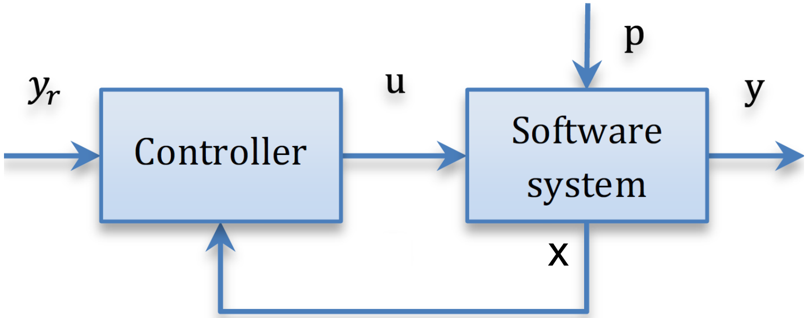
Robust Linear Quadratic Control for Software Systems

Marin Litoiu
Adaptive Systems Research Lab
Dept. of EE and CS
York University
http://ceraslabs.com

Agenda

- **Motivation**
- **Model Identification Adaptive Control**
- **Experimental Results**
- **Conclusions**

Linear Quadratic Control



- y_r: goal/setpoint/reference
- y: outputs
- u: inputs/commands (actionable by controller)
- x: states (internal variables)
- p: perturbations
  - Workloads, faults, etc…

Linear Quadratic Regulator(LQR):

maintains y_r and minimizes J

$$J = \sum_{0}^{\infty} x^T Q_x x + u^T Q_u u$$

Subject to y=f(x,u) being linear
Q_x, Q_u, weight matrices

Control Theory and Adaptive Systems

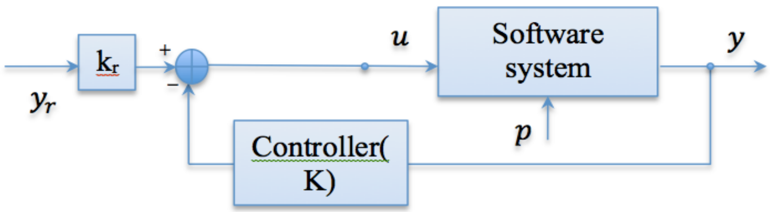
- **Methodology**
  - Have an adaptation goal (set point, objective function, etc…)
  - **Build a model (linear or linearized) of the software system**

$$\begin{cases} x(k+1) & = Ax(k) + Bu(k) \\ y(k) & = Cx(k) + Du(k) \end{cases}$$

  - Study the properties of system
    - Observability: can I estimate x if I only measure y?
    - Stability: bound inputs-> bound outputs?
    - Controllability: can I reach any x with set of inputs u?
  - **Synthesize a controller/Adaptation manage**r

Controller=h(goals, A, B,C,D)

LQR Structure: K, k_r



Big questions:

-y=f(u,x)?

-K=?

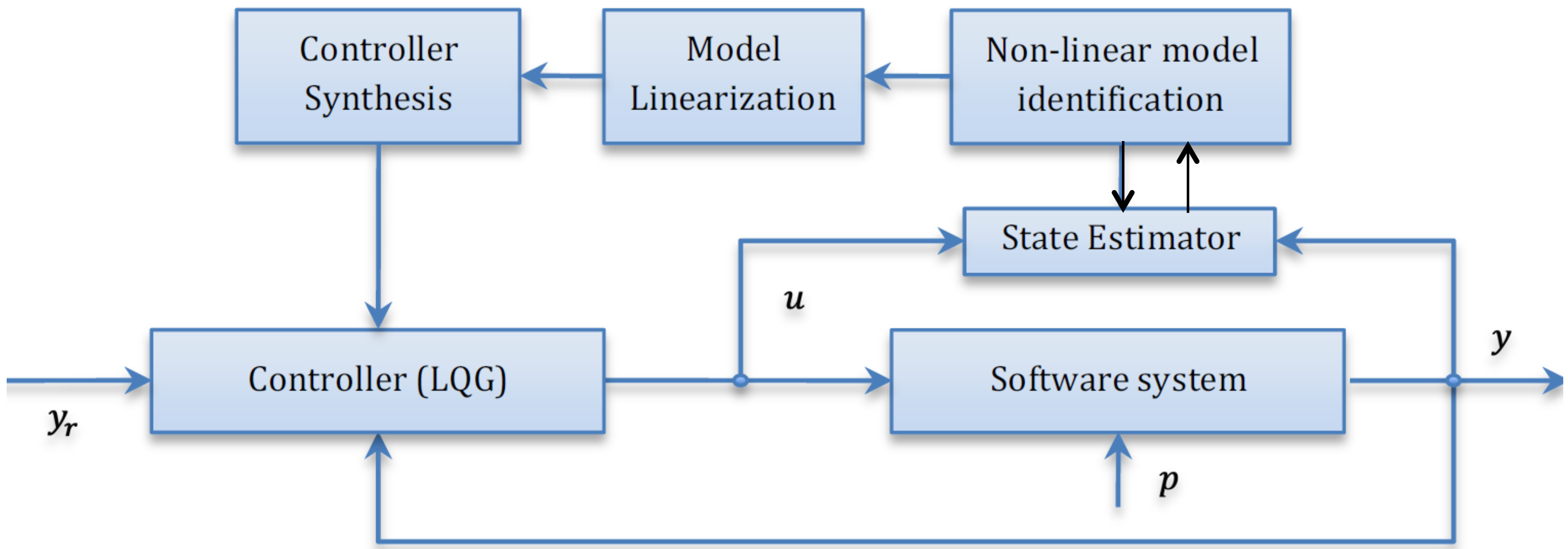
-kr=?

