Large Scale Parameter Estimation

John T. Betts
William P. Huffman

Introduction

Parameter Estimation
Estimate Parameters $\mathbf{p}$ so that
a mathematical model
"matches" observations.

Large Scale $\Leftrightarrow$ Lots of parameters

The Traditional Approach

1. Guess parameters $\mathbf{p}$

2. "Fly" a simulated trajectory

3. Does simulated trajectory "match" observed data?

   (a) Yes—estimated parameters are OK.

   (b) No—make new guess and repeat.

Estimating Aerodynamic Models

The Problem: Find parameters $\mathbf{p}$ that define aerodynamics, such that a simulated trajectory matches flight test(s).

Aerodynamics

$$D = \frac{1}{2} C_D S \rho v^2 = \frac{1}{2} \left[c_\ell(M) \alpha \right] S \rho v^2 \qquad \text{Drag}$$

$$L = \frac{1}{2} C_L S \rho v^2 = \frac{1}{2} \left[c_d(M) + \eta(M) c_\ell(M) \alpha^2 \right] S \rho v^2 \qquad \text{Lift}$$

with B-spline representations

$$c_\ell = \mathbf{p}_\ell^\top \mathbf{B}(M) \qquad c_d = \mathbf{p}_d^\top \mathbf{B}(M) \qquad \eta = \mathbf{p}_\eta^\top \mathbf{B}(M)$$

Estimating Aerodynamic Models (cont)

Trajectory Dynamics:

$$\begin{aligned} \dot{\mathbf{y}} &= \mathbf{f}[\mathbf{y}, \mathbf{u}, \mathbf{p}, t] = \mathbf{f}(Drag, Lift) \\ 0 &= c_\ell - \mathbf{p}_\ell^\mathsf{T}\mathbf{B} \\ 0 &= c_d - \mathbf{p}_d^\mathsf{T}\mathbf{B} \\ 0 &= \eta - \mathbf{p}_\eta^\mathsf{T}\mathbf{B} \end{aligned}$$

where

$$\mathbf{y}^\mathsf{T} = (h, \phi, \theta, v, \gamma, \psi) \qquad \mathbf{u}^\mathsf{T} = (c_\ell, c_d, \eta) \qquad \mathbf{p}^\mathsf{T} = (\mathbf{p}_\ell^\mathsf{T}, \mathbf{p}_d^\mathsf{T}, \mathbf{p}_\eta^\mathsf{T})$$

The Optimization Problem

Objective Function:

$$F = \frac{1}{2} \sum_{i=1}^{3} \sum_{j=1}^{N} \sum_{k=1}^{6} \left\{ \frac{1}{s_k} \left[y_k^{(i)}(\theta_{kj}) - \widehat{y}_{kj}^{(i)} \right] \right\}^2 .$$

where

$i = 1, 2, 3$	Three flight test trajectories
$j = 1, \ldots, N$	$N = 1000$ data points per trajectory
$k = 1, \ldots, 6$	Data measured for 6 dynamic variables

A "Dry Lab" Experiment

1. Assume given aero parameters $\tilde{\mathbf{p}}$ are "truth".

2. "Fly" (i.e. integrate) a trajectory and save $\tilde{y}_k$

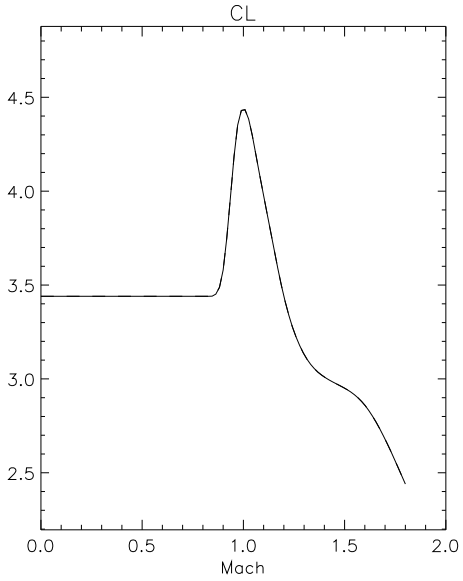
3. Add "noise"

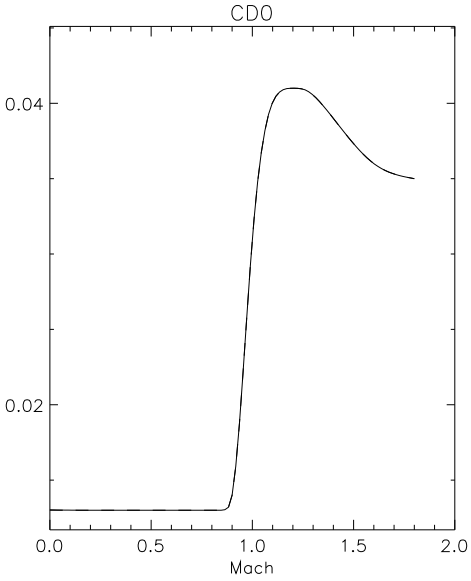
Simulated Flight Test Data

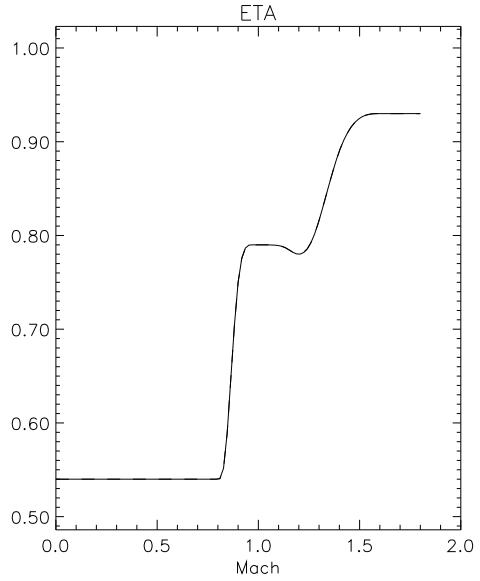
$$\widehat{y}_k = \tilde{y}_k + \sigma s_k \nu_k$$

