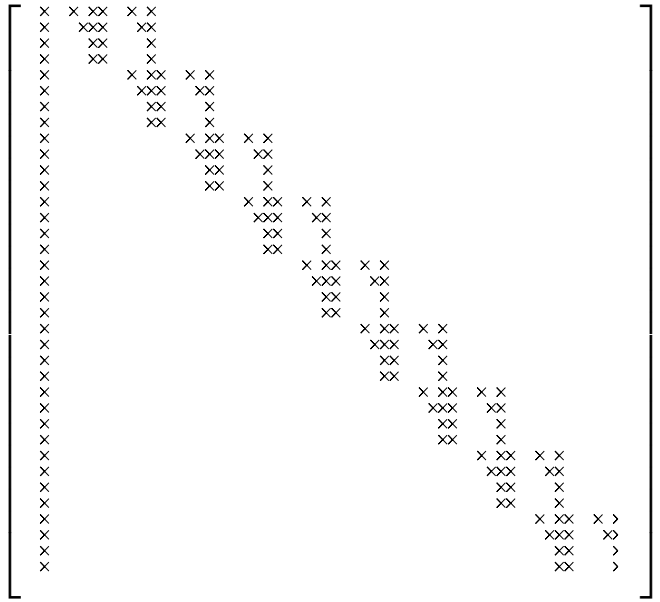
Sparsity template:

$$\left[\frac{\partial \mathbf{v}}{\partial \mathbf{y}} \middle| \frac{\partial \mathbf{v}}{\partial \mathbf{u}}\right] = \left[\begin{array}{cccc|c} . & . & x & . & x \\ . & . & . & x & x \\ . & . & . & . & x \\ . & . & . & . & x \end{array}\right]$$

where

$$\mathbf{v} = t_f \left(\mathbf{f}_{k+1} + 4\bar{\mathbf{f}}_{k+1} + \mathbf{f}_k\right)$$

$$\mathbf{D} =$$



$\gamma = 6$

Hermite-Simpson (Separated)

- Discretization constraints without local compression

$$\mathbf{0} = \bar{\mathbf{y}}_{k+1} - \frac{1}{2}(\mathbf{y}_{k+1} + \mathbf{y}_k) - \frac{\tau_k t_f}{8}(\mathbf{f}_k - \mathbf{f}_{k+1}) \qquad \textit{Hermite}$$

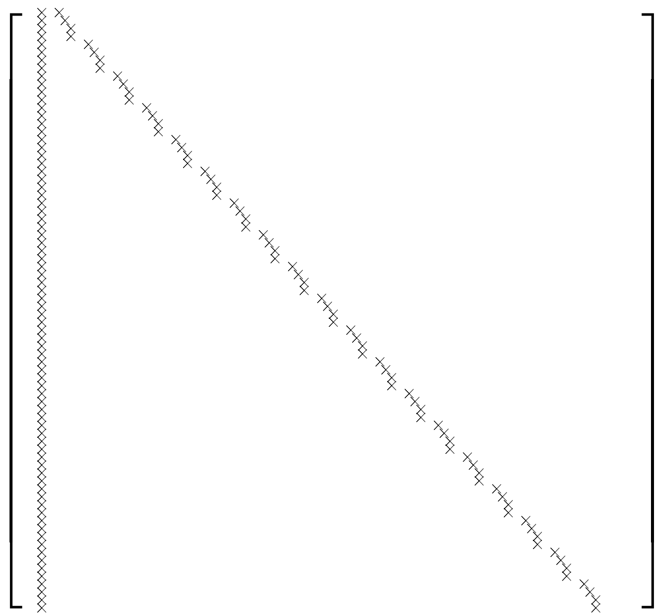
$$\mathbf{0} = \mathbf{y}_{k+1} - \mathbf{y}_k - \frac{\tau_k t_f}{6}\left[\mathbf{f}_{k+1} + 4\bar{\mathbf{f}}_{k+1} + \mathbf{f}_k\right] \qquad \textit{Simpson}$$

- NLP variables: $\mathbf{x} = (t_f, \mathbf{y}_1, \mathbf{u}_1, \boxed{\bar{\mathbf{y}}_2}, \boxed{\bar{\mathbf{u}}_2}, \mathbf{y}_2, \mathbf{u}_2, \ldots, \boxed{\bar{\mathbf{y}}_M}, \boxed{\bar{\mathbf{u}}_M}, \mathbf{y}_M, \mathbf{u}_M)$

