Introduction to simulation

- The Nature of Simulation
- Systems, Models, and Simulation

- Discrete-Event Simulation
- Advantages, Disadvantages, and Pitfalls of Simulation

- Exercise: design a simple simulator for RM scheduling
- Project: design a simulator for a Hierarchical Scheduling System

The Nature of Simulation

- *Simulation*: imitate the operations of *system*, usually using a computer
  - To study the system, often make assumptions/approximations, both logical and mathematical, about how it works
  - These assumptions form a *model* of the system
  - If model structure is simple enough, could use mathematical methods to get exact information on questions of interest — *analytical solution*

The Nature of Simulation, cont.

- But most complex systems require models that are also complex (to be valid)
  - Must be studied via simulation — evaluate model numerically and collect data to estimate model characteristics
- Example: Manufacturing company considering extending its plant
  - Build it and see if it works out?
  - Simulate current, expanded operations — could also investigate many other issues along the way, quickly and cheaply

The Nature of Simulation, cont.

- Some (not all) application areas
  - Systems engineering: supporting design decisions
  - Designing and analyzing manufacturing systems
  - Determining hardware requirements or protocols for communications networks
  - Determining hardware and software requirements for a computer system
  - Designing and operating transportation systems such as airports, freeways, ports, and subways
  - Evaluating designs for service organizations such as call centers, fast-food restaurants, hospitals, and post offices
  - Reengineering of business processes
  - Determining ordering policies for an inventory system
  - Analyzing financial or economic systems

The Nature of Simulation, cont.

- Impediments to acceptance, use of simulation
  - Models of large systems are usually very complex
    - But now we have better modeling software … more general, flexible, and still (relatively) easy to use
  - Can consume a lot of computer time
    - But now we have faster, bigger, cheaper hardware to allow for much better studies than just a few years ago … this trend will continue
    - However, simulation will also continue to push the envelope on computing power; we ask more and more of our simulation models
  - Impression that simulation is "just programming"
    - There's a lot more to a simulation study than just "coding" a model in some software and running it to get "the answer"
    - Need careful design and analysis of simulation models

Systems, Models and Simulation

- *System*: A collection of entities that act and interact together toward some end (Schmidt and Taylor, 1970)
  - In practice, depends on objectives of study
  - What are the boundaries of the system?
  - Judgment call: level of detail (e.g., what is an entity?)
  - Usually assume a time element – *dynamic* system