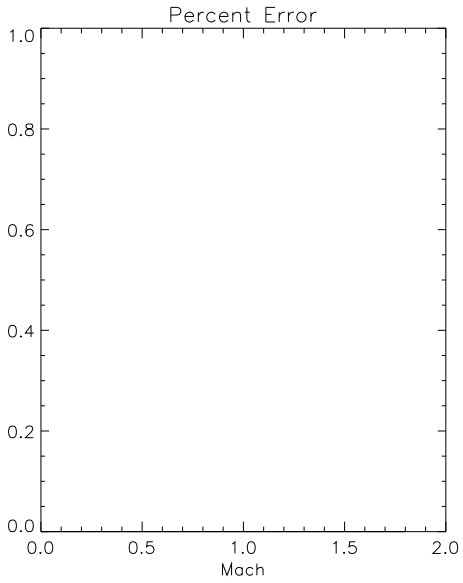
where $\tilde{y}$ is "truth," and noise, $\nu \sim \mathcal{N}(0, 1)$.

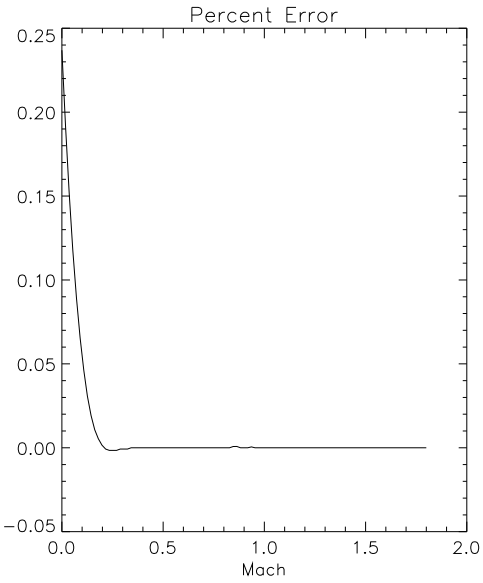
Aerodynamic Model Estimates (Error Free Data)

