Human Standing Control Parameter Identification with Direct Collocation

Jason K. Moore
Ton van den Bogert

CLEVELAND STATE UNIVERSITY

July 10, 2015
ISB TGCS 2015, Edinburgh, UK

Parameter Identification

Find the parameters $\mathbf{p}$ such that the difference between the model simulation, $\mathbf{y}$, and measurements, $\mathbf{y}_m$ is minimized.

Dynamic system

- Equations of Motion: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{p})$
- Measurement variables: $\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{p})$

Objective

$$\min_{\mathbf{p}} J(\mathbf{p})$$

where

$$J(\mathbf{p}) = \int [\mathbf{y}_m - \mathbf{y}(\mathbf{p})]^2 dt$$

Parameter Identification By Shooting

- Repeated simulations are computationally costly
- Systems may be unstable and thus have an ill-defined objective
- Local minima are inevitable
    - May requires a superb guess
    - May need time intensive global optimization methods

Local Minima Example: Simple pendulum

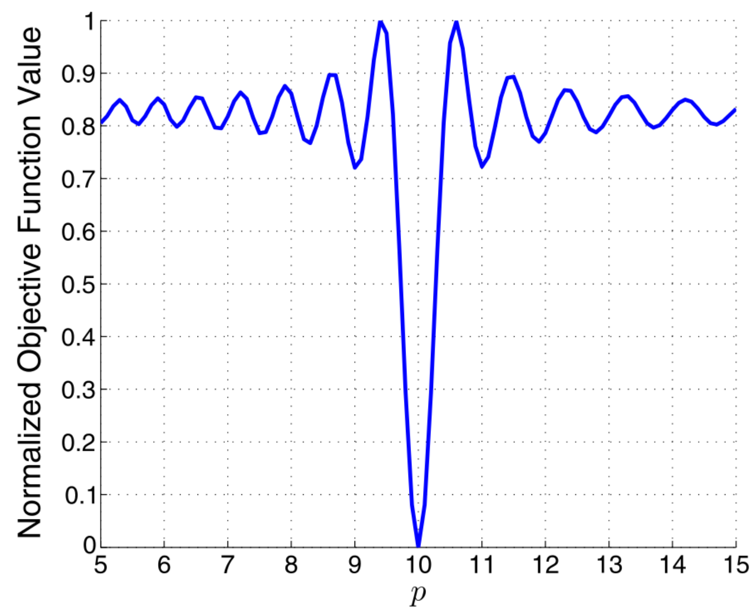
Vyasarayani, Chandrika P., Thomas Uchida, Ashwin Carvalho, and John McPhee. "Parameter Identification in Dynamic Systems Using the Homotopy Optimization Approach". Multibody System Dynamics 26, no. 4 (2011): 411-24.

1-DoF, 1 parameter pendulum equations of motion

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\theta}(t) \\ \dot{\omega}(t) \end{bmatrix} = \begin{bmatrix} \omega(t) \\ -p \sin\theta(t) \end{bmatrix}$$

Objective: Minimize least squares

$$J(p) = \min_{p} \int_{t_0}^{t_f} [\theta_m(t) - \theta(\mathbf{x}, p, t)]^2 \, dt$$



Direct Collocation

Betts, J., and W. Huffman. "Large Scale Parameter Estimation Using Sparse Nonlinear Programming Methods." SIAM Journal on Optimization 14, no. 1 (January 1, 2003): 223–44. doi:10.1137/S1052623401399216.

Benefits

- Fast computation times
- Handles unstable systems with ease
- Less susceptible to local minima

Disadvantages

- Accurate solution requires large number of nodes
- Memory management for large sparse matrices and operations
- Tedious and error prone to form gradients, Jacobians, and Hessians

Our Direct Collocation Implementation

Implicit Continous Equations of Motion

No need to solve for $\dot{\mathbf{x}}$.

$$\mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{r}, \mathbf{p}, t) = 0$$

- $\mathbf{x}$, $\dot{\mathbf{x}}$: states and their derivatives
- $\mathbf{r}$: exogenous inputs
- $\mathbf{p}$: constant parameters

Discretization

First order discrete integration options:

Backward Euler:

$$\mathbf{f}(\tfrac{\mathbf{x}_i-\mathbf{x}_{i-1}}{h}, \mathbf{x}_i, \mathbf{r}_i, \mathbf{p}, t_i) = 0$$

