



IEIIT-CNR



# Las Vegas and Monte Carlo Randomized Algorithms for Systems and Control

Roberto Tempo

IEIIT-CNR

Politecnico di Torino

[roberto.temp@polito.it](mailto:roberto.temp@polito.it)



IEIIT-CNR

# CSL UIUC



## ■ Six months at CSL in 2002...





- A Success Story
- Randomized Algorithms, Monte Carlo and Las Vegas
- Lyapunov Stability Analysis and Synthesis
- RandQuickSort for Switched Systems



IEIIT-CNR



# A Success Story



- Randomized Algorithms (RAs) are successfully used in various areas, including computer science, numerical analysis, optimization, ...

... but in systems and control their use is often limited to Monte Carlo simulations

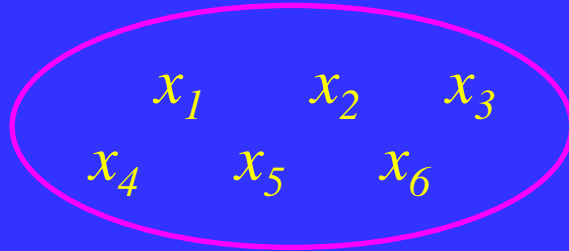
- Example: Sorting problem
- Algorithm: RandQuickSort (RQS)
- RQS is implemented in the *Linux* sorting command



# RandQuickSort (RQS)

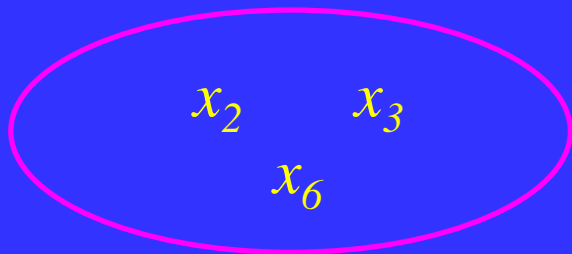


given  $n$  real numbers



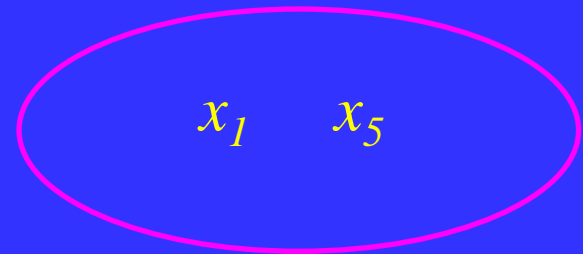
need to sort them in increasing order

- RQS is an iterative algorithm consisting of two phases
  1. randomly select a number  $x_i$  (e.g.  $x_4$ )
  2. perform deterministic comparisons between  $x_i$  and  $(n-1)$  remaining numbers



numbers smaller than  $x_4$

$\leq x_4 \leq$



numbers larger than  $x_4$



# Running Time of RQS

- Because of randomization, running time may be different from one execution of the algorithm to the next one
- RQS is very fast: average running time is  $O(n \log (n))$
- This is a major improvement compared to brute force approach for example when  $n = 2^m$
- Average running time is also a *highly probable* running time (Chernoff bound)



IEIIT-CNR



# Randomized Algorithms, Monte Carlo and Las Vegas



# Randomized Algorithm: Definition

- **Randomized Algorithm (RA):** An algorithm that makes random choices during execution to produce a result
- For hybrid systems, “random choices” could be switching between different states or logical operations
- For uncertain systems, “random choices” require (vector or matrix) random sample generation



# Monte Carlo and Las Vegas RA

- Monte Carlo Randomized Algorithm (MCRA): A randomized algorithm that may produce incorrect results, but with bounded error probability
- Las Vegas Randomized Algorithm (LVRA): A randomized algorithm that always produces correct results, the only variation from one run to another is the running time



- Consider random uncertainty  $\Delta$  and a bounding set  $B$
- $\Delta$  is a (real or complex) random vector (parametric uncertainty) or matrix (nonparametric uncertainty)
- Consider a performance function

$$J(\Delta): \mathbf{R}^{n,m} \rightarrow \mathbf{R}$$

and level  $\gamma > 0$

- Define worst case and average performance

$$J_{max} = \max_{\Delta \in B} J(\Delta) \qquad J_{ave} = E_{\Delta}(J(\Delta))$$



