Analyzing Simulation Results

Dr. John Mellor-Crummey

Department of Computer Science
Rice University

johnmc@cs.rice.edu

Topics for Today

Understand

- **Model verification**
- **Model validation**
- **Transient removal**
- **Terminating simulations**
- **Stopping criteria**

Model Goodness

Fidelity to modeled system

- **Measuring goodness**
  - —validation: are assumptions reasonable?
  - —verification: does model implement assumptions correctly?
- **Possible model states**

invalid, unverified	invalid, verified
valid, unverified	valid, verified

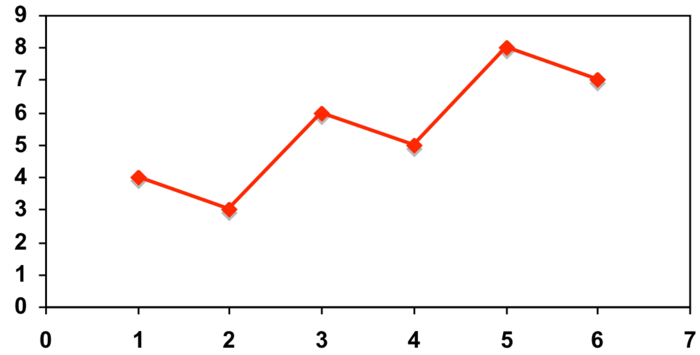
- — correctly implements bad assumptions
- — incorrectly implements good assumptions
- — correctly implements good assumptions

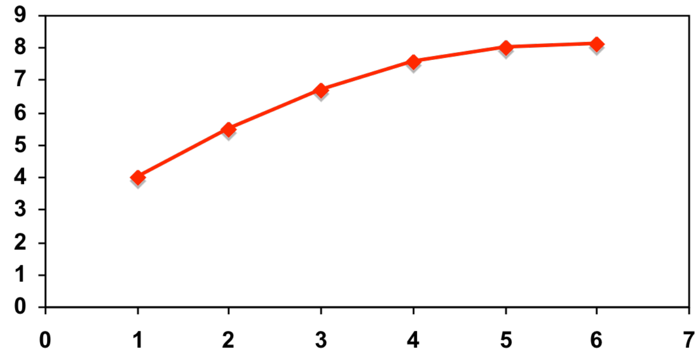
Model Verification Techniques I

- Strategies for avoiding bugs
  - —software engineering
    - – top-down design
      - layered (hierarchical) system structure
    - – modularity
      - well-defined interfaces
      - unit testing
  - —assertions to check invariants
    - – e.g., # packets received = # packets sent - # packets lost - # in flight
    - – entity accounting
  - —structured walk through
- Deterministic models
  - —run simulation with known distributions for random variates
- Simplified test cases with easily analyzed results
- Tracing: events, procedures, variables

Model Verification Techniques II

- **On-line graphical visualizations**
  - —**convey progress of simulation**
- **Continuity test**
  - —**test simulation with slightly different parameters**
  - —**investigate sudden changes in output**





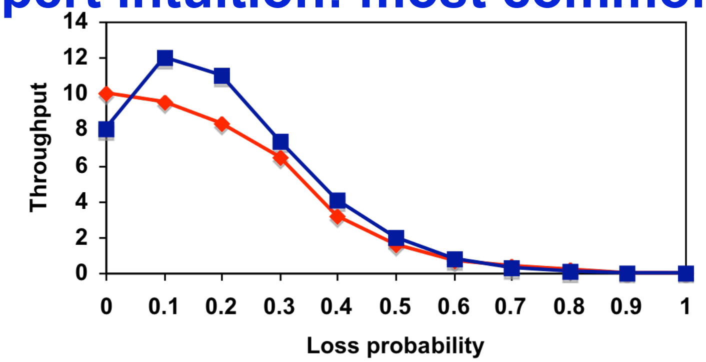
- **Degeneracy tests**
  - —**check model works for extreme cases**
  - —**e.g. networking: no routers, no router delays, no sources, …**

