Adaptive Control Loops for Advanced LIGO

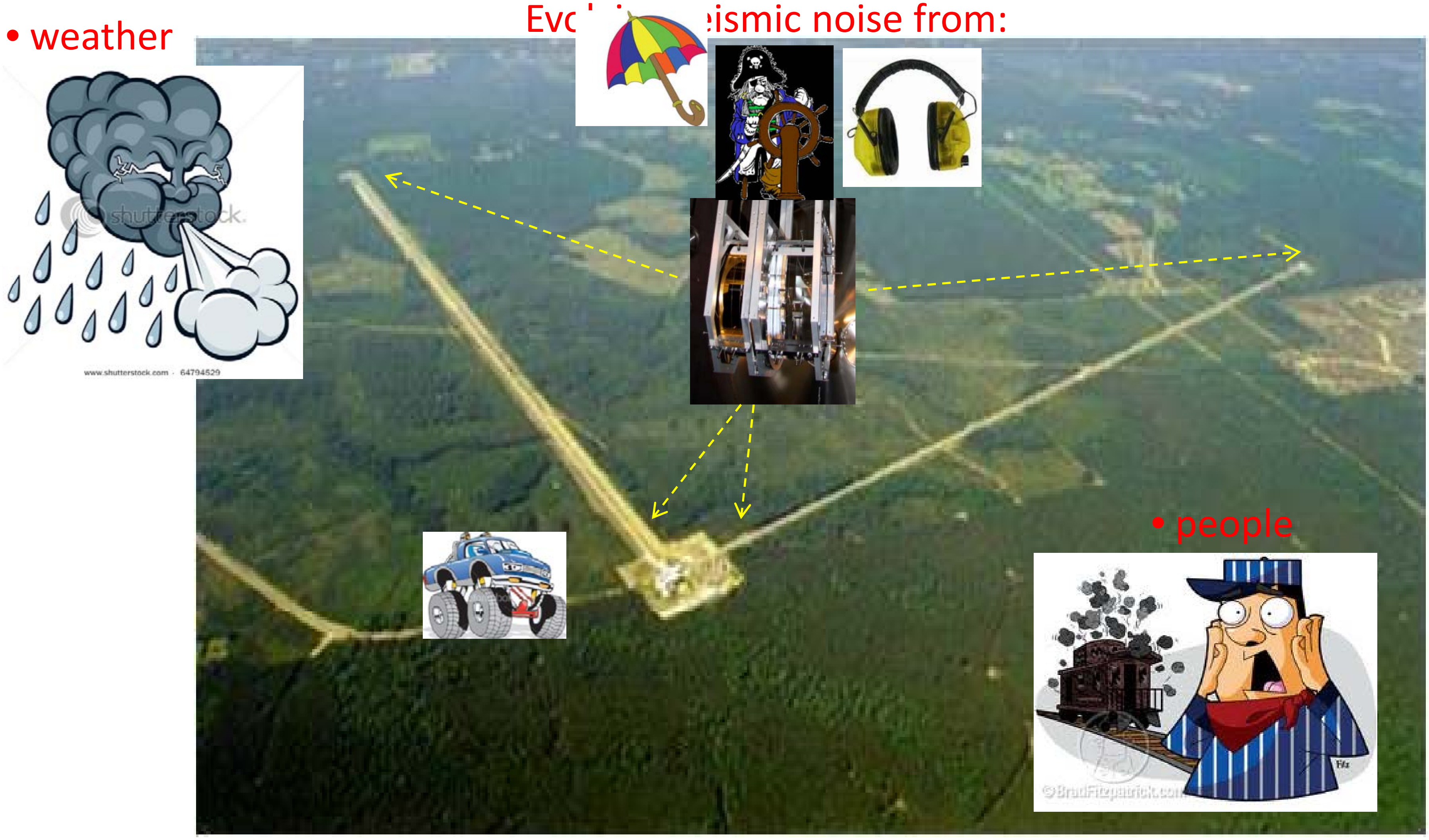
Brett Shapiro

25 February 2011

Control Loops Keep LIGO Running

Evc[illegible]eismic noise from:

- weather

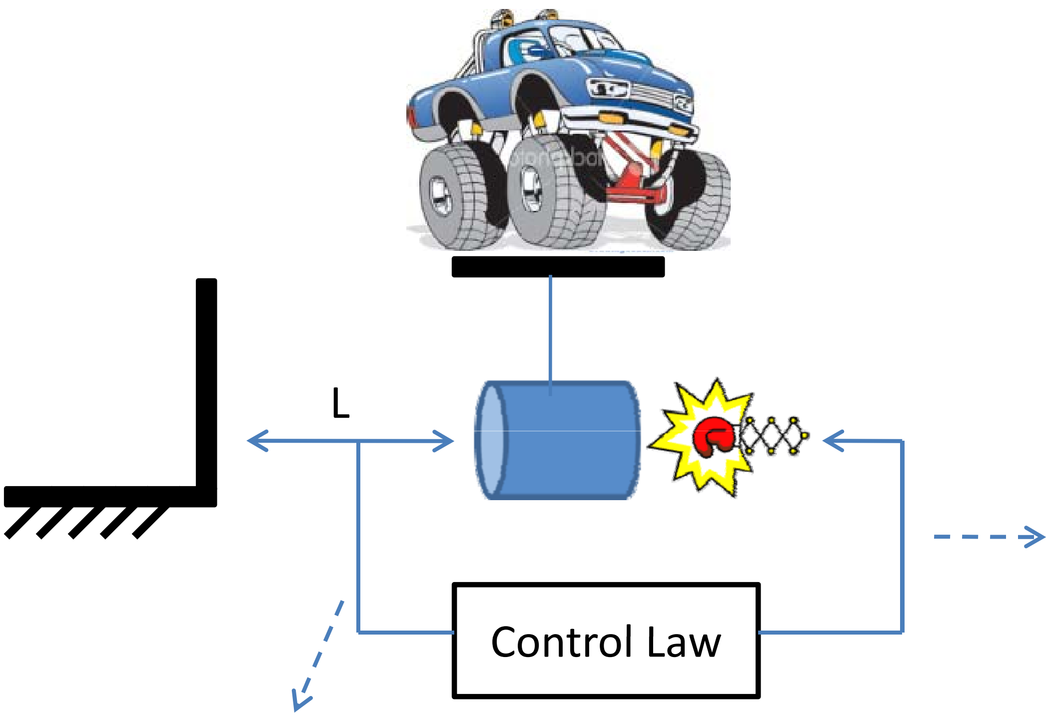


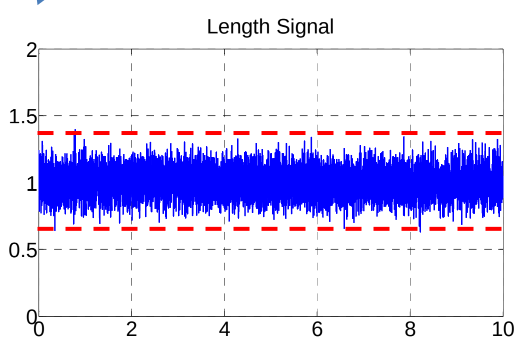
- people

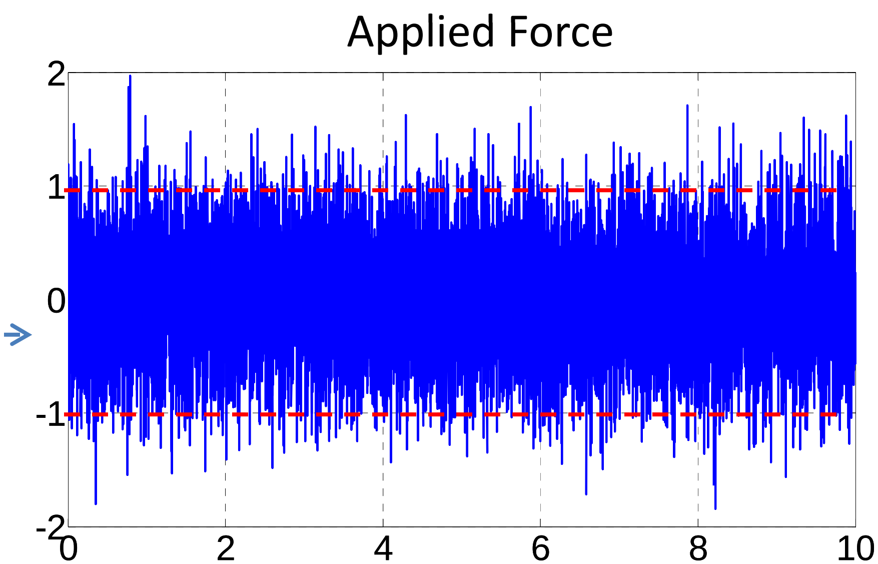
… adaptive control also makes a very good thesis topic…

How are Adaptive Loops Useful?

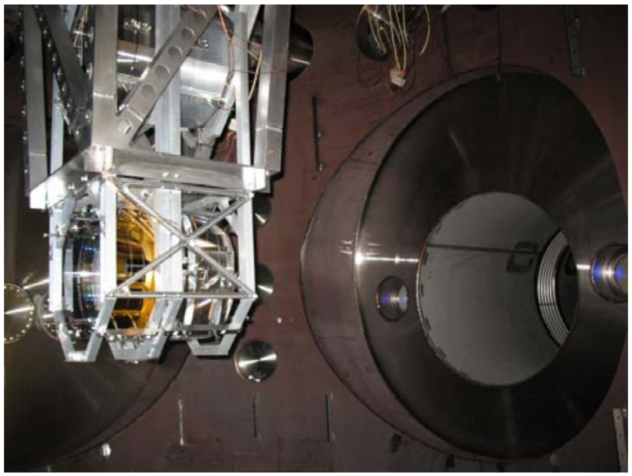
Simple example: Push on the pendulum to maintain a constant L.

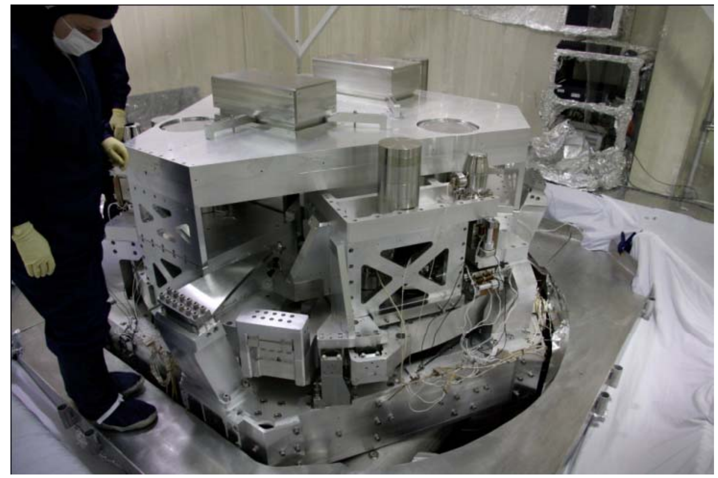






Adaptive Control for Isolation Systems





Suspensions:
- Angular mirror control
- Damping (modal or classical)
- Length control?

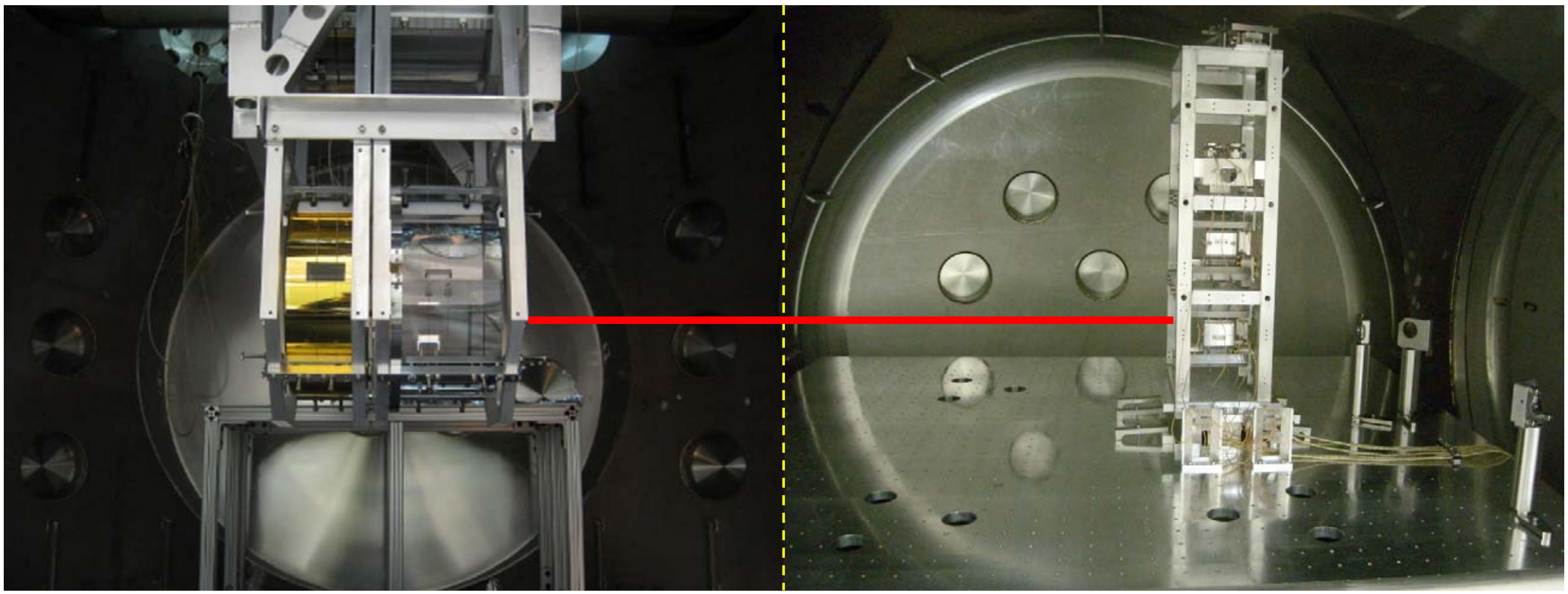
Seismic Control
- Isolation loops
- sensor blending?

