About OMICS Group

OMICS Group International is an amalgamation of Open Access publications and worldwide international science conferences and events. Established in the year 2007 with the sole aim of making the information on Sciences and technology 'Open Access', OMICS Group publishes 400 online open access scholarly journals in all aspects of Science, Engineering, Management and Technology journals. OMICS Group has been instrumental in taking the knowledge on Science & technology to the doorsteps of ordinary men and women. Research Scholars, Students, Libraries, Educational Institutions, Research centers and the industry are main stakeholders that benefitted greatly from this knowledge dissemination. OMICS Group also organizes 300 International conferences annually across the globe, where knowledge transfer takes place through debates, round table discussions, poster presentations, workshops, symposia and exhibitions.

About OMICS Group Conferences

OMICS Group International is a pioneer and leading science event organizer, which publishes around 400 open access journals and conducts over 300 Medical, Clinical, Engineering, Life Sciences, Pharma scientific conferences all over the globe annually with the support of more than 1000 scientific associations and 30,000 editorial board members and 3.5 million followers to its credit.

OMICS Group has organized 500 conferences, workshops and national symposiums across the major cities including San Francisco, Las Vegas, San Antonio, Omaha, Orlando, Raleigh, Santa Clara, Chicago, Philadelphia, Baltimore, United Kingdom, Valencia, Dubai, Beijing, Hyderabad, Bengaluru and Mumbai.

Robustification of Repetitive Control Systems for High Accuracy and High Speed Operation

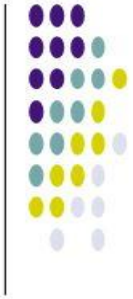


Richard W. Longman

Columbia University, New York City, USA

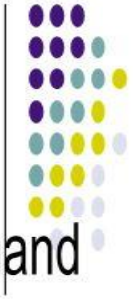


RC and ILC



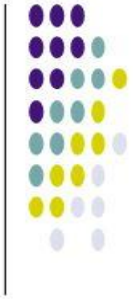
- ITERATIVE LEARNING CONTROL
  - A control systems performs same tracking task repeatedly
  - Examine error in last run, adjust command in next run
  - Aim: Converge to zero error – ***Perfect tracking***
- REPETITIVE CONTROL
  - A control system has periodic desired output
  - Or is subject to a periodic disturbance, period known
  - Aim: ***Perfect cancellation*** of disturbance
  - Aim: ***Perfect tracking*** of periodic trajectory

What Errors do We Try to Fix?



- The input to a control system is the command
  - It tells the control system what you want it to do
  - But control systems essentially ***NEVER*** do what you ask them to do !!!
- Recall frequency response
  - A tracking command can be written as a sum of sinusoids – Fourier expansion
  - The ***bandwidth*** of a control system defines up to what frequency does the system do a reasonable job (0.707 of amplitude you ask, phase error <45 deg)
- Repeating Disturbances
  - E.G. Gravity torque on robot link along desired path

Motivation of ILC



- ILC created because of robots doing repeated operations in manufacturing
  - In manufacturing a robot may perform the same task hundreds of times a day
  - And make the ***same tracking errors hundreds*** of times a day
- That is **STUPID**
  - ILC asks to learn from previous runs
  - Humans do, why not robots?

High Accuracy vs High Speed



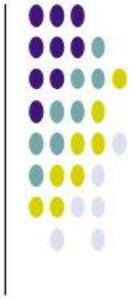
- Improving accuracy
  - Can produce improved products
  - Give a company competitive advantage
- High speed
  - If robots do their job faster assembly line can run faster
  - Increased productivity
- ILC and RC can help with both / trade off
  - Maybe can go at higher speed with same accuracy
  - Or stay at same speed with better accuracy

Limitations to achieving perfection



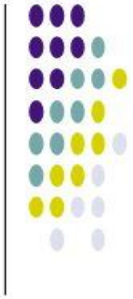
- How accurate is the sensor?
  - A sensor has limited number of digits accuracy
  - You cannot fix an error that you cannot see
- How reproducible is the hardware?
  - If you give the same command over and over, how different is the result from run to run?
  - ILC / RC can only approach this error limit
  - It tries to fix the part of error that does not repeat
  - Makes the error worse
  - The slower you learn, the smaller the error after you learn

What kind of learning?



