COMPLEXITY MADE SIMPLE*

* AT A SMALL PRICE

C. G. Cassandras

Boston University

cgc@bu.edu

http://vita.bu.edu/cgc

THREE FUNDAMENTAL *COMPLEXITY LIMITS*

$1/T^{1/2}$ LIMIT

NP-HARD LIMIT

one order increa

INFO.
SPACE

Tradeoff between **GENERALITY** and **EFFICIENCY** of an algorithm
[*Wolpert and Macready, 1997*]

NO-FREE-LUNCH LIMIT

THREE FUNDAMENTAL *COMPLEXITY LIMITS*

$1/T^{1/2}$ LIMIT

NP-HARD LIMIT

NO-FREE-LUNCH LIMIT

CENTRAL THEME OF THIS TALK...

Rather than tackling hopelessly complex problems by brute force...

- **Seek "*surrogate*" simpler problems whose solution is the same or "good enough"**

SMALL PRICE: *NEAR OPTIMALITY?*

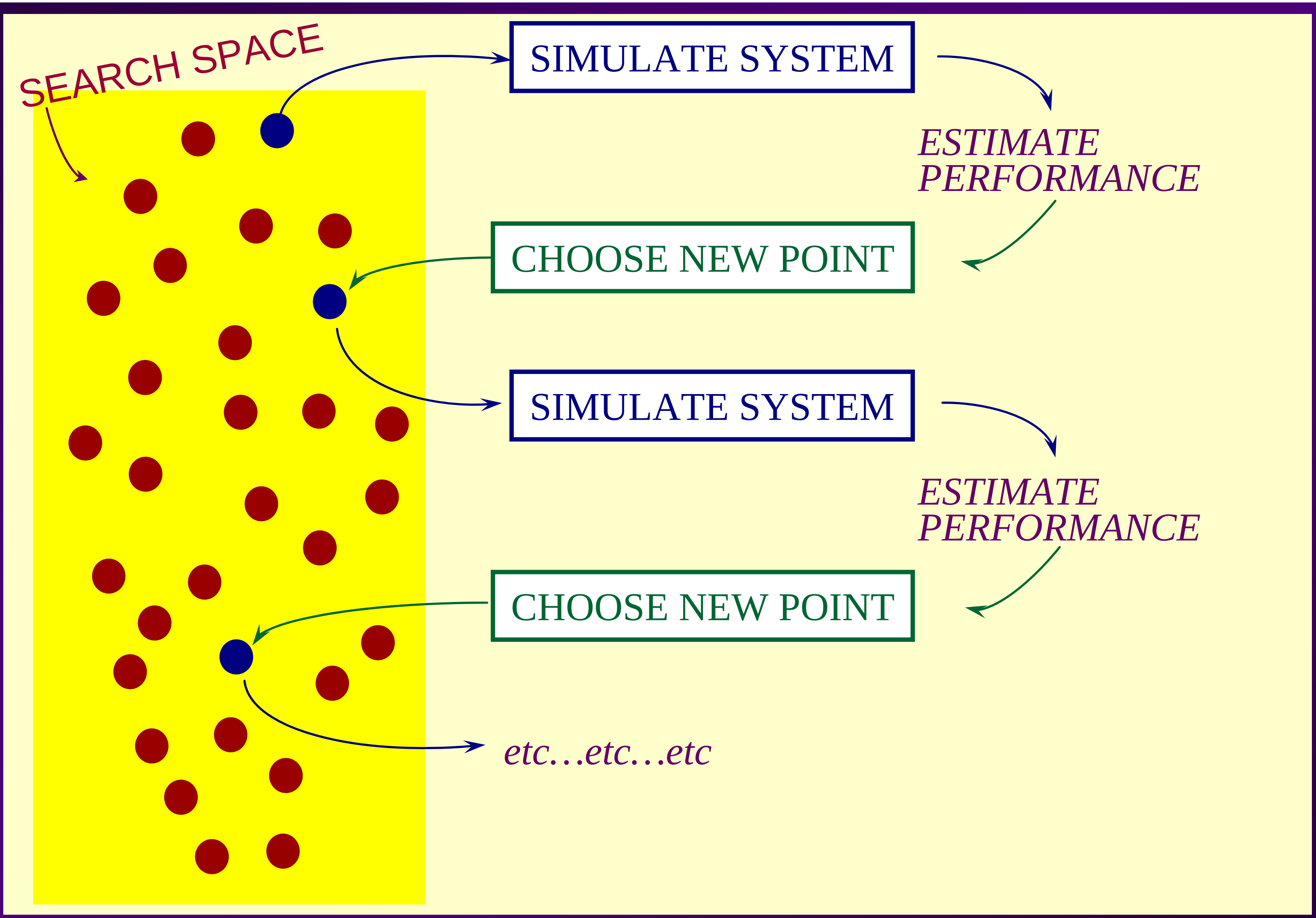
- **Exploit *problem structure* whenever possible**

SMALL PRICE: *SUFFICIENT GENERALITY?*

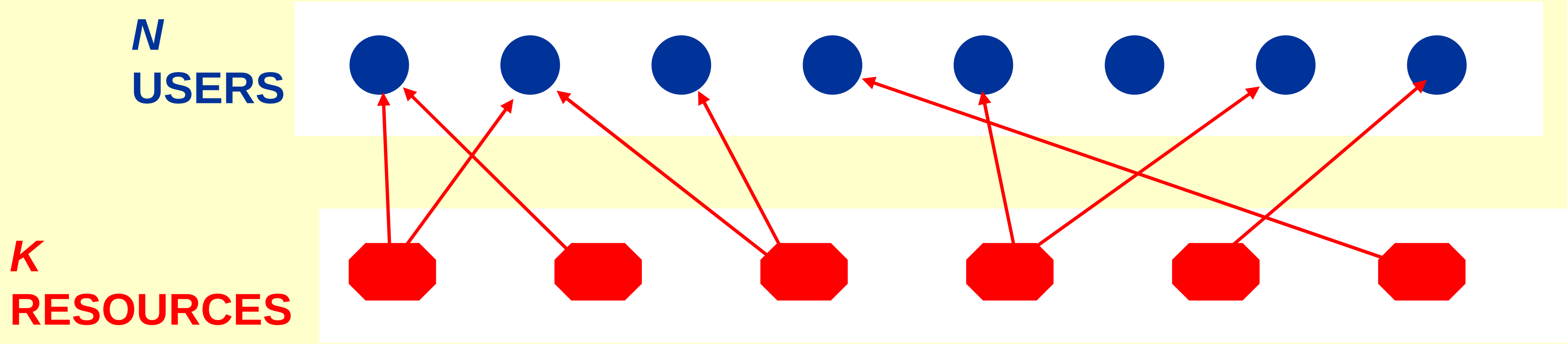
OUTLINE

- The classic complex optimization problem
- "SURROGATE PROBLEM" METHODS:
  Solutions to *hard* problems can be recovered from those of *simpler* problems
- CONCURRENT ESTIMATION:
  Answering *N* "what if" questions may *not* need *N* trials
- DECOMPOSITION:
  Solutions to *hard* problems can be recovered from those of *simpler* problems
- APPLICATIONS TO SOME COMPLEX SYSTEMS:
  - Resource Allocation
  - Hybrid Systems

TYPICAL OPTIMIZATION OF COMPLEX SYSTEMS: *REPEATED TRIAL AND ERROR*



RESOURCE ALLOCATION PROBLEMS



- **USERS** request **RESOURCES** at random points in time
- **USERS** hold **RESOURCES** for random periods of time

EXAMPLES:
- Buffers allocated to links in network switches
- Time slots allocated to tasks in computer systems
- Vehicles allocated to missions in transportation systems

PROBLEM FORMULATION

- ALLOCATION VECTOR: $r = [r_1, \ldots, r_N], \quad r_i \in \{0,1,2,\ldots\}$
- Determine r^* such that

$$J_d(r^*) = \min_{r \in A_d} J_d(r)$$

COST FUNCTION

CONSTRAINT SET

EXAMPLE: $A_d = \left\{ r : \sum_{i=1}^{N} r_i = K \right\}$ *Capacity Constraint*

WHY ARE THESE PROBLEMS HARD?

1. *Combinatorial complexity:* $$|S| = \frac{(K+N-1)!}{K!(N-1)!}$$

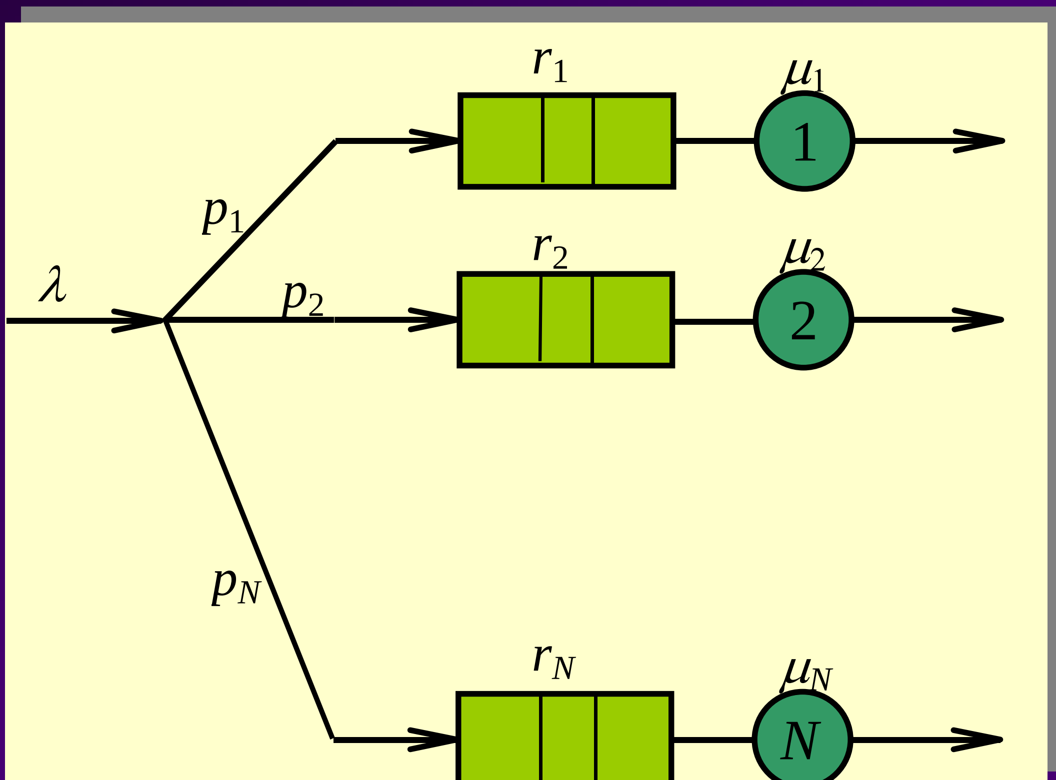
NP-HARD LIMIT

2. *Stochastic complexity:*

$J_d(r) = E[L_d(r)]$ is unknown and can only be obtained through simulation or direct observation of actual system sample paths