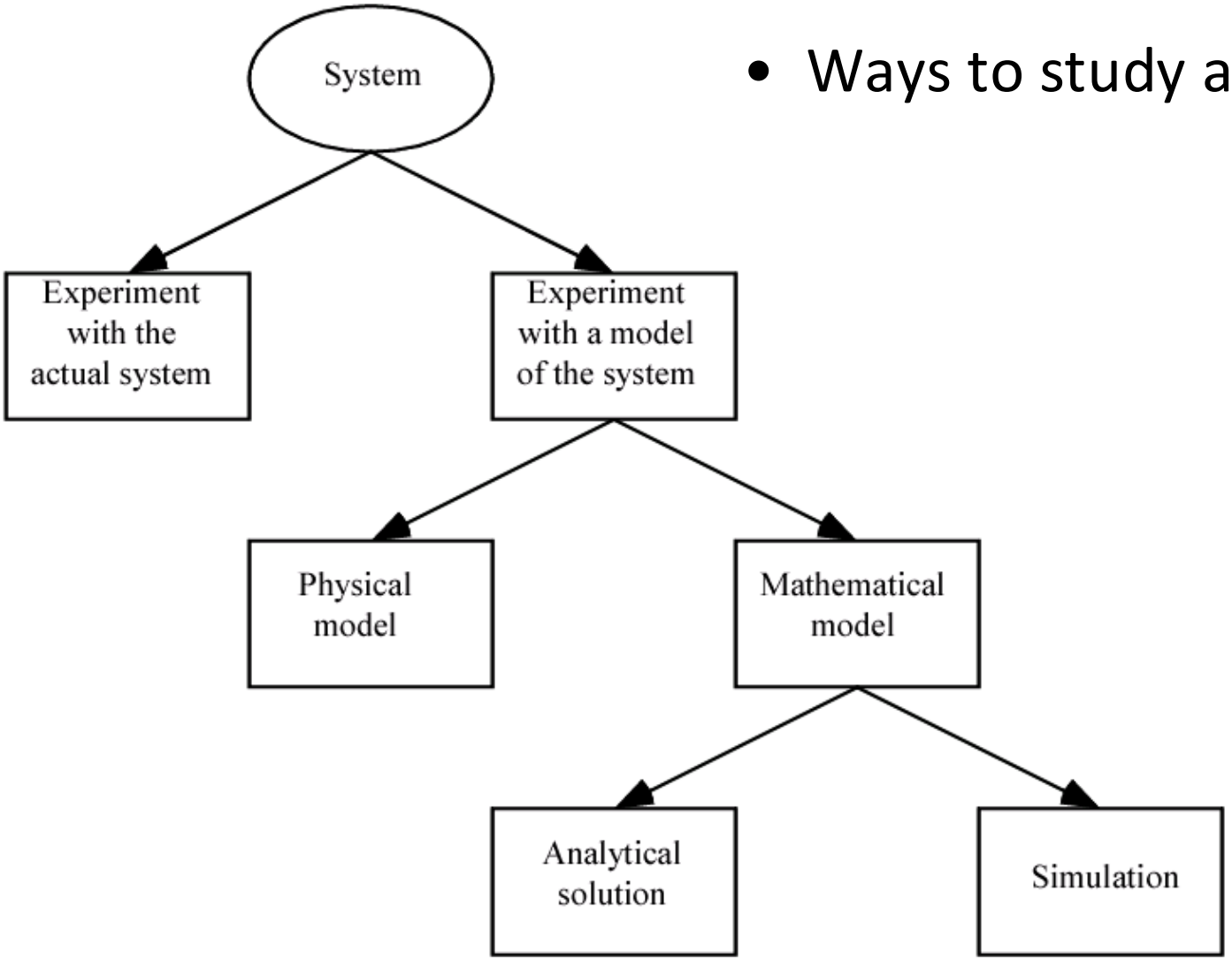
Systems, Models and Simulation, cont.

- *State* of a system: Collection of variables and their values necessary to describe the system at that time
  - Might depend on desired objectives, output performance measures
  - Bank model: Could include number of busy tellers, time of arrival of each customer, etc.

Systems, Models, and Simulation, cont.

- Types of systems
  - *Discrete*
    - State variables change instantaneously at separated points in time
    - Bank model: State changes occur only when a customer arrives or departs
  - *Continuous*
    - State variables change continuously as a function of time
    - Airplane flight: State variables like position, velocity change continuously
- Many systems are partly discrete, partly continuous

Systems, Models, and Simulation, cont.



- Ways to study a system

Systems, Models, and Simulation, cont.

- Classification of simulation models
  - *Static* vs. *dynamic*
  - *Deterministic* vs. *stochastic*
  - *Continuous* vs. *discrete*
- Most operational models are dynamic, stochastic, and discrete – will be called *discrete-event simulation models*

Discrete-even simulation

- *Discrete-event simulation*: Modeling of a system as it evolves over time by a representation where the state variables change instantaneously at separated points in time
  - State can change at only a *countable* number of points in time
  - These points in time are when *events* occur

- *Event*: Instantaneous occurrence that <u>may</u> change the state of the system

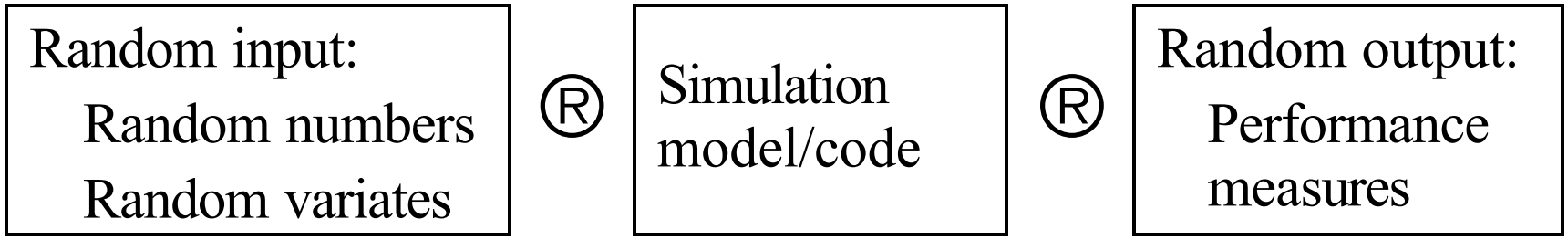
- Can in principle be done by hand; usually done on a computer