- NLP constraints: $\mathbf{c}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{q}(\mathbf{x})$ where $\mathbf{A}$ and $\mathbf{B}$ are constant matrices and

$$\mathbf{q} = \begin{bmatrix} t_f\mathbf{f}_1 \\ t_f\bar{\mathbf{f}}_2 \\ t_f\mathbf{f}_2 \\ \vdots \\ t_f\mathbf{f}_{M-1} \\ t_f\bar{\mathbf{f}}_M \\ t_f\mathbf{f}_M \end{bmatrix}$$

Hermite-Simpson (Separated) Sparsity

$\mathbf{D} =$ 

$\gamma = 2$

The General Approach

- For path constrained problems in semi-explicit form

$$\begin{array}{rcl} \dot{\mathbf{y}} & = & \mathbf{f}[\mathbf{y},\mathbf{u},t] \\ \mathbf{g}_\ell & \leq & \mathbf{g}[\mathbf{y},\mathbf{u},t] \quad \leq \quad \mathbf{g}_u, \end{array}$$

  write transcribed NLP functions as

$$\left[\begin{array}{c} \mathbf{c}(\mathbf{x}) \\ F(\mathbf{x}) \end{array} \right] = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{q}(\mathbf{x})$$

  where $\mathbf{A}$ and $\mathbf{B}$ are constant matrices and $\mathbf{q}$ involves right hand sides at grid points.

- Construct sparsity template for DAE right hand side

$$\left[\begin{array}{c|c} \frac{\partial \mathbf{f}}{\partial \mathbf{y}} & \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \\ \\ \frac{\partial \mathbf{g}}{\partial \mathbf{y}} & \frac{\partial \mathbf{g}}{\partial \mathbf{u}} \end{array} \right]$$

- Construct sparsity for $\mathbf{D}$ and compute index sets.

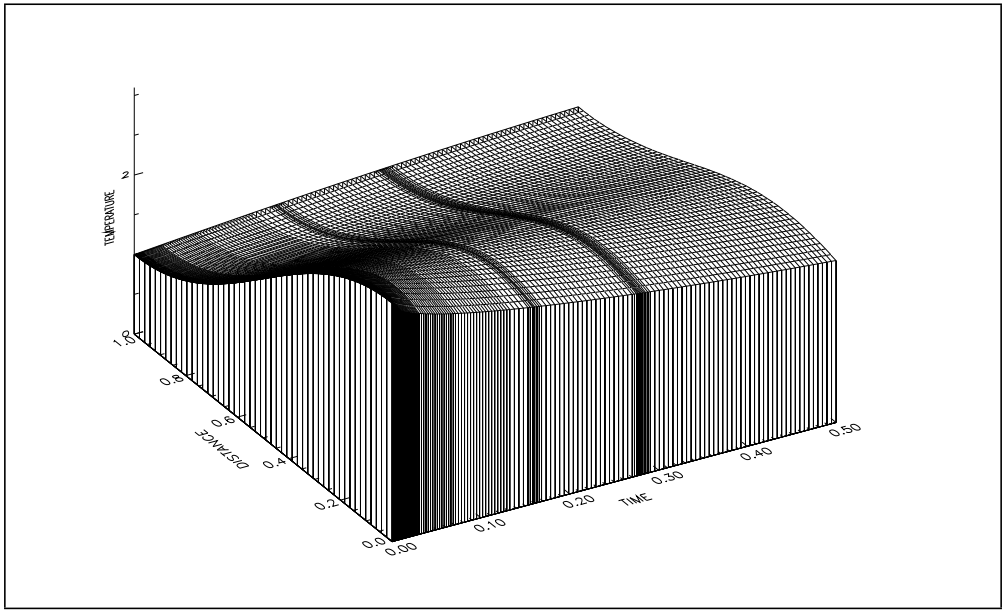
- Define NLP Jacobian sparsity from $\mathbf{G} = \mathbf{A} + \mathbf{B}\mathbf{D}$

- Define NLP Hessian sparsity from $(\mathbf{B}\mathbf{D})^{\mathsf{T}}(\mathbf{B}\mathbf{D})$.