Model Verification Techniques III

- **Consistency tests**
  - —similar results for parameters that should have similar effects
    - – e.g. router simulation: 2 sources, rate r ~ 1 source, rate 2r
- **Seed independence**
  - —similar results for different seed values

Model Validation Techniques I

- **What to check**
  - —assumptions
  - —input parameter values and distributions
  - —output values and conclusions
- **How**
  - —expert intuition: most common and practical



$$\sum_{k=1}^{n} \frac{(o_i - e_i)^2}{e_i} < \chi^2_{[\alpha;k-1]}$$

  - —measurements of real system
    - – are simulation results and measurements distinguishable?
      - can use statistical tests, e.g. paired observations
    - – verify input distributions, e.g. chi-square test

Model Validation Techniques II

- **How (continued)**
  - —theoretical results, e.g. queueing model
    - simplifying assumptions helps
    - validate a few simple cases of theoretical model with simulation or intuition
    - use analytical model to predict complex cases

Caution: myth of a fully-validated model

—generally possible only to prove model not wrong for some cases

—more comparisons increase confidence, but prove nothing!

Transient Removal

- **Transient state: prefix of simulation before steady state**
- **Steady state performance is usually that of interest**
  - —e.g. cache performance after cache is “warm”
- **Goal: results exclude transient state before steady state**
- **Problem: identifying end of transient state**
- **Heuristic approaches for removing transient state**
  - —long runs
  - —proper initialization
  - —truncation
  - —initial data deletion
  - —moving average of independent replications
  - —batch means

Transient Removal: Long Runs

- **Long run = steady state results long enough to dominate effects of initial transients**
- **Disadvantages**
  - —wastes resources (computer time and real time)
  - —difficult to ensure length of run is "long enough"
- **Recommendation: avoid this method**

Transient Removal: Proper Initialization

- **Proper initialization = starting simulation in state close to expected steady state**
  - —e.g. start CPU scheduling simulation with non-empty job queue
  - —e.g. start WWW cache trace-driven simulation with most frequently referenced files in cache
- **Effect: reduces length of transient behavior**

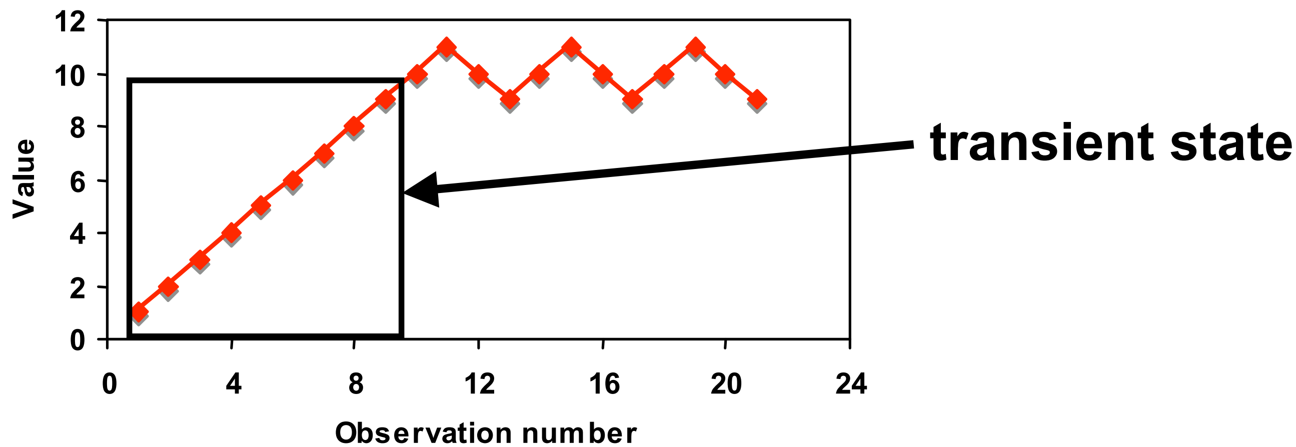
Transient Removal: Truncation

- **Assumption: variability of steady state < transient state**
- **Truncation method assumes variability = range**
- **Truncation algorithm**