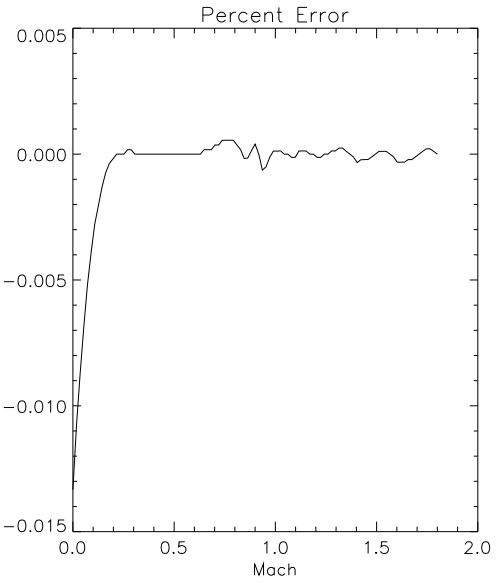




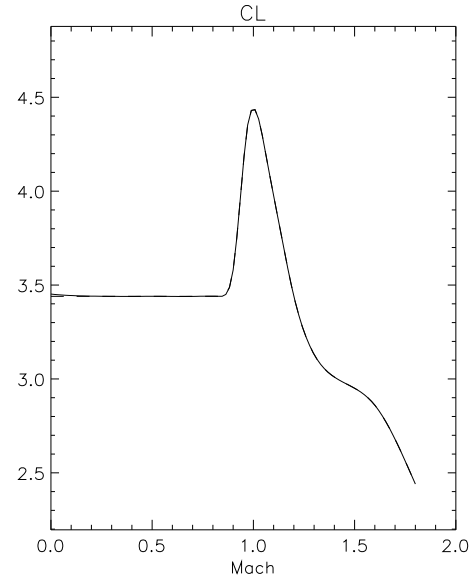


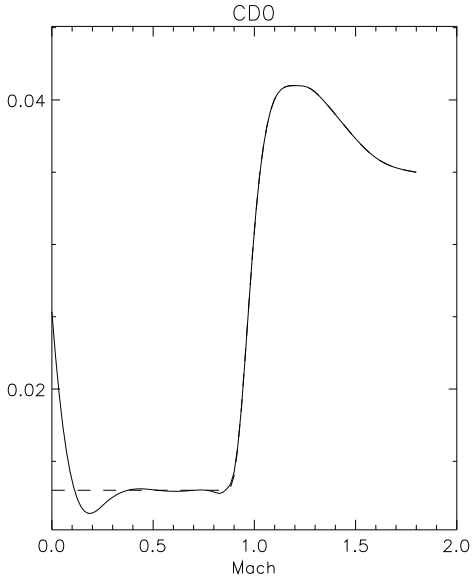


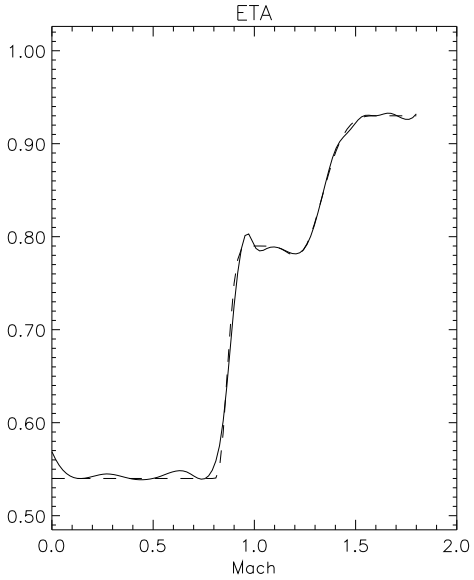


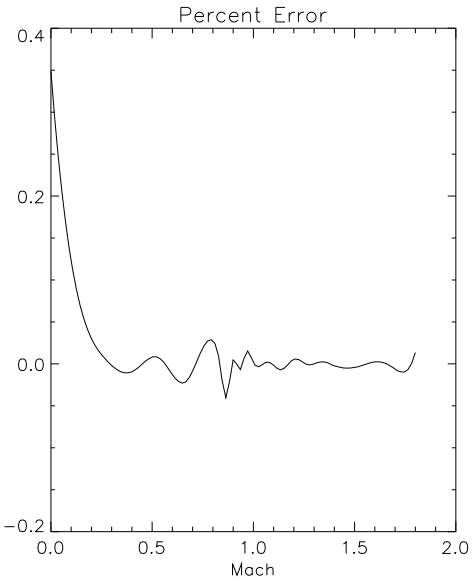


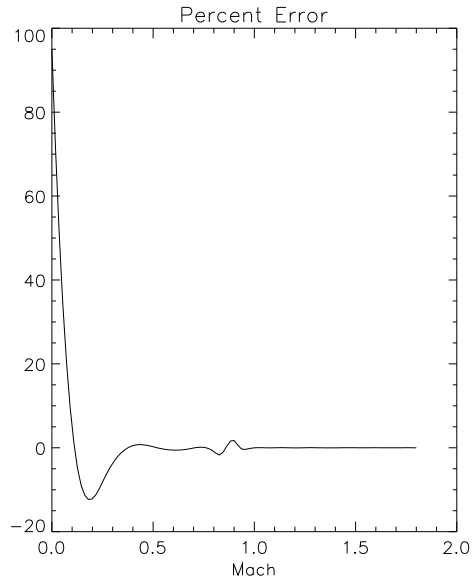
Unconstrained Model Estimates (Noisy Data)

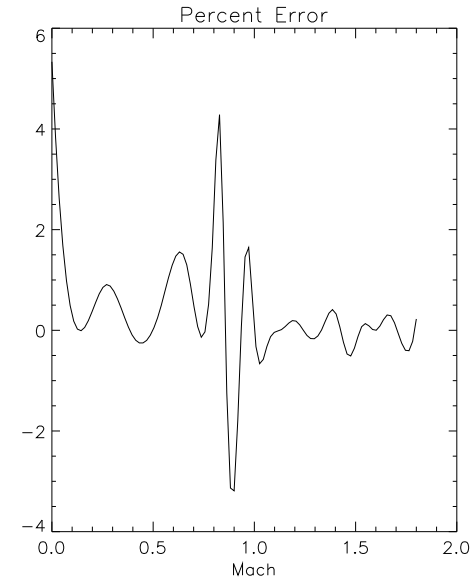












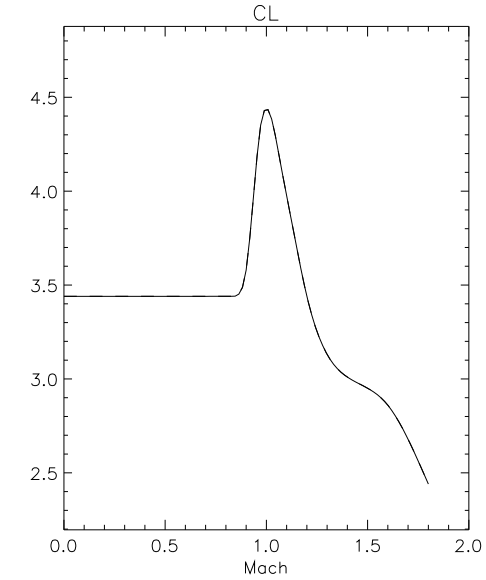
Model Behavior

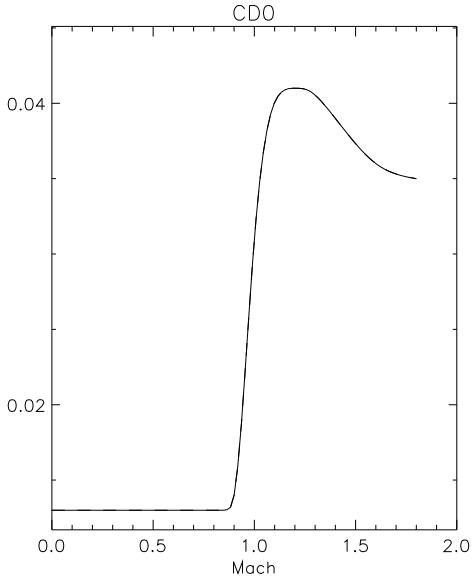
No data at $M = 0$ so add (typical) constraints;

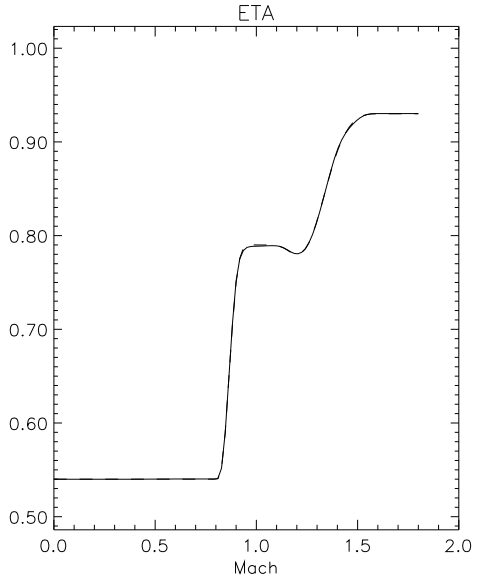
$$c'_\ell \,|_{M=0} = \mathbf{p}_\ell^{\mathsf{T}} \mathbf{B}' \,|_{M=0} = 0 \qquad \text{Asymptotic}$$