Specific Performance Highlights

- Spatial discretization of Nonlinear Parabolic PDE (Heinkenschloss)
- DAE system (50 states, 3 controls, 2 algebraic equations)
- Five mesh refinements; Final NLP—9181 variables, 9025 active constraints, and 156 degrees of freedom

	Dense	Sparse	Reduction (%)
γ_s	53	4	-92.5%
γ_c	109	15	-86.2%
Func. Eval.	42552	983	-97.7%
CPU Time	1851.43	377.86	-79.6%



Summary

- Write transcribed NLP functions as

$$\begin{bmatrix} \mathbf{c}(\mathbf{x}) \\ F(\mathbf{x}) \end{bmatrix} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{q}(\mathbf{x})$$

  to exploit sparsity in $\mathbf{D} = \frac{\partial \mathbf{q}}{\partial \mathbf{x}}$.

- Sparsity considerations lead to desirable discretizations:
  - Separated form of k-stage Runge-Kutta Schemes

- Construction and analysis of RHS sparsity
  - Numerical with limited analysis of rank

Optimal Control Algorithm

Direct Transcription Transcribe the optimal control problem into a nonlinear programming (NLP) problem by discretization;

Sparse Nonlinear Program Solve the sparse NLP using an SQP or Barrier Method

Mesh Refinement Assess the accuracy of the approximation (i.e. the finite dimensional problem), and if necessary refine the discretization, and then repeat the optimization steps.

Mesh Refinement

- **The Problem:** Approximate continuous state and control functions using values at discrete points
  - Many discrete values give “good” approximation
  - Many discrete values make numerical problem “hard”
- **Mesh Refinement:** Choose
  - discretization technique and
  - number and location of points

  to achieve
  - an “accurate” solution and
  - an “efficient” solution method.

The Challenge for Mesh Refinement

- Computation of solution accuracy depends on **order** of the discretization method
- **Order** depends on **stiffness** of ODE
- **Order** depends on **index** of DAE
- **Index** depends on which path constraints are **active**, e.g.

Index 1	Index 2
$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}, \mathbf{u}, t]$	$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}, \mathbf{u}, t]$
$0 = g_1[\mathbf{y}, \mathbf{u}, t]$	$0 = g_2[\mathbf{y}, \mathbf{u}, t]$

$$\Downarrow$$

- Order variation makes acceleration techniques (e.g. extrapolation, deferred correction) cumbersome

Discretization Selection Strategy

Trapezoidal Robust, $O(h^2)$, good for coarse grid

Separated Hermite-Simpson $O(h^4)$, fewer index sets than HSC, but larger NLP

Compressed Hermite-Simpson $O(h^4)$, more index sets than HSS, but smaller NLP

Representing the Solution

- **Goal:** Construct approximate continuous solution $[\widetilde{\mathbf{y}}(t), \widetilde{\mathbf{u}}(t)]$ from the discretization.

- **Spline Representation:**

$$\begin{bmatrix} \mathbf{y}(t) \\ \mathbf{u}(t) \end{bmatrix} \approx \begin{bmatrix} \widetilde{\mathbf{y}}(t) \\ \widetilde{\mathbf{u}}(t) \end{bmatrix} = \sum_{i=1}^{N} \alpha_i B_i(t)$$

  where the coefficients α_i are defined such that

  - *state variables*

$$\widetilde{\mathbf{y}}(t_k) = \mathbf{y}_k \qquad \tfrac{d}{dt}\widetilde{\mathbf{y}}(t_k) = \mathbf{f}_k$$

  - *control variables*

$$\widetilde{\mathbf{u}}(t_k) = \mathbf{u}_k \qquad \widetilde{\mathbf{u}}[(t_k + t_{k-1})/2] = \overline{\mathbf{u}}_k \quad \text{(H-S only)}$$

- **Properties:**

  - *state variables* C^1 cubic functions
  - *control variables* C^0 linear or quadratic functions

Estimating Discretization Error

- Define *absolute local error*

$$\eta_{i,k} = \int_{t_k}^{t_{k+1}} |\varepsilon_i(s)| ds.$$

where

$$\varepsilon(t) = \dot{\widetilde{\mathbf{y}}}(t) - \mathbf{f}[\widetilde{\mathbf{y}}(t), \widetilde{\mathbf{u}}(t), t]$$

- Define *relative local error*

$$\varepsilon_k \approx \max_i \frac{\eta_{i,k}}{(w_i + 1)}$$

where the scale weight

$$w_i = \max_{k=1}^{M} \left[|\tilde{y}_{i,k}|, |\dot{\tilde{y}}_{i,k}| \right]$$

defines the maximum value for the i-th state variable or its derivative over the M grid points.