- There are various kinds of learning
  - **Adaptive Control** updates a model
    - But maybe we can get better error level with forcing the system to behave like a chosen form of model
    - e.g. linear system of given order with constant coefficients, and disturbance model?
  - **Neural Networks**
    - Can take a very long time to learn (we get learning in 12 iterations)
  - **Integral Control** based learning
    - Integral control will not tolerate a constant error
    - Similarly we can not tolerate a periodic / repeating error

Inverse Problems



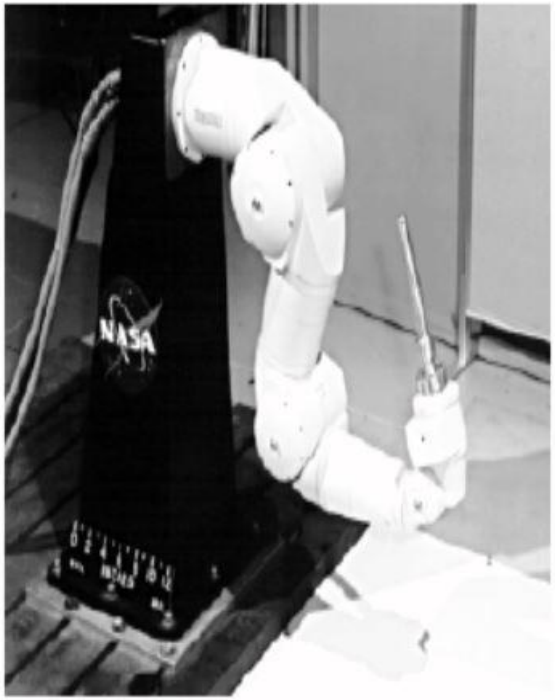
- Both RC and ILC are Inverse Problems
  - Know the desired solution of model / real world
  - Ask to find the input that produces this solution
  - Why do we have to learn, *just invert our model* ?
  - Because the model is not perfect !!
- Given a Periodic / Repeating Disturbance
  - Find input that cancels the effect of the disturbance
  - Much harder to have a model of the disturbance
  - We want perfect tracking without knowing or identifying the disturbance

Iterate With a Model
or
Iterate With the World

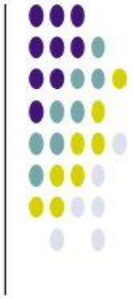


- As engineers we are used to making a model of the world, and using it to solve our problems
- Instead of a model of the world, we want to use the real world for our iterations
- CONCEPT:
  - We want perfect tracking in the real world
  - Not in our model of the world
- Error improvement can be **dramatic**

Experiments Run on a Robot at NASA Langley Research Center



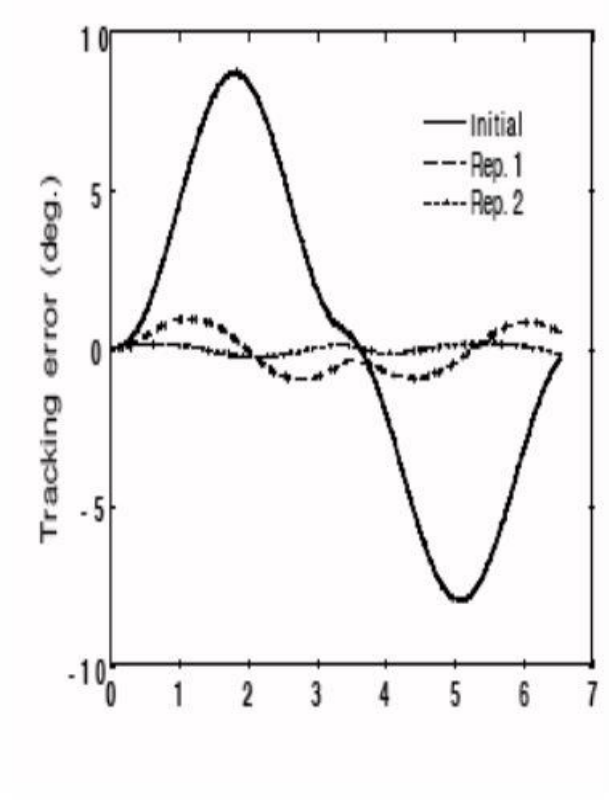
Command and Feedback Performance



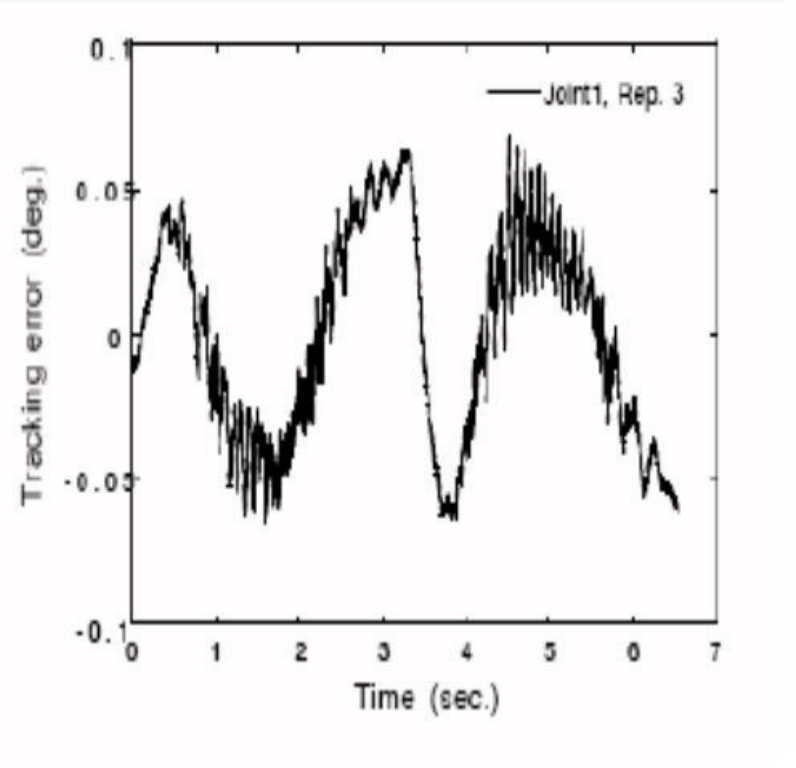
- Ask all 7 joints to rotate 90 deg and then rotate back
- At fastest rate allowed by manufacturer
- Well designed classical feedback controllers for each joint reach maximum error of 9 deg
- We show performance of one ILC design