$1/T^{1/2}$ LIMIT

RESOURCE ALLOCATION EXAMPLE



Allocate K buffer slots (RESOURCES) over N servers (USERS) so as to minimize *BLOCKING PROBABILITY*

subject to $\sum_{i=1}^{N} r_i = K$

For ***N*=6**, ***K*=24** → Possible allocations = **118,755**

"SURROGATE" PROBLEM APPROACH

PROBLEM:

$$\min_{r \in A_d} J_d(r) = \min_{r \in A_d} E_\omega\left[L_d(r,\omega)\right]$$

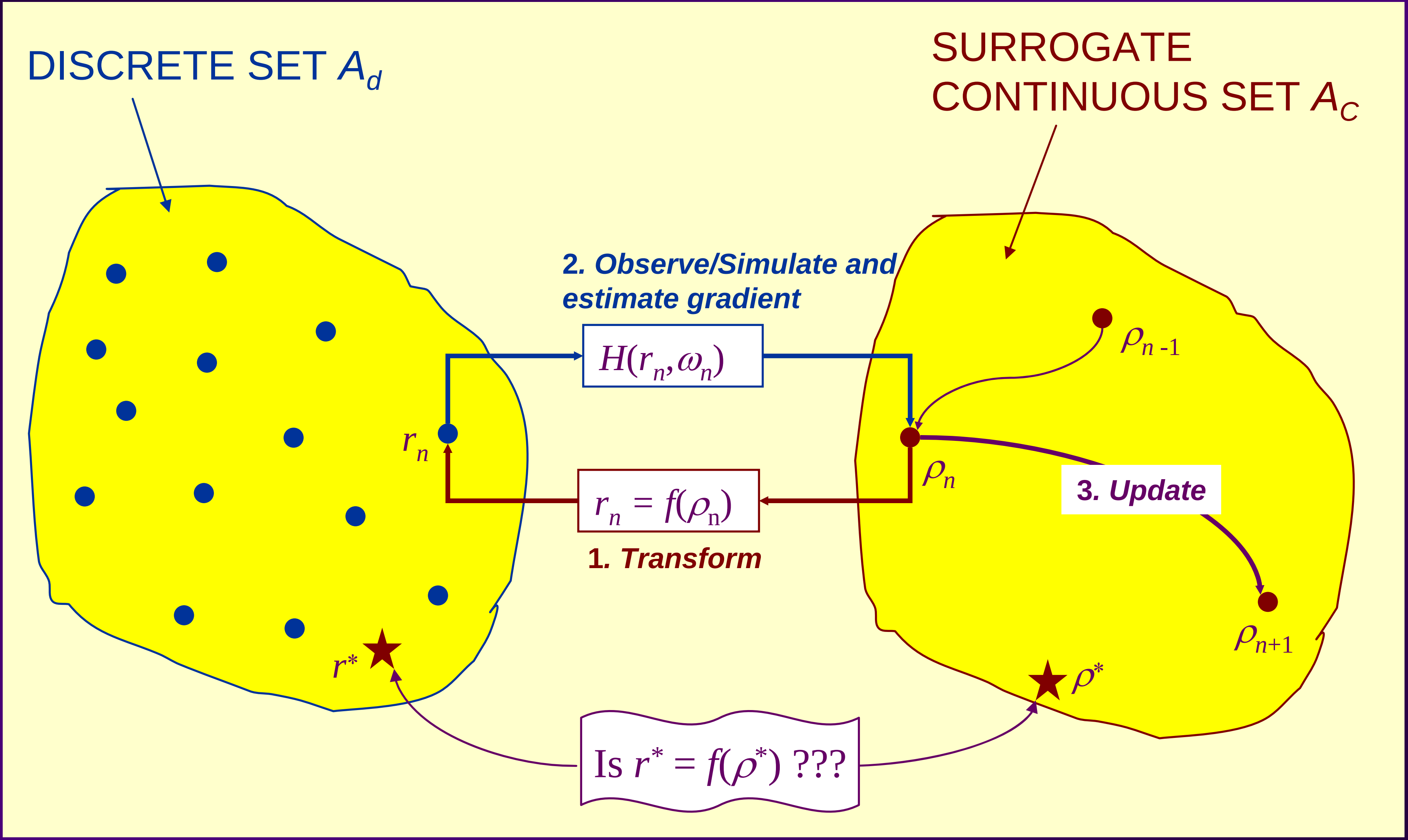
SURROGATE **PROBLEM:**

$$\min_{\rho \in A_c} J_c(\rho) = \min_{\rho \in A_c} E_\omega\left[L_c(\rho,\omega)\right]$$

$[\rho_1,\ldots,\rho_N], \quad \rho_i \in \Re_+$

CONTINUOUS SET CONTAINING A_d

"SURROGATE" PROBLEM APPROACH *CONTINUED*



IS $r^* = f(\rho^*)$???

ANSWER: ***YES...***

Theorem: Let ρ^* minimize $L_c(\rho)$. Then, there exists a discrete feasible neighbor $r^* \in N_N(\rho^*)$ which minimizes $L_d(r)$ and satisfies $L_d(r^*) = L_c(\rho^*)$.

[*Gokbayrak and Cassandras, JOTA, 2001*]

"SURROGATE" PROBLEM APPROACH *CONTINUED*

SOLUTION: Iterative algorithm, *n* = 0,1,…, with two steps:

1. Update ρ_n:

$$\rho_{n+1} = \pi_{n+1}\left[\rho_n - \eta_n H\left(r_n, \omega_n\right)\right]$$

PROJECTION INTO FEASIBLE SET A_c

STEP SIZE (LEARNING RATE)

*SENSITIVITY ESTIMATE FROM SIMULATION AT **DISCRETE** ALLOCATION*

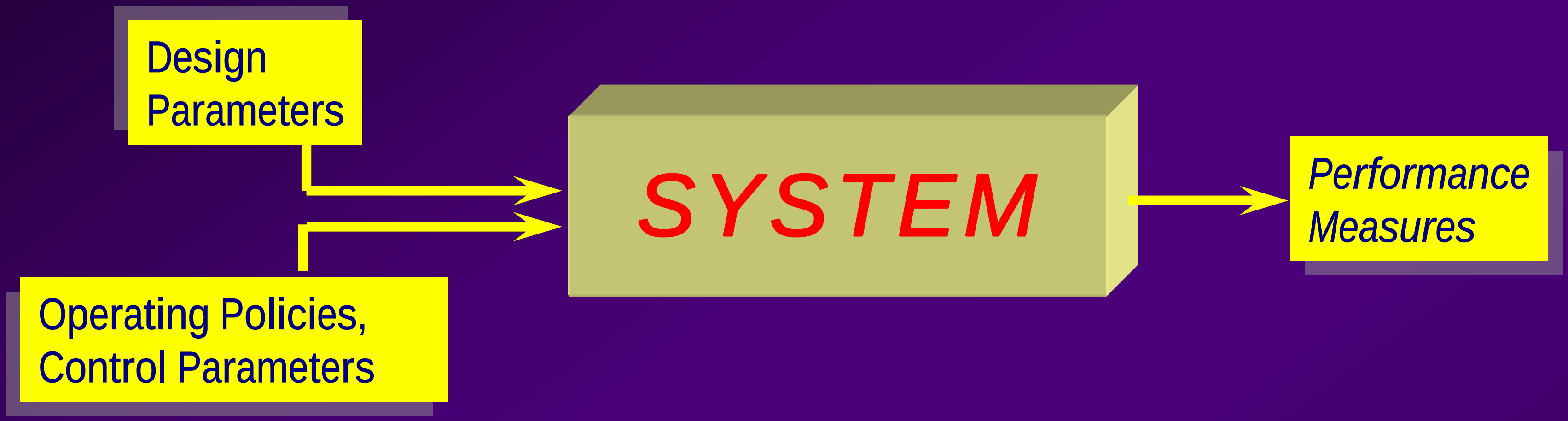
HOW DO YOU GET THIS?

2. Transform surrogate allocation ρ_{n+1} into ACTUAL feasible allocation:

$$r_{n+1} = f_{n+1}\left(\rho_{n+1}\right), \quad f_{n+1} : A_c \to A_d$$

SEVERAL POSSIBLE MAPPINGS

LEARNING BY TRIAL AND ERROR



CONVENTIONAL TRIAL-AND-ERROR ANALYSIS (e.g., simulation)

- *Repeatedly change parameters/operating policies*
- *Test different conditions*
- *Answer multiple WHAT IF questions*