# Example - $H_\infty$ Performance

- $H_\infty$  performance of sensitivity function

$$S(s, \Delta) = 1 / (1 + P(s, \Delta) C(s))$$

$$J(\Delta) = \|S(s, \Delta)\|_\infty$$

- Objective: Check if

$$J_{max} \leq \gamma \quad \text{and} \quad J_{ave} \leq \gamma$$

- These are uncertain decision problems



# Two Problem Instances

- We have two problem instances for worst case performance

$$J_{max} \leq \gamma \quad \text{and} \quad J_{max} > \gamma$$

and two problem instances for average case performance

$$J_{ave} \leq \gamma \quad \text{and} \quad J_{ave} > \gamma$$

- This leads to one-sided and two-sided MC randomized algorithms



- One-sided MCRA: Always provide a correct solution in one of the instances (they may provide a wrong solution in the other instance)
- Consider the empirical maximum

$$\hat{J}_{max} = \max_{i=1, \dots, N} J(\Delta^i)$$

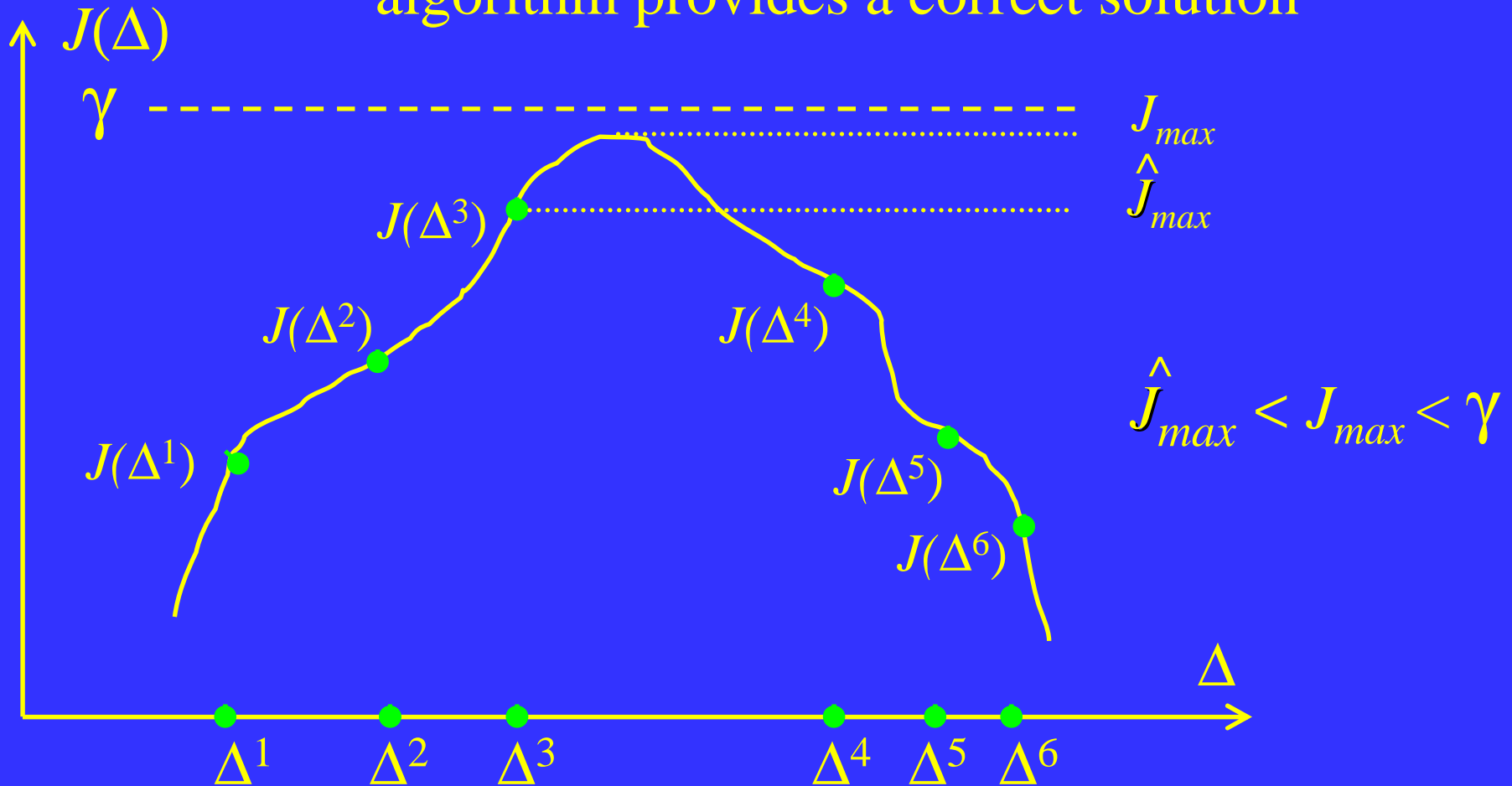
where  $N$  is the sample size

- Check if  $\hat{J}_{max} \leq \gamma$  or  $\hat{J}_{max} > \gamma$



# One-Sided MCRA: Case 1

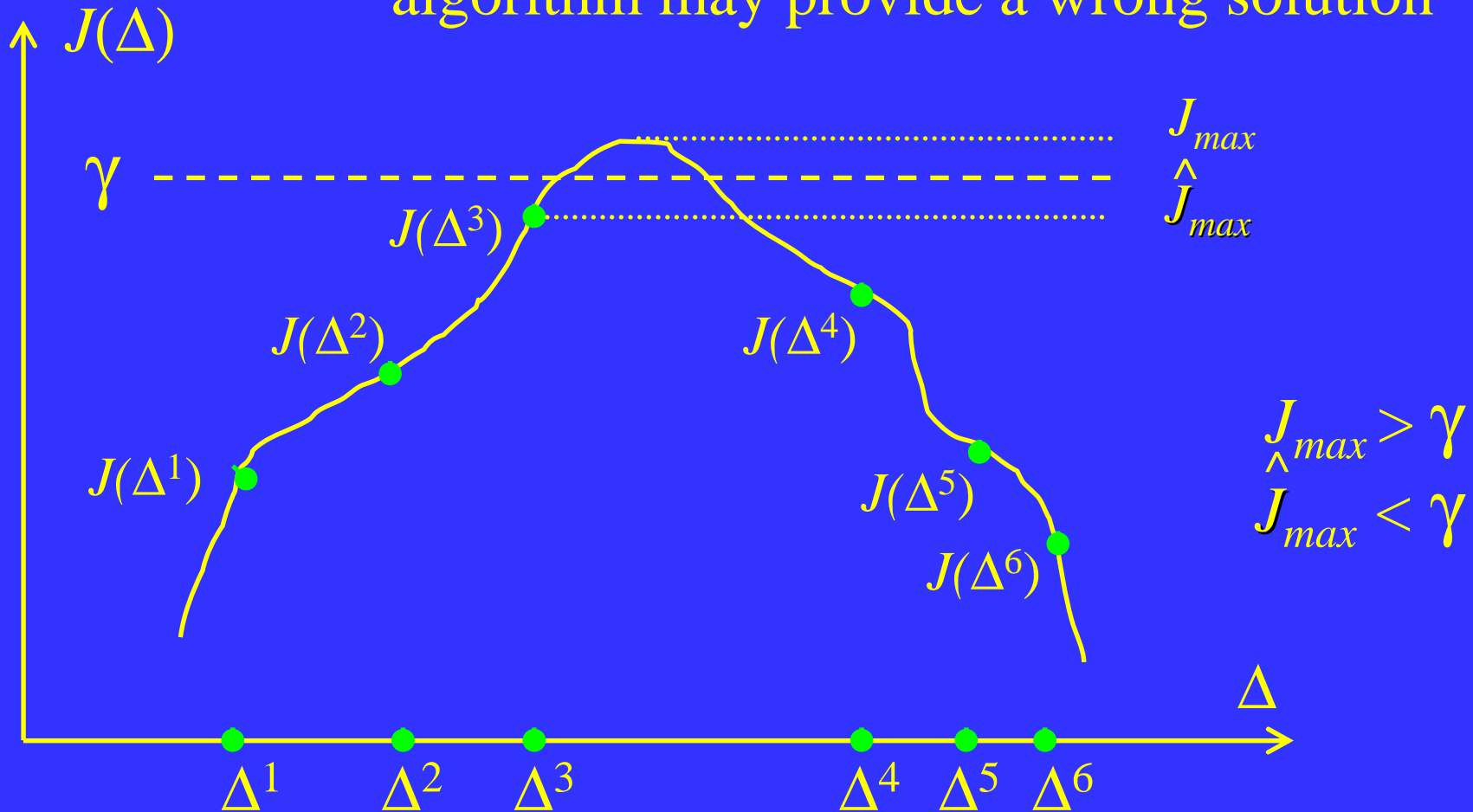
algorithm provides a correct solution





# One-Sided MCRA: Case 2

algorithm may provide a wrong solution





- Two-sided MCRA: They may provide a wrong solution in both instances
- Consider the empirical average