> **input: n observations $\{x_1, x_2, \ldots, x_n\}$**
>
> **for k = 2, n**
>
> > **$min_k = \min(\{x_k, \ldots, x_n\})$**
> >
> > **$max_k = \max(\{x_k, \ldots, x_n\})$**
> >
> > **if $min_k \neq x_k$ && $max_k \neq x_k$ break**
>
> **post condition: if $k \neq n$ then $k - 1$ = length of transient state**

is there a flaw?

can we fix it?



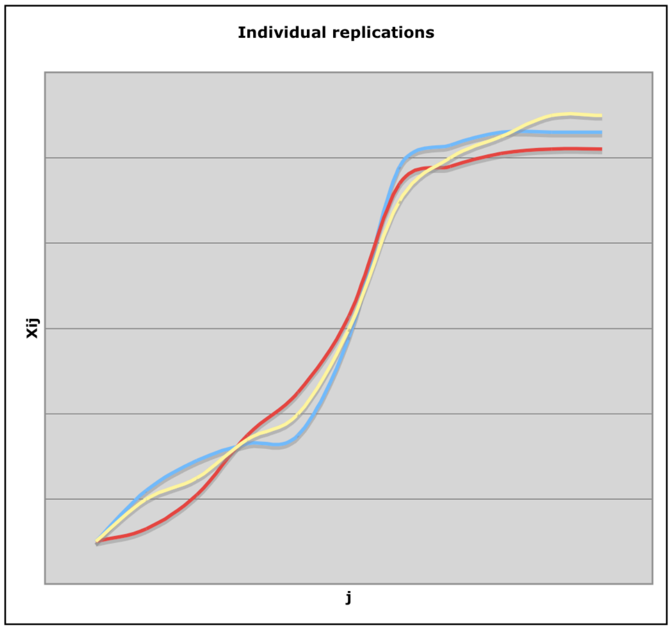
Terminating Simulations: Initial Data Deletion

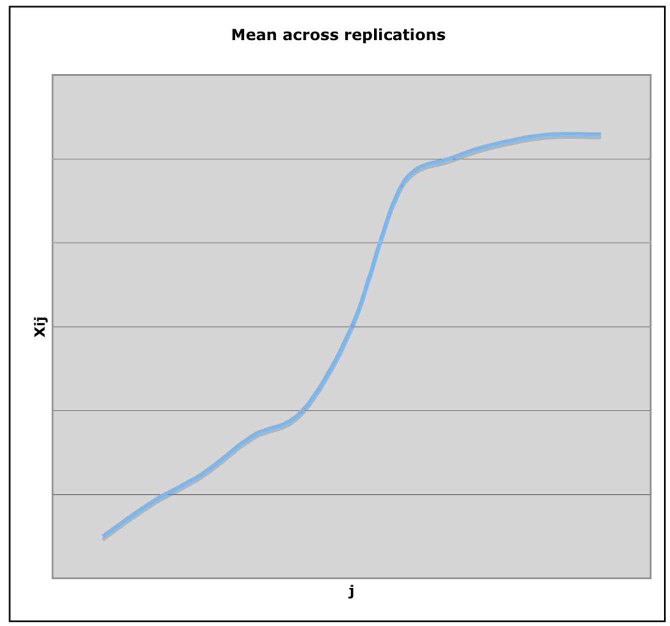
- **Conceptual idea**
  - —compute average after some of initial observations omitted
  - —during steady state average does not change much as additional observations are deleted
- **Problem**
  - —randomness in observations causes avg to change even in SS
- **Solution**
  - —average across several replications
    - – replication: same parameter values; only seed values differ
    - – rationale: smooths trajectory
- **Input: m replications, each of length n**