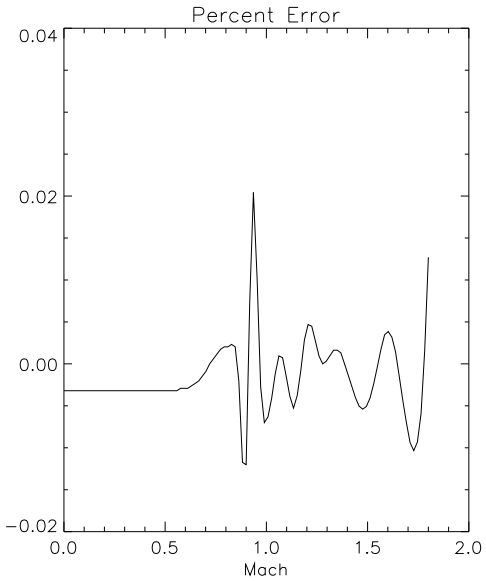
$$c'_\ell(M) \geq 0 \qquad \text{for}(0 \leq M < 1) \qquad \text{Monotonic}$$

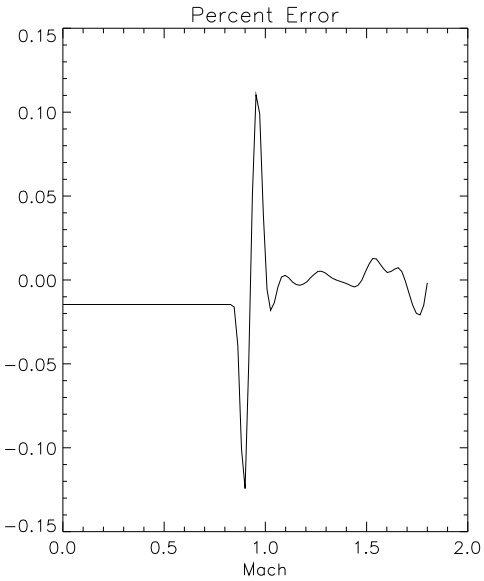
Constrained Model Estimates (Noisy Data)