Midpoint Rule:

$$\mathbf{f}(\tfrac{\mathbf{x}_{i+1}-\mathbf{x}_i}{h}, \tfrac{\mathbf{x}_i+\mathbf{x}_{i+1}}{2}, \tfrac{\mathbf{r}_i+\mathbf{r}_{i+1}}{2}, \mathbf{p}, t_i) = 0$$

Nonlinear Programming Formulation

$$\min_{\theta} J(\theta)$$
$$\text{where } \theta = [\mathbf{x}_i, \dots, \mathbf{x}_N, \mathbf{r}_i, \dots, \mathbf{r}_N, \mathbf{p}]$$
$$\text{subject to } \mathbf{f}(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{r}_i, \mathbf{r}_{i+1}, \mathbf{p}, t_i) = 0 \text{ and } \theta_L \leq \theta \leq \theta_U$$

Software Tool: opty

- User specifies continous symbolic:
  - objective
  - equations of motion (explicit or implicit)
  - additional constraints
  - bounds on free variables: $\mathbf{x}$, $\mathbf{r}$, $\mathbf{p}$
- EoMs can be generated with PyDy (http://pydy.org (http://pydy.org))
- Effficient just-in-time compiled C code is generated for functions that evaluate:
  - objective and its gradient
  - constraints and its Jacobian
- NLP problem automatically formed for IPOPT
- Open source: BSD license
- Written in Python
- http://github.com/csu-hmc/opty (http://github.com/csu-hmc/opty)

Example Code: 1 DoF, 1 Parameter Pendulum

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\theta}(t) \\ \dot{\omega}(t) \end{bmatrix} = \begin{bmatrix} \omega(t) \\ -p\sin\theta(t) \end{bmatrix}$$

Paraphrased from https://github.com/csu-hmc/opty/blob/master/examples/vyasarayani2011.py (https://github.com/csu-hmc/opty/blob/master/examples/vyasarayani2011.py)

Symbolics

```
# Specify symbols for the parameters
p, t = symbols('p, t')

# Specify the functions of time
theta, omega, theta_m = symbols('theta, omega, theta_m', cls=Function)

# Specify the symbolic equations of motion
eom = (Matrix([theta(t), omega(t)]).diff(t) -
       Matrix([omega(t), -p * sin(theta(t))]))

# Specify the symbolic objective function
obj = Integral((theta_m(t) - theta(t))**2, t)
```

Example Code: 1 DoF, 1 Parameter Pendulum

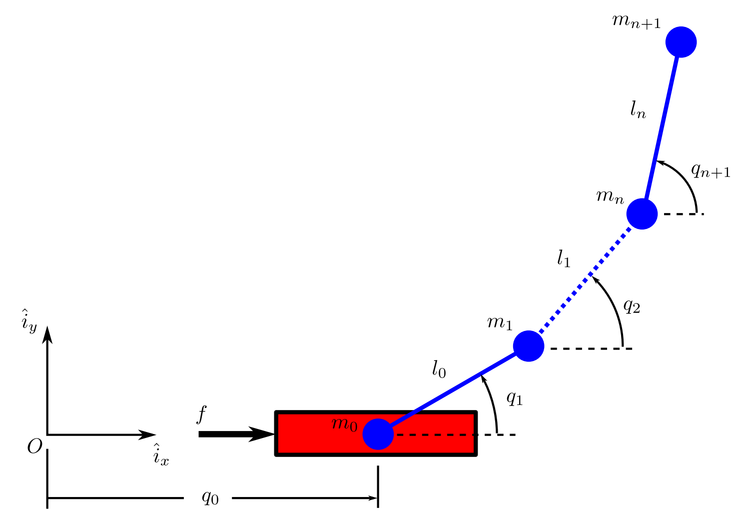
Numerics

```
# Choose discritzation values
num_nodes = 1000
interval = 0.01  # seconds

# Form the problem
prob = Problem(obj, eom, (theta(t), omega(t)), num_nodes, interval,
               known_trajectory_map={y1_m(t): measured_data},
               integration_method='midpoint')

# Set an initial guess
initial_guess = random(prob.num_free)

# Solve the system
solution, info = prob.solve(initial_guess)
```

Computational Speed

Example Larger System

10 link pendulum on sliding cart (not stiff)
11 DoF, 22 states, 22 parameters
12800 mathematical operations in constraint expressions
100 s sampled @ 100 hz



Computational Speed

Discretization Variables

- 10,000 collocation nodes
- 219,978 constraints
- 14,518,548 nonzero entries in the Jacobian
- 220,022 free variables