Initial Data Deletion: First Steps

- **Compute mean trajectory by averaging across replications**

$$\bar{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij}, \qquad j = 1,2,\ldots,n$$





- **Compute overall mean**

$$\bar{\bar{x}} = \frac{1}{n}\sum_{j=1}^{n} \bar{x}_j$$

Initial Data Deletion: Remaining Steps

for k = 1, n - 1

- **assume transient state is of length k**
- **delete first k observations from mean trajectory**
- **compute overall mean from remaining n - k values**

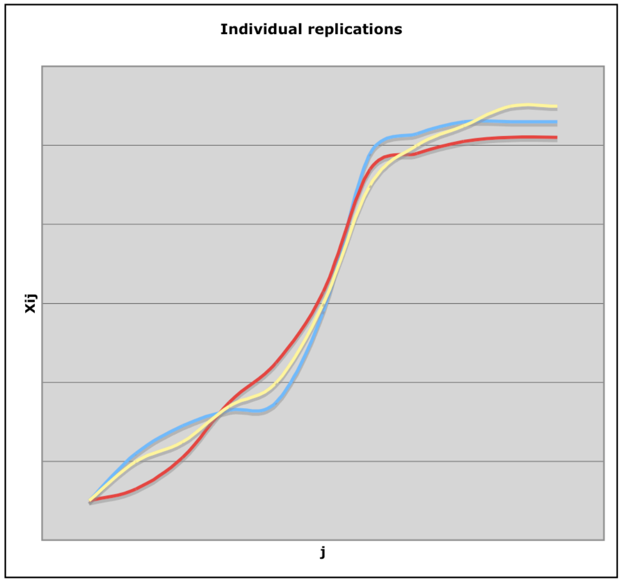
$$\bar{\bar{x}} = \frac{1}{n-k} \sum_{j=k+1}^{n} \bar{x}_j$$

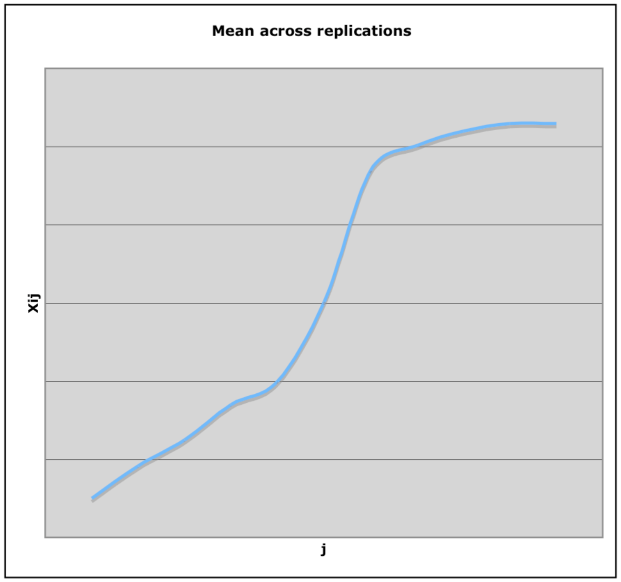
- **compute relative change in overall mean**

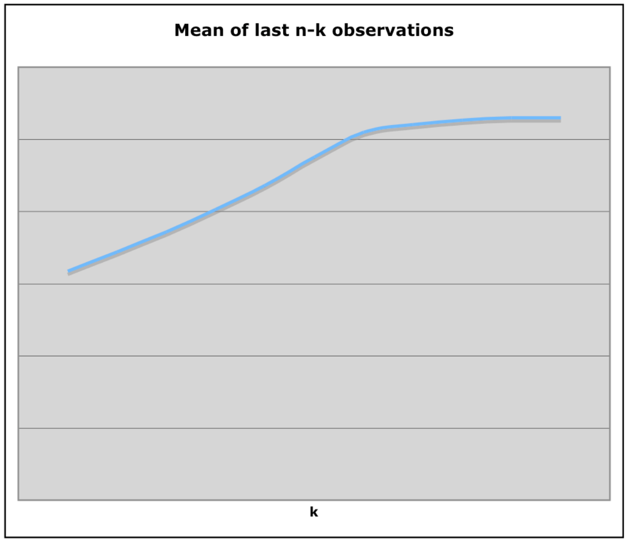
$$\text{Relative change } = \frac{\bar{\bar{x}}_k - \bar{\bar{x}}}{\bar{\bar{x}}}$$