ILC design with compensator and 18Hz Zero-Phase Low-Pass Filter Cutoff

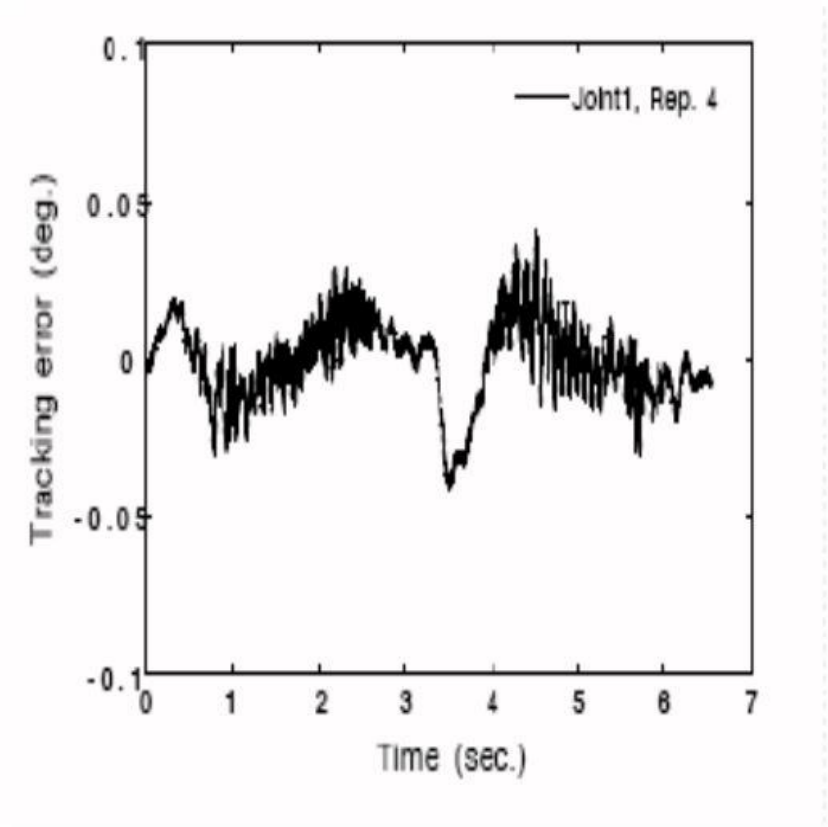
- Error histories, iterations 0, 1, 2



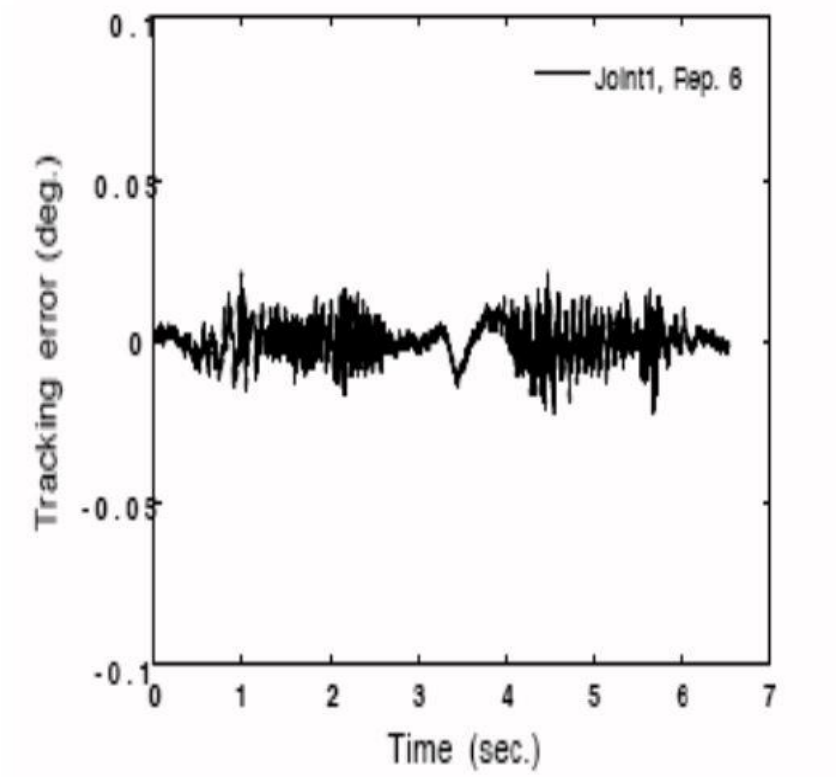
Error History on 3rd Iteration



Error History on 4th Iteration

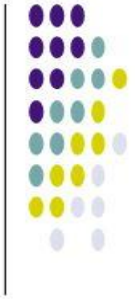


Error History on 6th Iteration



- RMS Error reduced by factor of 1000 in 12 iterations

Iterate With the World



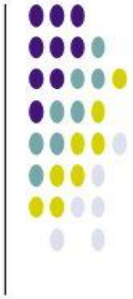
- Use Model or World Itself:
  - We made math model of robot and solved the inverse problem
  - Applied result to world, reduced robot tracking error by factor of 50
  - Then turn on ILC, and reduce the error by total factor of 1000
- We want to fix errors from sources that we cannot model

Fixing Errors