- Evaluate $\eta_{i,k}$ accurately using Romberg quadrature

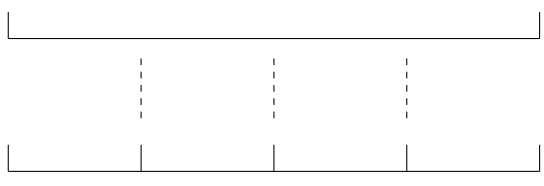
Estimating Order Reduction

- *Local Error* $O(h^{p-r+1})$ has the form

$$\varepsilon_k \approx \|\mathbf{c}_k h^{p-r+1}\|$$

where *order reduction* is r, and $p = 2$ for trapezoidal, $p = 4$ for Hermite-Simpson.

- Subdivide Interval by adding I points



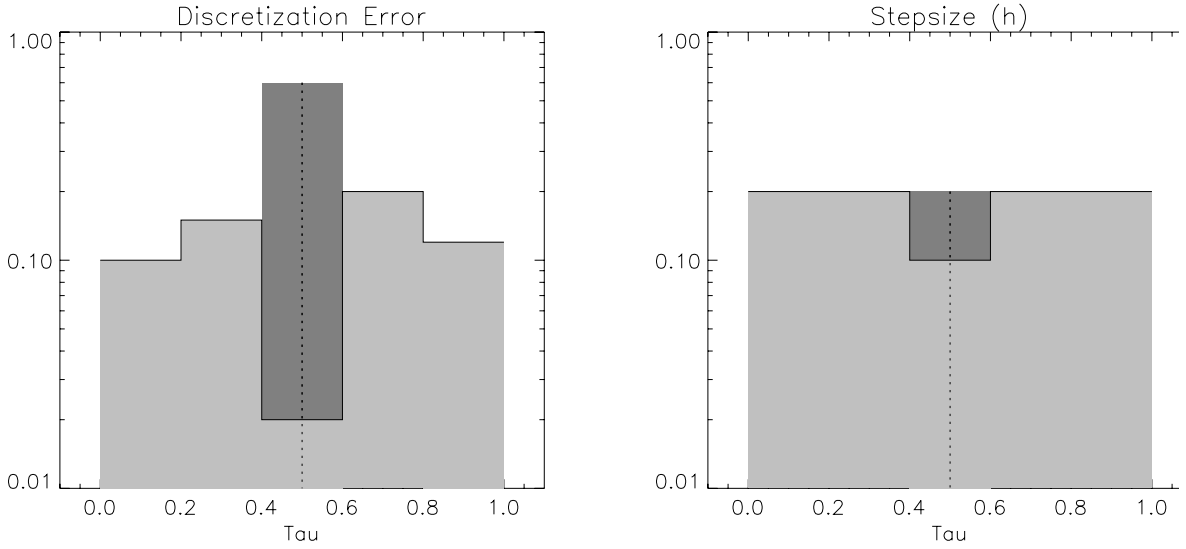
- Assume constant c and r solve for *Predicted Reduction*

$$r = \max\left[0, \min\left(\text{nint}(\hat{r}), p\right)\right]$$

where

$$\hat{r} = p + 1 - \frac{\log(\theta/\eta)}{\log(1+I)}.$$

Subdividing The Grid



Example: Add One Point to Third Interval

Constructing a New Mesh

- *Subdivide* current grid by adding I_k points to interval k.
- Assuming constant coefficients c_k across interval

$$\varepsilon_k \approx \|c_k\| \left(\frac{h}{1+I_k}\right)^{p-r_k+1} = \max_i \frac{\eta_{i,k}}{(w_i+1)} \left(\frac{1}{1+I_k}\right)^{p-r_k+1}$$

- *Integer Program:* Choose I_k to minimize

$$\phi(I_k) = \max_k \varepsilon_k$$

and satisfy the constraints

$$\sum_{k=1}^{n_s} I_k \leq M-1$$

and

$$0 \leq I_k \leq M_1$$

for $k = 1, \ldots, n_s$.