Simulation input

- Inappropriate input distribution(s) can lead to incorrect output, bad decisions

Use	**Pros**	**Cons**
Trace-driven Use actual data values to drive simulation	Valid *vis à vis* real world Direct	Not generalizable
Empirical distribution Use data values to define a "connect-the-dots" distribution (several specific ways)	Fairly valid Simple Fairly direct	May limit range of generated variates (depending on form)
Fitted "standard" distribution Use data to fit a classical distribution (exponential, uniform, Poisson, etc.)	Generalizable—fills in "holes" in data	May not be valid May be difficult

Simulation output



- Performance measures:
  - Interpreting simulation output: statistical analysis of output data
  - Observations from the probability distribution of output

- Outputs
  - *Standard reports* for the estimated performance measures
  - *Customized reports, Histograms, Time plots, Databases, Correlation plots*
  - *Export individual model output observations* to other software packages

Components and Organization of a Discrete-Event Simulation Model

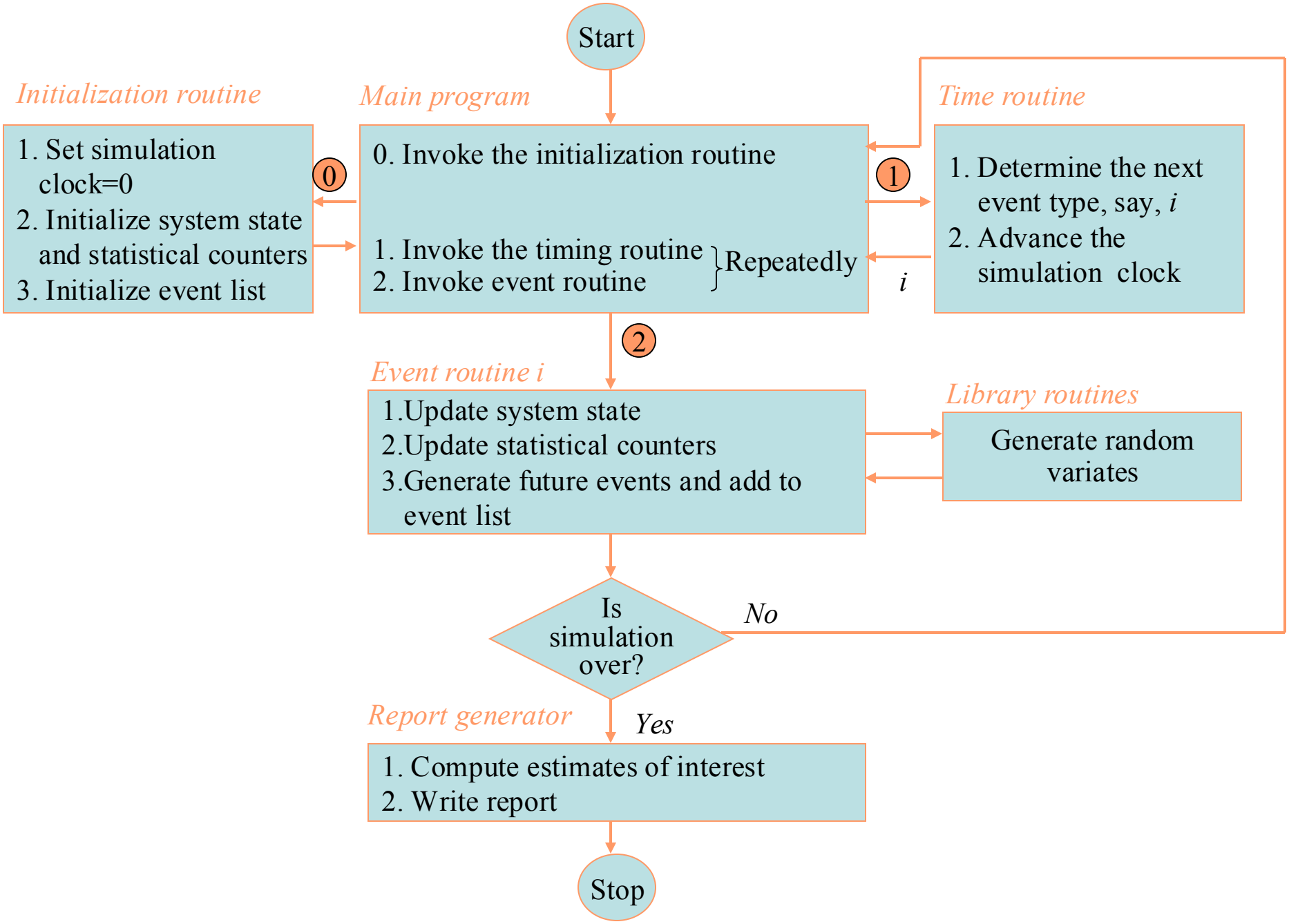
- Each simulation model must be customized to target system
- But there are several common components, general organization
  - *Systems state*: The collection of state variables necessary to describe the system at a particular time
  - *Simulation clock*: A variable giving the current value of simulated time
  - *Event list*: A list containing the next time when each type of event will occur
  - *Statistical counters*: Variables used for storing statistical information about system performance