- Theoretical Limit: Reproducibility Level of Hardware
  - Factor of 1000 is below the reproducibility level of robot from day to day (vs. minute to minute)
  - So ILC fixes errors of how different the robot behaves tomorrow compared to today
  - We do not know why the difference, do not know how to model
  - Implies: Keep RC running to track slow changes in behavior

Philosophical Question

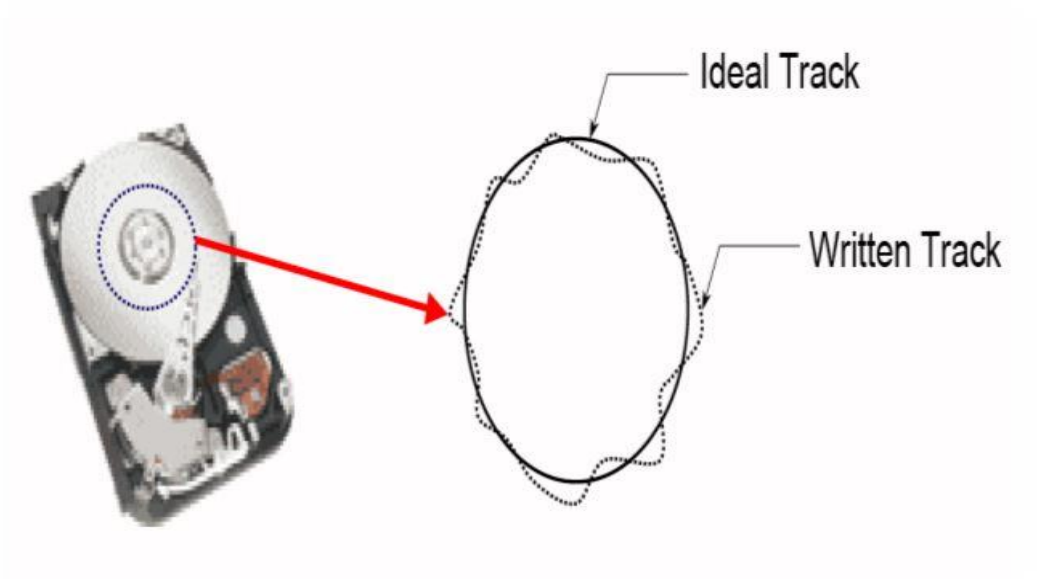


- If we were iterating with a model, maybe we could prove that the iterations converge
- But we are trying to solve an inverse problem in the world, not a model
- How much do we need to know about the world, to know that we will converge to zero error?
- Important Objective: make the iteration process as robust to model errors as possible