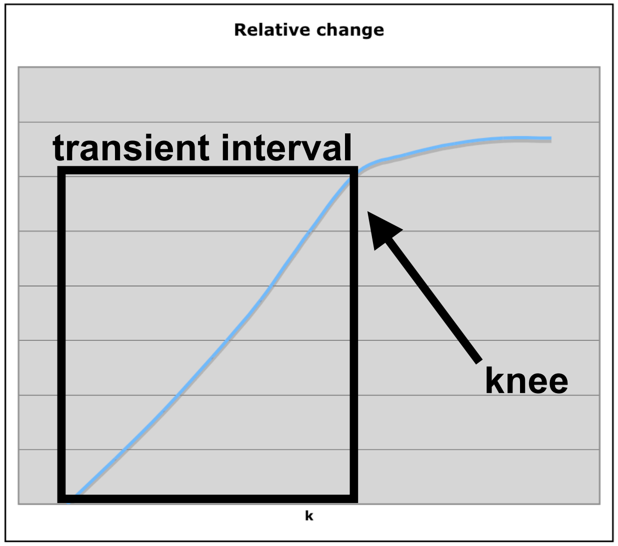
find knee in a curve showing the relative change in overall mean

Initial Deletion: Putting it all Together









Moving Average of Independent Replications

- **Compute mean trajectory by averaging across replications**

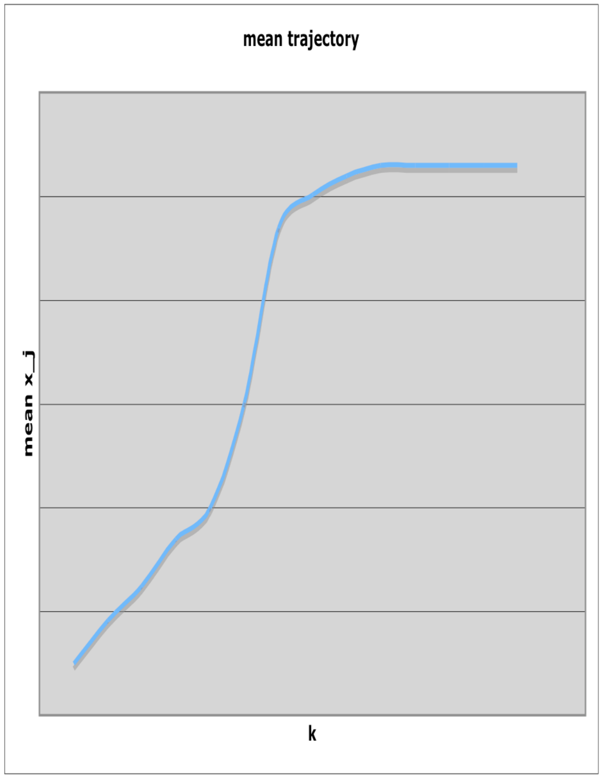
$$\bar{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij}, \qquad j = 1,2,\ldots,n$$