Components and Organization of a Discrete-Event Simulation Model, cont.

- *Initialization routine*: A subprogram to initialize the simulation model at time 0
- *Timing routine*: A subprogram that determines the next event from the event list and then advances the simulation clock to the time when that event is to occur
- *Event routine*: A subprogram that updates the system state when a particular type of event occurs (there is one event routine for each event type)
- *Library routines*: A set of subprograms used to generate random observations from probability distributions that were determined as part of the simulation model

Components and Organization of a Discrete-Event Simulation Model, cont.

- *Report generator*: A subprogram that computes estimates (from the statistical counters) of the desired measures of performance and produces a report when the simulation ends
- *Main program*: A subprogram that invokes the timing routine to determine the next event and then transfers control to the corresponding event routine to update the system state appropriately. The main program may also check for termination and invoke the report generator when the simulation is over.


Start
Initialization routine
Main program
Time routine
1. Set simulation clock=0
2. Initialize system state and statistical counters
3. Initialize event list
0
0. Invoke the initialization routine
1. Invoke the timing routine
2. Invoke event routine
Repeatedly
1
i
1. Determine the next event type, say, i
2. Advance the simulation clock
2
Event routine i
Library routines
1.Update system state
2.Update statistical counters
3.Generate future events and add to event list
Generate random variates
Is simulation over?
No
Yes
Report generator
1. Compute estimates of interest
2. Write report
Stop

Time-Advance Mechanisms

- *Simulation clock*: Variable that keeps the current value of (simulated) time in the model
  - Must decide on, be consistent about, time units
  - Usually no relation between simulated time and (real) time needed to run a model on a computer

- Two approaches for time advance
  - *Next-event time advance* (usually used)
  - *Fixed-increment time advance* (seldom used)

Time-Advance Mechanisms, cont.

- More on next-event time advance
  - Initialize simulation clock to 0
  - Determine times of occurrence of future events – *event list*
  - Clock advances to next (most imminent) event, which is executed
    - Event execution may involve updating event list
  - Continue until stopping rule is satisfied
  - Clock "jumps" from one event time to the next, and doesn't "exist" for times between successive events … periods of inactivity are ignored

Advantages, Disadvantages, Pitfalls

- Advantages
  - Simulation allows great flexibility in modeling complex systems, so simulation models can be highly valid
  - Easy to compare alternatives
  - Control experimental conditions
  - Can study system with a very long time frame
- Disadvantages
  - Stochastic simulations produce only estimates – with noise
  - Simulation models can be expensive to develop
  - Simulations usually produce large volumes of output – need to summarize, statistically analyze appropriately
- Pitfalls
  - Failure to identify objectives clearly up front
  - Inappropriate level of detail (both ways)
  - Inadequate design and analysis of simulation experiments
  - Inadequate education, training