- **Consider**
  - y - response time
  - u - change in waiting time (threads, replicas, etc..)
  - y_r= 2 seconds
  - K=1, k_r=1(to keep it simple)
- **Sampling 1**
  - y=2→u=0; nothing to change
- **Sampling 2**
  - y=3→ u=-1→decrease waiting time
- **Sampling 3**
  - y=1.8→u=0.2→ increase waiting time

What Limits LQR Robustness?

- **y=f(u) is considered static and linear**
  - not accurate for real systems
- **u has one dimension (threads or #servers..)**
  - SISO
  - Has limited influence, cannot address large perturbations
- **The controller is designed statically**
- **In practice, engineers still use “ON condition, DO action”**

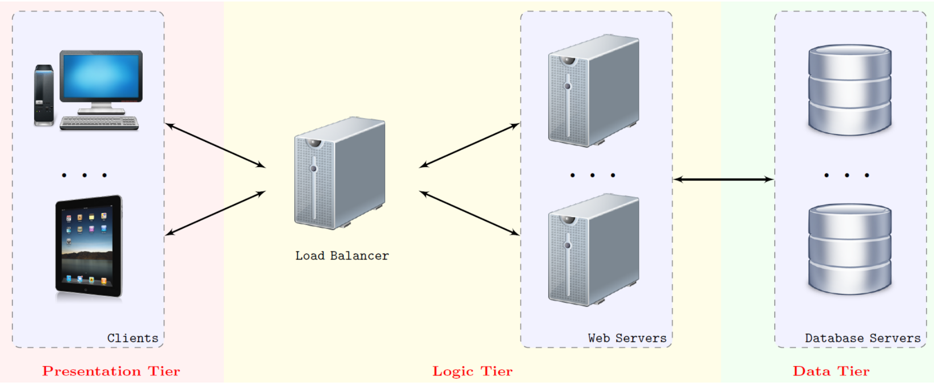
Model Identification Adaptive Control (for performance and cost)



Non-Linear Model

- **Structure(known)**
  - u is multi-dimensional [threads, no of instances, bandwith, etc…]
  - y is multi-dimensional [response time, throughput, cost, etc..]
  - y=f(u, x) is non-linear (LQM, ML, etc..)
- **Uncertainties (unknown) in the model**
  - Model parameter uncertainties
    - use estimators, like Kalman filter, to estimate them at runtime
  - Perturbation uncertainties
    - the controller is designed to address these
  - Non-modeled dynamics

Case Study



u=[webServers, threadsWS, DBServers, threadsDB]
y=x= [response time]
p= [workload]

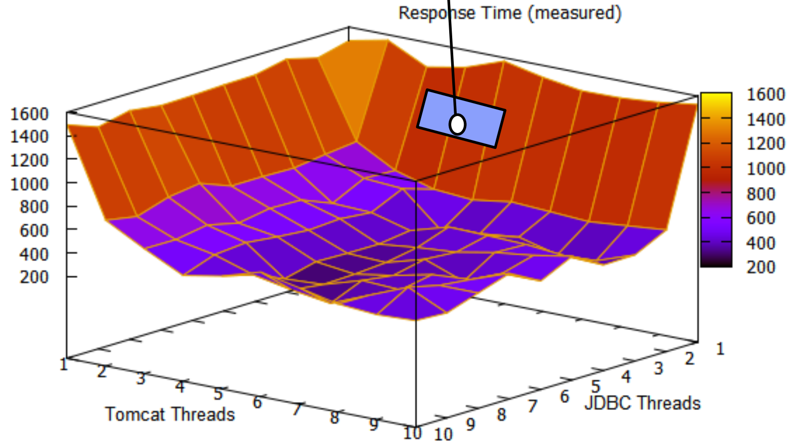
$x(k+1)=Ax(k)+Bu(k)$ C=1, D=0;
$y(k)=x(k)$