LEARNING THROUGH *PERTURBATION ANALYSIS*

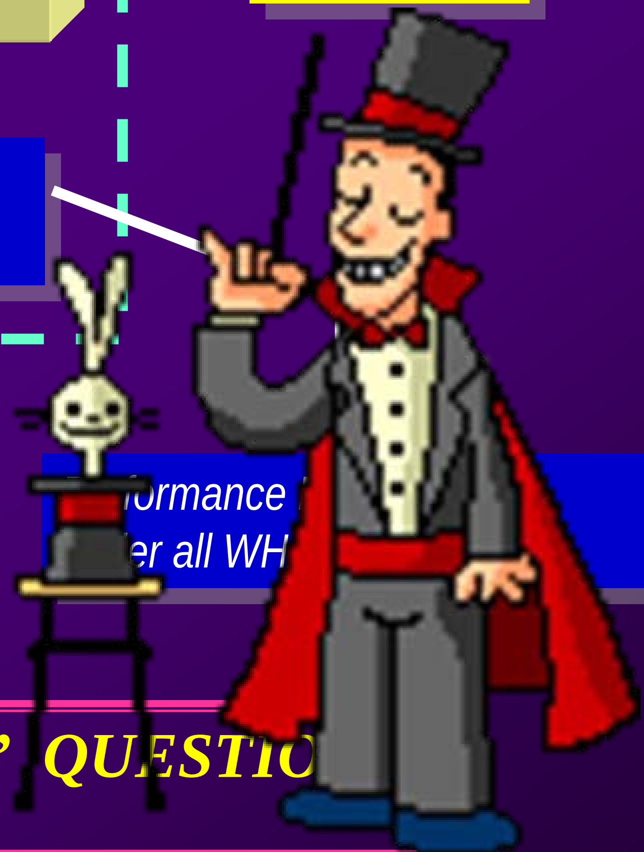
Design Parameters

Operating Policies, Control Parameters

SYSTEM

Performance Measures

PERTURBATION ANALYSIS



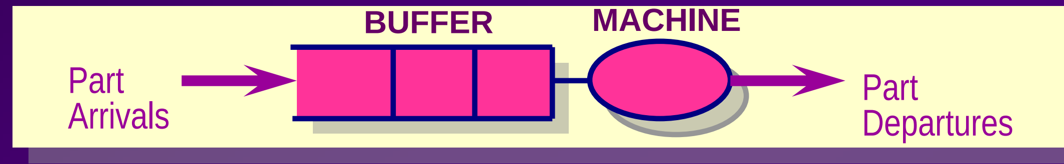
WHAT IF…

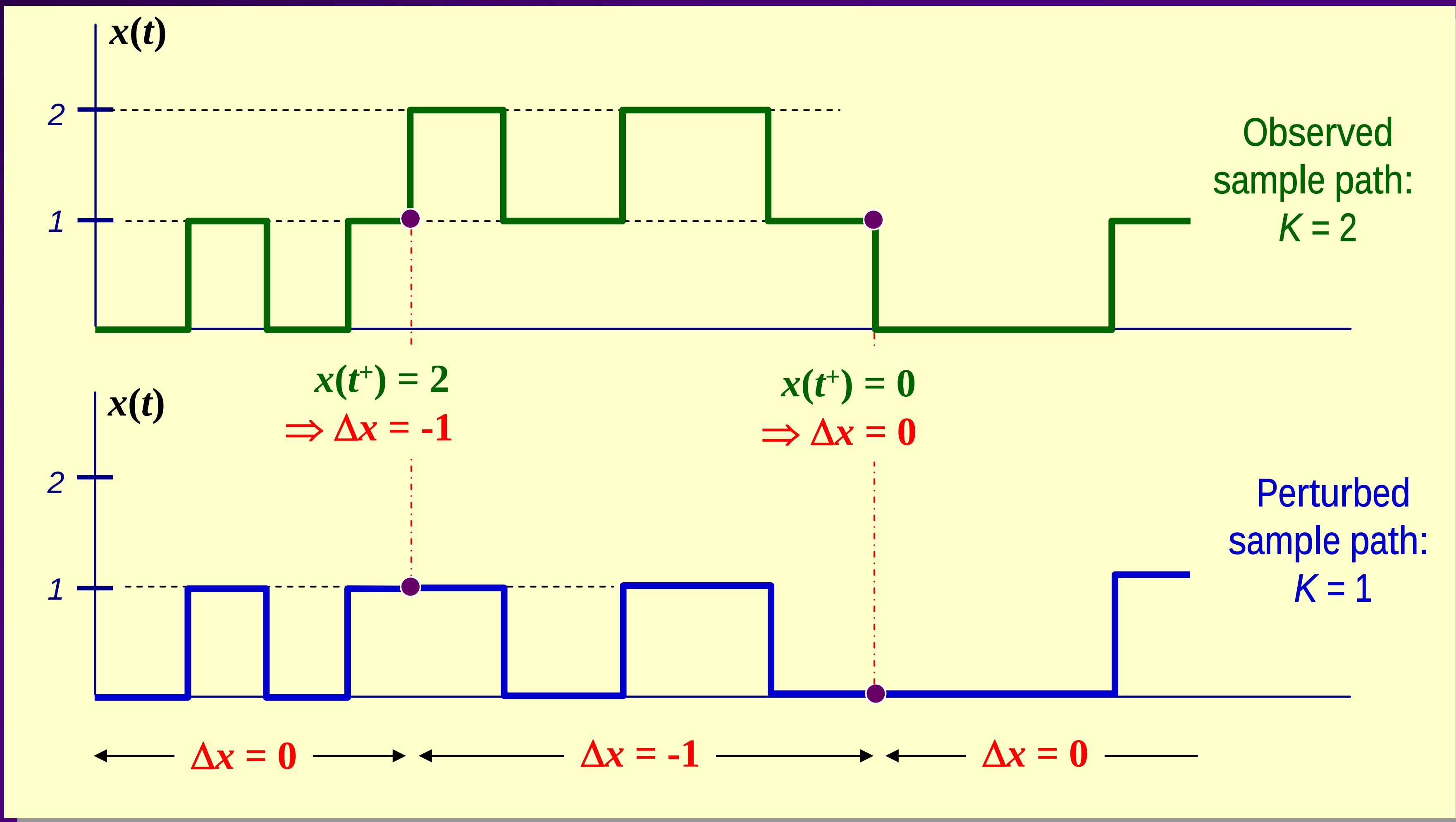
- Parameter $p_1 = a$ were replaced by $p_1 = b$
- Parameter $p_2 = c$ were replaced by $p_2 = d$

  ⋮

ANSWERS TO MULTIPLE "WHAT IF" QUESTIO AUTOMATICALLY PROVIDED

PERTURBATION ANALYSIS





PERTURBATION ANALYSIS

CONTINUED

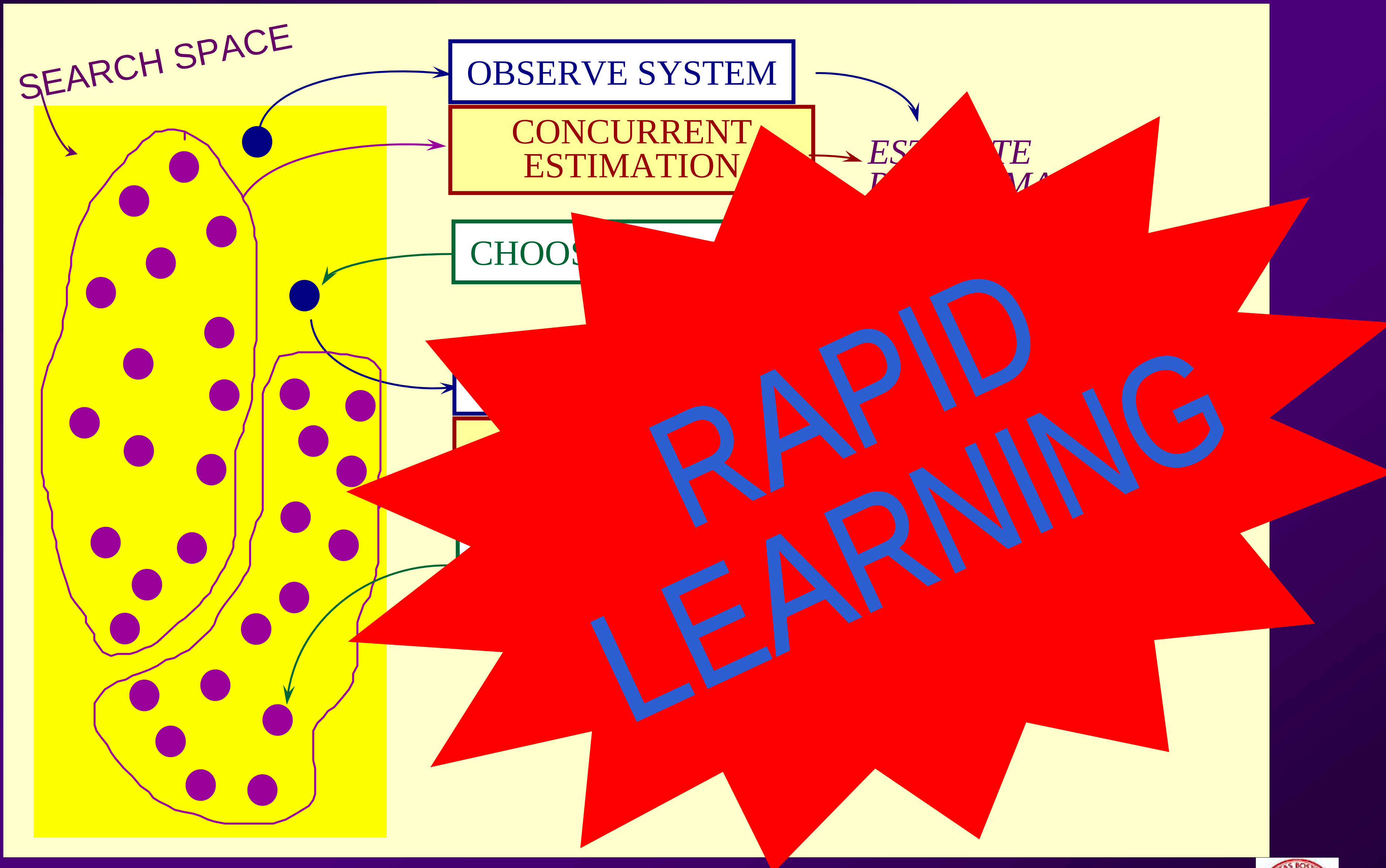
PERTURBATION DYNAMICS OBTAINED FROM OBSERVED NOMINAL SAMPLE PATH ONLY!

$$\Delta x(t+\delta;\theta,\Delta\theta) = f[\Delta x(t;\theta,\Delta\theta), x(t;\theta); \theta, \Delta\theta]$$