Can be used to optimize in real time:
- aLIGO noise budget amplification (from sensor noise, barkhausen noise, etc)
- actuator forces
- RMS error signals
- just about anything else

LASTI Quad-Triple Cavity

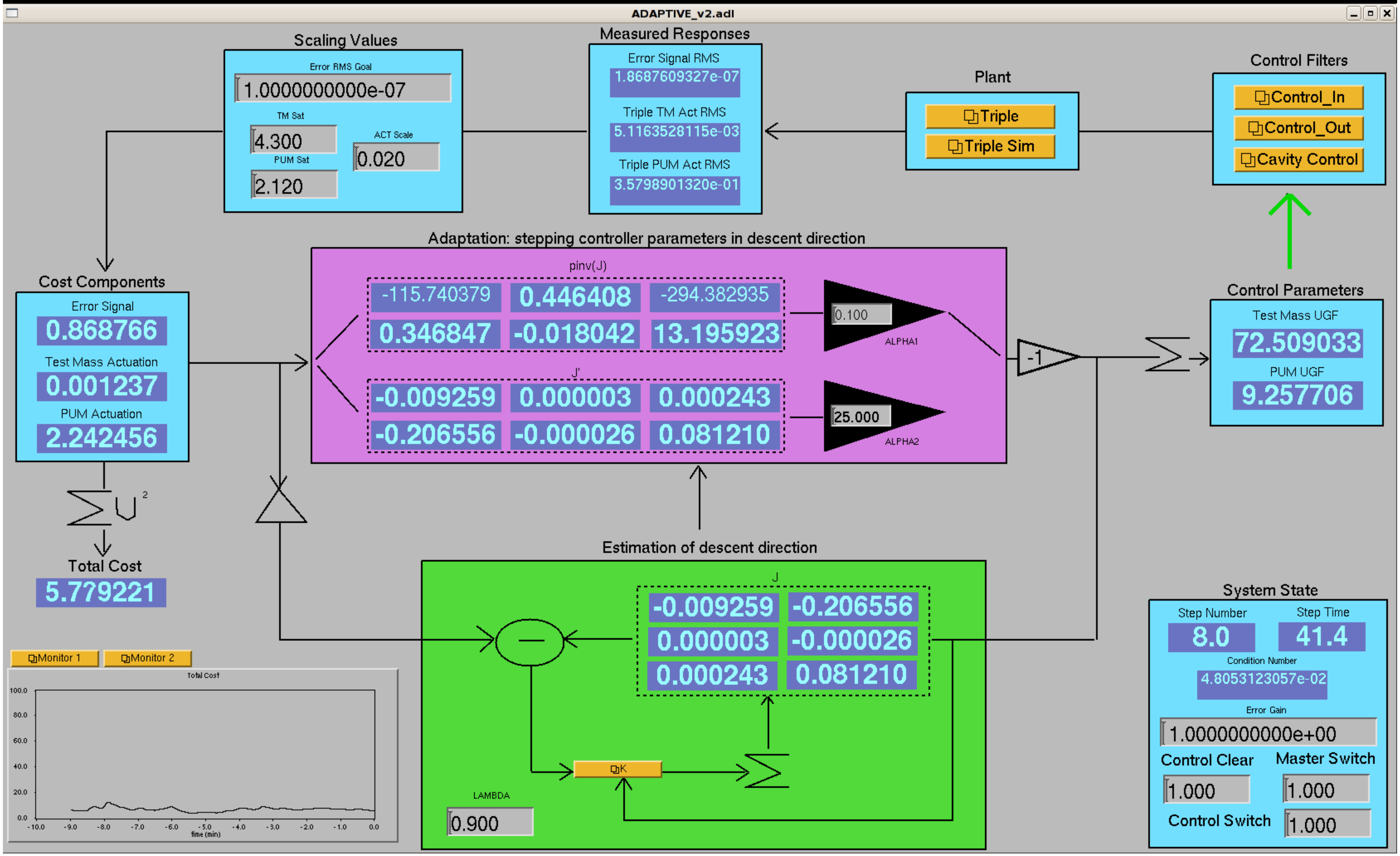
Quad Pendulum

Triple Pendulum



16 meters

Adaptive Control MEDM Screen



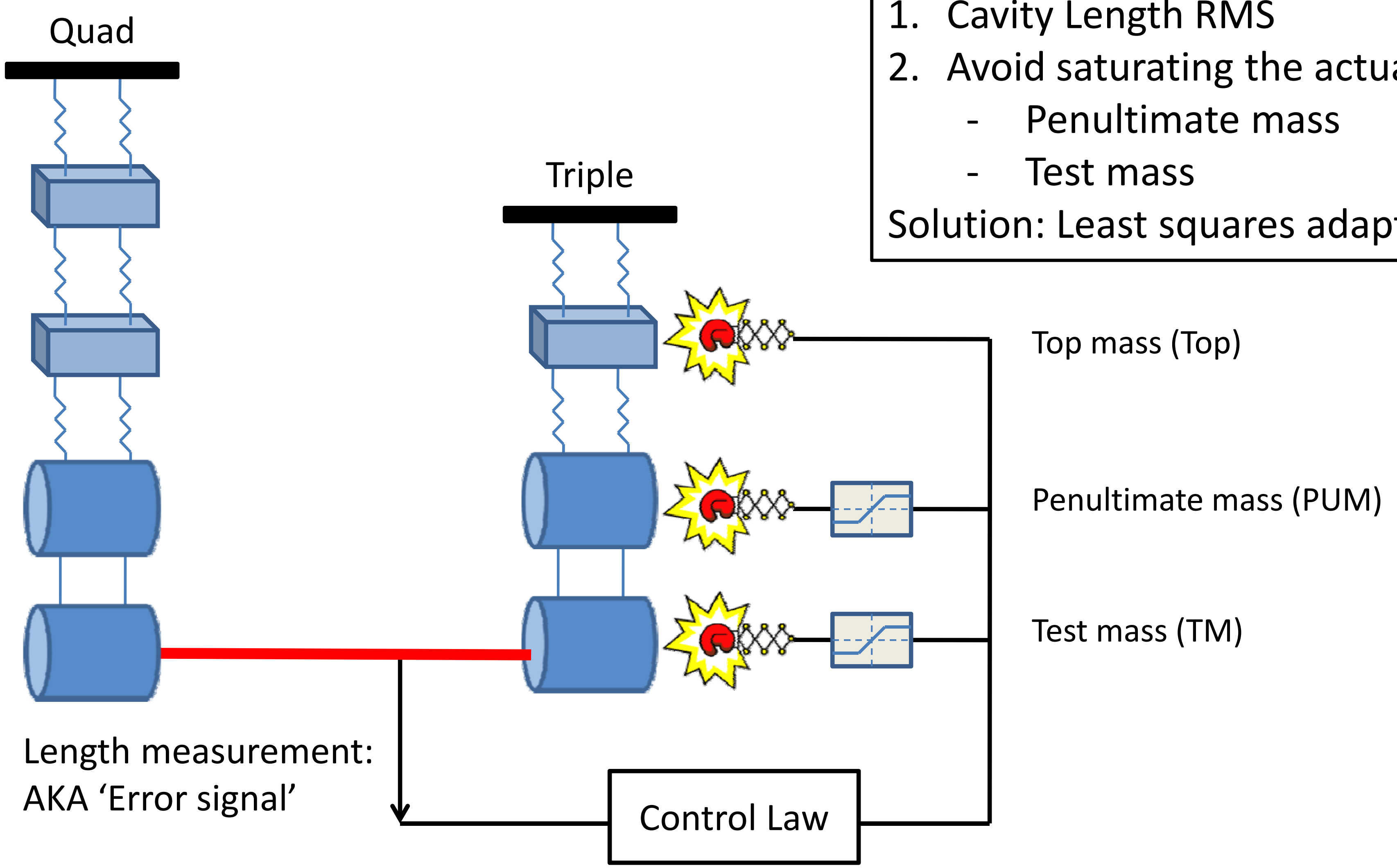
Questions to Answer

1. What is being controlled in this experiment?
2. What is the adaptation optimizing?
3. What parameters are being adapted?
4. How are they being adapted?

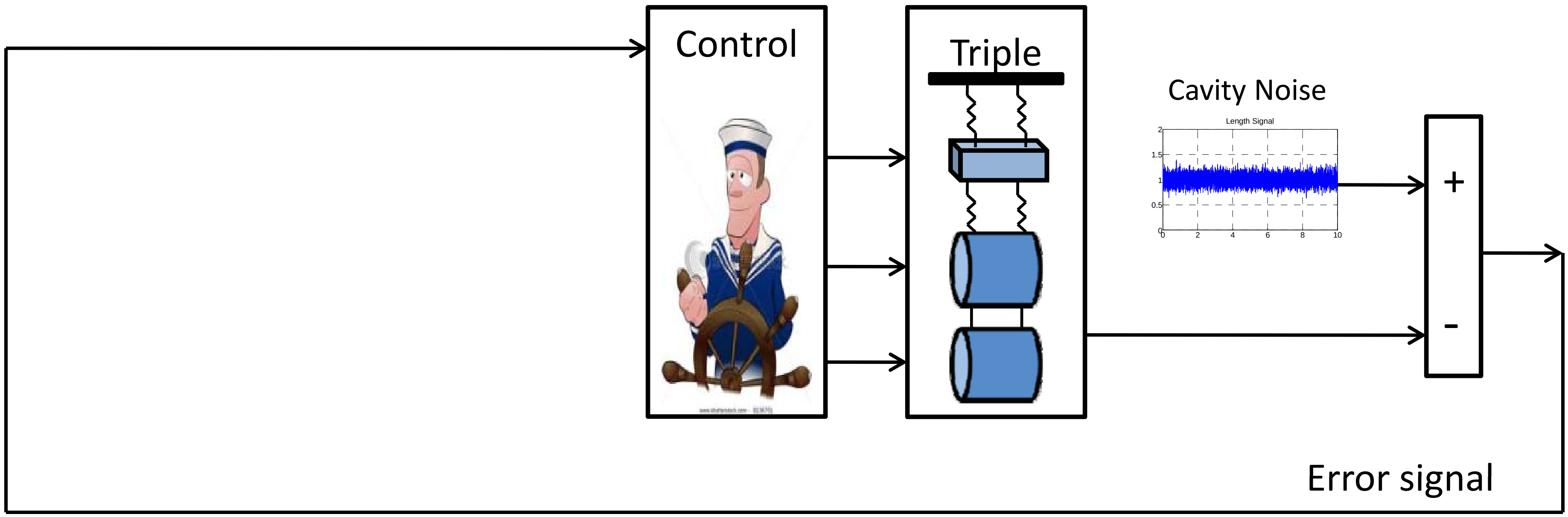
LASTI Experimental Adaptive Setup

Optimization Goals:
1. Cavity Length RMS
2. Avoid saturating the actuators
   - Penultimate mass
   - Test mass

Solution: Least squares adaptation



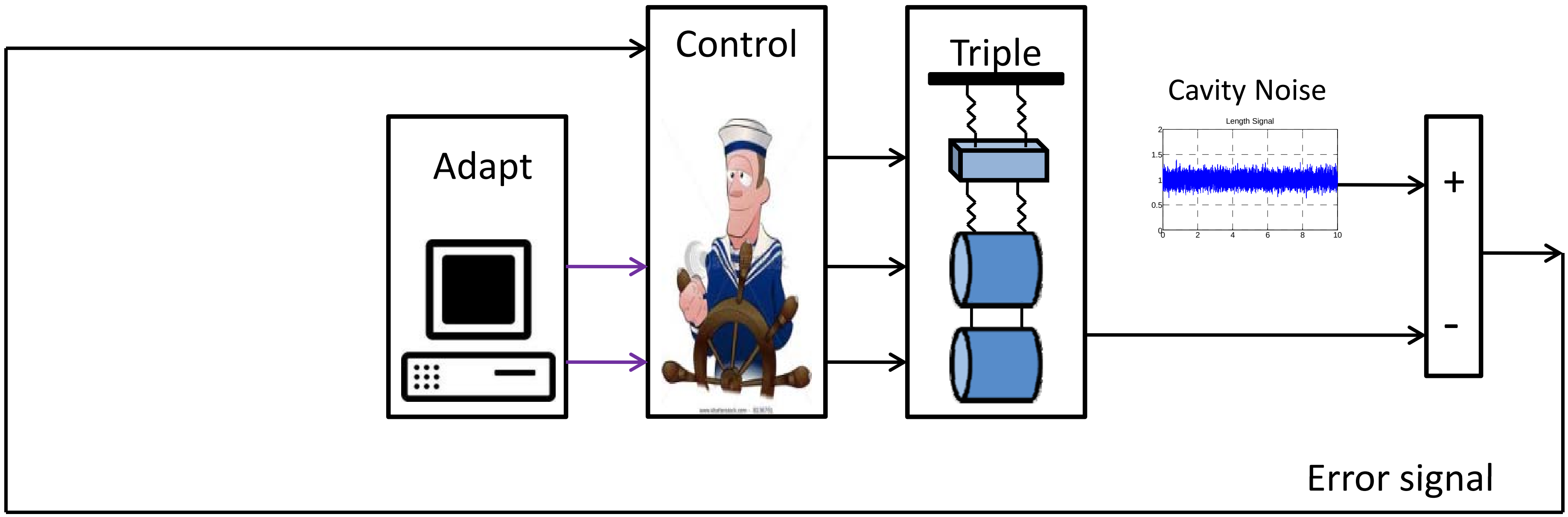
Control Block Diagram Diagram



3 primary components to adaptive control:

1. Control

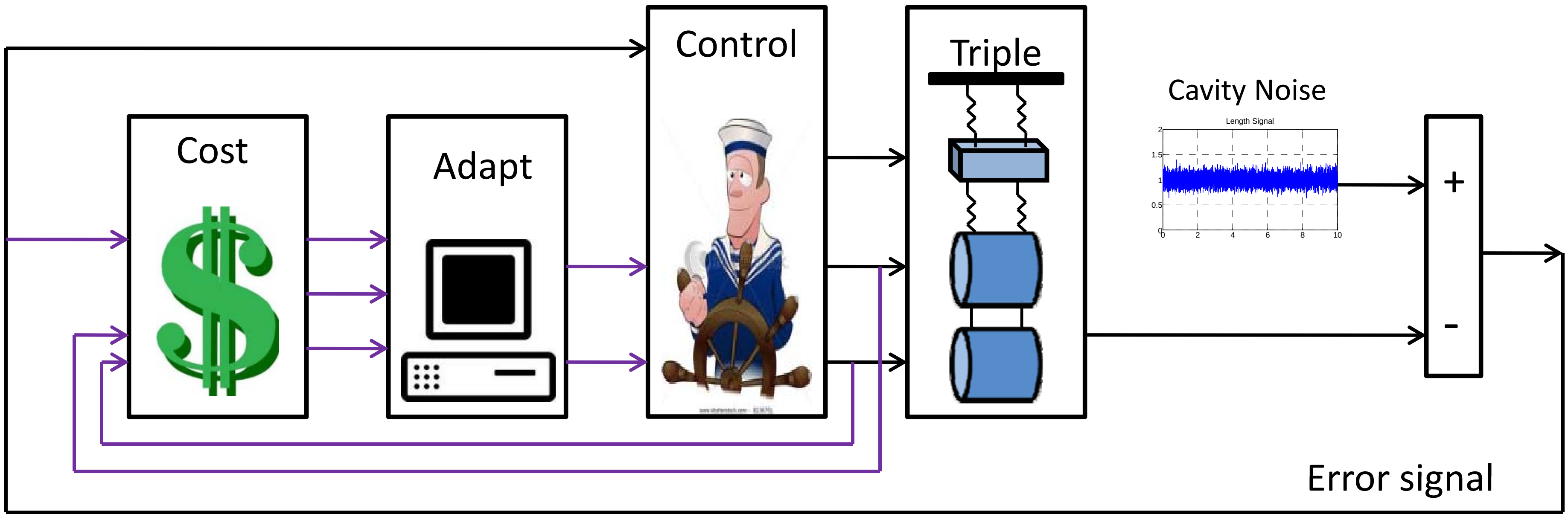
Adaptive Control Architecture



3 primary components to adaptive control:

1. Control – parameterized in terms of adapting parameters
2. Adaptation algorithm – updates control parameters

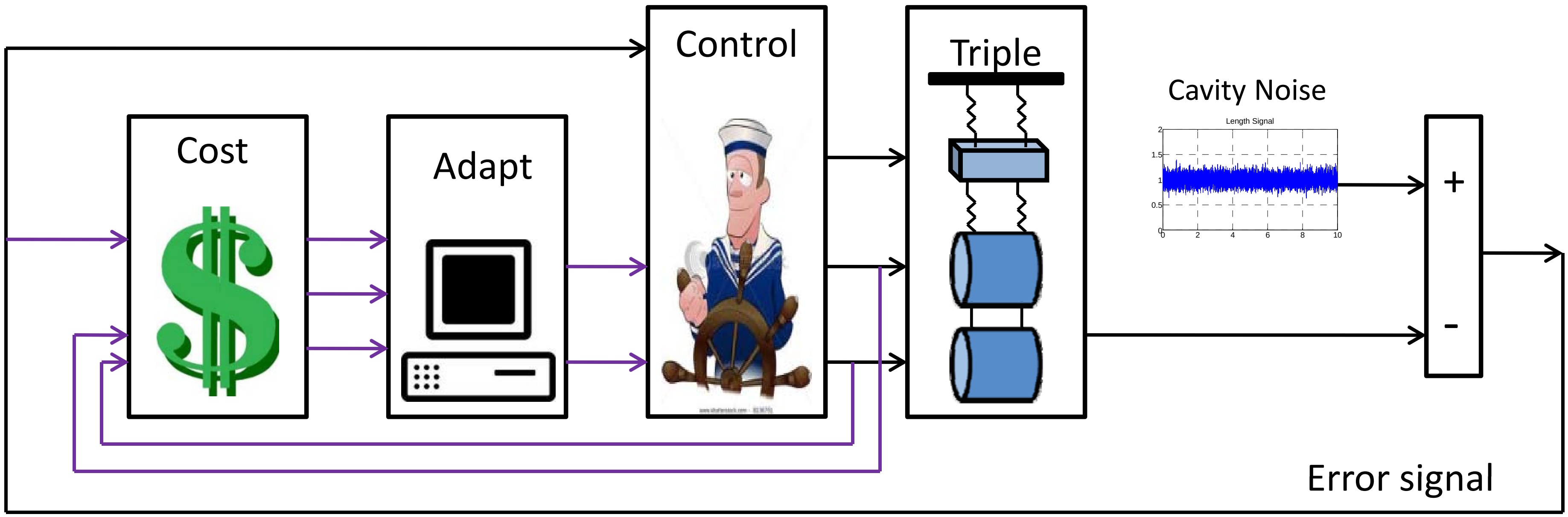
Adaptive Control Architecture



3 primary components to adaptive control:

1. Control – parameterized in terms of adapting parameters
2. Adaptation algorithm – updates control parameters
3. Costs – variables that are optimized with adaptation

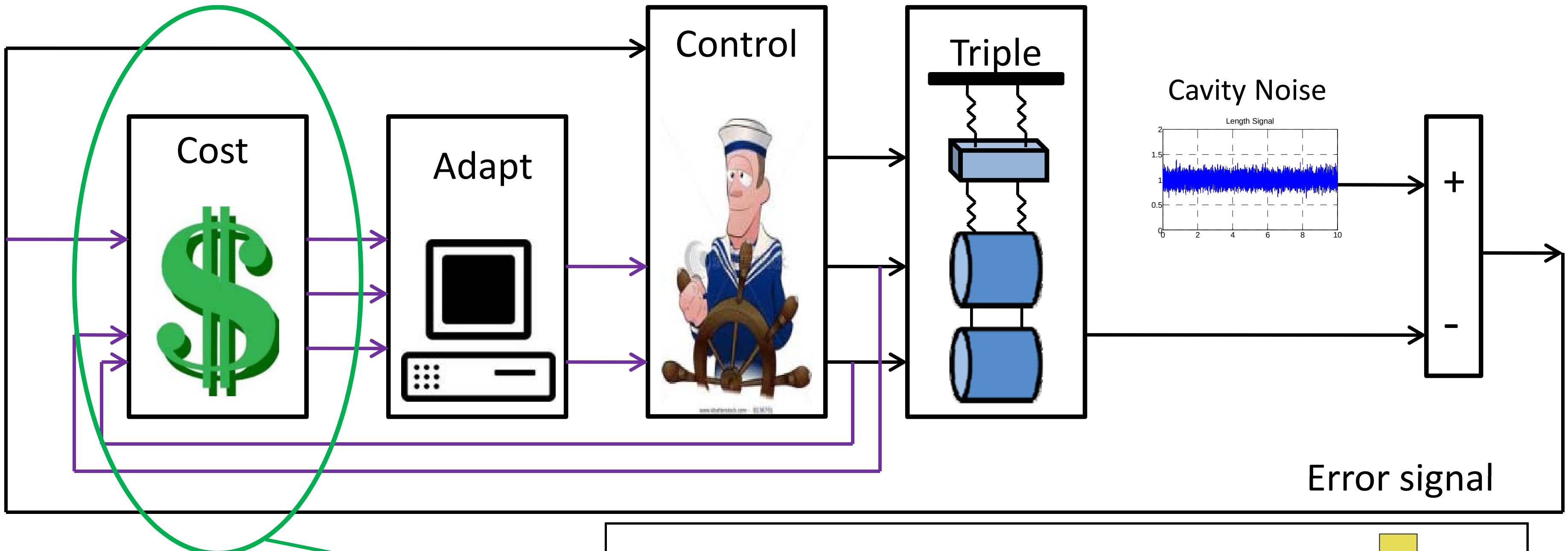
Adaptive Control Architecture

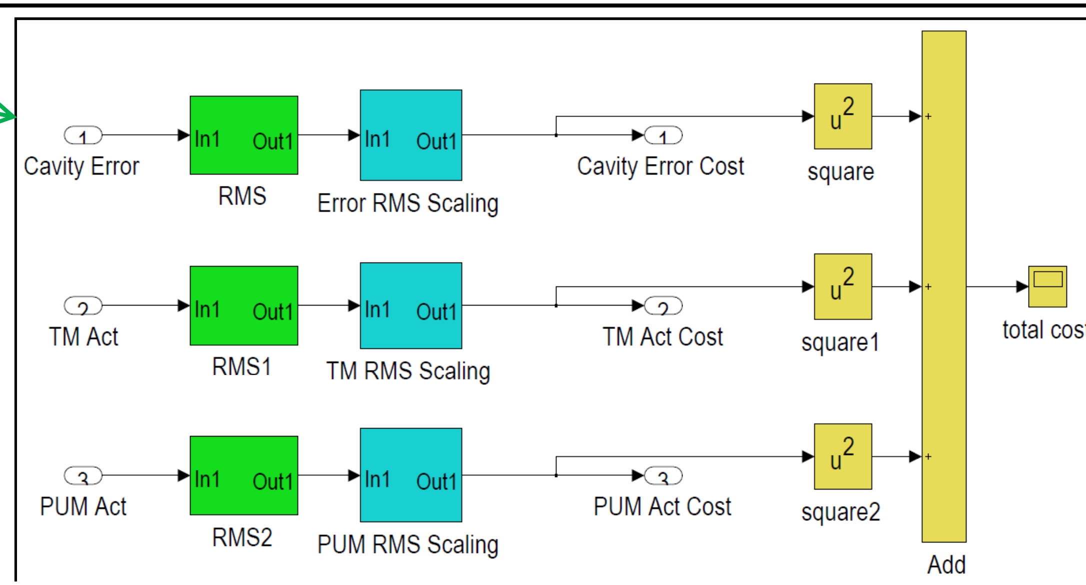


3 primary components to adaptive control:

1. Control – parameterized in terms of adapting parameters
2. Adaptation algorithm – updates control parameters
   - uses a least squares optimization routine
3. Costs – variables that are optimized with adaptation

Cost Box: Summation of Costs

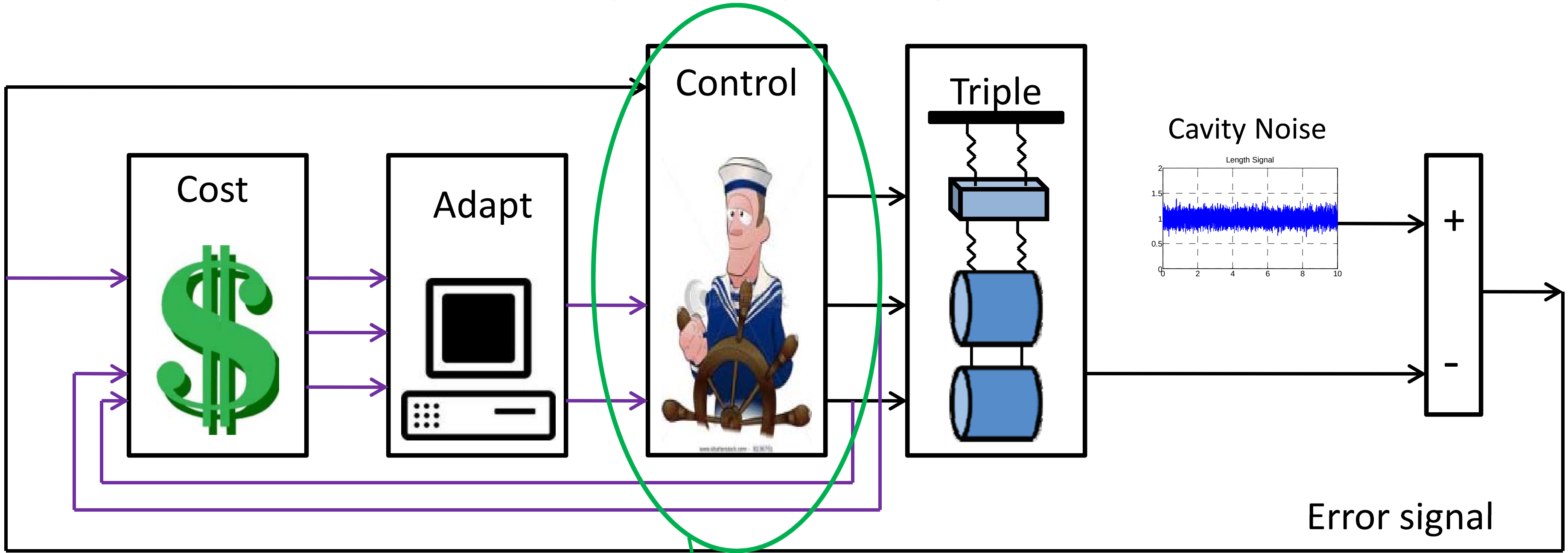




The cost box measures the performance values we care about and scales them to get the costs:

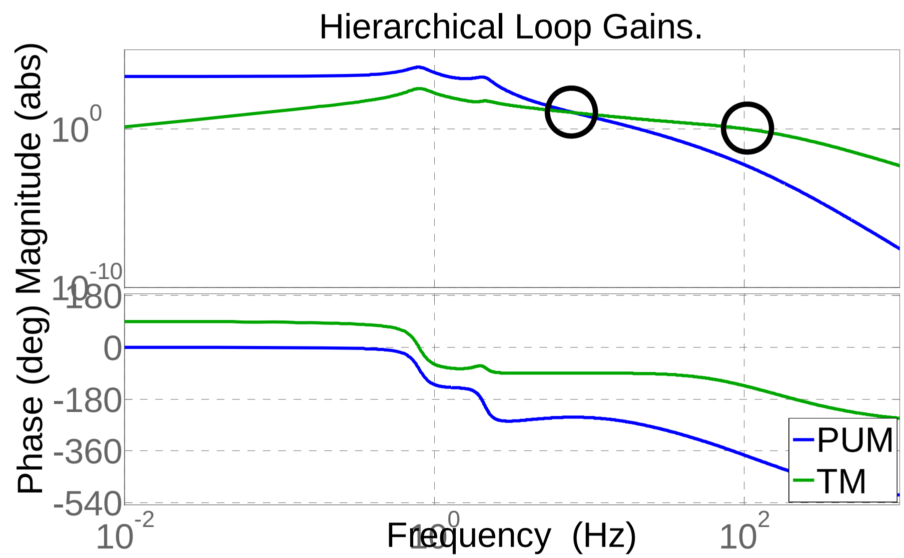
1. Error RMS
2. PUM force RMS
3. Test mass force RMS

Control Box

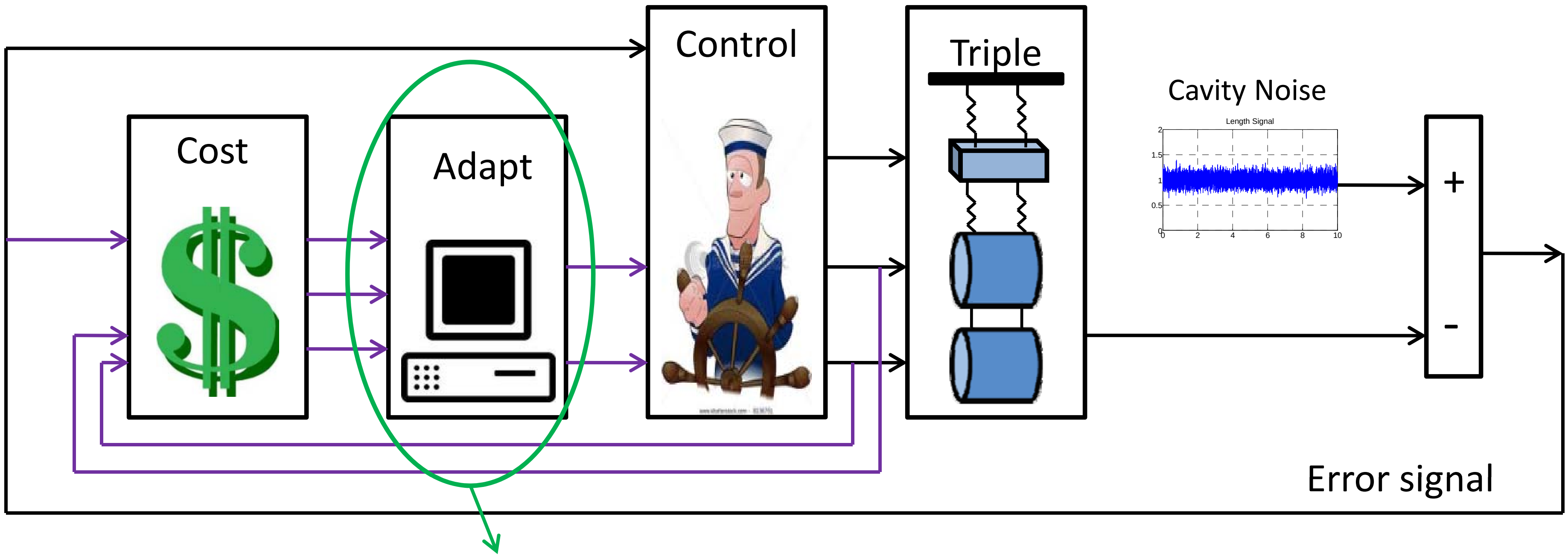


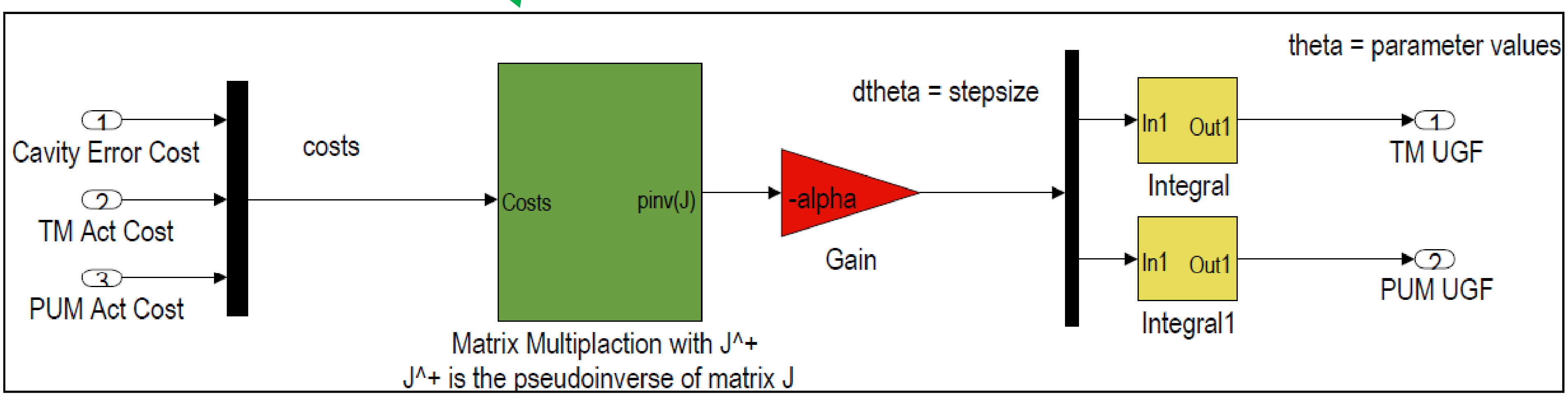
Control filters are parameterized in terms of:

- PUM crossover frequencies
- Test mass crossover frequency

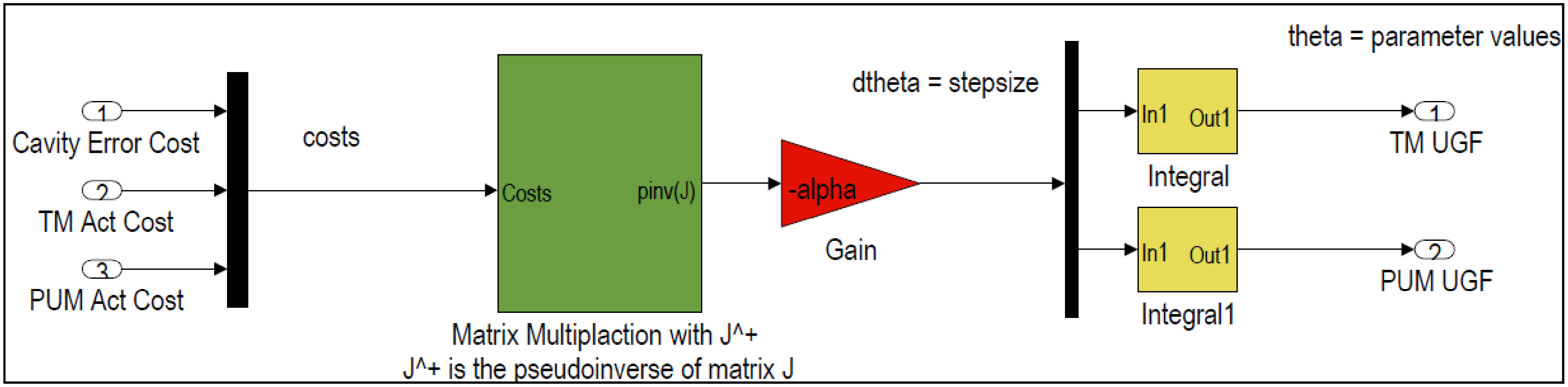


Adaptation Block





Adaptation Algorithm: Least Squares Minimization Approach



Definitions for optimization

- Total system cost ----- $v = \frac{1}{2}\sum_i c_i^2 = \frac{1}{2}\vec{c}^T\vec{c}$
- Cost gradient ------------ $\frac{\partial v}{\partial \vec{\theta}} = \left(\frac{\partial \vec{c}}{\partial \vec{\theta}}\right)^T \vec{c}$
- System Jacobian matrix ------- $J = \left(\frac{\partial \vec{c}}{\partial \vec{\theta}}\right)$

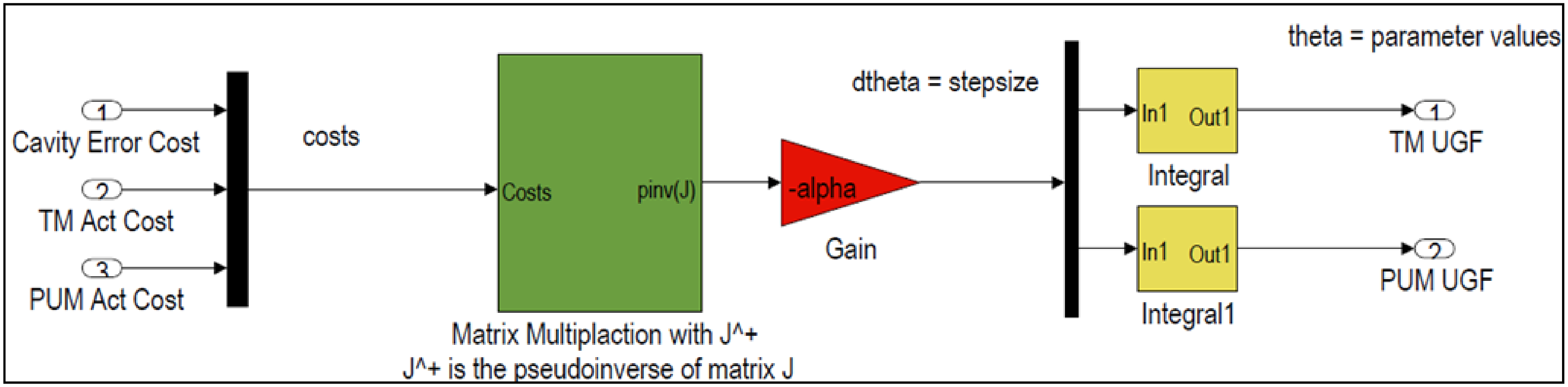
Optimization routine

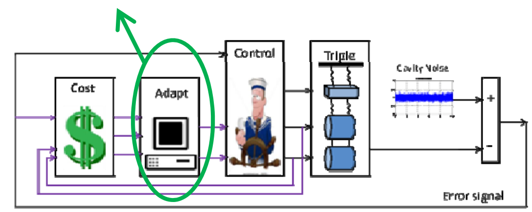
J^T yields gradient descent (1st order).
J^{-1} yields Gauss-Newton (2nd order).

Variable list

v = total cost
c = list of costs we want to optimize
θ = list of adjustable control parameters
J = System Jacobian matrix

Adaptation Algorithm: quadratic minimization approach





Adaptation Problem

- Question: What is J?