$$J=\sum_{0}^{\infty} x^T Q_x x + u^T Q_u u$$

$$Q_x=[1] \qquad Q_u=\begin{bmatrix} 100\,000 & 0 & 0 & 0 \\ 0 & 1\,000 & 0 & 0 \\ 0 & 0 & 100\,000 & 0 \\ 0 & 0 & 0 & 1\,500 \end{bmatrix}$$

Linearization

At time k, we linearize around this point, op



$$\begin{cases} y(k) = y_a(k) - y_{op} \\ x(k) = x_a(k) - x_{op} \\ u(k) = u_a(k) - u_{op} \end{cases}$$

$$\begin{cases} x(k+1) & = Ax(k) + Bu(k) \\ y(k) & = Cx(k) + Du(k) \end{cases}$$

Controller Design: K and k_r

- **If we have the model**

$$\begin{cases} x(k+1) & = Ax(k) + Bu(k) \\ y(k) & = Cx(k) + Du(k) \end{cases}$$

- **And the objective function**

$$J = \sum_{0}^{\infty} x^T Q_x x + u^T Q_u u$$

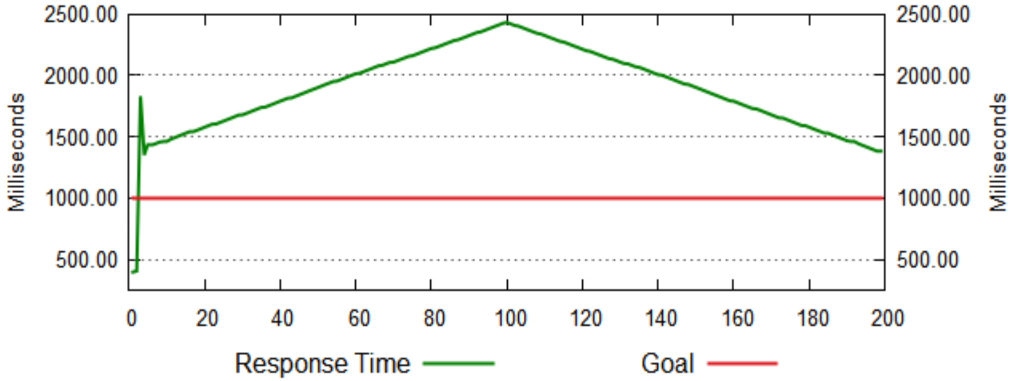
- **Then we apply a lot of algebraic formulas and find**