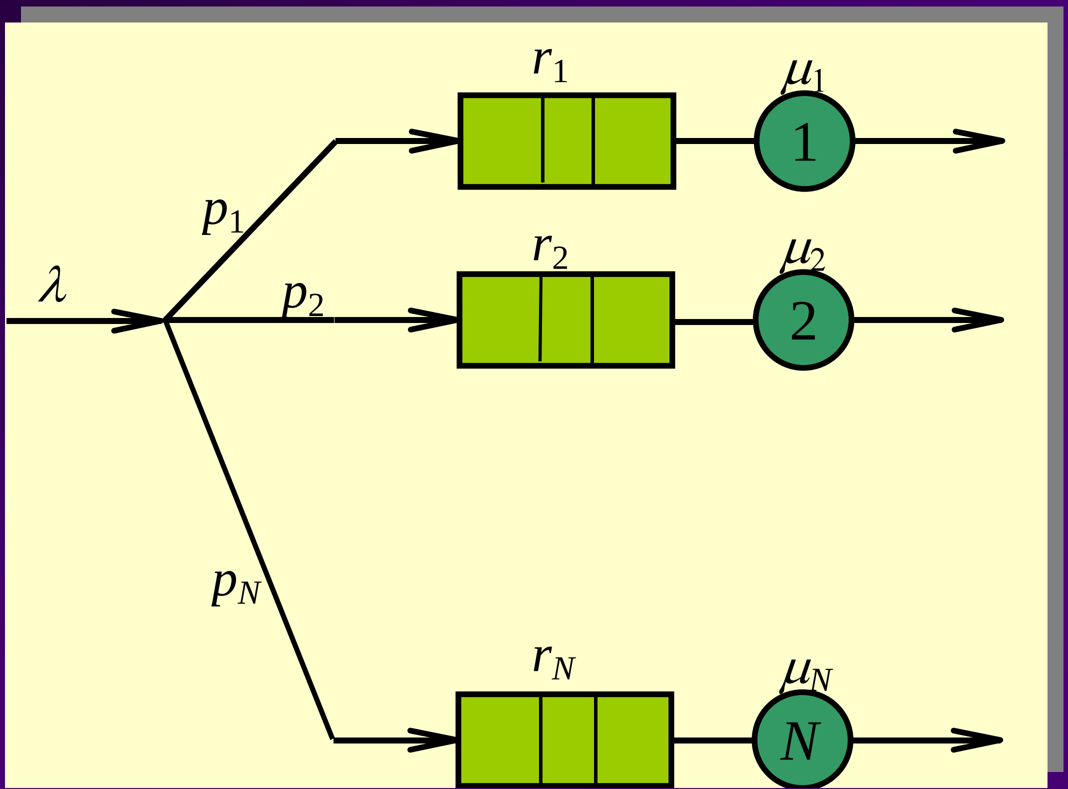
Why does this work?
Because structural knowledge of *nominal system dynamics* is also used

- Constructability Theory: Conditions under which this is possible and methods for constructing perturbed sample paths
- Concurrent Estimation: $\Delta J(t+\delta;\theta,\Delta\theta) = f[\Delta J(t;\theta,\Delta\theta), x(t;\theta); \theta, \Delta\theta]$
- Perturbation Analysis: Obtaining unbiased, consistent estimators for $\dfrac{dJ}{d\theta}$

CONCURRENT ESTIMATION



RESOURCE ALLOCATION EXAMPLE

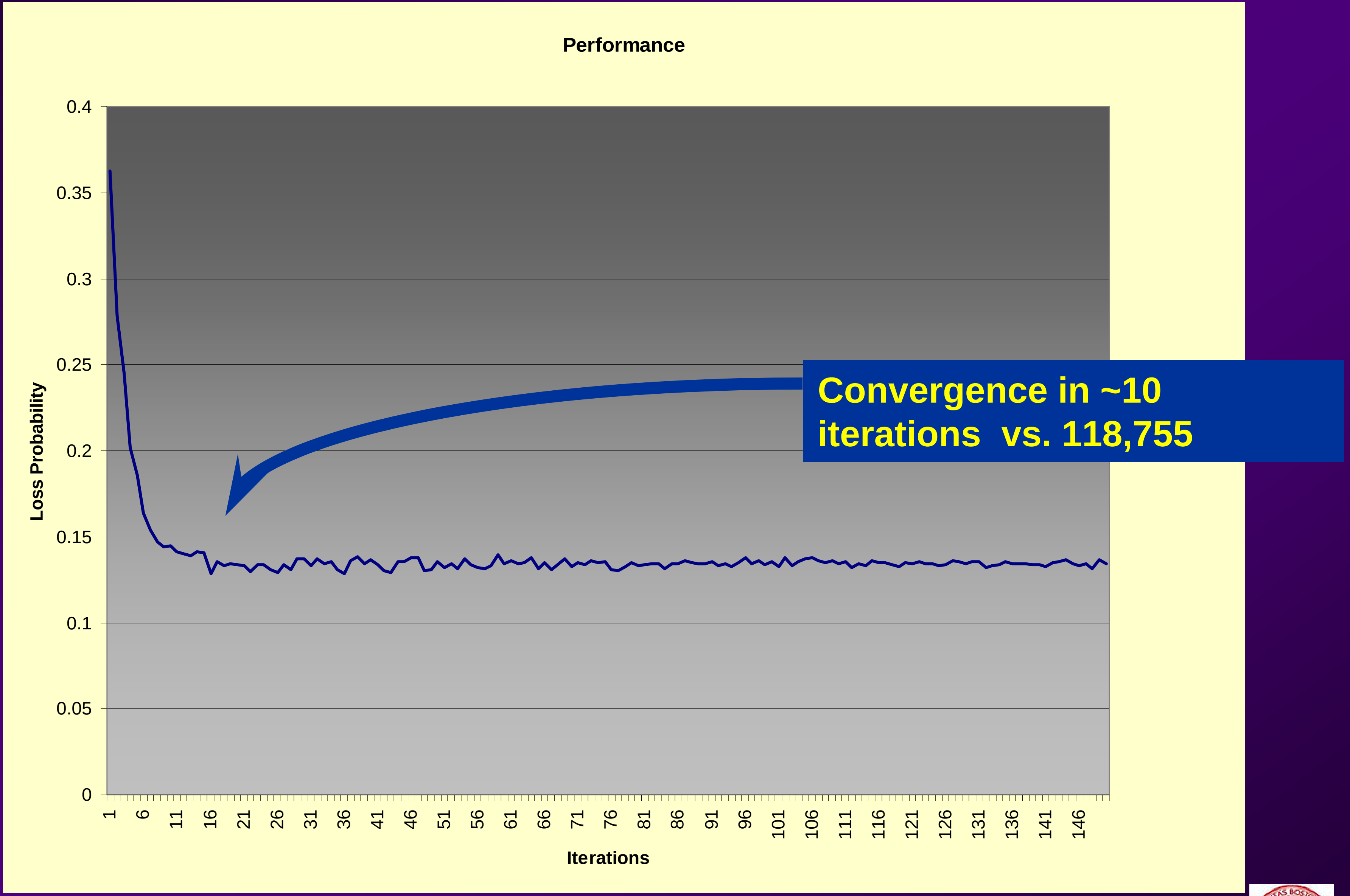


Allocate *K* buffer slots (RESOURCES) over *N* servers (USERS) so as to minimize *BLOCKING PROBABILITY*

subject to $\sum_{i=1}^{N} r_i = K$

For ***N*=6**, ***K*=24** $\rightarrow$ Possible allocations = **118,755**

"SURROGATE METHOD" RESULTS



OUTLINE

- The classic complex optimization problem
- "SURROGATE PROBLEM" METHODS:

  Solutions to *hard* problems can be recovered from those of *simpler* problems

- CONCURRENT ESTIMATION:

  Answering *N* "what if" questions may *not* need *N* trials

- DECOMPOSITION:

  Solutions to *hard* problems can be recovered from those of *simpler* problems

- APPLICATIONS TO SOME COMPLEX SYSTEMS:

  - Resource Allocation
  - Hybrid Systems

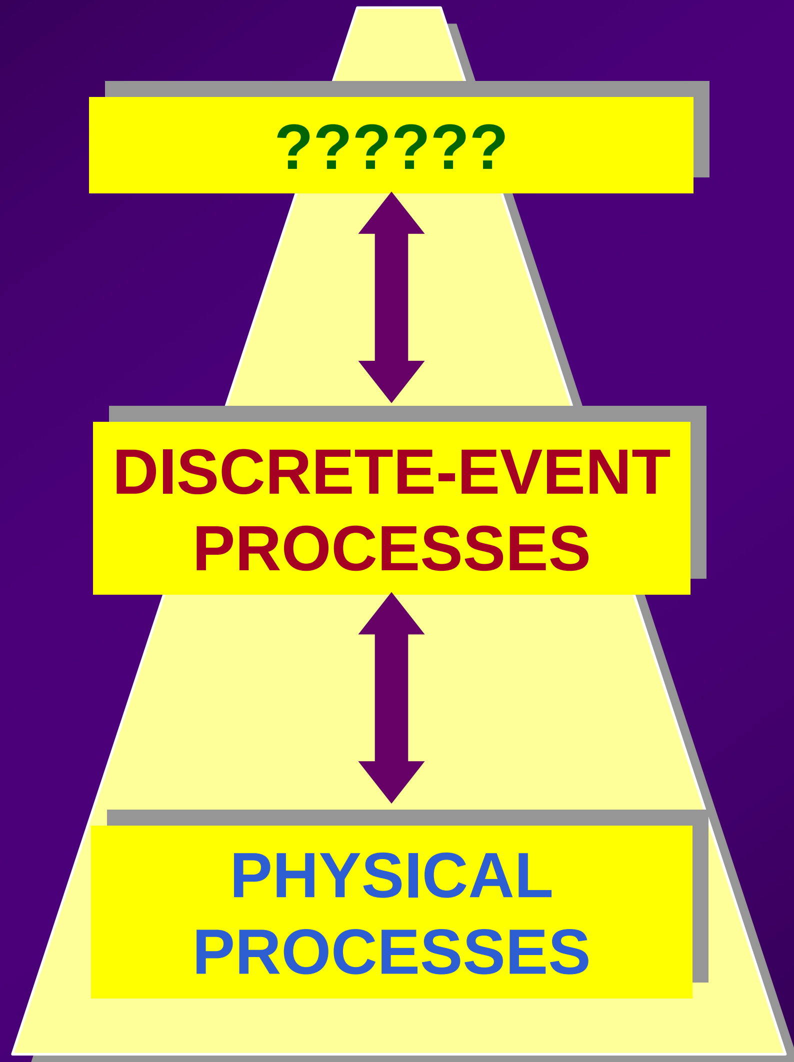
HIERARCHICAL DECOMPOSITION OF COMPLEX SYSTEMS

TIME SCALE

Weeks - Months

Minutes - Weeks

*m*sec - Hours



MODEL

Diff. Eq's, LP

Automata, Simulation, Queueing

Diff. Eq's, Detailed Simulation

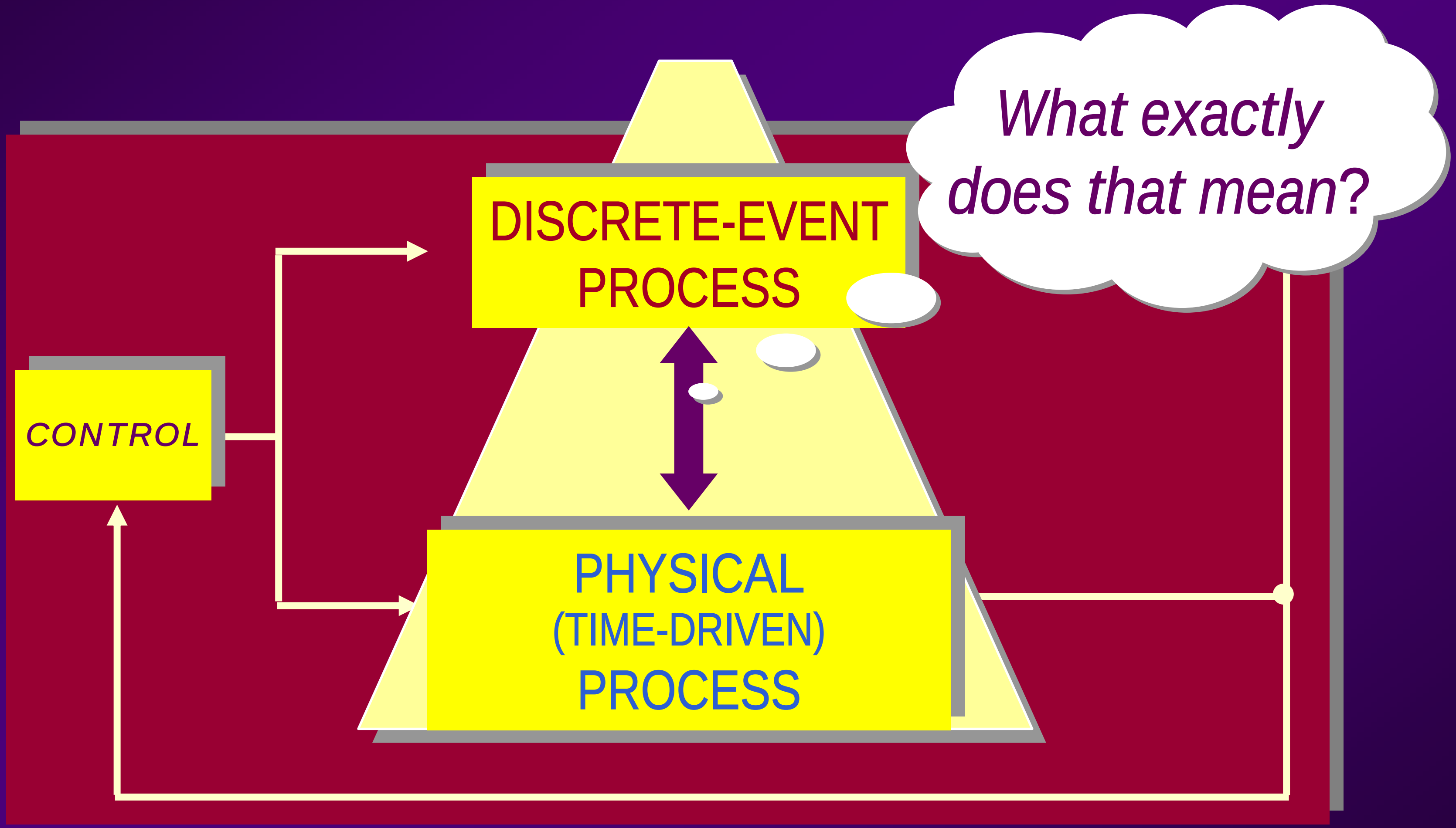
EXAMPLES

- **POWER SYSTEMS**: Unit Dynamics
  - → Operating Condition Changes
  - → Power Flows

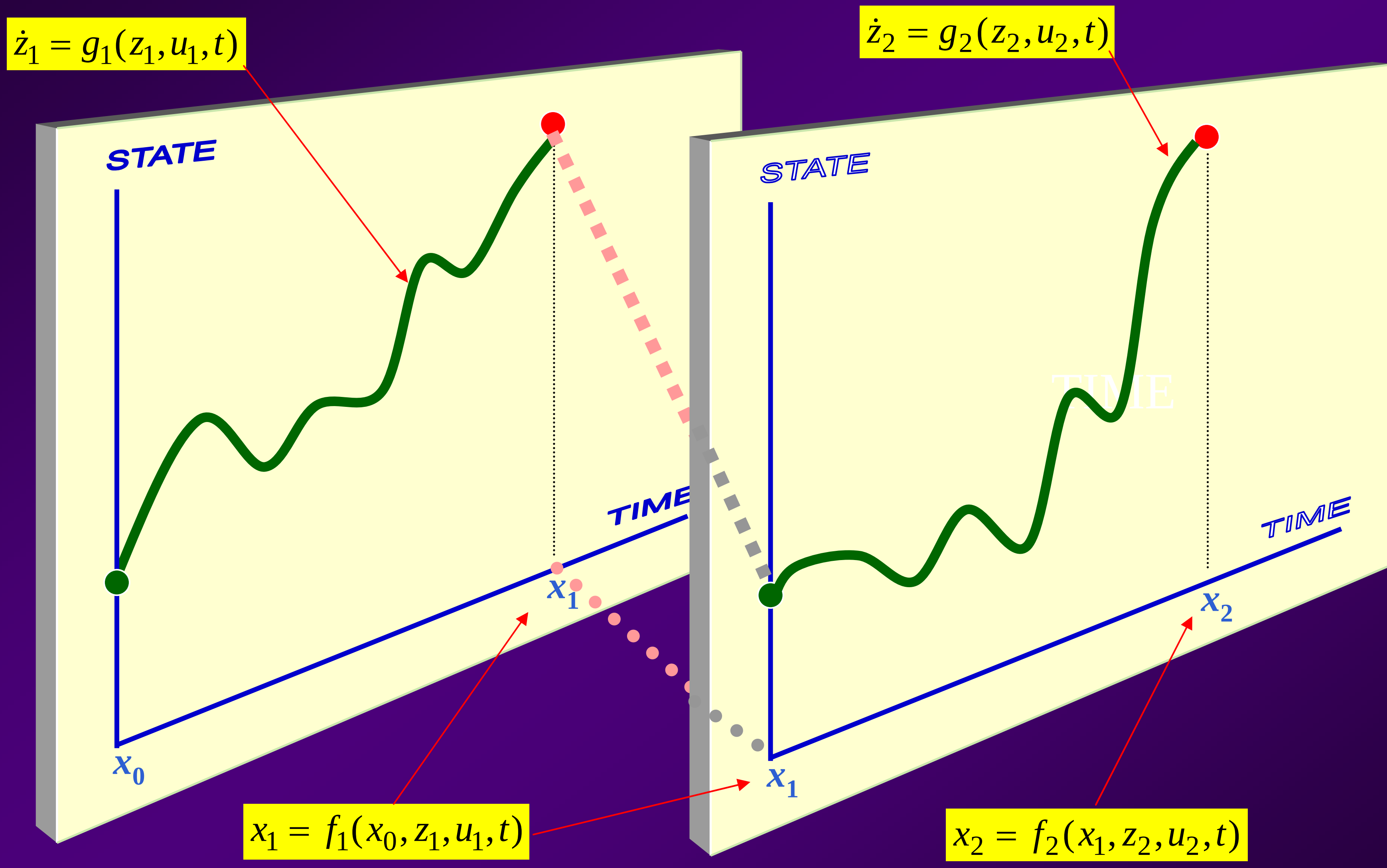
- **MANUFACTURING**: Physical Part Processing
  - → Start/Stop Control
  - → Strategic Planning