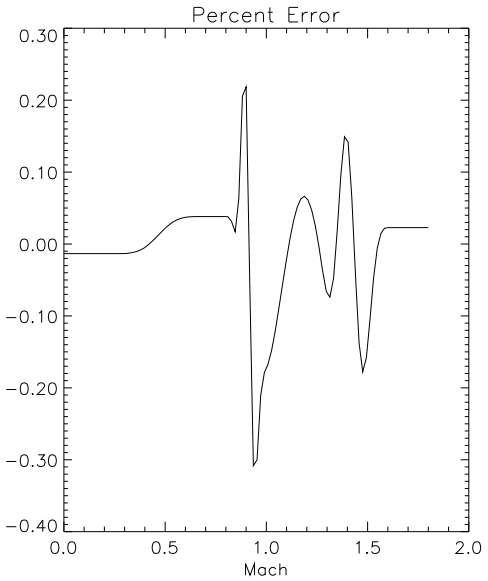




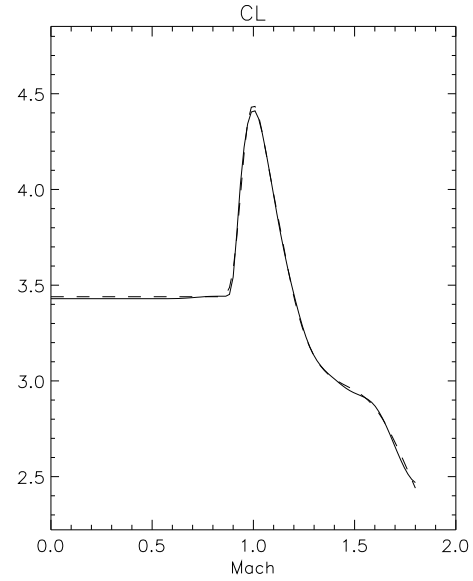


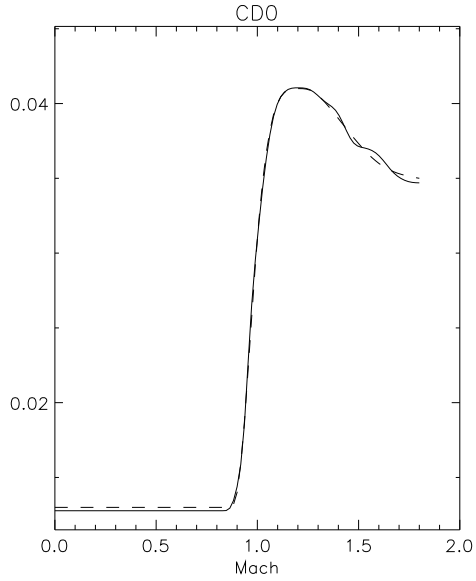


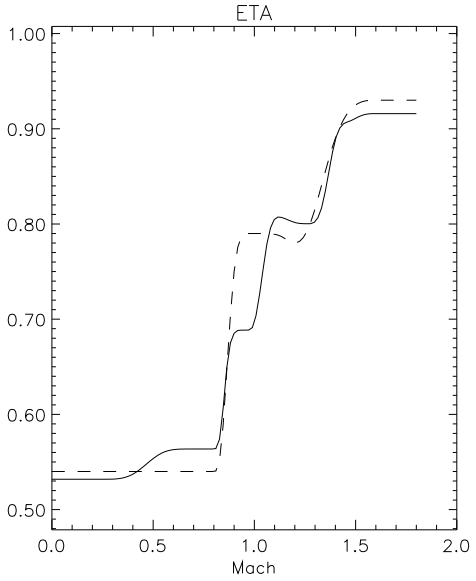


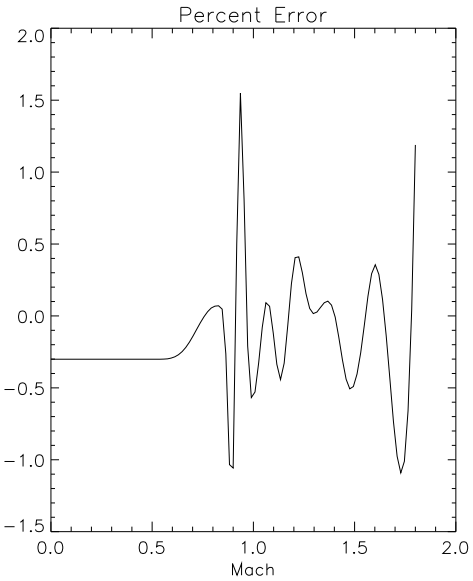


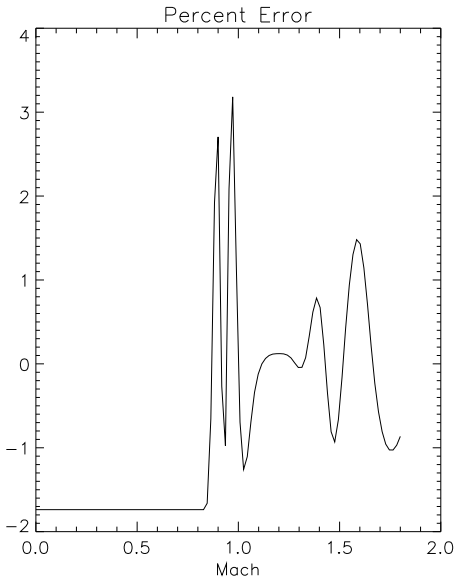
Constrained Model Estimates (Lousy Data)

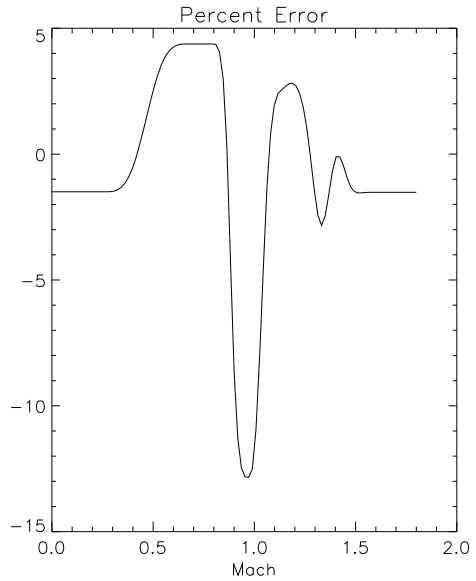












Solving the Large Scale Estimation Problem

The Parameter Estimation Problem

- *State equations*

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

state variables	$\mathbf{y}(t)$
algebraic variables	$\mathbf{u}(t)$
parameters	$\mathbf{p}$

- *Bounds*

$$\mathbf{y}_L \leq \mathbf{y}(t) \leq \mathbf{y}_U \qquad \mathbf{u}_L \leq \mathbf{u}(t) \leq \mathbf{u}_U \qquad \mathbf{p}_L \leq \mathbf{p} \leq \mathbf{p}_U$$

- *Boundary conditions*

$$\boldsymbol{\psi}_L \leq \boldsymbol{\psi}[\mathbf{y}(t_I), \mathbf{u}(t_I), t_I, \mathbf{y}(t_F), \mathbf{u}(t_F), t_F, \mathbf{p}] \leq \boldsymbol{\psi}_U,$$

The Parameter Estimation Problem (cont)

- *Path constraints*

$$g_L \leq \alpha_0 g[\mathbf{v}, t] + \boldsymbol{\alpha}^{\mathsf{T}}\mathbf{v} + \boldsymbol{\beta}^{\mathsf{T}}\mathbf{a}[\mathbf{v}, t] \leq g_U,$$

- Linear Combination of:

analytic terms	$\boldsymbol{\alpha}^{\mathsf{T}}\mathbf{v}$
auxiliary functions	$\boldsymbol{\beta}^{\mathsf{T}}\mathbf{a}[\mathbf{v}, t]$

where $\mathbf{v} = (\mathbf{y}, \mathbf{u}, \mathbf{p})$.

The Least Squares Objective Function

- Find $\mathbf{v}$ subject to constraints and minimize:

- *Objective Function*

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

- *Residuals*

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

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

Objective Function (cont)

- *Maximum Likelihood Objective*

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

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

- *Transformation to Standard Form*

$\mathbf{s}(t) = \mathbf{Q}\mathbf{g}(\mathbf{y}, \mathbf{u}, \mathbf{p}, t)$	Algebraic Constraint
$\hat{\mathbf{s}}_k = \mathbf{Q}\hat{\mathbf{g}}_k$	Normalized Data

- *Transformed Objective Function* (minimize)

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

Parameter Estimation Algorithm

Direct Transcription Transcribe the parameter estimation problem into a nonlinear programming (NLP) problem by discretization;

Sparse Nonlinear Program Solve the sparse NLP using sequential quadratic programming

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.