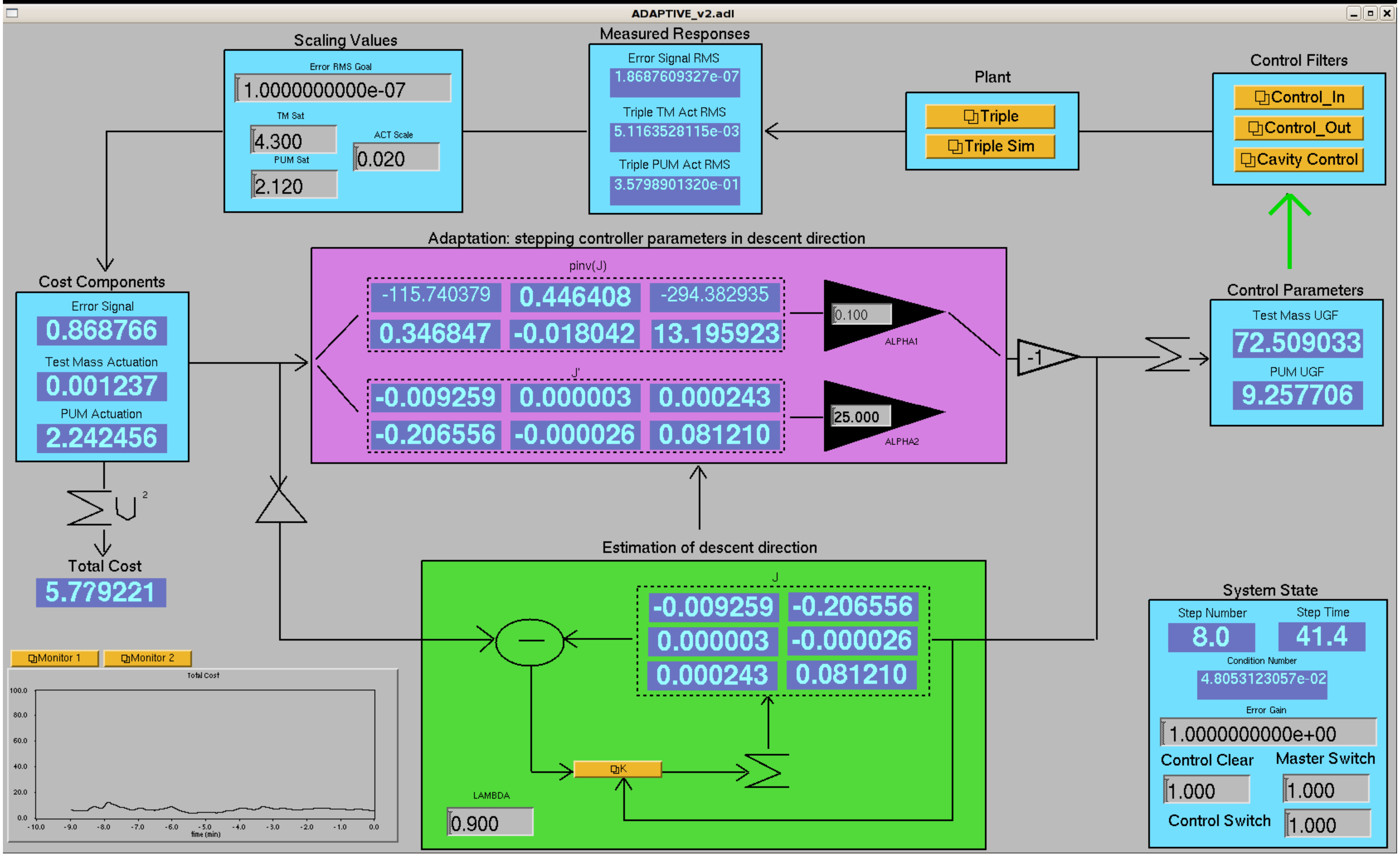
- Answer: We do not know. It depends on θ and c, and other unknown or unmodeled parameters.

- Good news: we can estimate it in real-time.
A recursive least squares algorithm (RLS) is used.

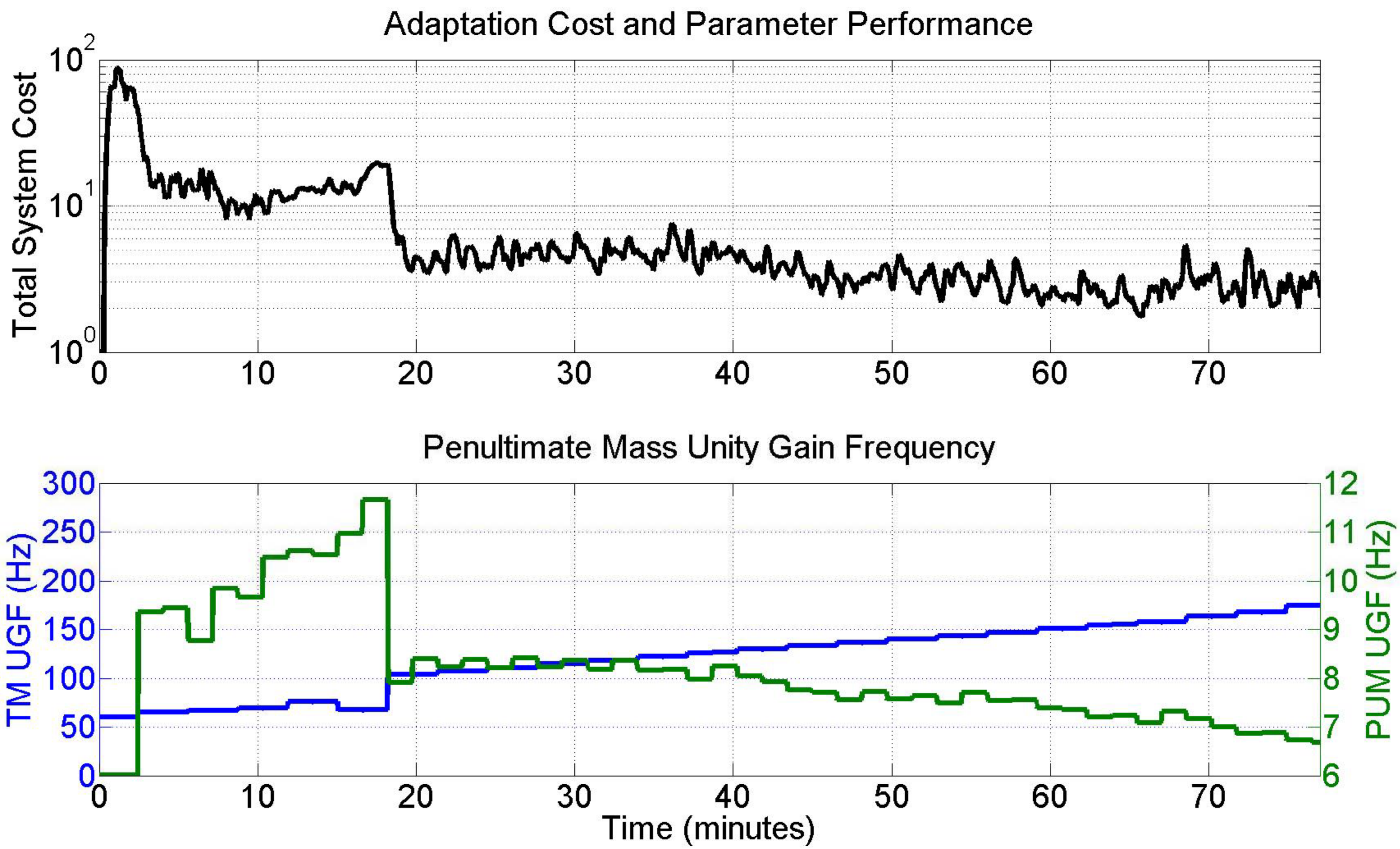
Variable list

v = total cost
c = list of costs we want to optimize
θ = list of adjustable control parameters
J = system Jacobian matrix
α = user defined scalar step size

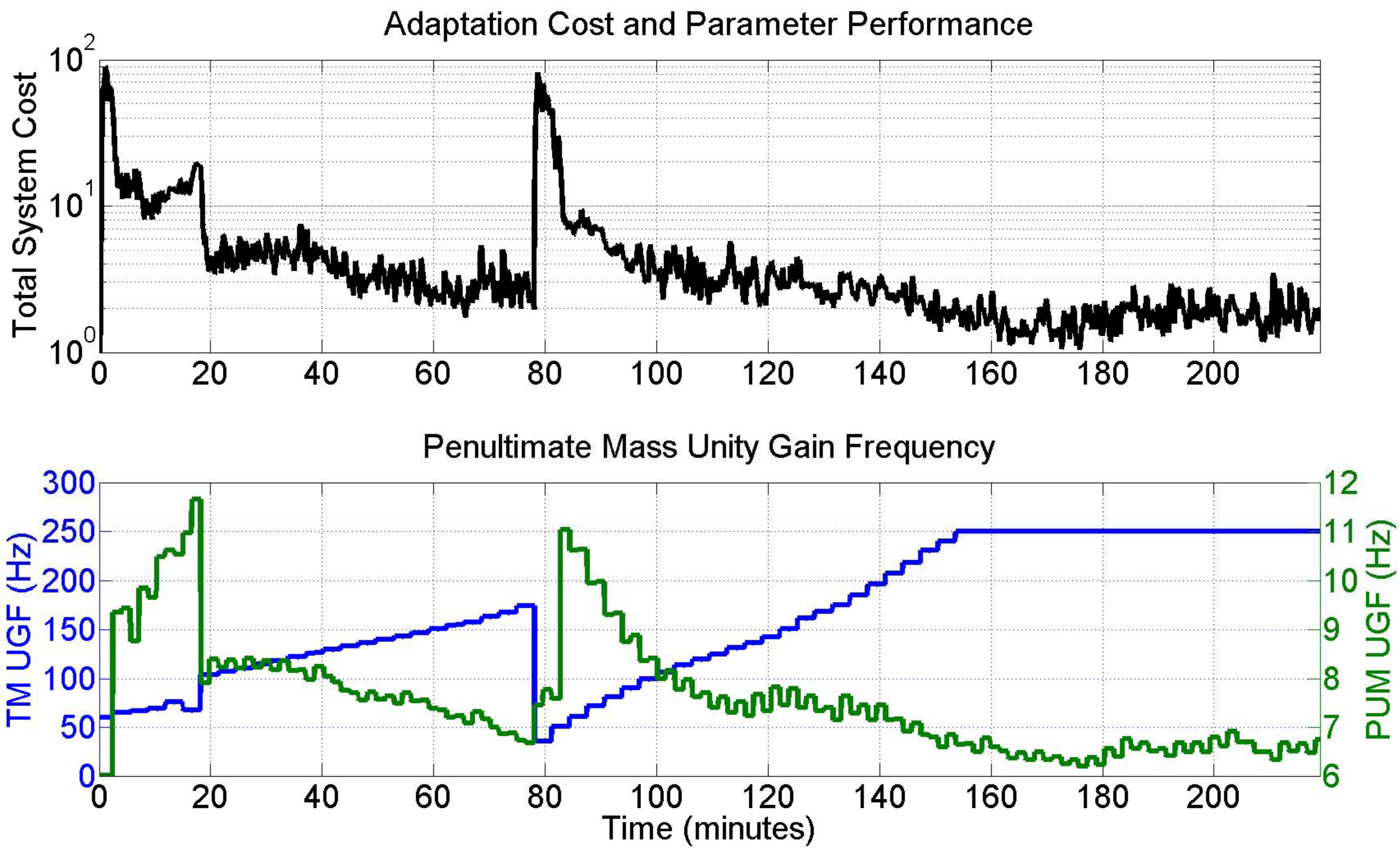
Results


ADAPTIVE_v2.adl
Scaling Values
Error RMS Goal
1.0000000000e-07
TM Sat
4.300
PUM Sat
2.120
ACT Scale
0.020
Measured Responses
Error Signal RMS
1.8687609327e-07
Triple TM Act RMS
5.1163528115e-03
Triple PUM Act RMS
3.5798901320e-01
Plant
Triple
Triple Sim
Control Filters
Control_In
Control_Out
Cavity Control
Adaptation: stepping controller parameters in descent direction
pinv(J)
-115.740379
0.446408
-294.382935
0.346847
-0.018042
13.195923
J'
-0.009259
0.000003
0.000243
-0.206556
-0.000026
0.081210
0.100
ALPHA1
25.000
ALPHA2
-1
Cost Components
Error Signal
0.868766
Test Mass Actuation
0.001237
PUM Actuation
2.242456
Total Cost
5.779221
Control Parameters
Test Mass UGF
72.509033
PUM UGF
9.257706
Estimation of descent direction
J
-0.009259
-0.206556
0.000003
-0.000026
0.000243
0.081210
K
LAMBDA
0.900
Monitor 1
Monitor 2
Total Cost
time (min)
System State
Step Number
8.0
Step Time
41.4
Condition Number
4.8053123057e-02
Error Gain
1.0000000000e+00
Control Clear
1.000
Master Switch
1.000
Control Switch
1.000