- **HYDRODYNAMICS**: Particle Dynamics
  - → Navier-Stokes Equations

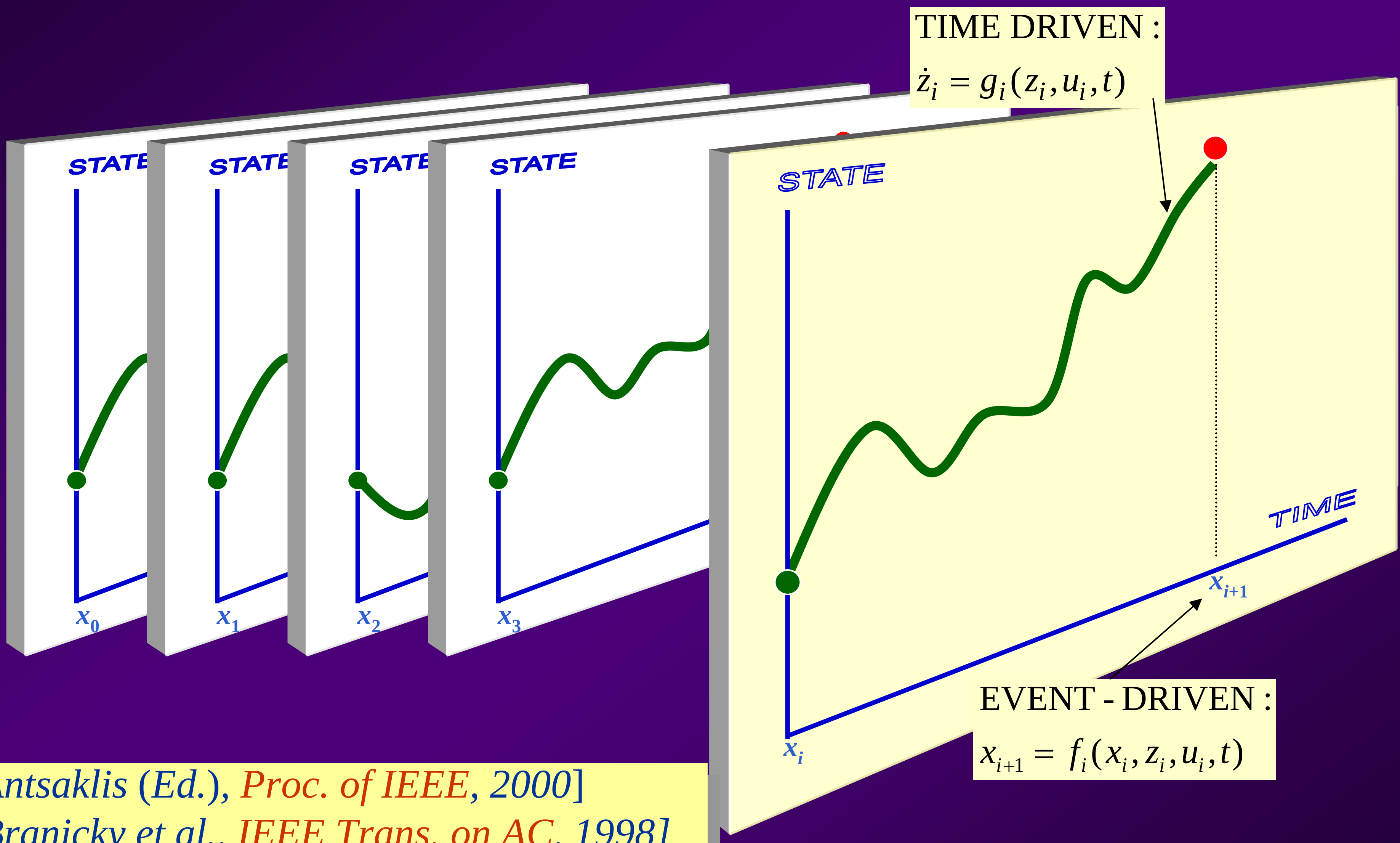
HYBRID SYSTEMS



HYBRID SYSTEM FRAMEWORK



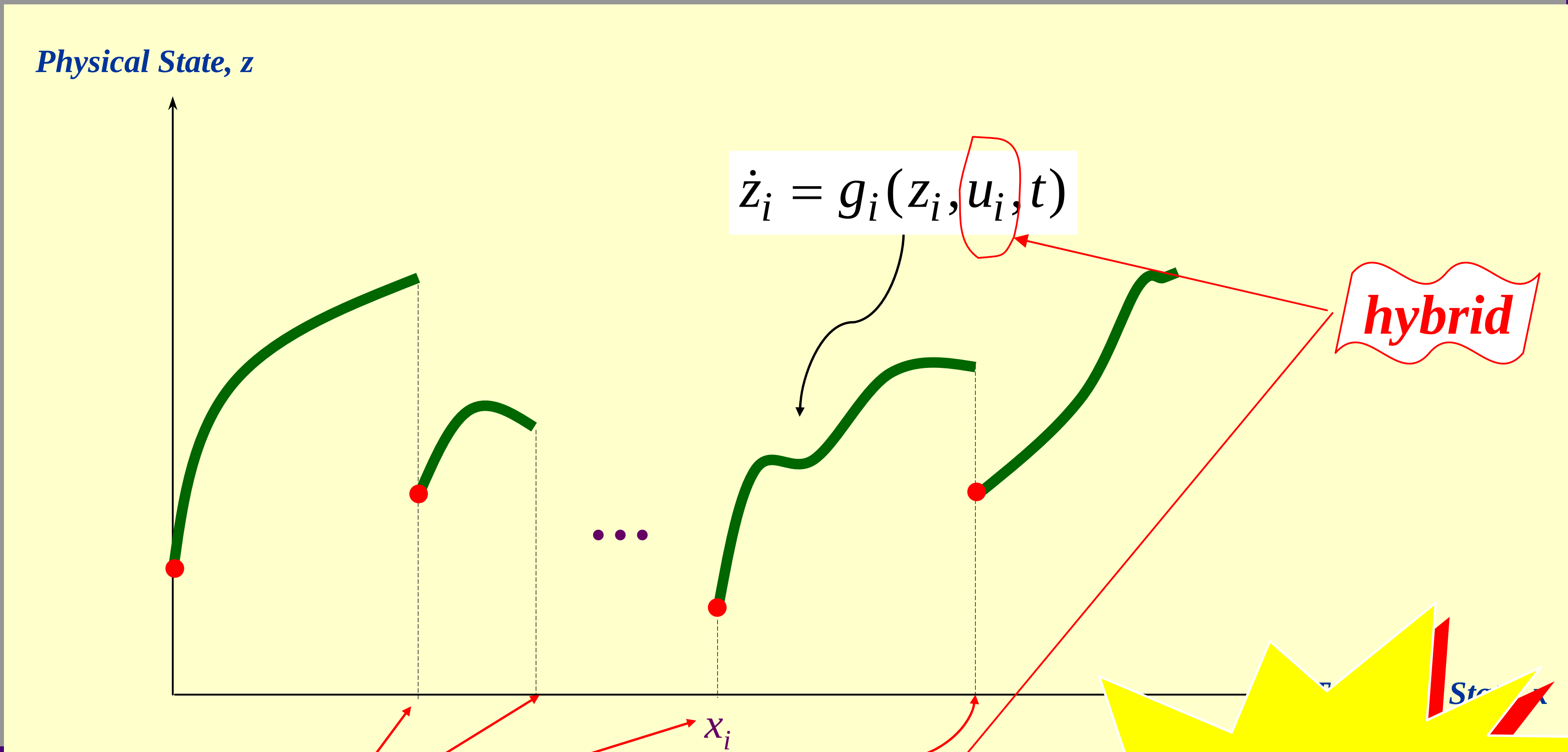
HYBRID SYSTEM FRAMEWORK



[*Antsaklis* (*Ed.*), *Proc. of IEEE*, *2000*]

[*Branicky et al.*, *IEEE Trans. on AC*, *1998*]

HYBRID SYSTEM FRAMEWORK



$$\dot{z}_i = g_i(z_i, u_i, t)$$

hybrid

Switching Times

$$x_{i+1} = f_i(x_i, u_i, t)$$

SWITCHING TIMES HAVE THEIR OWN DYNAMICS!

OPTIMAL CONTROL PROBLEMS

- Get to a desired final physical state z_N in minimum time x_N, subject to N-1 switching events

- Minimize deviations from N desired physical states: $(z_i - q_i)^2$
  and
  deviations from target desired times: $(x_i - \tau_i)^2$

In general:

$$\min_{\mathbf{u}} \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} L_i(z_i(t), x_i, u_i(t))dt \quad s.t. \begin{cases} \dot{z}_i = g_i(z_i, u_i, t) \\ x_{i+1} = f_i(x_i, u_i, t) \end{cases}$$

Physical state

Temporal state

TIME-DRIVEN AND EVENT-DRIVEN DYNAMICS

Time-Driven **Dynamics (*STATE* = *z*):**

$$\dot{z}(t) = g(z, u, t) \qquad \text{or:} \qquad z_{k+1} = g_k(z_k, u_k)$$

Event-Driven **Dynamics (*STATE* = Event Times $x_{k,i}$):**

$$x_{k+1,i} = \max_{j \in \Gamma_i} \{ x_{k,j} + \mathbf{a}_{k,j} \mathbf{u}_{k,j} \}$$

Event counter k = 1,2,...

Event index $i \in E = \{1,\ldots,n\}$

HYBRID SYSTEMS IN MANUFACTURING

Key questions facing manufacturing system integrators:

- How to integrate *'process control'* with *'operations control'* ?
- How to improve product [illegible]*Y* within reasonable ***TIME*** ?

PROCESS CONTROL
- Physicists
- Material Scientists
- Chemical Engineers
- …

Time-Driven World

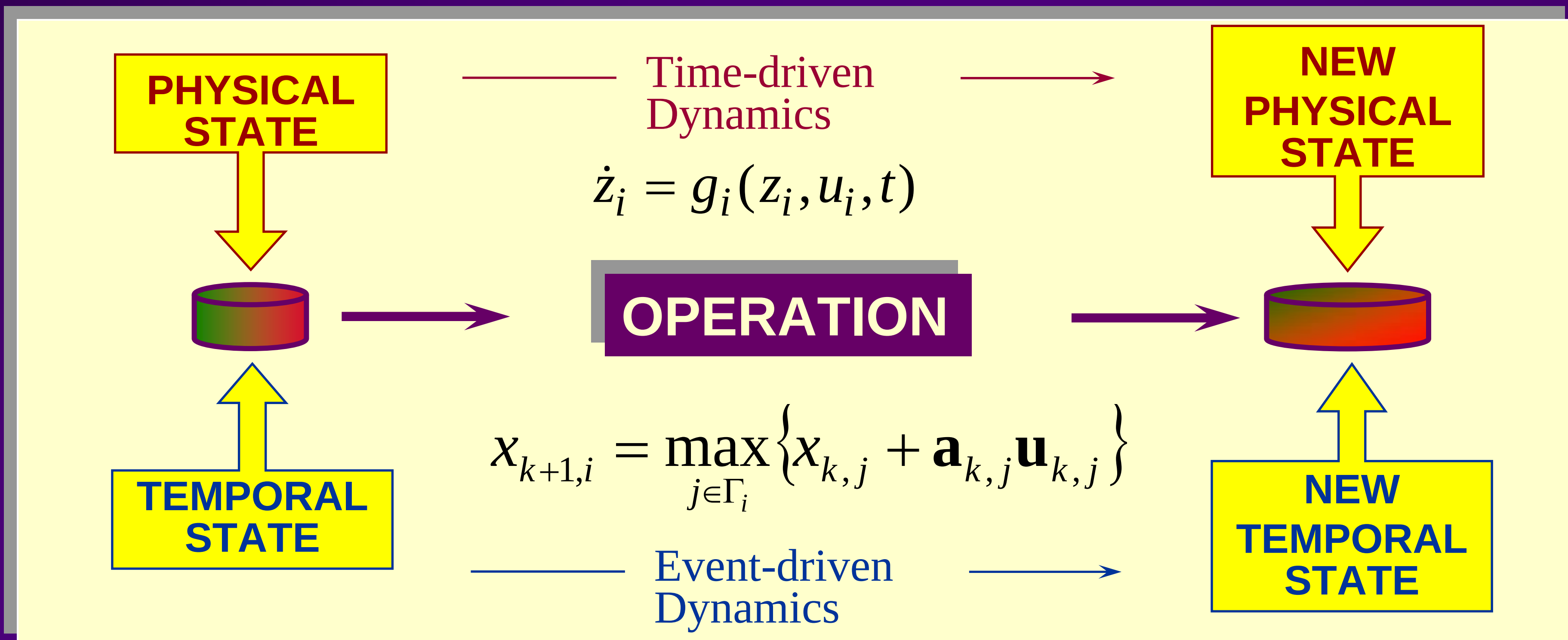
OPERATIONS CONTROL
- Industrial Engineers, OR
- Schedulers
- Inventory Control
- …