Simulating RM

- Simulate by hand the following tasks running on a single processor using Rate Monotonic scheduling

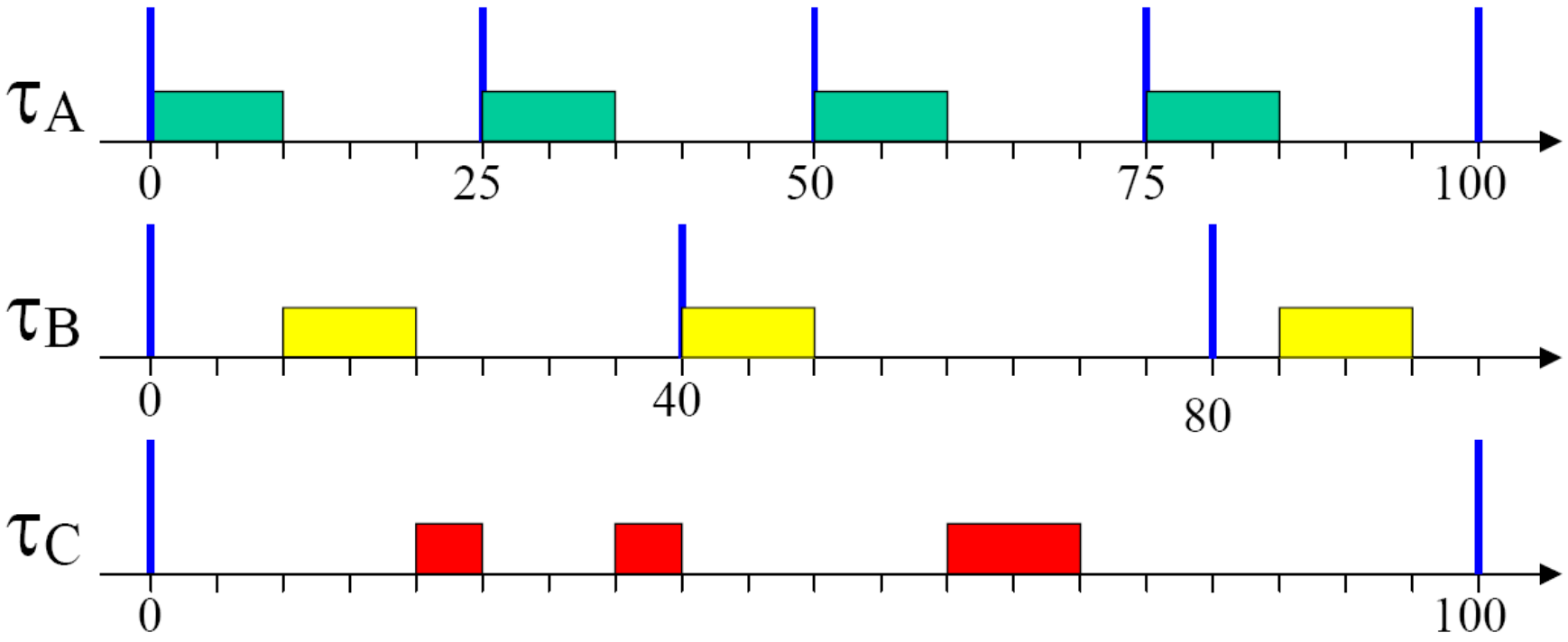
Task	WCET	Period = Deadline
A	10	25
B	10	40
C	20	100

Assumptions

- Single processor
- Strictly periodic tasks
- Deadlines are equal to the periods
- All tasks are independent
- All tasks are released as soon as they arrive
- All tasks arrive at time zero