Another Application Wanting High Speed and High Accuracy

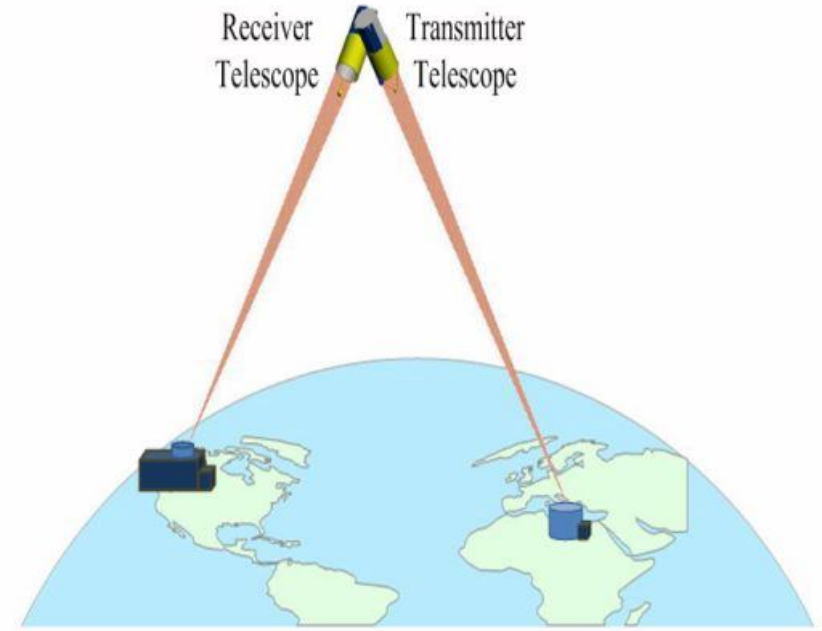


- Computer disk storage
- Improved track following allows closer tracks and more storage



Another Application:
Laser Communication Relay Satellite

NASA Lunar Laser Demo, LADEE Spacecraft, 622 Megabits/sec // Also Space Station Test is Planned



Surprise: The Inverse Problem is Usually Unstable

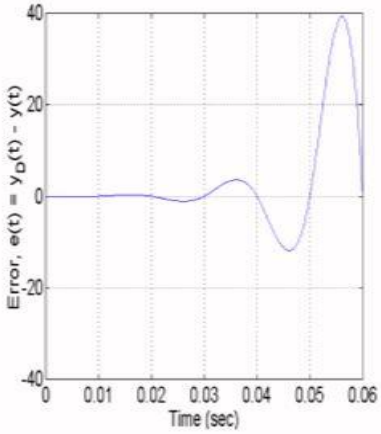
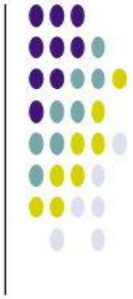


Figure 17: Intersample error for the third order system showing zero error at the sample times from the converged ILC command history.

- Error at sample times is zero, error between sample times grows
- RC solves this problem:
  - RC can ask for convergence as time step goes to infinity
  - RC can use
    - inverse of steady state frequency response
    - instead of inverse of difference equation model

Frequency Response and z-Transfer Functions



- Linear difference equation
  - Input: $\cos(\omega kT)$
  - Steady state output: $M(\omega)\cos(\omega kT + \phi(\omega))$
  - Transfer function $G(z)$: $M(\omega) = \left|G(e^{i\omega T})\right|$

$$\phi(\omega) = \measuredangle G(e^{i\omega T})$$

RC Design: Create Inverse Frequency Response Model



- Minimize cost summed over frequencies zero to Nyquist

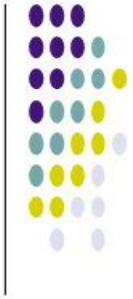
$$J = \sum_{j=0}^{N} [1 - G(e^{i\omega_j T})F(e^{i\omega_j T})]W_j[1 - G(e^{i\omega_j T})F(e^{i\omega_j T})]^*$$

- Fit FIR model

$$F(z) = a_1 z^{m-1} + a_2 z^{m-2} + \cdots + a_m z^0 + \cdots a_{n-1} z^{-(n-m-1)} + a_n z^{-(n-m)}$$
$$= (a_1 z^{n-1} + a_2 z^{n-2} + \cdots + a_m z^{n-m} + \cdots + a_{n-1} z^1 + a_n z^0) / z^{(n-m)}$$

- Stated in words: Compute linear combination of several errors in previous period
- Control Law: $U(z) = [z^{-p}U(z)] + F(z)[z^{-p}E(z)]$

Some Design Choices

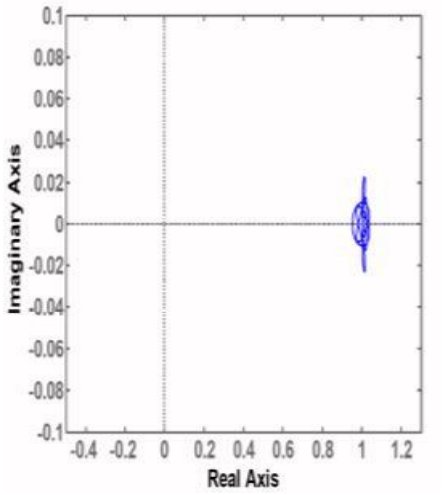


- Number of gains in FIR filter
- Sample rate
- Introduce penalty on gain size if too large