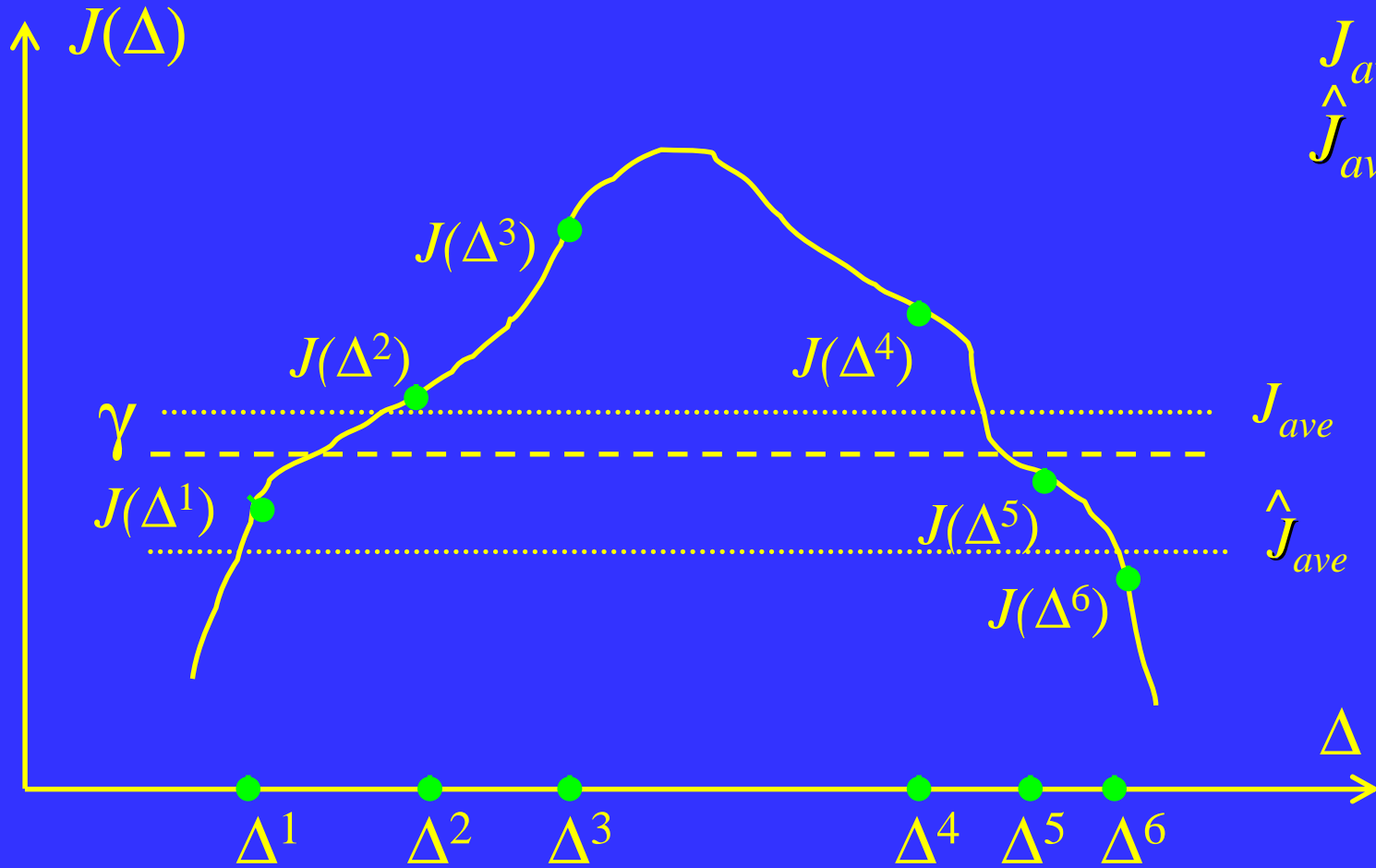
$$\hat{J}_{ave} = \text{ave}_{i=1, \dots, N} J(\Delta^i)$$