Refinement Algorithm

Given a discrete solution on the current grid (from NLP), parameters δ, $\kappa \leq 1$, and M_1 (typically $\delta = 10^{-7}$, $\kappa = .1$, $M_1 = 5$)

1. Construct Continuous Representation. Compute the cubic spline representation from the discrete solution $\mathbf{x}^*$

2. Estimate Discretization Error. Compute an estimate for the discretization error ε_k in each segment of the current mesh

3. Compute Primary Order of New Mesh.

- If error is equidistributed for low order method increase order; (If $p = 2$ and $\varepsilon_{max} \leq 2\bar{\varepsilon}$ then set $p = 4$ and terminate).
- otherwise if more than two refinements with low order method increase order; (if $p = 2$ and *ref. iter.* > 2 then set $p = 4$ and terminate).

4. Estimate Order Reduction. Compare current and old grid to compute r_k

Refinement Algorithm (Cont.)

5. Construct New Mesh Solve integer program

1. Compute interval α with maximum error, i.e.

$$\varepsilon_\alpha = \max_k \varepsilon_k$$

2. Terminate if:
   - M' points have been added ($M' \geq \min[M_1, .1M]$)

     and:
   - *Error within tolerance:* $\varepsilon_\alpha \leq \delta$ and $I_\alpha = 0$ or;
   - *Predicted error safely within tolerance:* $\varepsilon_\alpha \leq \kappa\delta$ and $0 < I_\alpha < M_1$ or;
   - $M - 1$ points have been added or;
   - M_1 points have been added to a single interval.
3. Add a point to interval α, i.e. $I_\alpha \leftarrow I_\alpha + 1$;
4. Update predicted error for interval α, and
5. Repeat steps 1-5

Transcription Accuracy

- Mesh Refinement Algorithm
  - B-spline representation for optimal solution
  - Discretization error based Romberg quadrature
- Interval subdivision by solving integer programming problem
  - Minimizes the maximum relative error
  - Tends to equidistribute error by clustering points where error is large
  - Tends to reduce overall error when error is equidistributed
- Predicted order reduction
  - Computed by comparing successive grids
  - Tends to cluster points where order is low
  - Alters where points are added, but
  - Is not used when computing discretization accuracy!
- Overall strategy uses
  - lower order trapezoidal method for early refinment iterations
  - higher order Hermite-Simpson method for final refinment iterations
  - Separated vs Compressed H-S selected dynamically

Parameter Estimation Using Direct Transcription Methods

The “Forward Problem”

- Given a set of parameters $\mathbf{p}$ find the resulting trajectory.
- “Integrate” the differential equations from t_I to t_F

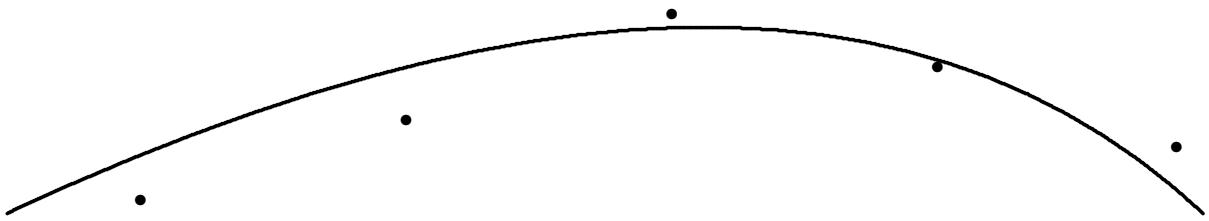
$$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t]$$

  state variables $\mathbf{y}$, algebraic variables $\mathbf{u}$.

The “Inverse Problem”

- Given the trajectory find the parameters $\mathbf{p}$.
- Trajectory is “given” by

$$\widehat{\mathbf{x}}(t_1), \widehat{\mathbf{x}}(t_2), \ldots, \widehat{\mathbf{x}}(t_N)$$



The Parameter Estimation Problem

- *State equations*

$$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t]$$

state variables $\mathbf{y}$, algebraic variables $\mathbf{u}$, for $t_I \leq t \leq t_F$.