Timings

- Integrating with ODEPACK lsoda: **5.6 s**
- Constraint evaluation: **33 ms (0.033 s)**
- Jacobian evaluation: **128 ms (0.128 s)**

Case Study: Human Control Parameter Identification

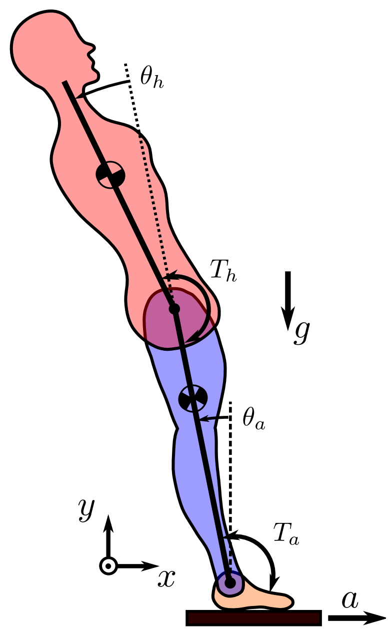
Plant

Torque driven two-link inverted pendulum with an accelerating base.
States: $\mathbf{x} = [\theta_a \quad \theta_h \quad \omega_a \quad \omega_h]^T$

Exogoneous inputs:

- Controlled: $\mathbf{r}_c = [T_a \quad T_h]^T$
- Specified: $\mathbf{r}_k = [a]$

Known parameters: $\mathbf{p}_k$



Open Loop Equations of Motion

$$\dot{\mathbf{x}} = \mathbf{f}_o(\mathbf{x}, \mathbf{r}_c, \mathbf{r}_k, \mathbf{p}_k, t)$$

Lumped Passive+Active Controller

- True human controller is practically impossible to isolate and identify
- Identify a controller for a similar system that causes the same behavior as the real system

Simple State Feedback

$$\mathbf{r}_c(t) = -\mathbf{K}\mathbf{x}(t)$$

Unknown Parameters

$$\mathbf{p}_u = \text{vec}(\mathbf{K})$$

Closed Loop Equations of Motion

$$\dot{\mathbf{x}} = \mathbf{f}_c(\mathbf{x}, \mathbf{r}_k, \mathbf{p}_k, \mathbf{p}_u, t)$$

Generate Data

Specify the psuedo-random platform acceleration

$$a(t) = \sum_{i=1}^{12} A_i \sin(\omega_i t) \quad \text{where,} \quad 0.15\text{rad/s} < \omega_i < 15.0\text{rad/s}$$

Choose a stable controller

$$\mathbf{K} = \begin{bmatrix} 950 & 175 & 185 & 50 \\ 45 & 290 & 60 & 26 \end{bmatrix}$$

Generate Data

Simulate closed loop system under the influence of perturbations for 60 seconds, sampled at 100 Hz

$$\dot{\mathbf{x}} = \mathbf{f}_c(\mathbf{x}, \mathbf{r}_k, \mathbf{p}_k, \mathbf{p}_u, t)$$

Add Gaussian measurement noise

$$\mathbf{x}_m(t) = \mathbf{x}(t) + \mathbf{v}_x(t)$$
$$a_m(t) = a(t) + v_a(t)$$

- σ_θ = 0.3 deg
- σ_ω = 4 deg/s
- σ_a = 0.42 ms^{-2}

Parameter Identification Problem Specification

Given noisy measurements of the states, $\mathbf{x}_m$, and the platform acceleration, a_m, can we identify the controller parameters $\mathbf{K}$?

$$\min_{\theta} J(\theta), \quad J(\theta) = \sum_{i=1}^{N} h[\mathbf{x}_{mi} - \mathbf{x}_i]^2$$

where

$$\theta = [\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{p}_u]$$

Subject to the constraints:

$$\mathbf{f}_c(\mathbf{x}_i, \mathbf{x}_{i+1}, a_{mi}, a_{mi+1}, \mathbf{p}_u) = 0, \quad i = 1 \ldots N$$

And the initial guess:

$$\theta_0 = [\mathbf{x}_{m1}, \ldots, \mathbf{x}_{mN}, \mathbf{0}]$$

For, N = 6000:

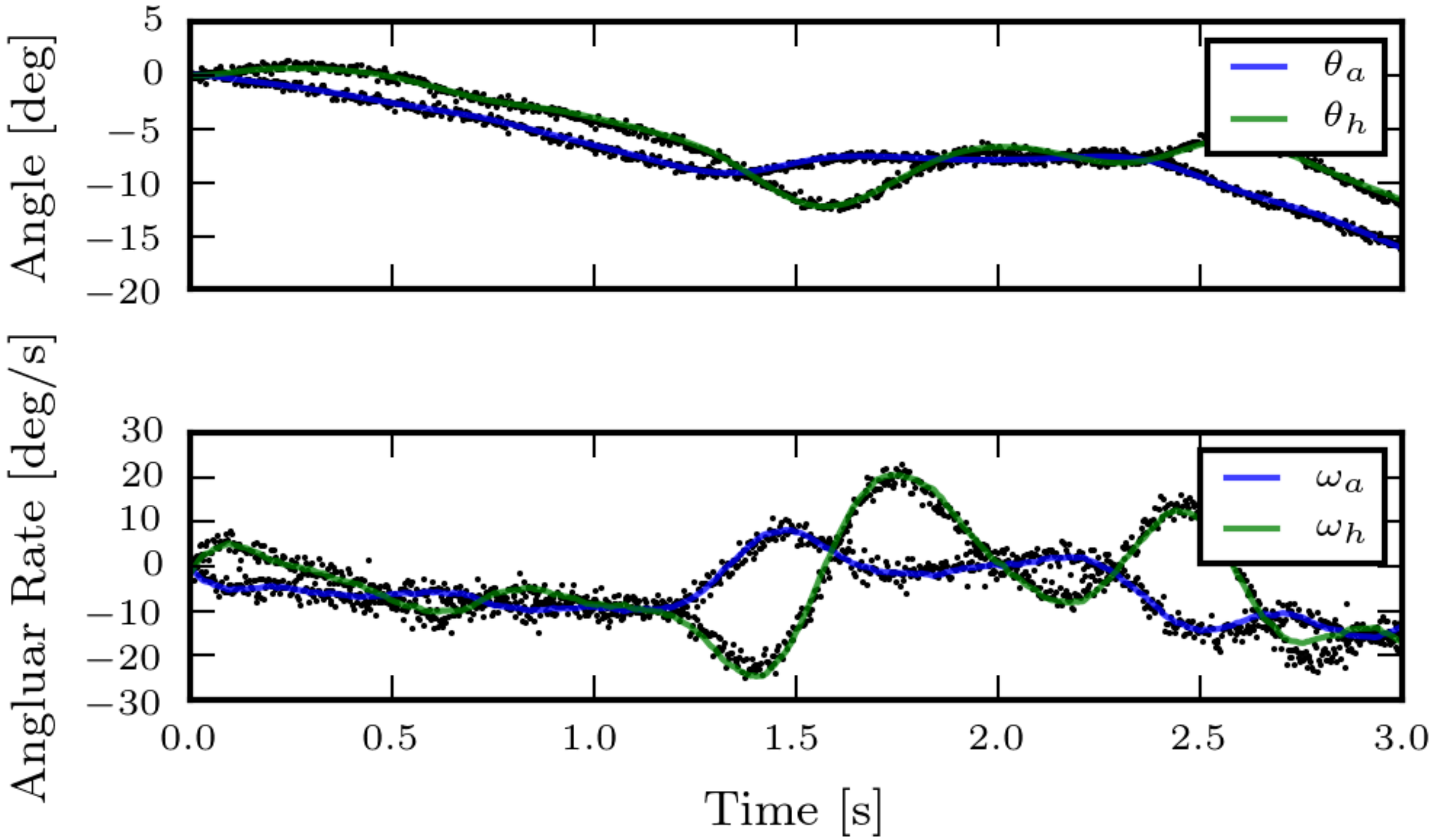
- 24008 free variables
- 23996 x 24008 Jacobian matrix with 384000 non-zero entries

Results

Converges in **11 iterations** in **2.8 seconds** of computation time.

	Known	**Identified**	**Error**
k_{00}	950	946	-0.4%
k_{01}	175	177	1.4%
k_{02}	185	185	-0.2%
k_{03}	50	55	9.4%
k_{10}	45	45	1.1%
k_{11}	290	289	-0.3%
k_{12}	60	59	-2.1%
k_{13}	26	27	4.2%

Identified State Trajectories



Conclusion

- Direct collocation is suitable for biomechanical parameter identification
- Computation speeds are orders of magnitude faster than shooting
- Parameter identification accuracy improves with # nodes
- Complex problems can be solved with few lines of code and high level mathematical abstractions