Transcription Formulation

- Discretization: M grid points; stepsize $h_j \equiv t_{j+1} - t_j$

$$\begin{aligned} \mathbf{y}_j &\equiv \mathbf{y}(t_j) & \qquad \overline{\mathbf{u}}_j &\equiv \mathbf{u}[(t_j + t_{j-1})/2] \\ \mathbf{u}_j &\equiv \mathbf{u}(t_j) & \qquad \mathbf{f}_j &\equiv \mathbf{f}[\mathbf{y}(t_j), \mathbf{u}(t_j), t_j] \end{aligned}$$

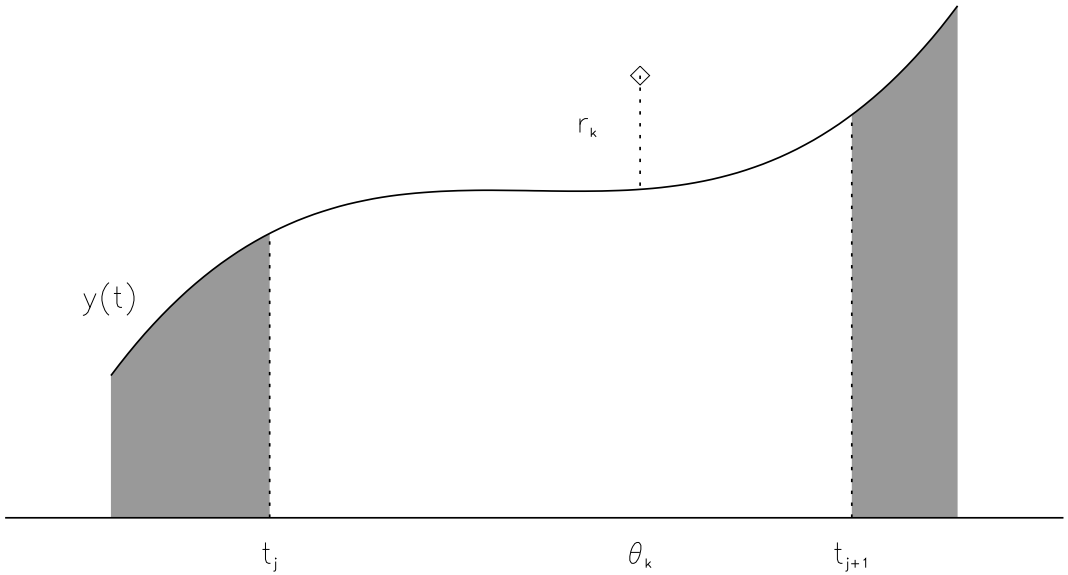
- Hermite-Simpson
  - NLP variables

$$\mathbf{x} = [\mathbf{y}_1, \mathbf{u}_1, \overline{\mathbf{u}}_2, \mathbf{y}_2, \mathbf{u}_2, \overline{\mathbf{u}}_3, \ldots, \mathbf{y}_M, \mathbf{u}_M, \mathbf{p}, t_I, t_F]^\top .$$

  - Defect constraints for $j = 1, \ldots, (M-1)$

$$\zeta_j = \mathbf{y}_{j+1} - \mathbf{y}_j - \frac{h_j}{6}\left[\mathbf{f}_{j+1} + 4\overline{\mathbf{f}}_{j+1} + \mathbf{f}_j\right] = 0$$

Computing The Residuals



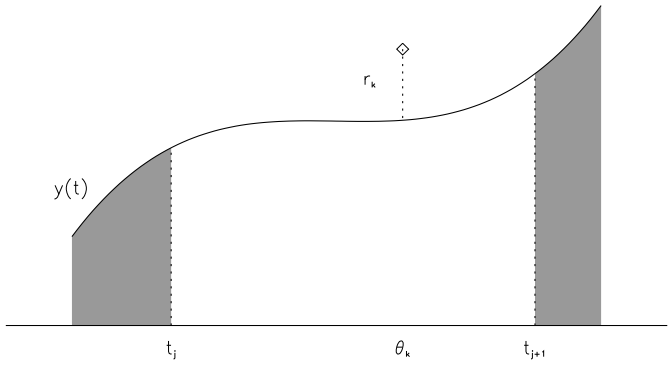
Residual $r_k = [y(\theta_k) - \widehat{y}_k]$

State Residual : $y(\theta_k) \longrightarrow$ Cubic Interpolant

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

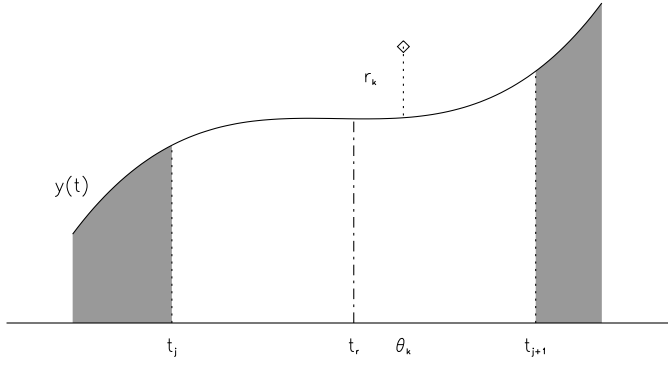
Mesh Refinement and Jacobian Sparsity

Original Mesh



$$\frac{\partial r_k}{\partial y(t_j)} \neq 0$$

Refined Mesh



$$\frac{\partial r_k}{\partial y(t_r)} \neq 0 \quad \text{but} \quad \frac{\partial r_k}{\partial y(t_j)} = 0$$