Simulated Results

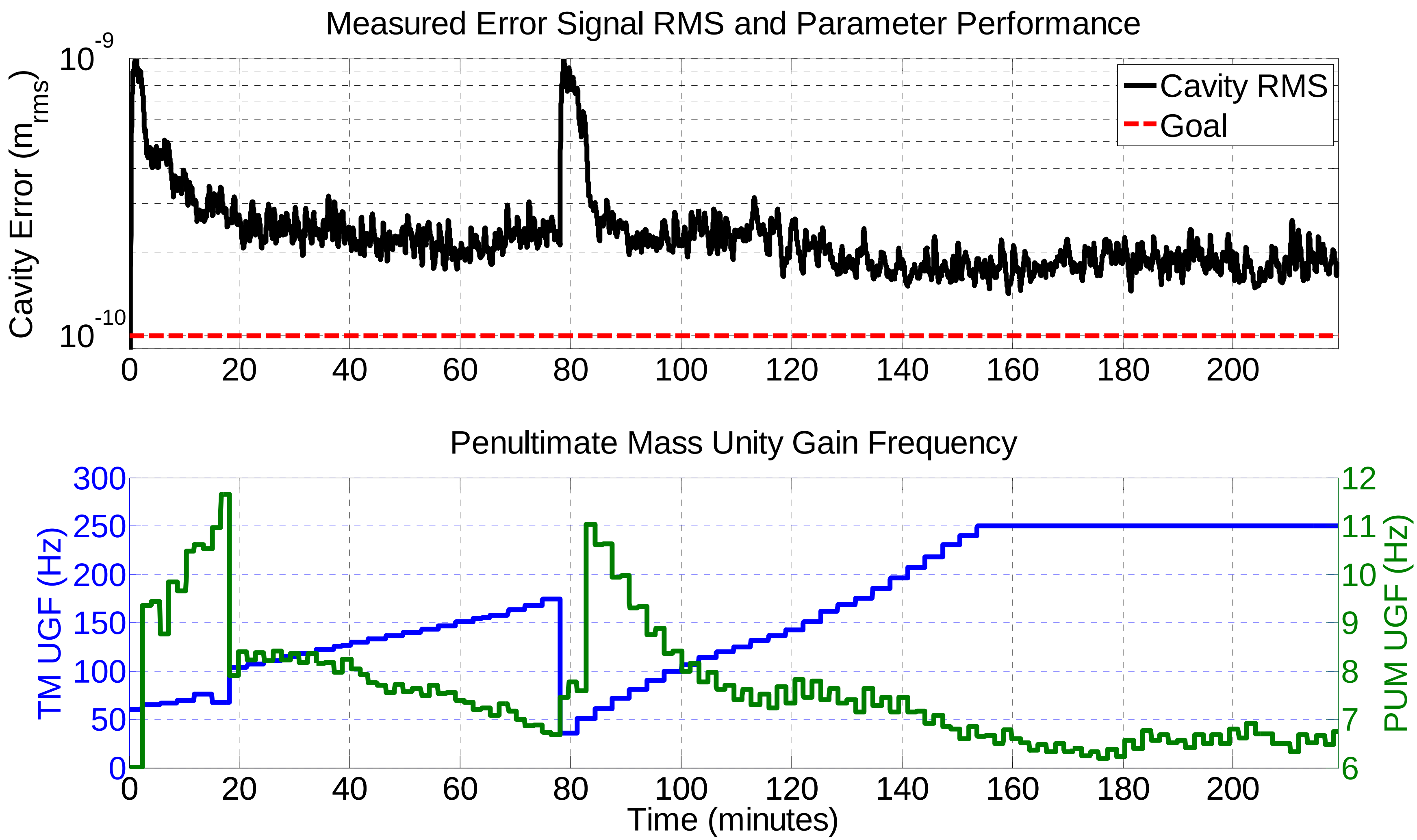


Results



Adaptation Cost and Parameter Performance
Total System Cost
10^2
10^1
10^0
0
20
40
60
80
100
120
140
160
180
200
Penultimate Mass Unity Gain Frequency
TM UGF (Hz)
300
250
200
150
100
50
0
PUM UGF (Hz)
12
11
10
9
8
7
6
Time (minutes)

Results



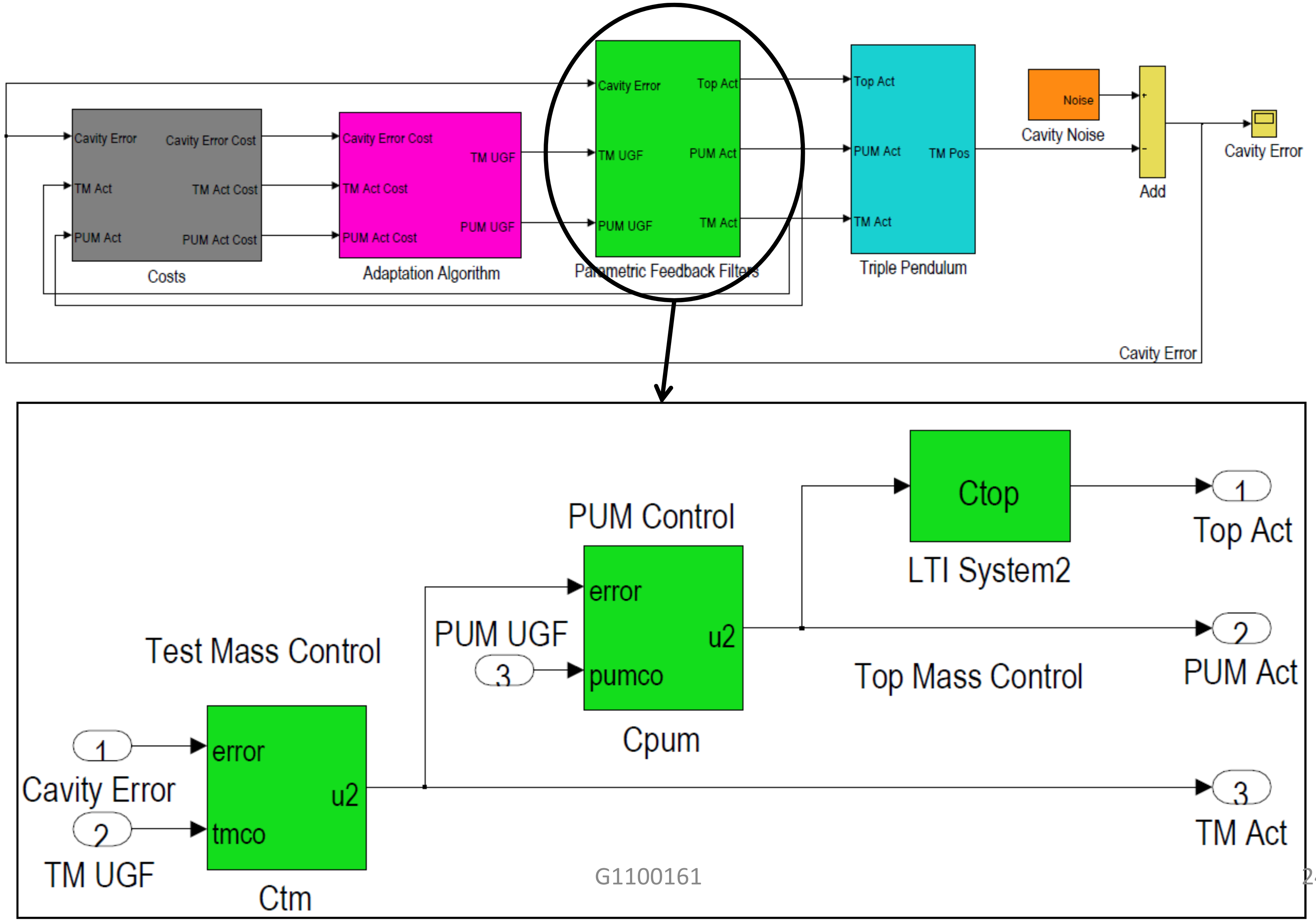
Measured Error Signal RMS and Parameter Performance
Cavity Error (m_{rms})
10^{-9}
10^{-10}
Cavity RMS
Goal
0
20
40
60
80
100
120
140
160
180
200
Penultimate Mass Unity Gain Frequency
TM UGF (Hz)
PUM UGF (Hz)
300
250
200
150
100
50
0
12
11
10
9
8
7
6
Time (minutes)

Conclusions

- Adaptive control is a powerful real-time self-tuning method for many aLIGO loops.
  - Can target arbitrary performance requirements:
    - Avoiding actuator saturations
    - Minimizing noise amplification
- Real-time RLS estimation of system response compensates for unknowns adequately.
- Adaptation speed is limited by RMS averaging
- Complexities beyond this talk exist
  - Achieving good estimates with the Jacobian
  - Setting good stopping and starting conditions

Backups

Feedback Filter Box



Transfer Function Canonical forms

$$\mathbf{G}(s) = \frac{n_1 s^3 + n_2 s^2 + n_3 s + n_4}{s^4 + d_1 s^3 + d_2 s^2 + d_3 s + d_4}.$$

Laplace Transfer function

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -d_1 & 1 & 0 & 0 \\ -d_2 & 0 & 1 & 0 \\ -d_3 & 0 & 0 & 1 \\ -d_4 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix} \mathbf{u}(t)$$

$$\mathbf{y}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}(t).$$

State space transfer function in observer canonical form

Parametric Transfer Functions

Test mass feedback filter: Laplace form

$$\frac{y(s)}{u(s)} = K\omega_{ugf}^3 \frac{s + \frac{\omega_{ugf}}{p}}{\left(s + p\omega_{ugf}\right)\left(s + f\omega_{ugf}\right)} = K\omega_{ugf}^3 \frac{s + \frac{\omega_{ugf}}{p}}{s^2 + (f + p)\omega_{ugf} s + fp\omega_{ugf}^2}$$

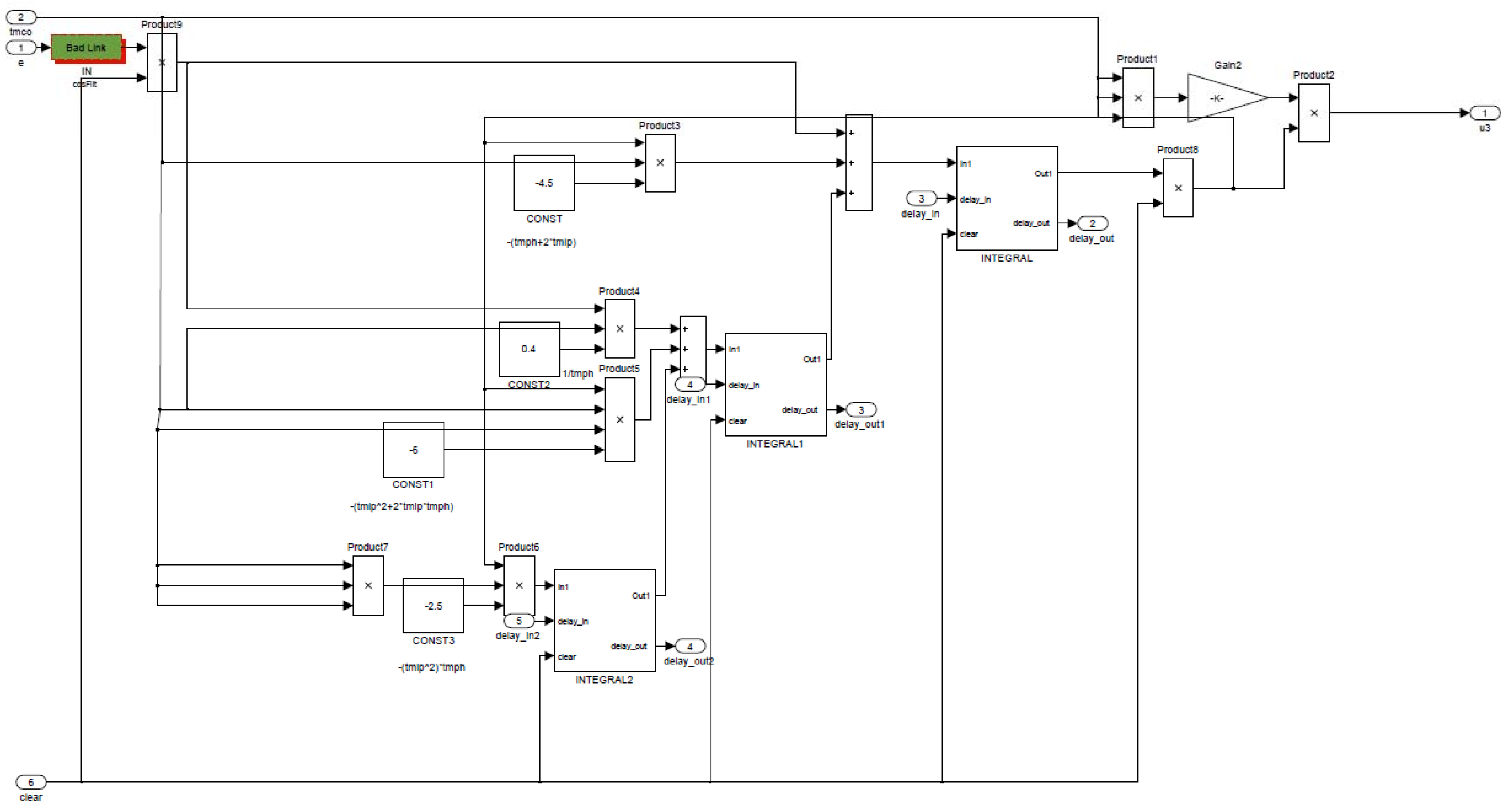
Test mass feedback filter: State space form

$$\dot{x}(t) = \begin{bmatrix} -(f + p)\omega_{ugf} & 1 \\ -fp\omega_{ugf}^2 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ \omega_{ugf}/p \end{bmatrix} u(t)$$