- *Path constraints*

$$\mathbf{g}_L \leq \mathbf{g}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t] \leq \mathbf{g}_U,$$

- *Bounds*

$$\begin{array}{ccccc} \mathbf{y}_L & \leq & \mathbf{y}(t) & \leq & \mathbf{y}_U \\ \mathbf{u}_L & \leq & \mathbf{u}(t) & \leq & \mathbf{u}_U \\ \mathbf{p}_L & \leq & \mathbf{p} & \leq & \mathbf{p}_U. \end{array}$$

- *Boundary conditions*

$$\psi_L \leq \psi[\mathbf{y}(t_I), \mathbf{u}(t_I), t_I, \mathbf{y}(t_F), \mathbf{u}(t_F), t_F, \mathbf{p}] \leq \psi_U,$$

- *Objective Function* (minimize)

$$F = \frac{1}{2}\mathbf{r}^{\mathsf{T}}\mathbf{r} = \frac{1}{2}\sum_{k=1}^{\ell} r_k^2.$$

The Least Squares Objective Function

- *Residuals*

$$r_k = w_{ij}\,[y_i(\theta_{ij}) - \hat{y}_{ij}] \qquad \text{State}$$
$$r_k = w_{ij}\,[u_i(\vartheta_{ij}) - \hat{u}_{ij}] \qquad \text{Algebraic}$$

  for $t_I \leq \theta_{ij} \leq t_F$ and $t_I \leq \vartheta_{ij} \leq t_F$.

- *Maximum Likelihood Objective*

$$F = \frac{1}{2}\sum_{k=1}^{N}[\mathbf{g}(\mathbf{y},\mathbf{u},\mathbf{p},\theta_k) - \widehat{\mathbf{g}}_k]^{\mathsf{T}}\Lambda[\mathbf{g}(\mathbf{y},\mathbf{u},\mathbf{p},\theta_k) - \widehat{\mathbf{g}}_k]$$

  where $\widehat{\mathbf{g}}_k$ are the observed values of the function $\mathbf{g}$, and the positive definite inverse covariance matrix $\Lambda = \mathbf{Q}^{\mathsf{T}}\mathbf{Q}$.

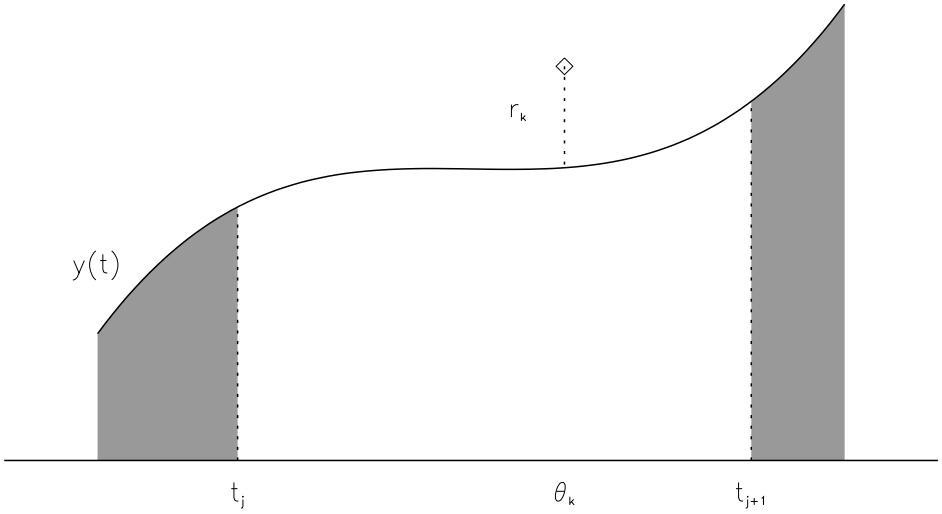
- *Transformation to Standard Form*

$$\mathbf{v}(t) = \mathbf{Q}\mathbf{g}(\mathbf{y},\mathbf{u},\mathbf{p},t) \qquad \text{Algebraic Constraint}$$
$$\widehat{\mathbf{v}}_k = \mathbf{Q}\widehat{\mathbf{g}}_k \qquad \text{Normalized Data}$$

- *Transformed Objective Function* (minimize)

$$F = \frac{1}{2}\sum_{k=1}^{N}[\mathbf{v}_k - \widehat{\mathbf{v}}_k]^{\mathsf{T}}[\mathbf{v}_k - \widehat{\mathbf{v}}_k]$$