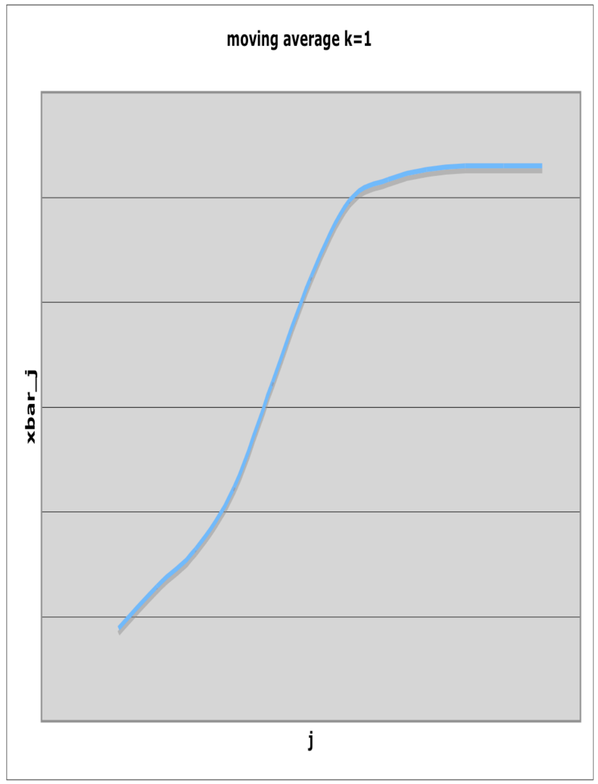
- **for k = 1 to n**
  - **—plot trajectory of moving average of successive 2k+1 values**

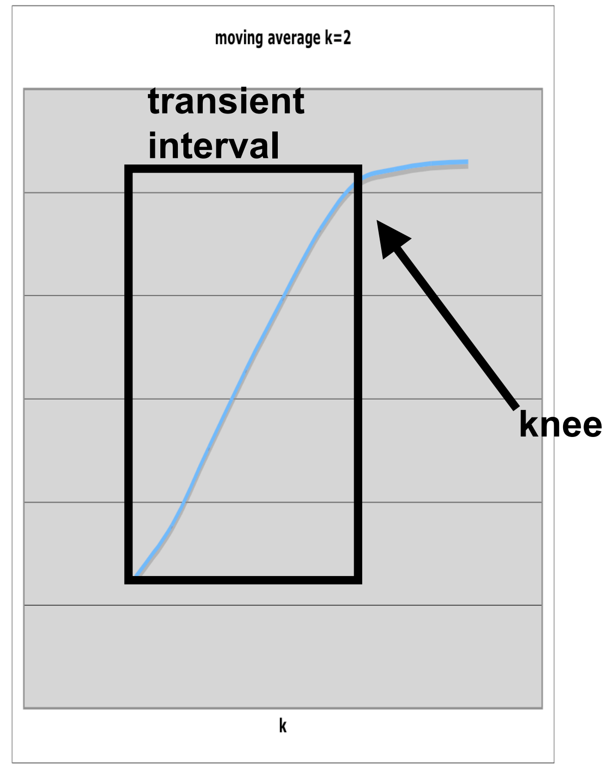
$$\bar{x}_j = \frac{1}{2k+1}\sum_{l=-k}^{k} \bar{x}_{j+l}, \qquad j = k+1, k+2, \ldots, n-k$$

  - **—if trajectory is “sufficiently smooth”, break**
- **find the knee in the curve.**
- **j at the knee gives the length of the transient phase**

Moving Average of Independent Replications







Batch Means

- **Run a very long simulation**
- **Afterward, divide it into several parts of equal duration**
- **Each part is a <u>batch</u>**
- **Batch mean = mean of observations in each batch**

Input: m batches of floor(M/n)

Algorithm

- **for each batch, compute a batch mean** $$\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij}, \qquad i = 1,2,\ldots,m$$
- **compute the overall mean across all batches**

$$\bar{\bar{x}} = \frac{1}{m}\sum_{i=1}^{m} \bar{x}_i$$

- **compute variance of batch means** $$\mathrm{Var}(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m}(\bar{x}_i - \bar{\bar{x}})^2$$
- **repeat for increasing n=3,4,5,…**
- **plot variance as function of batch size**
- **length of transient interval is length at which variance starts decreasing**