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

Sparse Least Squares Method

Gradient Vector $(F = \frac{1}{2}\mathbf{r}^\mathsf{T}\mathbf{r})$

$$\mathbf{g} = \mathbf{R}^\mathsf{T}\mathbf{r} = \sum_{i=1}^{\ell} r_i \nabla \mathbf{r}_i.$$

Residual Jacobian

$$\mathbf{R}^\mathsf{T} = [\nabla \mathbf{r}_1, \ldots, \nabla \mathbf{r}_\ell]$$

Hessian of the Lagrangian

$$\begin{aligned} \mathbf{H}_L(\mathbf{x}, \boldsymbol{\lambda}) &= \sum_{i=1}^{\ell} r_i \nabla^2 r_i - \sum_{i=1}^{m} \lambda_i \nabla^2 c_i + \mathbf{R}^\mathsf{T}\mathbf{R} \\ &\equiv \mathbf{V} + \mathbf{R}^\mathsf{T}\mathbf{R}. \end{aligned}$$

Sparse Least Squares Method

Gauss Method Assumption: $\mathbf{V} = \mathbf{0}$

- OK if $\mathbf{r} \to \mathbf{0}$ *zero residuals*
- OK if $\nabla^2 r_i = 0$ *linear least squares*

Linear Convergence for nonlinear, nonzero residuals.

Quadratic Convergence Sparse NLP does not assume $\mathbf{r} \to \mathbf{0}$ or $\nabla^2 r_i = 0$ as in Gauss method.

Least Squares – Residual Hessian

Residual Hessian $\mathbf{V}$ is computed using sparse finite differences; number of perturbations $\ll n$.

Separability Write NLP functions in terms of quantities at grid points,

$$\begin{bmatrix} \mathbf{c}(\mathbf{x}) \\ \mathbf{r}(\mathbf{x}) \end{bmatrix} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{q}(\mathbf{x}) + \boldsymbol{\zeta},$$

compute $\frac{\partial \mathbf{q}}{\partial \mathbf{x}}$ by differencing.

Least Squares – Normal Matrix

Normal Matrix $\mathbf{R}^T\mathbf{R}$ is not formed explicitly.

Solve KKT system

$$\begin{bmatrix} \mathbf{V} & \mathbf{R}^T & \mathbf{G}^T \\ \mathbf{R} & \mathbf{I} & \mathbf{0} \\ \mathbf{G} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} -\boldsymbol{\Delta}\mathbf{x} \\ -\boldsymbol{\Delta}\mathbf{z} \\ \overline{\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{0} \\ \mathbf{c} \end{bmatrix}$$

instead of

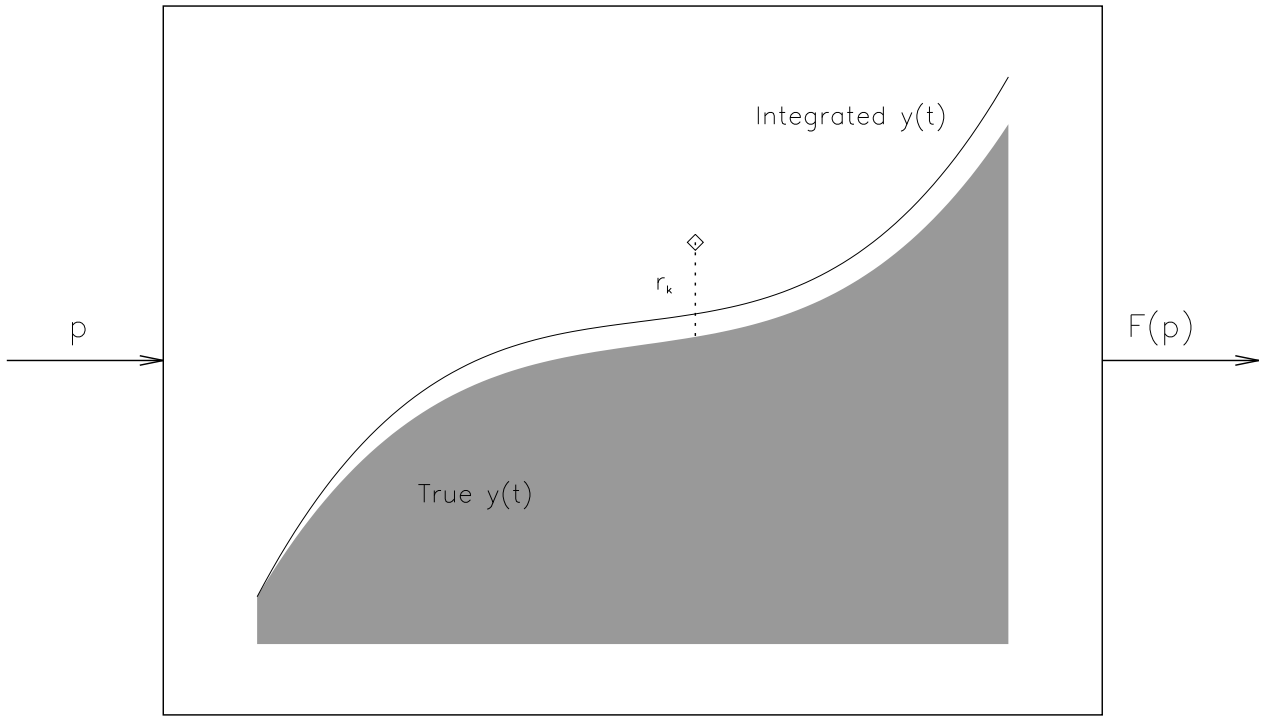
$$\begin{bmatrix} \mathbf{V}+\mathbf{R}^T\mathbf{R} & \mathbf{G}^T \\ \mathbf{G} & \mathbf{0} \end{bmatrix} \begin{bmatrix} -\boldsymbol{\Delta}\mathbf{x} \\ \overline{\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{c} \end{bmatrix}$$

Some Numbers

	Unconstrained		Constrained	
σ	0	.0001	.0001	.01
Residuals	18000	18000	18000	18000
Grid Points	2563	2346	2565	2588
NLP Var.	30972	28368	30996	31272
Act. Const.	30879	28276	30931	31216
Iterations	14	13	13	16
F^*	3.6×10^{-13}	$.9002 \times 10^{-4}$	$.9017 \times 10^{-4}$	.9

What's Wrong With The Traditional Approach?