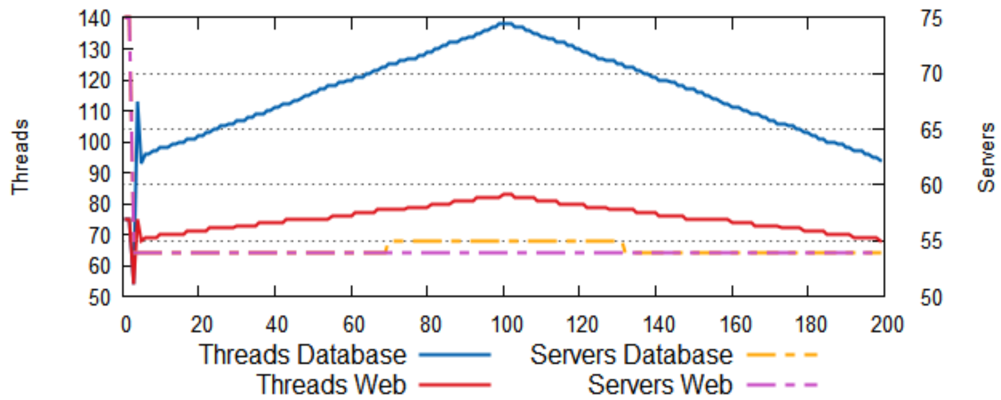
$$K = -Q_u^{-1} B^T P$$

$$PA + A^T P - PBQ_u^{-1} B^T P + Q_x = 0$$

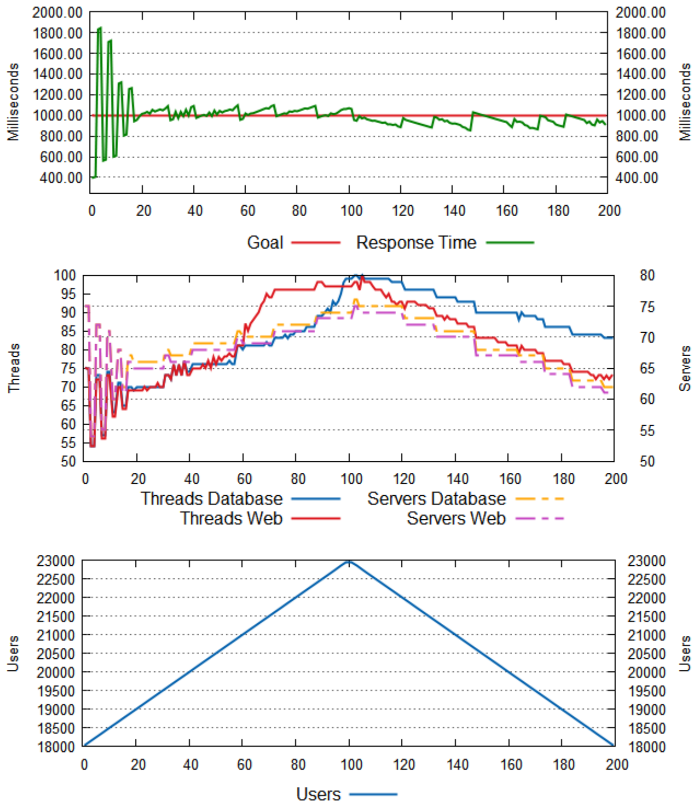
$$1 = C(A - BK)^{-1} Bk_r$$

One Linear Model Fails to Control

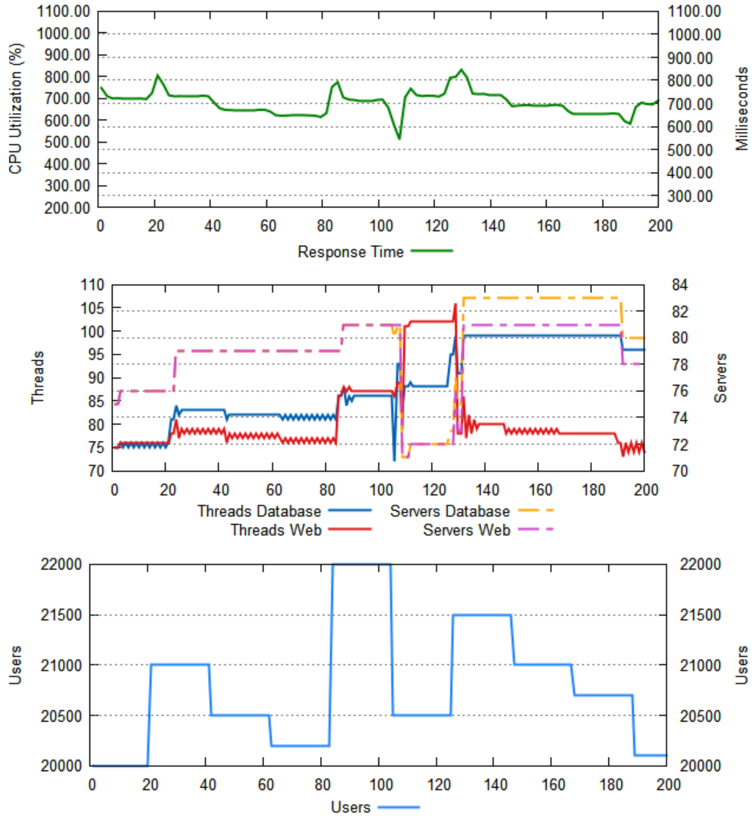




However, MIAC works!!! Evrika!!!!



..even with sudden changes in workload(perturbations)



Conclusions…

- **Model Identification Adaptive Controller**
  - Identifies a nonlinear model
  - At runtime, linearizes it
  - At runtime synthesizes a LQR controller
- **Limitations**
  - Some experiments done through simulations, do they hold on real system?
  - Sensitivity analysis
  - How do we know the bounds of robustness
  - Other type of controllers