Terminating Simulations

- Most simulations reach a steady state, but some don't
  - —Example
    - – network traffic consists of xfer of small files (1-3 packets each)
    - – steady state simulations using large files give results of no interest to typical user
- Necessary to study such systems in transient state
- Terminating simulations: ones that don't reach steady state
- Other terminating simulations
  - —one that shuts down at 10PM every day
  - —systems with parameters that change over time
- Terminating simulations don't require transient removal
- Final conditions
  - —may not be typical. can remove like "initial conditions"

Stopping Criteria: Variance Estimation

- **Choosing proper simulation length is important**
  - —too short: results highly variable
  - —too long: wastes time and resources
- **Simulation should be run until confidence interval for mean response narrows to desired width**

$$\bar{x} \pm z_{1-\alpha/2}\sqrt{\mathrm{Var}(\bar{x})}$$

- **Problem: how to estimate the variance**
  - —observations in simulation are not independent
  - —e.g. waiting time for job I+1 depends on time for job I

Variance Estimation: Independent Replications

- **Replications obtained by repeating simulation with different seed**
- **Method assumption: means of independent replications are independent even though observations within a replication are correlated**
- **Input: m replications of size n + n_o (n_o is size transient phase)**
- **Algorithm**
  - —compute mean for each replication, excluding transient phase
  - —compute overall mean for all replications $\bar{\bar{x}}$
  - —calculate variance of replicate means

$$\mathrm{Var}(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m}(\bar{x}_i - \bar{\bar{x}})^2$$

  - —confidence interval is then

$\bar{\bar{x}} \pm z_{1-\alpha/2}\sqrt{\mathrm{Var}(\bar{x})}$ Note: conf interval inversely proportional to $\sqrt{\mathrm{mn}}$

waste less by increasing n rather than m

Variance Estimation: Batch Means

- **Run long simulation; remove transient & divide into batches**
- **Algorithm**
  - —compute mean for each batch
  - —compute overall mean for all batches $\bar{\bar{x}}$
  - —calculate variance of batch means

$$\mathrm{Var}(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m}(\bar{x}_i - \bar{\bar{x}})^2$$

  - —confidence interval is then

$$\bar{\bar{x}} \pm z_{1-\alpha/2}\sqrt{\frac{\mathrm{Var}(\bar{x})}{m}}$$

- **Notes**
  - —increase confidence by increasing # batches (m) or batch size (n)
  - —batch size must be large so batch means have little correlation
  - —finding correct n
    - – increase batch size until autocovariance between batch means is small w.r.t. variance
    - – autocovariance = $\mathrm{Cov}(\bar{x}_i, \bar{x}_{i+1}) = \frac{1}{m-2}\sum_{i=1}^{m}(\bar{x}_i - \bar{\bar{x}})(\bar{x}_{i+1} - \bar{\bar{x}})$

Variance Estimation: Batch Means

- **Run long simulation; remove transient & divide into batches**
- **Algorithm**
  - —compute mean for each batch
  - —compute overall mean for all batches $\bar{\bar{x}}$
  - —calculate variance of batch means

$$\mathrm{Var}(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m}(\bar{x}_i - \bar{\bar{x}})^2$$