Optimization Algorithms Need Derivatives—Why not this?



What's Wrong With The Traditional Approach?

- Integration error contaminates Residuals:

$$r_k = [y(\theta) - \widehat{y}] + \text{“integration error”}$$

- Jacobian Matrix is:
  - large (because $\mathbf{p}$ is large)
  - dense (because of trajectory integration)
  - expensive to compute

- Hessian Matrix is approximated—bad news!

"Notorious Test Problem" *

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

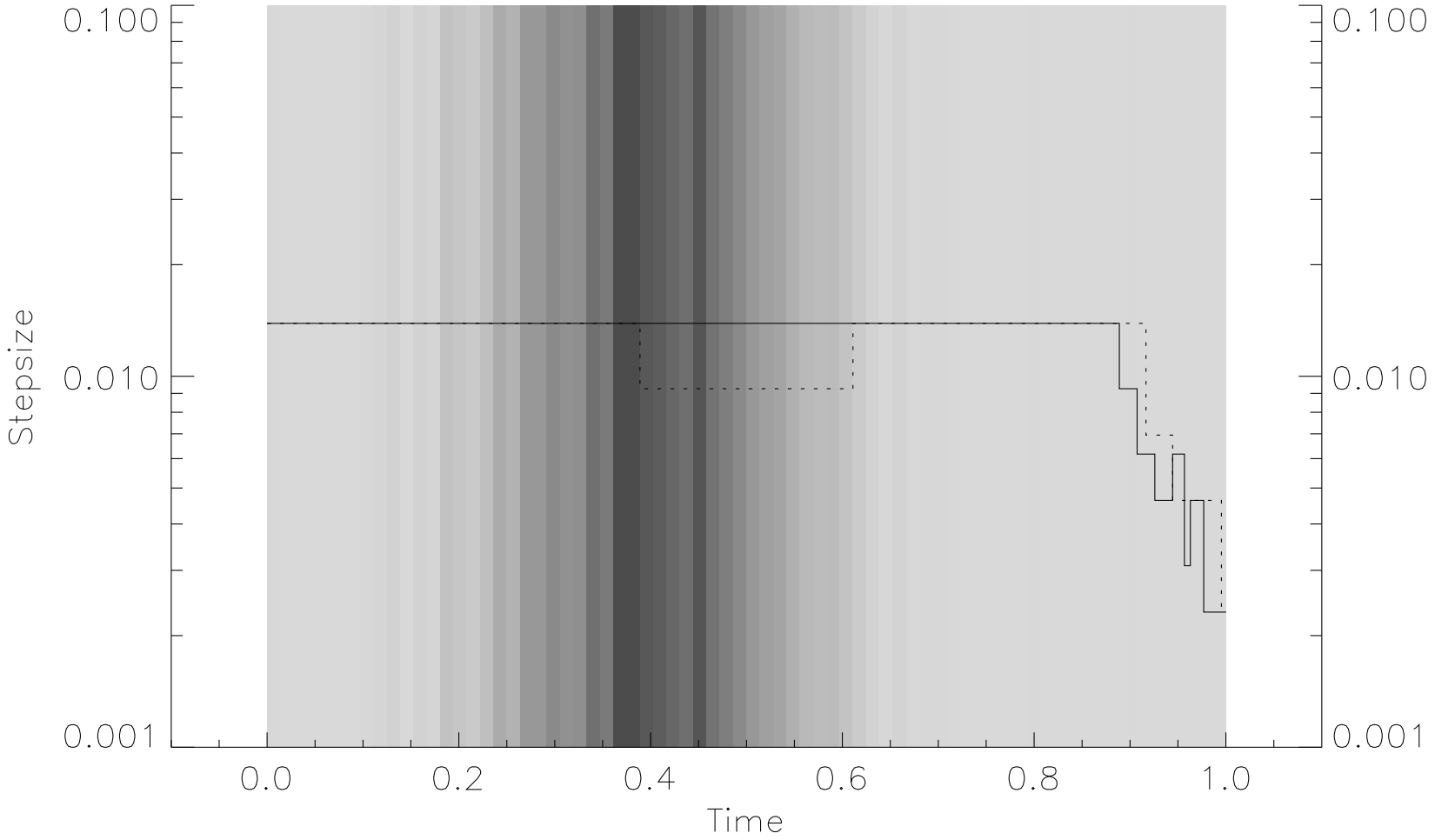
$$y_1(0)=0,\ y_2(0)=\pi,\ \mu=60,\ 0\le t\le 1;\ p^*=\pi$$

Observations

- Traditional method is unstable (i.e. can't integrate ODE's)
- Mesh refinement is driven by ODE accuracy, <u>not</u> data

*Bulirsch

"Notorious Test Problem"



Summary

- Quadratic Convergence for Nonlinear, Nonzero Residuals
- Full Hessian Makes Fast Convergence Possible
- Jacobian and Hessian Computed Using Sparse Finite Differences
- Aerodynamic Models Constructed with
  - Asymptotic and Monotonicity Constraints
  - Multiple Flight Test Trajectories