FIR Design Can Be Simple

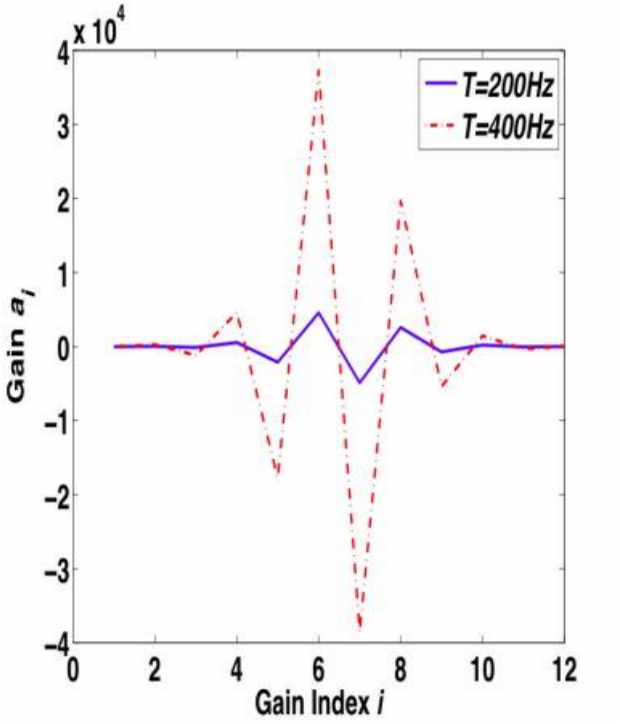


- Product FG for 8 gains a_i



- Just add errors from 8 time steps from previous period to perform inversion for current step

Design Choice: Sample Rate

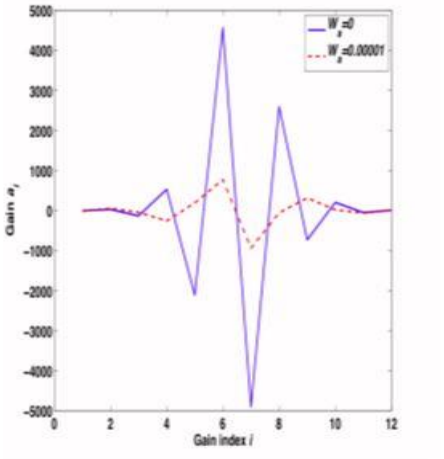


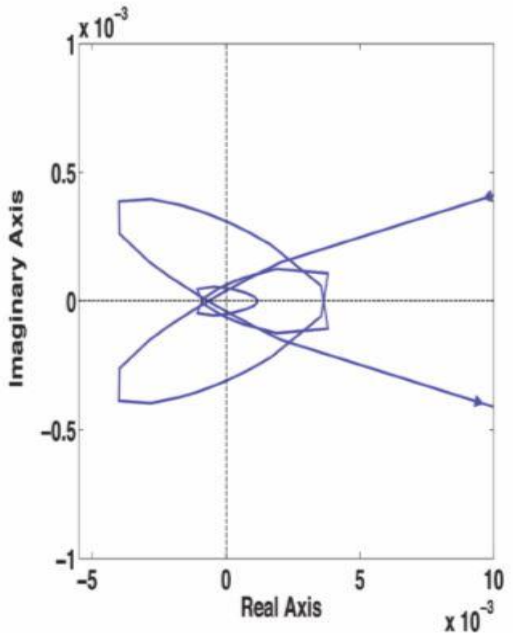
Design Choice: Penalty on Gain Size

- Penalize large gains

$$J = \sum_{j=0}^{N} [1 - G(e^{i\omega_j T}) F(e^{i\omega_j T})] W_j [1 - G(e^{i\omega_j T}) F(e^{i\omega_j T})]^* + W_a (a_i)^2$$

- Slows learning at high frequencies
- Possible stability issue

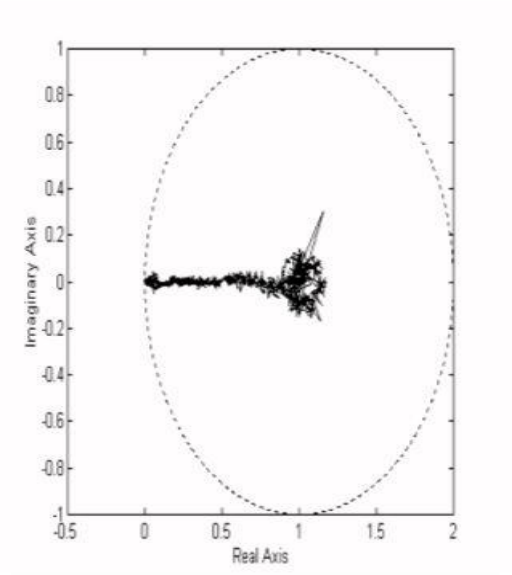




Stability

- Frequency response inverse needs to be good enough

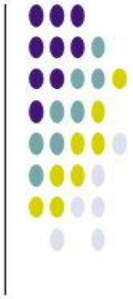
$$\left|1 - G(z)F(z)\right| < 1 \quad \forall \quad z = e^{i\omega T}$$



Typical instability at high frequency

Instability grows very slowly (*unusual*)

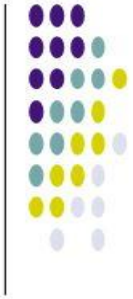
What is Robustness



- Stability Robustness – means RC is stable for substantial model error

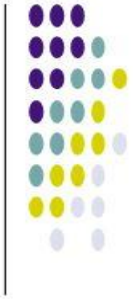
- Stability Robustness = Convergence Robustness – RC converges for substantial model error