where  $N$  is the sample size

- Check if  $\hat{J}_{ave} \leq \gamma$  or  $\hat{J}_{ave} > \gamma$



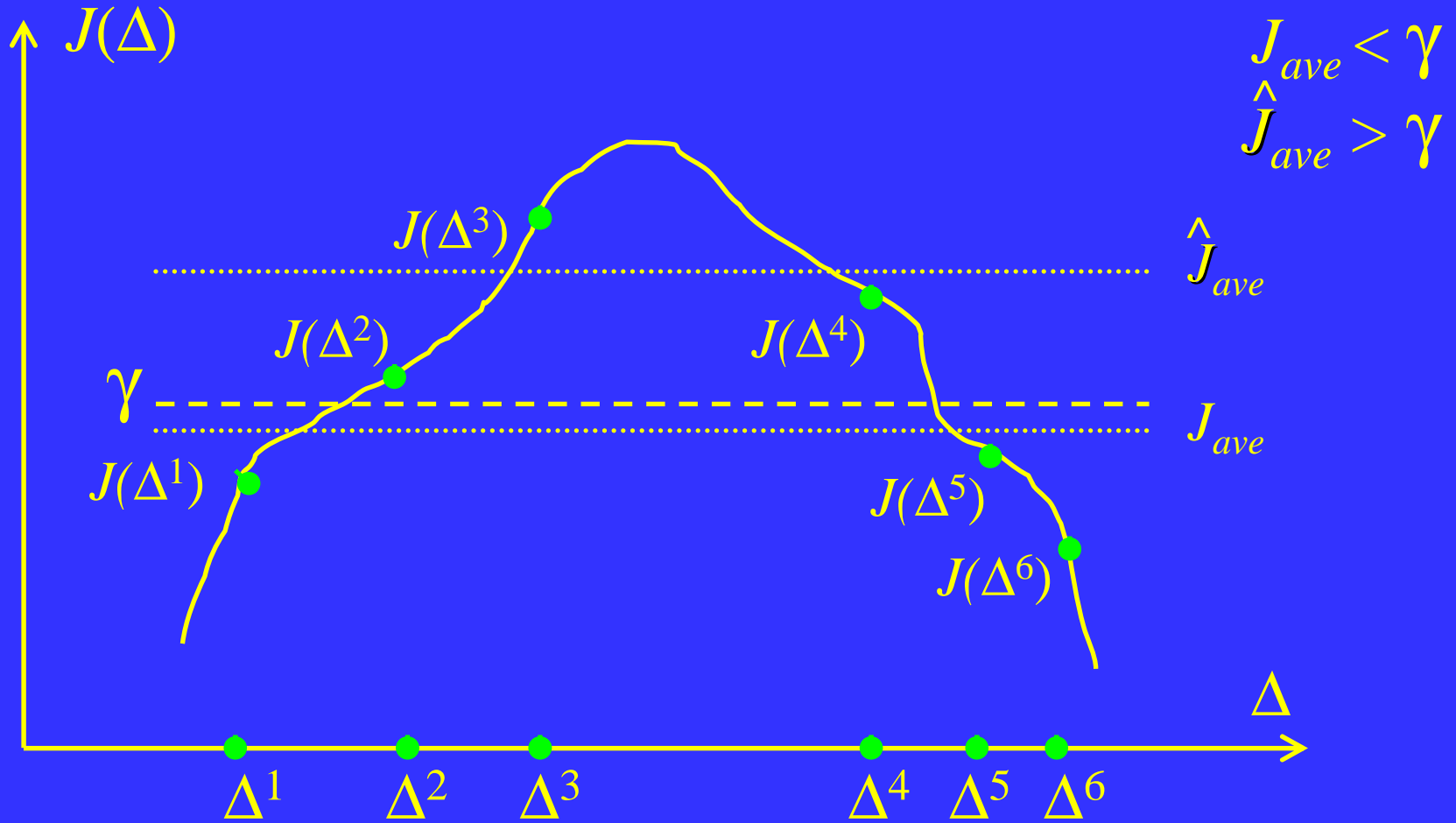
# Two-Sided MCRA



$$J_{ave} > \gamma$$
$$\hat{J}_{ave} < \gamma$$



# Two-Sided MCRA





# Las Vegas Randomized Algorithms

- We also have zero-sided (Las Vegas) randomized algorithms
- Las Vegas Randomized Algorithm (LVRA): Always give the correct solution
- Running time may be different from one run to another
- LVRA has more limited applicability than MCRA
  