Event-Driven World

HYBRID SYSTEMS IN MANUFACTURING *CONTINUED*

Throughout a manuf. process, each part is characterized by

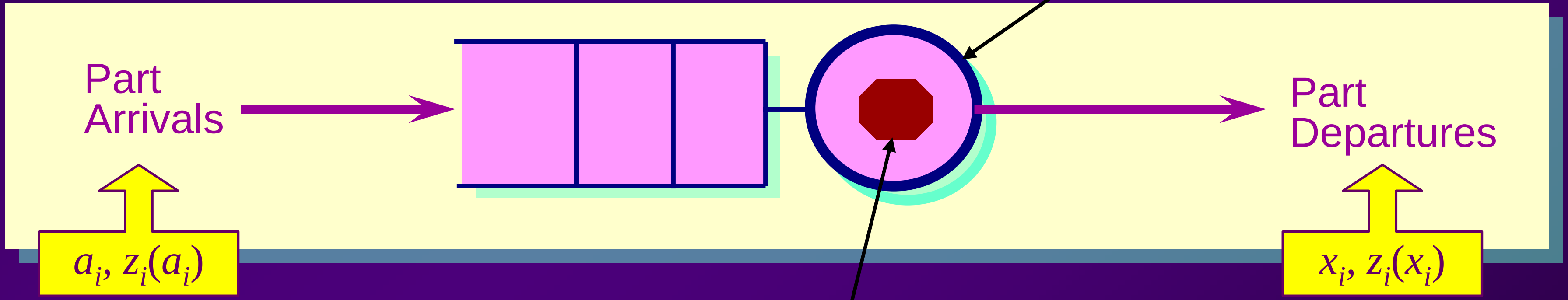
- A **PHYSICAL** state (e.g., size, temperature, strain)
- A **TEMPORAL** state (e.g., total time in system, total time to due-date)



HYBRID SYSTEMS IN MANUFACTURING *CONTINUED*

EVENT-DRIVEN COMPONENT

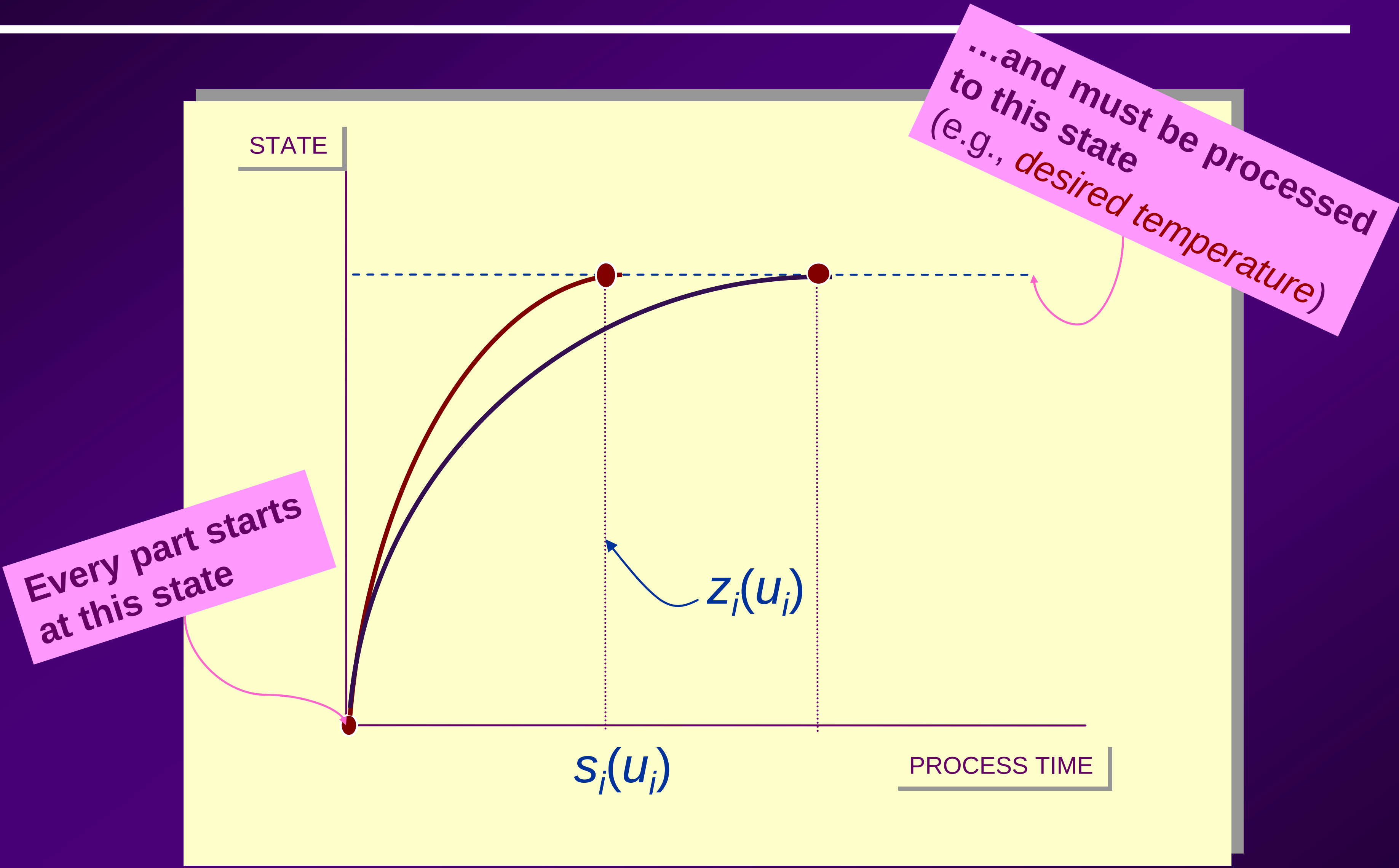
$$x_i = \max\{x_{i-1}, a_i\} + s_i(u_i)$$



TIME-DRIVEN COMPONENT

$$u_i \longrightarrow \dot{z}_i(t) = g(z_i, u_i, t)$$

EXAMPLE


STATE
PROCESS TIME
Every part starts at this state
...and must be processed to this state (e.g., desired temperature)
$z_i(u_i)$
$s_i(u_i)$

HIERARCHICAL DECOMPOSITION

$$\min_{\mathbf{u}} \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} L_i(z_i(t), x_i, u_i(t))dt \quad s.t. \quad \begin{cases} \dot{z}_i = g_i(z_i, u_i, t) \\ x_{i+1} = f_i(x_i, u_i, t) \end{cases}$$

Let: $s_i = x_i - x_{i-1}$

Time spent at ith operating region (mode)

Consider objective functions:

$$\min_{\mathbf{u}} \sum_{i=1}^{N} \left[\phi_i(z_i, u_i, s_i) + \psi_i(x_i, s_i)\right]$$

Physical process

Switching time process

HIERARCHICAL DECOMPOSITION

CONTINUED

$$\min_{\mathbf{u}} \sum_{i=1}^{N} \left[\phi_i(z_i, u_i, s_i) + \psi_i(x_i, s_i) \right] \quad s.t. \quad \begin{cases} \dot{z}_i = g_i(z_i, u_i, t) \\ x_{i+1} = f_i(x_i, u_i, t) \end{cases}$$

HIGHER LEVEL PROBLEM:

$$\min_{\mathbf{s}} \sum_{i=1}^{N} \left[\phi_i^*(s_i) + \psi_i(x_i, s_i) \right]$$

s.t.

$$x_{i+1} = f_i(x_i, s_i, t)$$

LOWER LEVEL PROBLEMS:

$$\min_{u_i} \phi_i(z_i, u_i, s_i)$$

s.t.

$$\dot{z}_i = g_i(z_i, u_i, t)$$

FIXED s_i

LOWER LEVEL PROBLEM

LQ PROBLEM:

Parameterized by switching times

$$\min_{u_i} \phi_i(z_i, u_i, s_i) = \frac{1}{2} h (z_{fi} - z_{di})^2 + \int_0^{s_i} \frac{1}{2} r u_i^2(t) dt$$

$s.t.$ $\dot{z}_i = a z_i + b u_i, \quad z_i(0) = \varsigma_i$

Penalize final state deviation

STANDARD LQ SOLUTION METHOD:

$$\phi_i^*(s_i) = \frac{1}{2} h (z_{fi}^* - z_{di})^2 + \int_0^{s_i} \frac{1}{2} r u_i^*(t) dt$$

HIGHER LEVEL PROBLEM

$$\min_{\mathbf{s}} \sum_{i=1}^{N} \left[\phi_i^*(s_i) + \psi_i(x_i) \right] \quad s.t. \quad x_i = \max\{x_{i-1}, a_i\} + s_i(u_i)$$

Cost of optimal process control over interval [0, s_i]

Cost related to event timing

Given arrival sequence (INPUT)

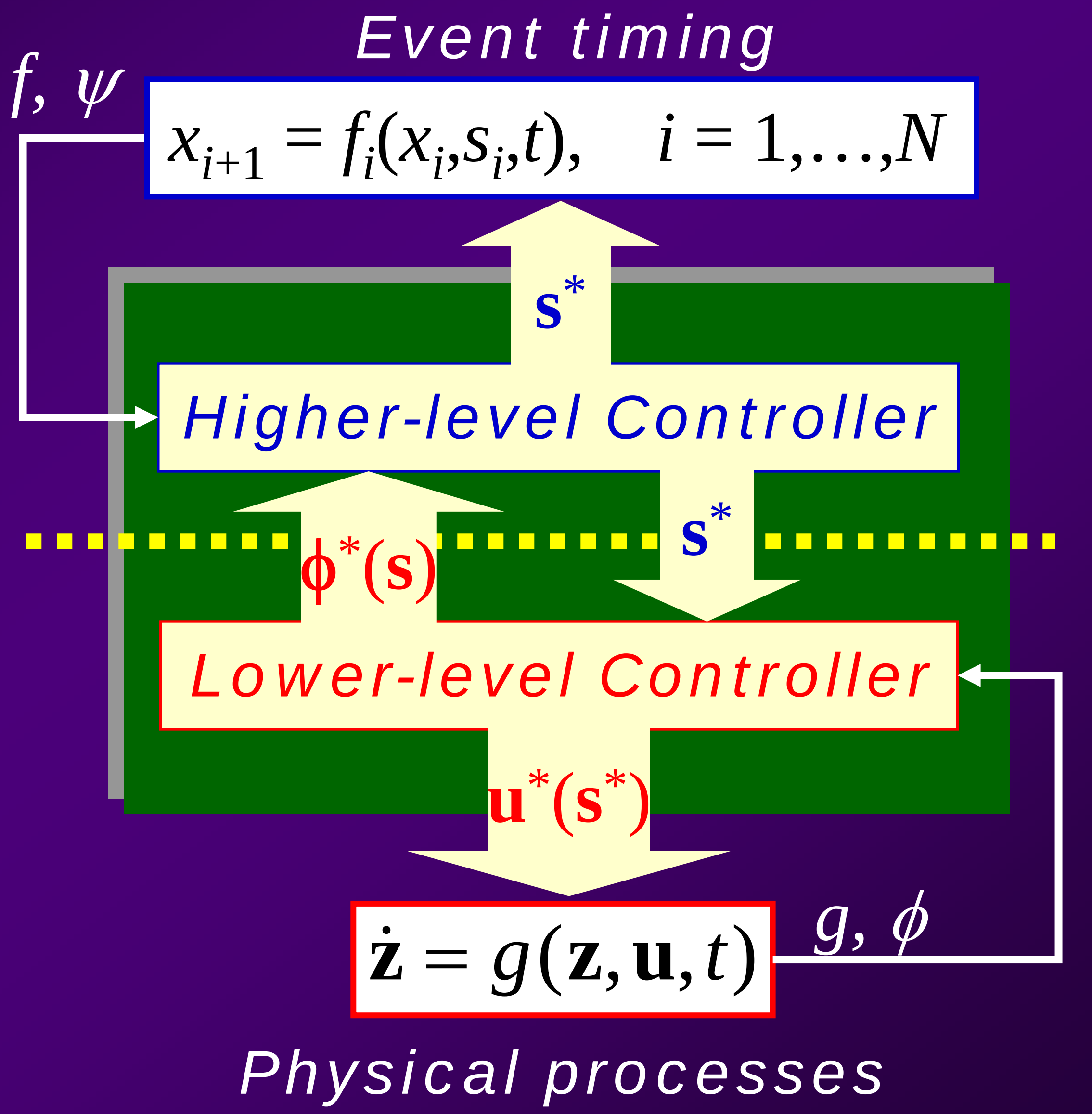
Processing time (CONTOLLABLE)