The Model Difficulty



- Stability of feedback control design can usually be established at relatively low frequency (e.g. gain margin, phase margin)
- RC want zero error for fundamental and every harmonic up to Nyquist frequency
- Langley robot:
  - Nyquist frequency 200 Hz
  - Bandwidth of feedback control systems 1.4 Hz
  - Cannot even get any data at 200 Hz input !!
  - But RC can grow error at high frequencies - slowly

Methods of Robustification



- Use a frequency cutoff of the learning
- Can use error growth data to adaptively fix frequency response model error
- Ask for zero error below a “cutoff” and quadratic cost control above
- Given parameter uncertainties in model, design RC to minimize cost function over set of models
- Improve by reducing gain at some frequencies for slower learning but better robustness

Robustify to High Frequency Model Errors



- Robustify to inaccurate or missing high frequency dynamics
  - Parasitic poles / Residual Modes / Dynamics not visible in noisy data

$$U(z) = z^{-p} H(z)[U(z) + F(z)E(z)]$$

- One can optimize the **frequency cutoff filter**
- Minimize cost

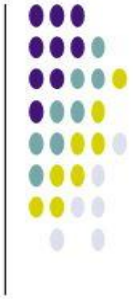
$$H(z) = \sum_{k=-n}^{n} a_k z^k$$

- Parameters

$$J_H = \alpha \sum_{j=0}^{j_p} [1 - H(e^{i\omega_j T})][1 - H(e^{i\omega_j T})]^* + \sum_{j=j_s}^{N-1} [H(e^{i\omega_j T})][H(e^{i\omega_j T})]^*$$

  - Filter order
  - Passband vs. stopband weights
  - Introduce transition zone
  - Introduce equality constraint, DC gain of unity
  - Introduce inequality constraints, to prevent magnitudes above unity

Picking the Cutoff Frequency



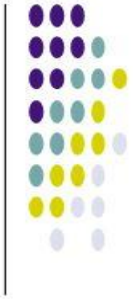
- Must adjust cutoff in hardware
- Cutoff is needed because model is not good enough for convergence at high frequency
- If we knew what was wrong with our model we would fix it

Robustify by Updating Frequency Response Information



- Instability is typically slow
- DFT can show what frequencies are growing
- And data for growing data tells what wrong with frequency response at those frequencies

Robustify: ask for zero error up to a frequency, then quadratic cost control above that



- ILC cost, use Parsival to convert to frequency domain

$$J_j = \underline{e}_j^T \underline{e}_j + \delta_j \underline{u}^T R_d \delta_j \underline{u} + \underline{u}_j^T R_u \underline{u}_j$$

- Middle term controls learning speed
- Last term prevents reaching zero error

$$U_j(e^{i\omega_k T}) = H(e^{i\omega_k T})[U_{j-1}(e^{i\omega_k T}) + F(e^{i\omega_k T})E_{j-1}(e^{i\omega_k T})]$$

$$H(e^{i\omega_k T}) = \frac{\left|G(e^{i\omega_k T})\right|^2 + r_d(k)}{\left|G(e^{i\omega_k T})\right|^2 + r_d(k) + r_u(k)} \; ; \; F(e^{i\omega_k T}) = \frac{G(e^{-i\omega_k T})}{\left|G(e^{i\omega_k T})\right|^2 + r_d(k)}$$

- Design changes radius of stability circle as function of frequency – no longer getting zero error at that frequency

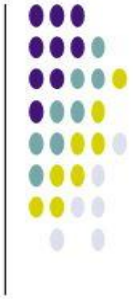
Robustify by Averaging Cost Over Model Probability Distributions



- Robustify to parameter uncertainties
- Given uncertainty interval on parameters, pick set of models from distribution
- Minimize cost that is average over models

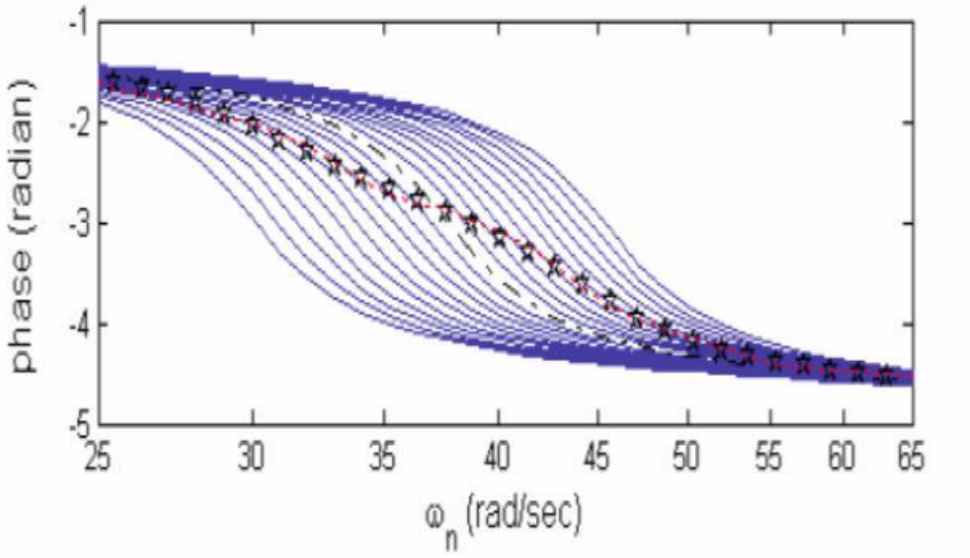
$$J_{mm} = \sum_{k=1}^{2M+1} \sum_{j=1}^{N} [1 - G_k(e^{i\omega_j T}) F(e^{i\omega_j T})][1 - G_k(e^{i\omega_j T}) F(e^{i\omega_j T})] *$$