Computing The Residuals



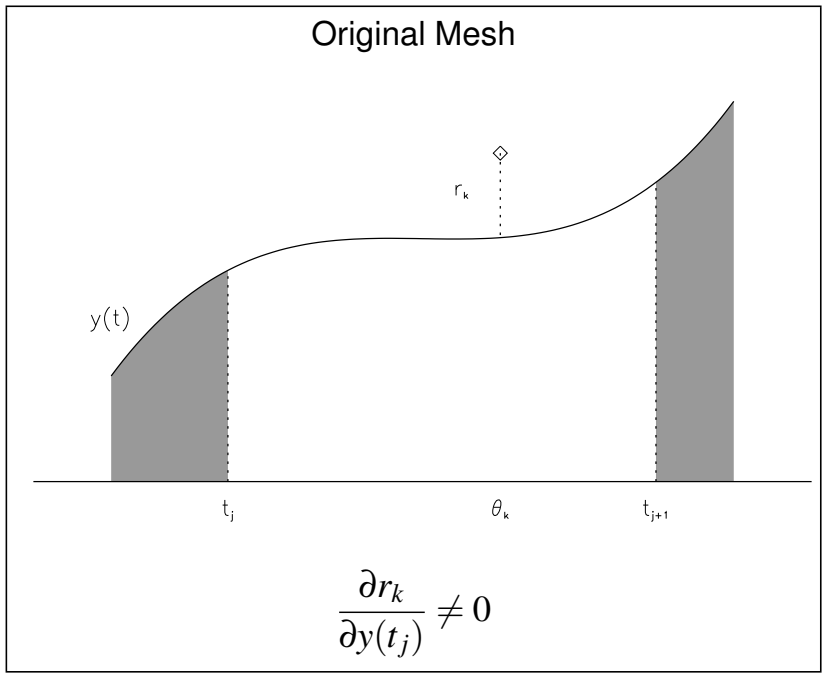
Residual $r_k = [y(\theta_k) - \hat{y}_k]$

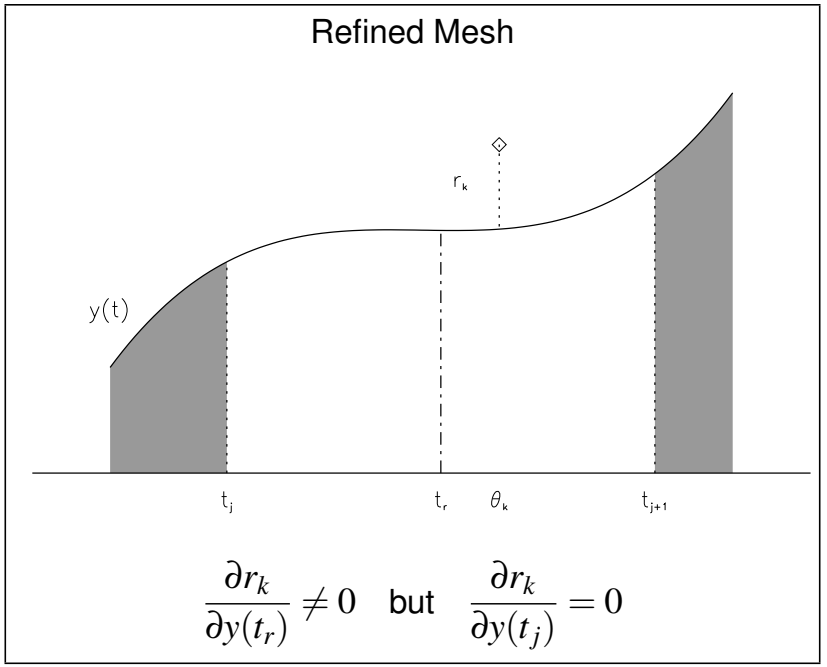
State Residual : $y(\theta_k) \longrightarrow$ Cubic Interpolant using $y_j, y_{j+1}, f_j, f_{j+1}$

Algebraic Residual : $u(\vartheta_k) \longrightarrow$

- Quadratic Interpolant using $u_j, \bar{u}_{j+1}, u_{j+1}$
- Linear Interpolant using u_j, u_{j+1}

Mesh Refinement and Jacobian Sparsity





Observation: Jacobian and Hessian Sparsity Pattern Change as Mesh is Refined!