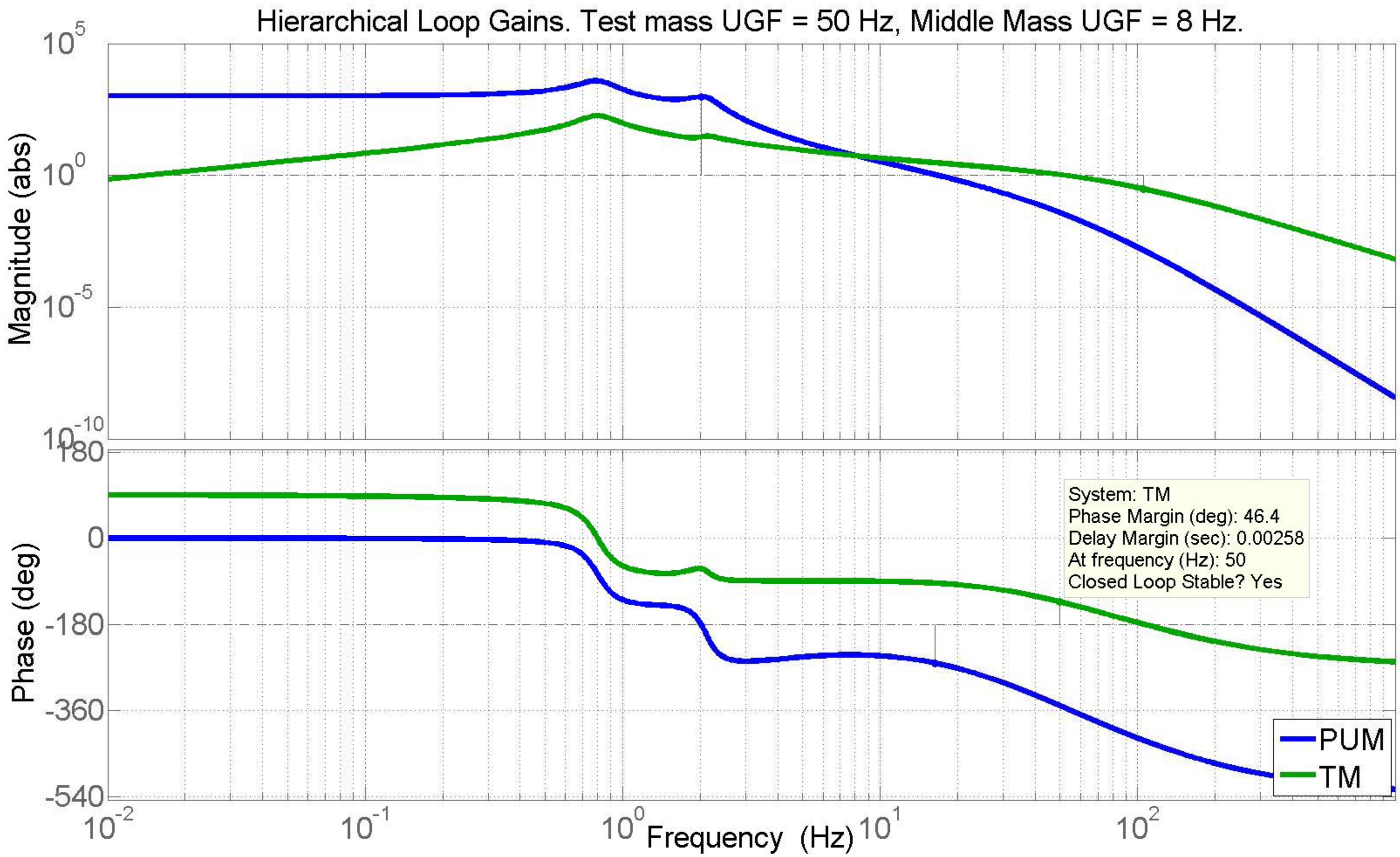
$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t)$$

ω_{ugf} = unity gain freq.
p -> gives phase margin around ugf
f -> f*ω_{ugf} is the freq. of a low pass pole
K is a constant scaling factor which compensates for the plant gain

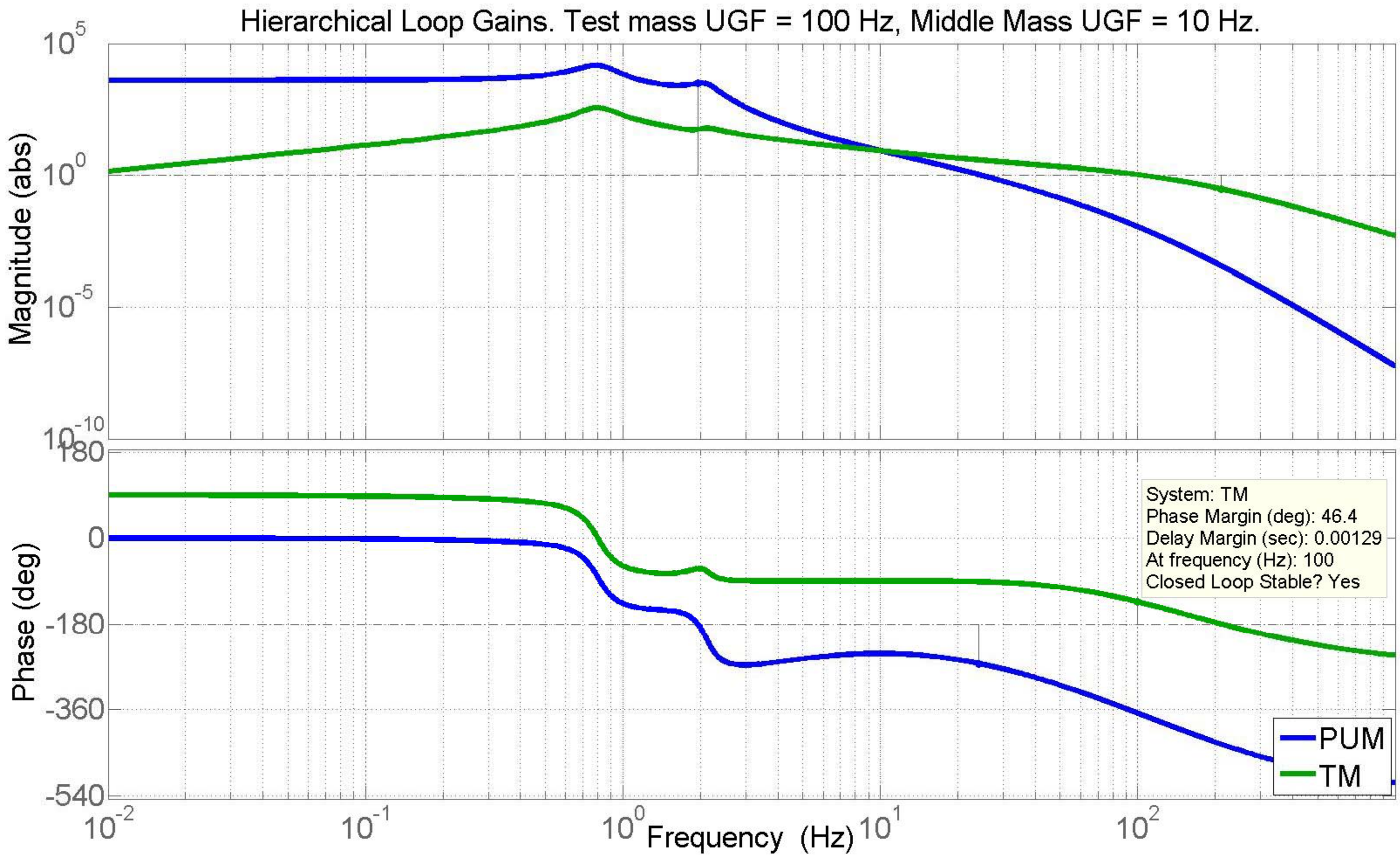
Parametric Filter in LIGO's RCG Simulink Environment



Parametric Controller TFs

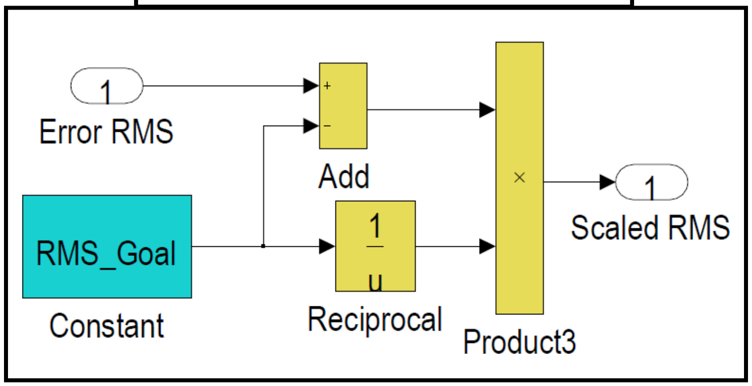


Parametric Controller TFs

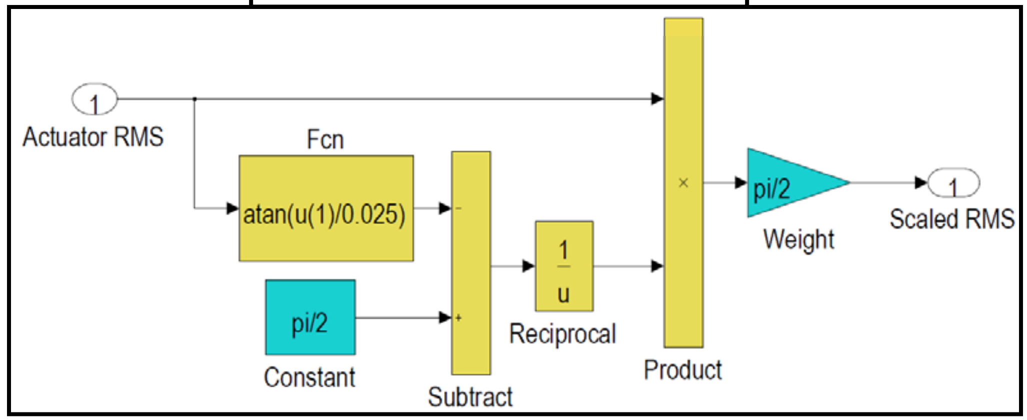


Cost Box Details

Error RMS scaling



Actuator RMS scaling



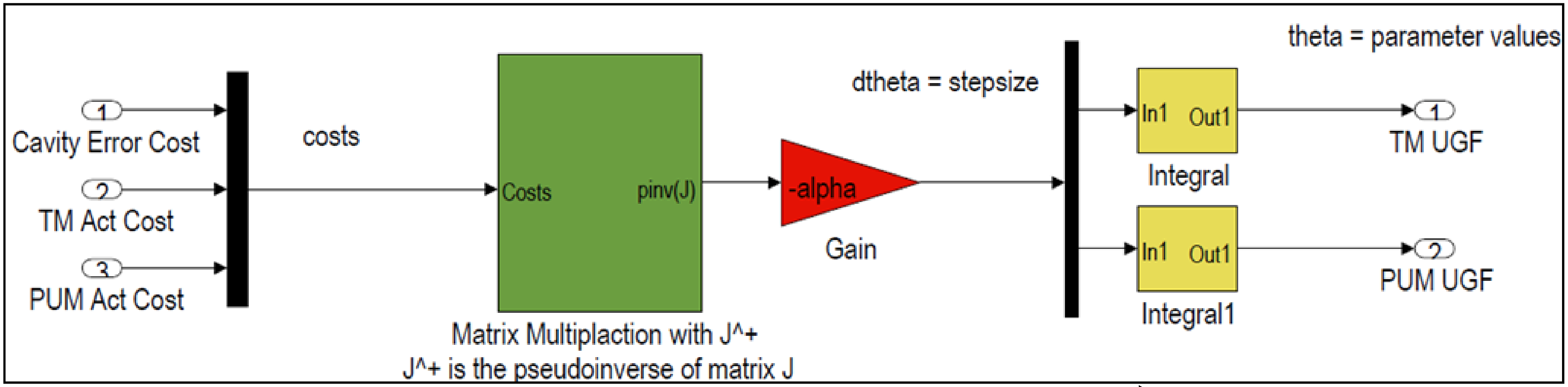
$$E_{Scaled,RMS} = \frac{E_{RMS} - E_{Goal,RMS}}{E_{Goal,RMS}}$$

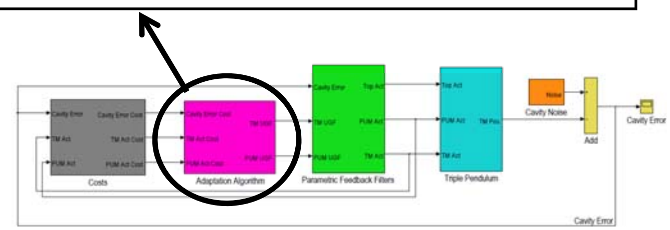
$$U_{Scaled,RMS} = \frac{\frac{\pi}{2} U_{RMS}}{\frac{\pi}{2} - \tan^{-1}(U_{RMS} / 0.025)}$$

The error RMS is scaled relative to a target value (such as the design value).

The actuator RMS is scaled relative to its saturation limit. In this case, the scaled value is set to ‘explode’ when the RMS reaches the saturation limit.