- Execution time is always equal to the WCET

Example simulation run



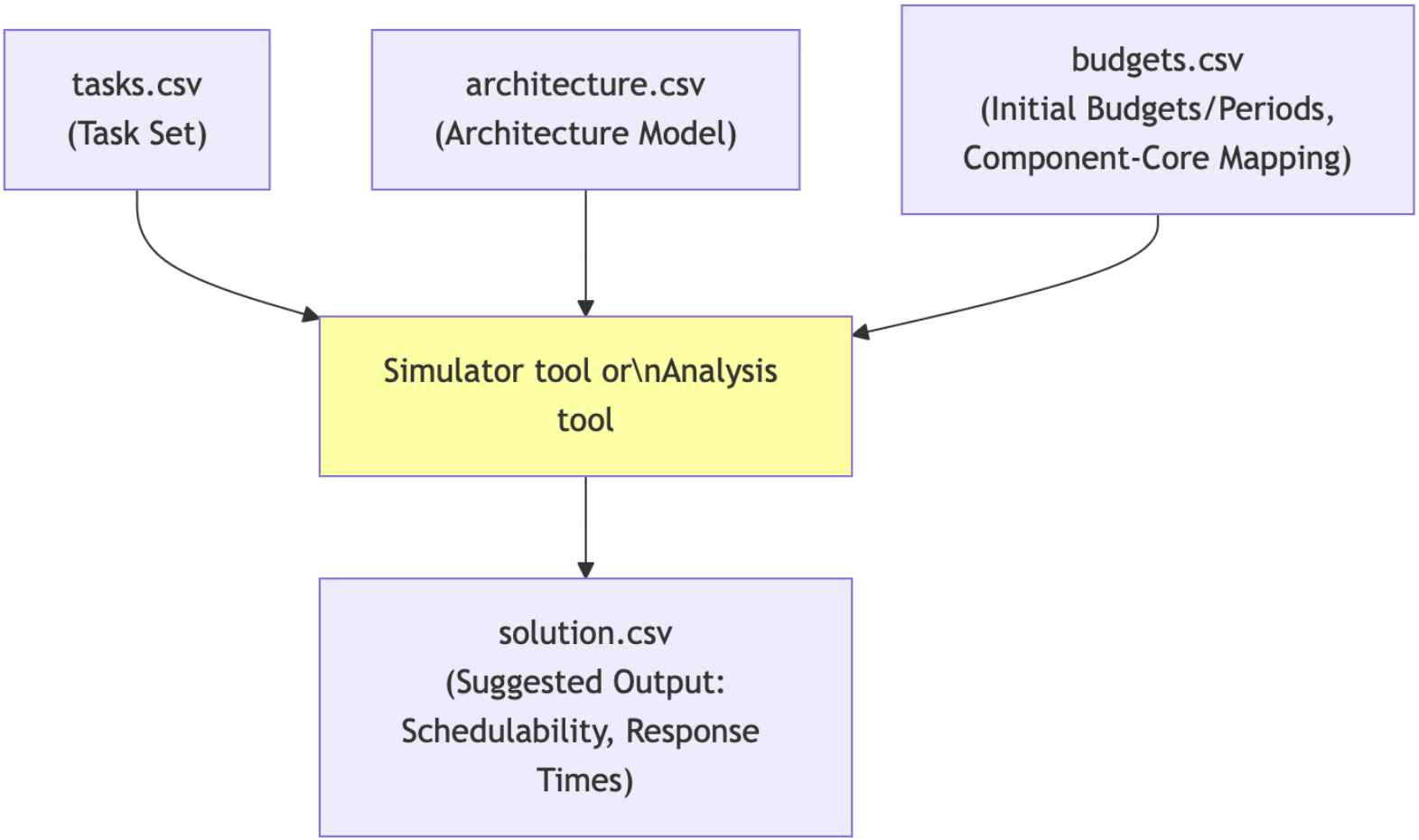
RM simulation

- What is the system state?
  - What are the entities?
  - What are their attributes?
- Event routine(s)
  - What are the events?
  - How do you organize your event list?
  - How do you update the system state when an event occurs?
- Timing routine:
  - How do you determine the next event?
  - How do you advance time?
- Report generator:
  - What information do you collect?
  - How do you report it?

Project Simulation: Overview

- The project requires you to develop a **simulator** for a Hierarchical Scheduling System.
- The simulator should model the execution of tasks on a **multicore platform** with **hierarchical scheduling**.
- You will use **CSV input files** to configure the system and tasks.
- The simulator should produce **output metrics** to analyze system behavior and schedulability.