$$\text{EXAMPLE :} \quad \psi_i(x_i) = (x_i - \tau_i)^2$$

HYBRID CONTROLLER STRUCTURE

Hybrid controller steps:

- **System identification**
- **Lower-level solution**
- **Higher-level solution**
- **Lower-level solution**
- **Operation...**



HOW DO WE SOLVE THE HIGHER LEVEL PROBLEM?

$$\min_{\mathbf{s}} \sum_{i=1}^{N} \left[\phi_i^*(s_i) + \psi_i(x_i) \right] \quad s.t. \quad \boxed{x_i = \max\{x_{i-1}, a_i\} + s_i(u_i)}$$

Causes nondifferentiabilities!

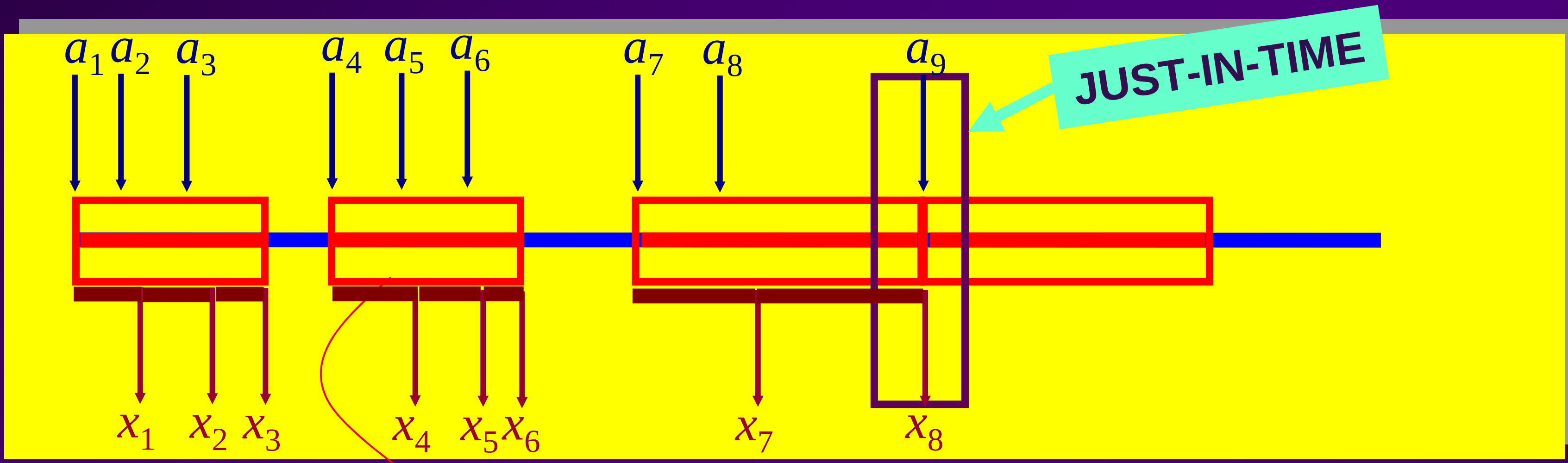
Even if these are convex,
problem is still NOT convex in s!

1. Even though problem is NONDIFFERENTIABLE and NONCONVEX, optimal solution shown to be ***unique***.

[*Cassandras et al., IEEE Trans. on AC, 2001*]

SOLVING THE HIGHER LEVEL PROBLEM *CONTINUED*

2. Optimal state trajectory can be ***decomposed*** into “blocks”



Each “block” corresponds to a ***Constrained Convex Optimization*** problem

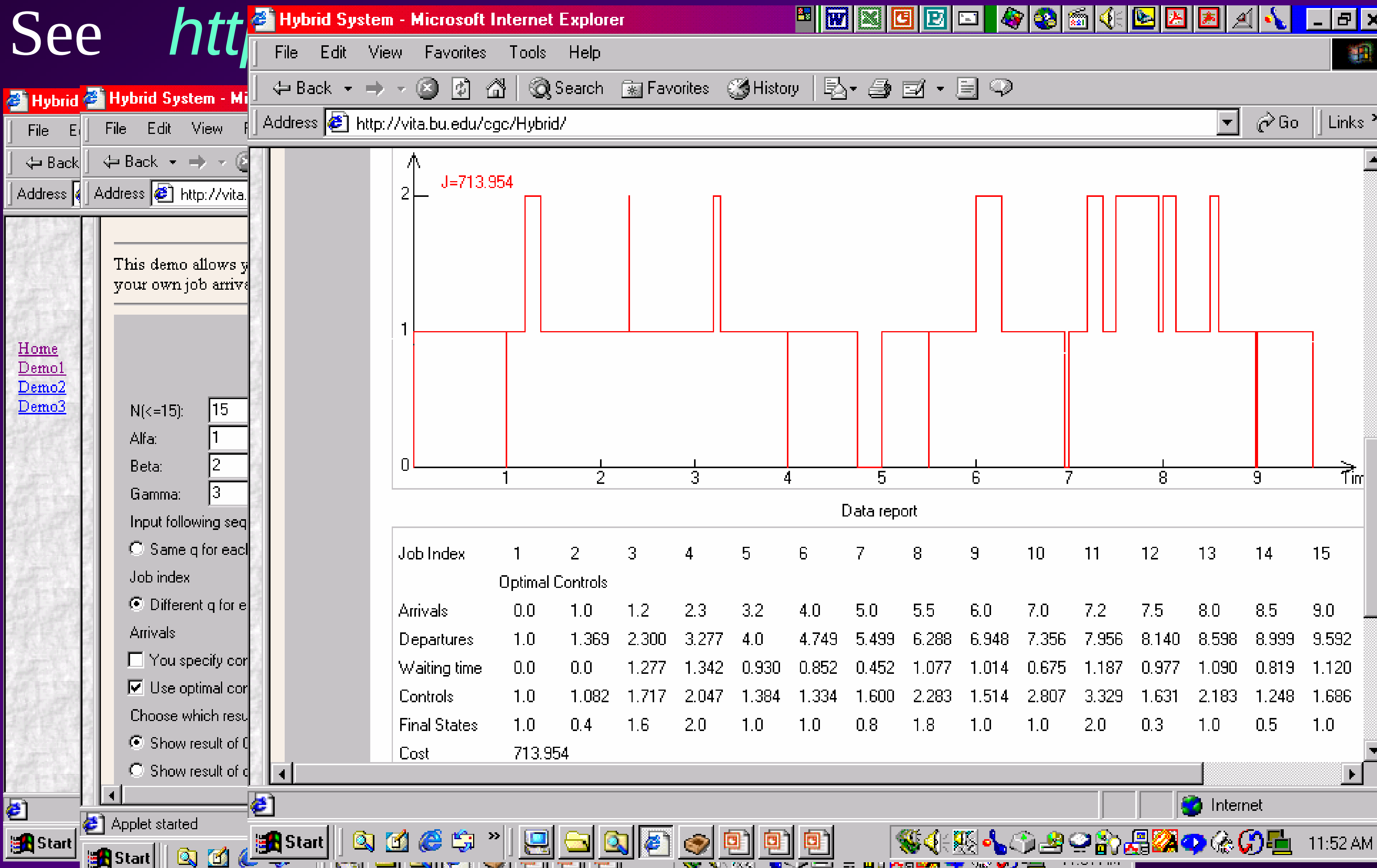
$\Rightarrow$ search over 2^{N-1} possible ***Constrained Convex Optimization*** problems

BUT

algorithms that only need N ***Constrained Convex Optimization*** problems have been developed $\Rightarrow$ SCALEABILITY

EXAMPLES: INTERACTIVE JAVA APPLETS

See *htt*



SOME OPEN PROBLEMS, RESEARCH AREAS

- *Ordinal Optimization*
- *Generality of "Surrogate" approaches*
- *Dynamic Optimization of Complex Systems (beyond Dynamic Programming…)*
- *Hybrid Systems: Modeling, Verification, Stochastic Optimization, Computation Methods*
- *Robustness vs. Optimality in Complex Systems*
- *What is the right resolution level for modeling Complex Systems?*

ACKNOWLEDGEMENTS

Thanks to several co-workers whose contributions are reflected in this talk…

… All 13 Former
+ Current PhD Students…

…Colleagues…

Y.C. [illegible] Ho | **L[illegible]i Da[illegible]**
We[illegible] | **X[illegible]en C[illegible]o**
Yor[illegible]d | **T[illegible]d Dja[illegible]eri[illegible]**
You[illegible] | **D[illegible]ug L[illegible]oz[illegible]**
P[illegible]a [illegible]azq[illegible]-Abad

MULTUMESC

…and Sponsors…

NSF, AFOSR, AFRL, DARPA, NASA, EPRI, ARO, UT, ALCOA, NOKIA, GE