Adaptation Algorithm: quadratic minimization approach





Cost Reduction Choices

1st order algorithm: gradient descent method

$$\vec{\theta}_{t+1} = \vec{\theta}_t - \alpha J^T \vec{c}$$

2nd order algorithm: Gauss-Newton method

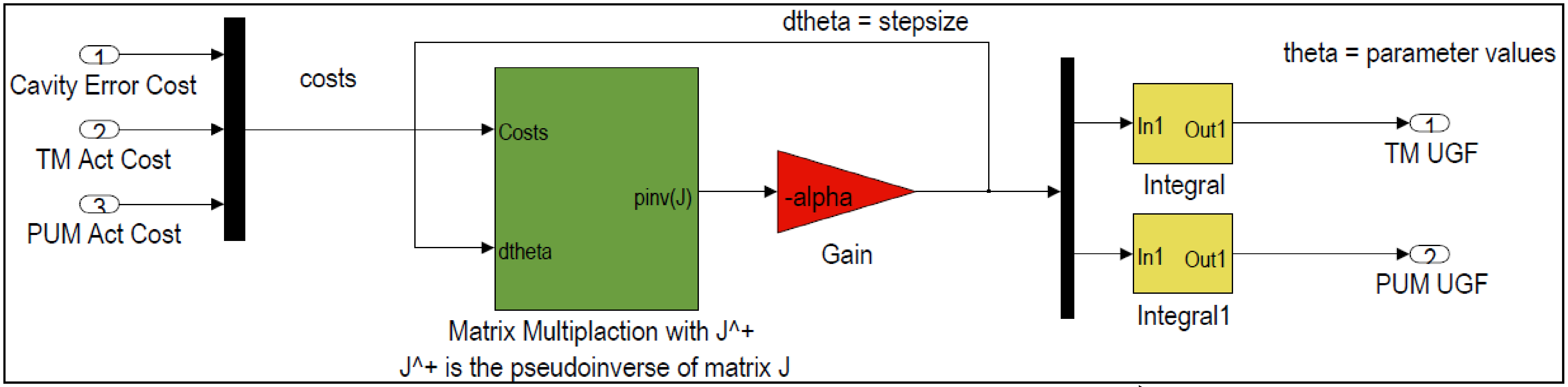
$$\vec{\theta}_{t+1} = \vec{\theta}_t - \alpha J^{-1} \vec{c}$$ For invertible J

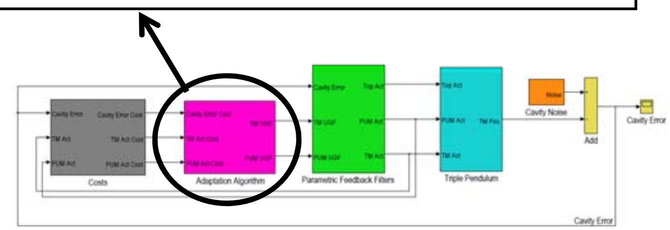
$$\boxed{\vec{\theta}_{t+1} = \vec{\theta}_t - \alpha J^{+} \vec{c}}$$ For noninvertible J (+ -> pseudoinverse)

Variable list

v = total cost
c = list of costs we want to optimize
θ = list of adjustable control parameters
J = system Jacobian matrix
α = user defined scalar step size
t = current time step

Adaptation Algorithm: quadratic minimization approach





Adaptation Problem

Jacobian definition ------------------- $J = \left(\dfrac{\partial \vec{c}}{\partial \vec{\theta}} \right)$

Approximate error --------- $e = \Delta\vec{c} - J\Delta\vec{\theta} \approx 0$

RLS updating ------ $J_{t+1} = J_t + K(\Delta\vec{c} - J_t\Delta\vec{\theta})$

K = Jacobian update gain

Variable list

v = total cost
c = list of costs we want to optimize
θ = list of adjustable control parameters
J = system Jacobian matrix
α = user defined scalar step size
t = current time step

RLS for Jacobian Estimation

$$V(\theta) = \frac{1}{2}\sum_{n=1}^{t} \lambda^{t-n}(\Delta\vec{c}_n - \Delta\vec{\theta}_n^T J_{i,n})^2$$

Cost function optimized by RLS

Iterative RLS algorithm

0. Initialize J and P

1. $$J_{i,t+1} = J_{i,t} + K(\Delta\vec{c}_t - \Delta\vec{\theta}_t^T J_{i,t})$$

2. $$K = \frac{P_t \Delta\vec{\theta}_t}{\lambda + \Delta\vec{\theta}_t^T P_t \Delta\vec{\theta}_t}$$

3. $$P_{t+1} = \frac{1}{\lambda}\left(I - K\Delta\vec{\theta}_t^T\right)P_t$$

4. Go back to 1.

Definitions:
i = row index of the Jacobian matrix
λ = exponential forgetting factor.
0 < λ ≤ 1

Convergence of Jacobian

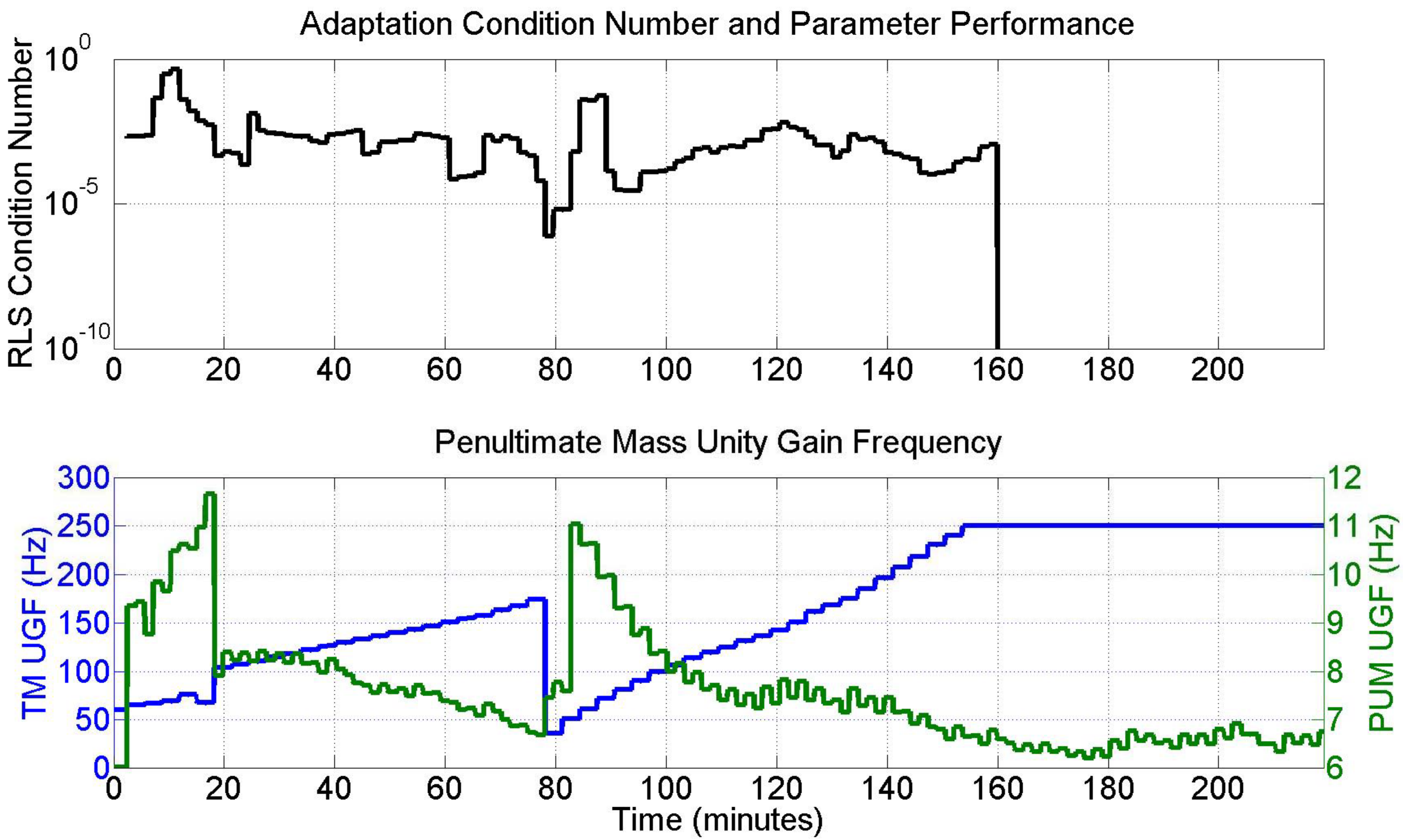
$$\Theta = \sum_{n=1}^{t} \theta_n \theta_n^T$$

A necessary condition for the convergence of RLS algorithm to the true J is that this matrix is nonsingular.

A real-time estimate of the invertiblility of this matrix is to calculate this matrix over a finite number of time steps and then calculate its condition number. The condition number is the smallest eigenvalue divided by the largest.

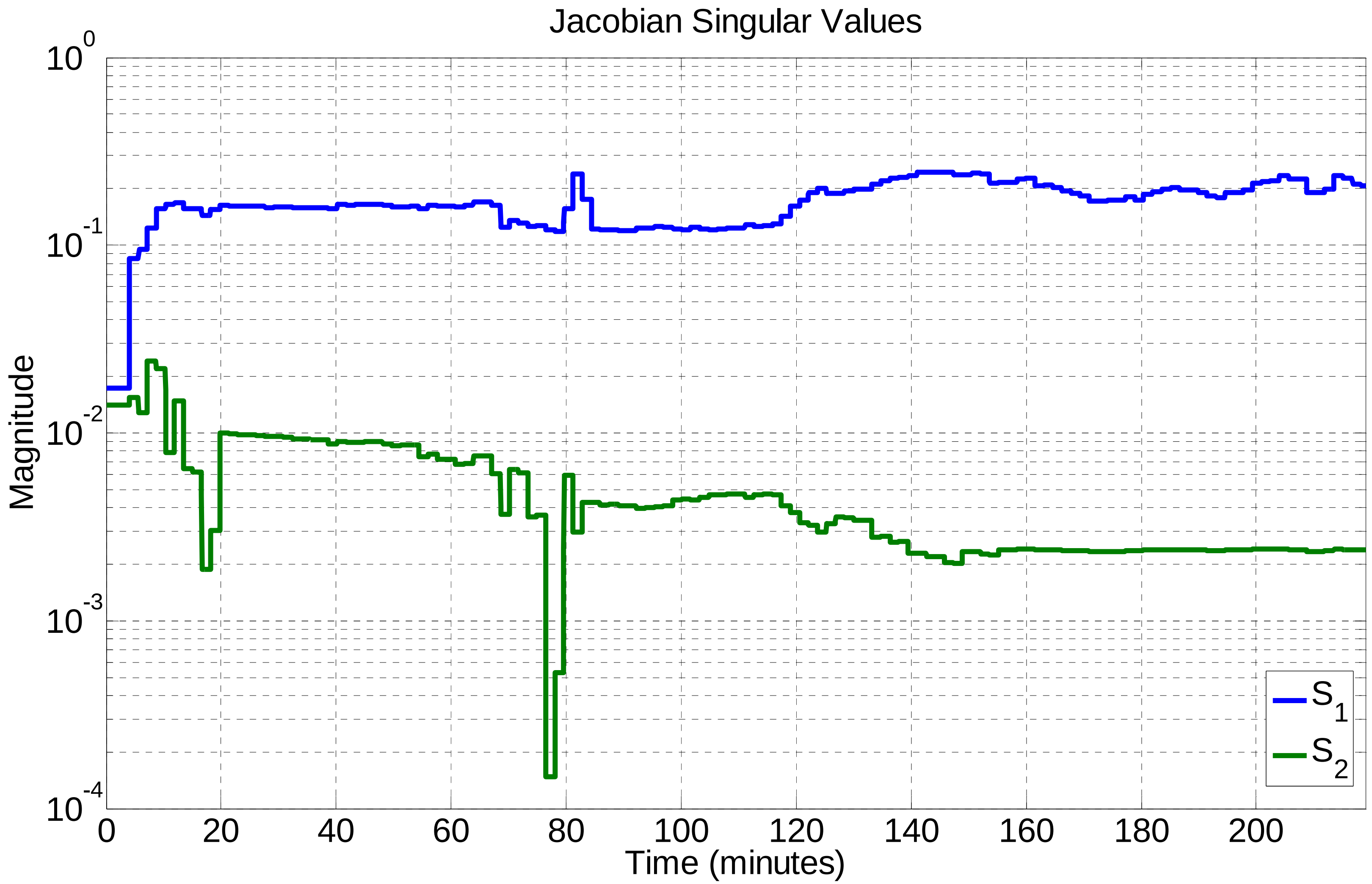
Condition number R = (max eigenvalue of Θ)/(min eigenvalue Θ)

Results



Adaptation Condition Number and Parameter Performance
RLS Condition Number
10^{0}
10^{-5}
10^{-10}
0 20 40 60 80 100 120 140 160 180 200
Penultimate Mass Unity Gain Frequency
TM UGF (Hz)
300 250 200 150 100 50 0
PUM UGF (Hz)
12 11 10 9 8 7 6
0 20 40 60 80 100 120 140 160 180 200
Time (minutes)

Results



Jacobian Singular Values
Magnitude
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
20
40
60
80
100
120
140
160
180
200
Time (minutes)
S_1
S_2

Thesis Contributions

- The literature has very little on generating real-time estimates for how controller parameters influence the statistics of linear system performance.
- Similarly there is little in the literature on using real-time function minimization techniques to optimize linear system performance.
- Optimization of Jacobian estimation accuracy.
- Optimizing adaptation rate using the measured statistics of the stochastic system performance.
- Use of singular value decomposition to quantify behavior of control adaption and system Jacobian.