  - —confidence interval is then

$$\bar{\bar{x}} \pm z_{1-\alpha/2}\sqrt{\frac{\mathrm{Var}(\bar{x})}{m}}$$

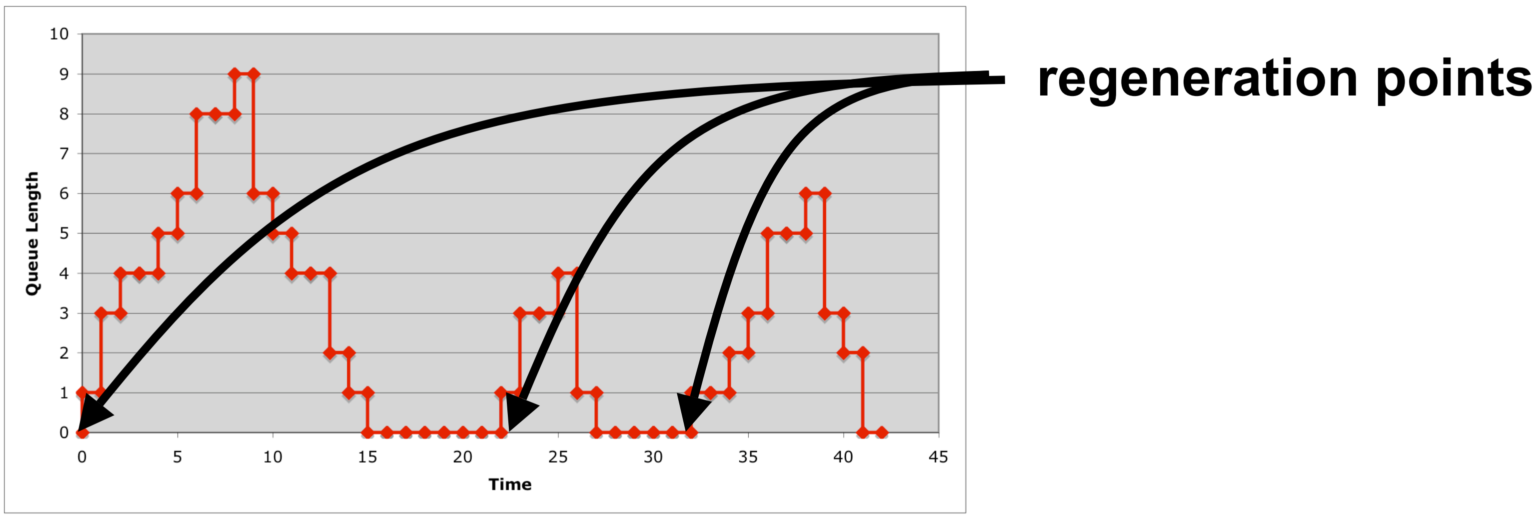
- **Notes**
  - —increase confidence by increasing # batches (m) or batch size (n)
  - —batch size must be large so batch means have little correlation
  - —finding correct n
    - – increase batch size until autocovariance between batch means is small w.r.t. variance
    - – autocovariance =

$$\mathrm{Cov}(\bar{x}_i, \bar{x}_{i+1}) = \frac{1}{m-2}\sum_{i=1}^{m}(\bar{x}_i - \bar{\bar{x}})(\bar{x}_{i+1} - \bar{\bar{x}})$$

Variance Estimation: Method of Regeneration

- **Consider CPU scheduling algorithm**
  - —every time queue is empty, it is like a fresh start for the simulation
    - – trajectory in interval after empty state does not depend on prior trajectory
  - —this phenomenon called regeneration
- **Regeneration point:**

  when a simulation enters an independent phase



- **Regenerative period: duration between 2 regeneration points**
- **Not all systems are regenerative**
  - —system with many queues regenerates only when all are empty

Variance Estimation: Method of Regeneration

m cycles of size $n_1, n_2, \ldots, n_m$

- **Algorithm**
  - —compute cycle sums $y_i = \sum_{j=1}^{n_i} x_{ij}$
  - —compute the overall mean $\bar{\bar{x}} = \sum y_i / \sum n_i$
  - —calculate difference between expected and observed cycle sums

$$w_i = y_i - n_i \bar{\bar{x}}, \qquad i = 1,2,\ldots,m \quad (w_i \text{ IID mean } 0)$$

  - —calculate variance of differences $\mathrm{Var}(w) = \frac{1}{m-1} \sum_{i=1}^{m} w_i^2$
  - —compute the mean cycle length $\bar{n} = \frac{1}{m} \sum_{i=1}^{m} n_i$
  - —confidence interval for mean response

$$\bar{\bar{x}} \pm z_{1-\alpha/2} \frac{1}{\bar{\mathrm{n}}} \sqrt{\frac{\mathrm{Var}(w)}{m}}$$