- Example: RandQuickSort



IEIIT-CNR



# Lyapunov Stability Analysis and Synthesis



# Lyapunov Stability of Interval Systems

- Example: Quadratic Lyapunov stability analysis and synthesis of interval systems
- These are *NP*-hard problems
- They can be solved in one-shot with convex optimization, but the number of constraints is exponential
- We can use relaxation or randomization



# Uncertain Systems in State Space

- Consider the uncertain system

$$\dot{x}(t) = A(\Delta) x(t) + B u(t)$$

and state feedback

$$u(t) = K x(t)$$

with  $x(0) = x_0$ ,  $x \in \mathbf{R}^n$ ,  $u \in \mathbf{R}^m$ ,  $A(\Delta) \in A$



# Interval and Vertex Matrices

- Uncertain matrix  $A$  is interval with radius  $\rho$ .
- That is,  $a_{ik}$  ranges in the interval for all  $i, k$

$$|a_{ik} - a_{ik}^*| \leq w_{ik} \rho$$

where  $a_{ik}^*$  are nominal values and  $w_{ik}$  are weights

- Define the  $N = 2^{n^2}$  vertex matrices  $A^1, A^2, \dots, A^N$

$$a_{ik} = a_{ik}^* + w_{ik} \rho \quad \text{or} \quad a_{ik} = a_{ik}^* - w_{ik} \rho$$

for all  $i, k = 1, 2, \dots, n$



# Lyapunov Stability Analysis

- Given matrices  $A^*$ ,  $W$  and feedback  $K$ , find the supremum of  $\rho \geq 0$  such that there exists a common quadratic Lyapunov function  $P$  for the system

$$\dot{x}(t) = (A + B K) x(t) \quad A \in A$$

- That is,

$$\rho^{qs} = \sup_{\rho, P} \{ \rho : L(P) < 0, P > 0 \text{ for all } A \in A \}$$

where  $L(P) = (A+BK)^T P + P (A+BK)$



# Relaxation Approach: $\pi/2$ Theorem

- Conservative approximation  $\rho^-$  of  $\rho^{qs}$  can be computed in polynomial-time solving a generalized eigenvalue problem
- Using the  $\pi/2$  Theorem<sup>[1]</sup>, we obtain<sup>[2]</sup>

$$\rho^- \leq \rho^{qs} \leq \pi/2 \rho^-$$

[1] Yu. Nesterov (1997)

[2] A. Ben-Tal and A. Nemirovski (2002)



- Various methods have been recently proposed for finding a *probability-one* common solution  $P$  of the Lyapunov equation<sup>[1]</sup>

$$L(P) = A^T P + P A < 0$$

for all  $A \in A$

- Is this probability-one solution a robust solution?

[1] B. Polyak and R. Tempo (2001)



# Las Vegas Randomized Algorithm

- Check if  $L(P) < 0$  for all vertex matrices
- This check needs to be performed (in the worst case)  $N = 2^{n^2}$  times, but the answer is always correct
- If we select the vertices in random order, it is a Las Vegas Randomized Algorithm
- Question: Do we really need to check all the vertex matrices ( $N = 2^{n^2}$ )?



- Answer: No, it suffices to check “only” a subset of  $2^{2n}$  vertex matrices<sup>[1]</sup>
- This is still exponential (the problem is *NP*-hard), but there is a major computational improvement for medium size problems

[1] T. Alamo, R. Tempo, D. Rodriguez and E.F. Camacho (2006)



# Diagonal Matrices and Extensions

- Transform the original problem from full square matrices  $A$  to diagonal matrices  $Z \in \mathbf{R}^{2n,2n}$
- It suffices to check the vertices of  $Z$
- Extensions for  $L_2$ -gain minimization and other related LMI problems



- Similar complexity reduction for checking interval matrix nonsingularity<sup>[1]</sup>

$$\det(A) \neq 0 \text{ for all } A \in A$$

needs  $2^{2n}$  vertex matrices

- Nonconservative solution of the full structured  $\mu$  problem requires  $2^{2n}$  real parameters<sup>[2]</sup>