Input Files Flow Diagram



Simulation Input Files (CSV Format)

Your simulator tool (and analysis tool) will read configuration from three CSV files:

1. **`tasks.csv`**: Task Set Definition
2. **`architecture.csv`**: Hardware Platform Model
3. **`budgets.csv`**: Initial Component Budgets & Core Mapping

These CSV files provide all the necessary information to set up your simulation environment.

1. `tasks.csv`: Task Set Definition

Defines the tasks in the system and their properties.

Columns:

- `task_name`: Unique name of the task (e.g., “Task_1”, “CameraTask”)
- `wcet`: Worst-Case Execution Time (at nominal core speed) (e.g., 10.0, 5.0)
- `period`: Task period (e.g., 20.0, 100.0)
- `component_id`: Component to which the task belongs (e.g., “Component_1”, “ADAS_Component”)
- `scheduler`: Scheduler of the component (“EDF” or “RM”)
- `priority`: Task priority (for RM tasks; leave empty for EDF)

See `tasks.csv` example file provided.

2. `architecture.csv`: Hardware Model

Describes the multicore hardware platform.

Columns:

- `core_id`: Unique ID of the core (e.g., “Core_1”, “CPU0”)
- `speed_factor`: Core speed relative to nominal (e.g., 1.0, 0.8, 1.2)

Important:

- Task `wcet` from `tasks.csv` is at *nominal speed* (`speed_factor = 1.0`).
- In your simulator, adjust task WCET based on the `speed_factor` of the core it runs on: `adjusted_wcet = wcet / speed_factor`

See `architecture.csv` example file provided.

3. `budgets.csv`: Component Budgets & Core Mapping

Defines initial resource budgets (Theta, Pi) for each component and assigns components to cores.

Columns:

- `component_id`: Component ID (matches `tasks.csv`)
- `budget`: Initial budget (Theta) for the component (e.g., 7.0, 10.0)
- `period`: Initial period (Pi) for the component (e.g., 10.0, 20.0)
- `core_id`: Core ID where the component is assigned (matches `architecture.csv`)

Important:

- These budgets are *initial values* derived from PRM analysis.
- You may need to optimize these budgets and convert them to BDR parameters (Alpha, Delta) for analysis and simulation.

See `budgets.csv` example file provided.

Simulator Output: `solution.csv`

A suggested CSV format for your simulator (and analysis tool) output:

Columns:

- `task_name`: Task name
- `component_id`: Component ID
- `task_schedulable`: Task schedulability (1=Yes, 0=No) - from analysis & sim
- `avg_response_time`: Average response time (from simulator)
- `max_response_time`: Maximum response time (from simulator)
- `component_schedulable`: Component schedulability (all tasks schedulable?)

See `solution.csv` example file provided.

Simulator Implementation: Key Aspects

- **Discrete-Event Simulation:** Model system as a sequence of events (task arrival, task completion, resource allocation, etc.).
- **Hierarchical Scheduling:** Implement the hierarchical scheduling approach with components and cores.
- **Scheduling Policies:** Support EDF and RM scheduling within components.
- **BDR Resource Model:** Model resource supply using the Bounded Delay Resource (BDR) model with (Alpha, Delta) parameters.
- **Performance Metrics:** Collect and report relevant metrics (response times, schedulability, utilization).

Flexibility and Customization

- **CSV Formats are Suggestions:** You can modify the CSV input and output formats as needed for your project.
- **Output Metrics:** Extend the output to include other relevant metrics for your analysis (e.g., BDR parameters, core utilization, preemptions).
- **Focus on Project Goals:** Ensure your simulator and output provide the information necessary to address the project requirements and analysis tasks.

Document any changes you make to the input/output formats in your project report.

Hierarchical Schedulability Analysis

02225 Distributed Real-Time Systems Project

Paul Pop and Silviu Craciunas

In the project you will use the compositional analysis techniques from the chapter "Hierarchical Scheduling" from *Handbook of Real-Time Computing*, 2022 (available in `DTU Learn/Content/Project`), Section 3 "Compositional Framework for HSS"

Outline

Hierarchical Scheduling Systems (HSS)

- Real-time systems are becoming increasingly complex.
- **Hierarchical Scheduling Systems (HSS)** provide a way to manage this complexity.
- Key Idea: Decompose the system into *components*.
  - Components are arranged in a **hierarchy**.
  - Each component has its own scheduler (e.g., EDF, RM).
  - Components share resources (e.g., CPU time) provided by their **parent** component.
  - Child components expose resource **requirements** to their parent through an **interface**. The parent component **allocates** resources to its children.