Robustify by Fine Tuning Robustness For Each Frequency

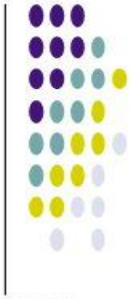


- Averaging works based on phase, use learning rate adjustment vs. frequency also

$$J_F = \sum_{j=1}^{N} [1 - F(e^{i\omega_j T})(1/\hat{r}_{F,j})e^{-i\theta_{F,j}}][1 - F(e^{i\omega_j T})(1/\hat{r}_{F,j})e^{-i\theta_{F,j}}]^*$$

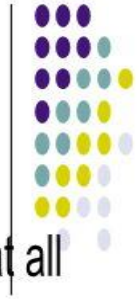


Conclusions



- Repetitive Control initially asks to converge to zero error in the presence of a periodic disturbance of known period
- The instability of the inverse model of discrete time linear systems fed by a zero order hold is circumvented by creating an inverse model of the steady state frequency response
- RC is iterating with the real world, not with a model of the world
- Aims to eliminate effects of disturbances of given period: the fundamental frequency and all harmonics to Nyquist

Conclusions (cont)



- Control systems do not normally ask for zero error at all addressed frequencies – pushes all analysis methods to limits
- Basic question: How much do we have to know about the system to know the RC will converge?
- RC eliminates periodic errors, so characterizing uncertainty in the frequency domain is natural
- Robustification to model error becomes fundamental for applications
- A set of robustification tools is presented to help produce convergence for wide range of models
- Because RC iterates with the world, tuning of the design choices often needs to be make in hardware

ILC and RC Must Be Digital



- ILC looks back at the error in a previous run
- RC looks back at the error in the previous period
- This is data that must be stored, and therefore control must be digital
- In this talk we consider linear models, with zero order hold inputs

Exact Conversion Differential to Difference Equation



- A linear differential equation

$$\frac{d^3y(t)}{dt^3}+\alpha_2\frac{d^2y(t)}{dt^2}+\alpha_1\frac{dy(t)}{dt}+\alpha_0 y(t)=\alpha_0 u(t)$$

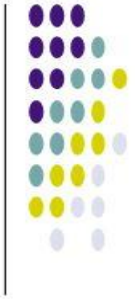
  - Fed by a zero order hold $u(t)=u(kT) \qquad kT \le t < (k+1)T$

- Can be converted to a difference equation

$$y((k+3)T)+a_2y((k+2)T)+a_1y((k+1)T)+a_0y(kT)$$
$$=b_2u((k+2)T)+b_1u((k+1)T)+b_0u(kT)$$

- Whose solution is identical at the sample times
- No approximation

The Inverse Problem



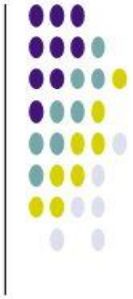
- Given desired output sequence $y(kT)$
- Find the input sequence $u(kT)$ to produce it
- Solve difference equation

$$f(kT) = b_2 u((k+2)T) + b_1 u((k+1)T) + b_0 u(kT)$$

- Solution: Particular solution plus solution of homogeneous equation
- Characteristic polynomial $b_2 z^2 + b_1 z + b_0 = 0$
- Solution homogeneous equation (for small T)

$$C_1(-3.7)^k + C_2(-1/3.7)^k$$

Inverse Problem UNSTABLE



- Inverse problem is unstable for 3rd order or higher systems (for reasonable sample rates)
- QUESTION: If we use interpolation instead of zero order hold?
- Then inverse problem is unstable for 4th order or higher systems
- Fundamental property: Any implications for use of discretizing the control input between shooting nodes?

Design Choice: Adjust Cost Function Frequency Upper Limit



- If the RC has upper limit on frequency, then the inverse model of frequency response can use modified upper limit *N*

$$J = \sum_{j=0}^{N} [1 - G(e^{i\omega_j T}) F(e^{i\omega_j T})] W_j [1 - G(e^{i\omega_j T}) F(e^{i\omega_j T})]^*$$