[1] J. Rohn (1989)

[2] P. Young (1997)



# Las Vegas and Monte Carlo Algorithms

- We can *soften the goals*, and perform the check only for  $K \ll N$  vertices
- This would be a MCRA with a non zero probability of failure
- If we consider a nonlinear (e.g., polynomial) function  $A(\Delta)$  of  $\Delta$ , we cannot construct a LVRA
- In this case, checking vertex matrices does not suffice
- LVRA have a more limited applicability



IEIIT-CNR



# RandQuickSort for Switched Systems



# Sorting for Switched Systems<sup>[1]</sup>

- Consider Lyapunov equations

$$L(P) = (A^i)^T P + P A^i$$

for  $i = 1, 2, \dots, N$

- Given  $P$ , the objective is to sort these  $N$  Lyapunov equations<sup>[1]</sup>
- Motivations: Deciding which systems are more stable than others is useful information for the controller

[1] H. Ishii and R. Tempo (2006)



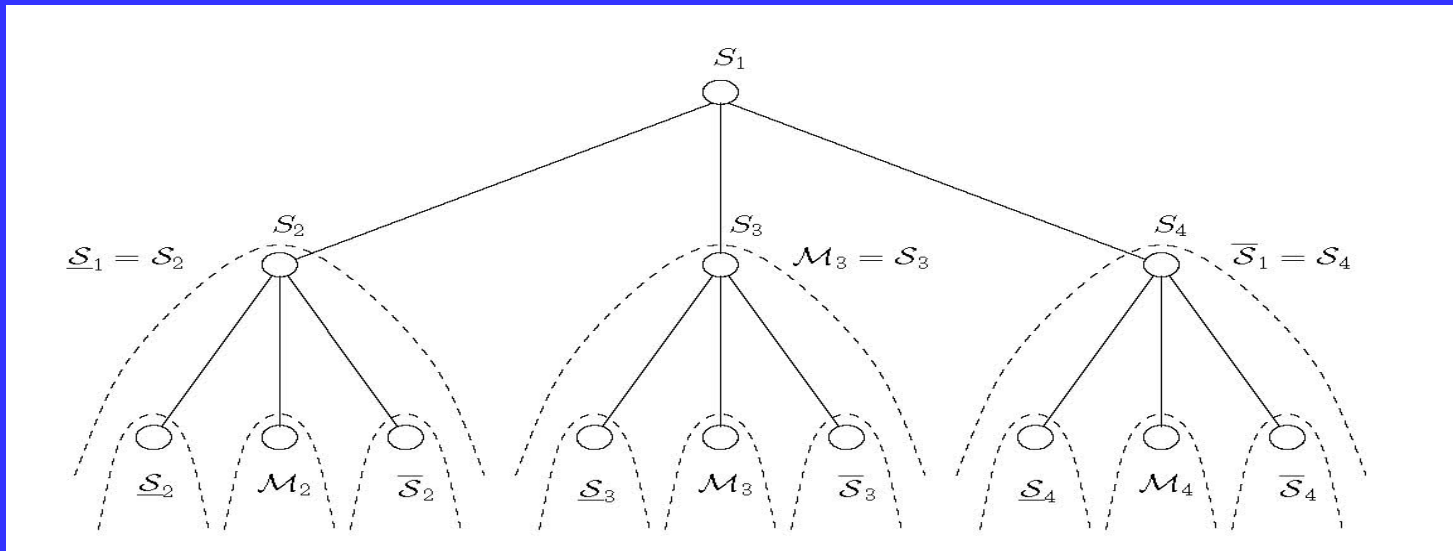
## Sorting for Switched Systems - 2

- The sorting operation should be performed quickly because we are switching between  $N = 2^{2n}$  systems
- Technical difficulty: Equations  $L(P)$  may be not completely sortable because of sign indefiniteness



# Las Vegas Randomized Algorithm - 1

- We propose a variation on RandQuickSort for sorting  $N = 2^{2n}$  Lyapunov equations
- Construction of a trinary tree (instead of binary) is necessary to detect if there are non-sortable matrices





# Las Vegas Randomized Algorithm - 2

- If the Lyapunov equations are completely sortable, then the expected running time is  $O(N \log (N))$
- If the Lyapunov equations are not completely sortable, then additional comparisons should be performed
- The worst case number of additional comparisons is  $N(N-1)/2$