"Notorious Test Problem"*

$$
\begin{aligned}
F(p) &= \frac{1}{2}\sum_{k=1}^{N}\left[(y_{1k}-\hat{y}_{1k})^2+(y_{2k}-\hat{y}_{2k})^2\right] \\
\dot{y}_1 &= y_2 \\
\dot{y}_2 &= \mu^2 y_1-(\mu^2+p^2)\sin(pt)
\end{aligned}
$$

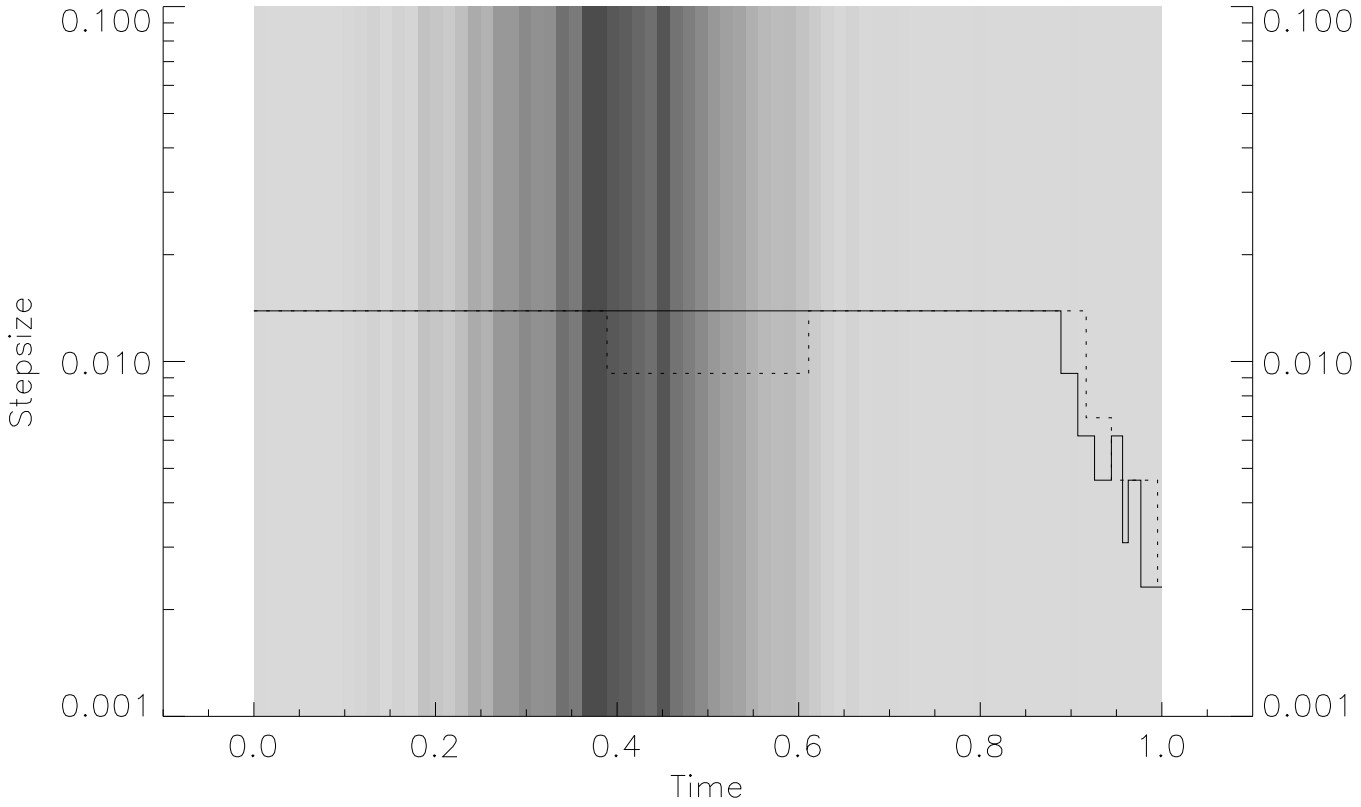
$$y_1(0)=0,\ y_2(0)=\pi,\ \mu=60,\ 0\le t\le 1;\ p^*=\pi$$

Observations

- Shooting method (i.e. "forward problem") is unstable .
- Residual Jacobian sparsity changes with mesh refinement
- Mesh refinement is driven by ODE accuracy, not data

*Bulirsch

“Notorious Test Problem”



Dashed	
Refn	5
Grid	92
Data (equidist)	10

Solid	
Refn	5
Grid	91
Data (random)	2000

Summary

- Mesh refinement is driven by DAE accuracy, not data
- Quadratic Convergence for Nonlinear, Nonzero Residuals
- Sparsity for Least Squares Jacobian and Hessian determined by
  - Discrete data location relative to mesh points
  - Right hand side sparsity
- Jacobian and Hessian Computed Using Sparse Finite Differences