HSS Example

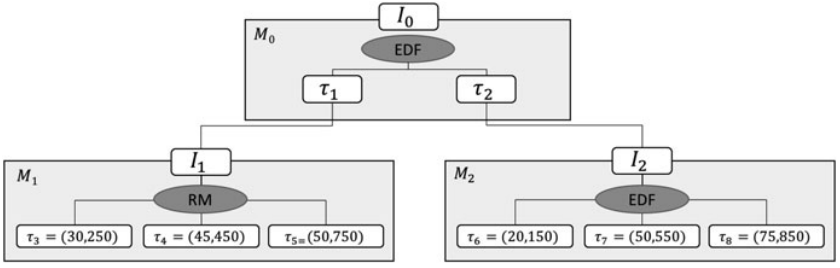


Figure: Example Hierarchical Scheduling System (HSS)

System-Level Schedulability (Top-Level)

- Components' interface consists of budgets (Q) and periods (P), and they act like periodic tasks at the system level.
- We need to check if the **entire system** is schedulable. This means verifying that the components themselves (treated as tasks) can be scheduled on the cores.
- **Simple Utilization-Based Tests:** We can use standard utilization-based schedulability tests for the top-level scheduler:
  - $\sum_{i=1}^{n} \frac{Q_i}{P_i} \leq 1$ for EDF.
  - $\sum_{i=1}^{n} \frac{Q_i}{P_i} \leq n(2^{1/n} - 1)$ for RM.
- You would perform these utilization tests (EDF or RM, depending on the core's scheduler) for the components assigned to each core independently.

Implementing the Analysis for each Component: Key Functions

- You'll need to implement the following functions for your analysis:
  1. **Supply Bound Function (SBF) for BDR:** Equation (6).
  2. **Half-Half Algorithm**: Theorem 3. For converting a PRM interface (Q, P) into an equivalent BDR interface (α, Δ).
  3. **Demand Bound Function (DBF) for RM:** Equation (4).
  4. **Demand Bound Function (DBF) for EDF:** Equations (2) and (3).
- These equation and theorem numbers refer to the numbers in the "Hierarchical Scheduling" Handbook Chapter.

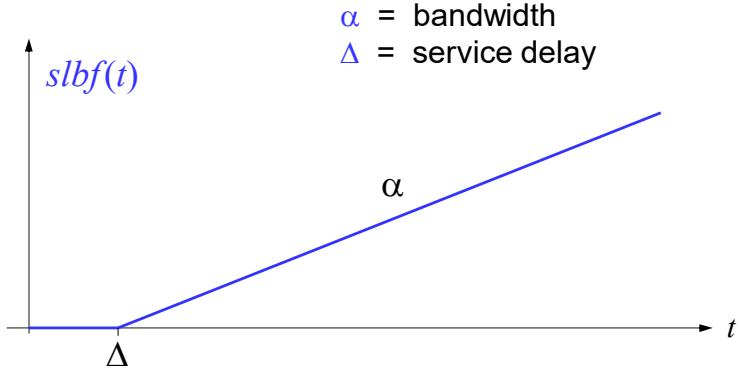
Bounded Delay Resource (BDR) Model

- A way to model resource allocation in hierarchical scheduling systems.
- Characterized by two parameters:
  - α: **Resource availability factor**. The fraction of processing time guaranteed.
  - Δ: **Maximum delay in resource allocation**. Bounds the worst-case delay in getting the allocated resource.
- Provides *temporal isolation* between components.
- A component behaves as if it has a dedicated processor with a slower speed (α) and some initial delay (Δ).

Supply Bound Function (SBF) for BDR

- The **Supply Bound Function (SBF)** gives the *minimum* resource supply provided by a BDR resource $R = (\alpha, \Delta)$ within a time interval t.
- Equation:

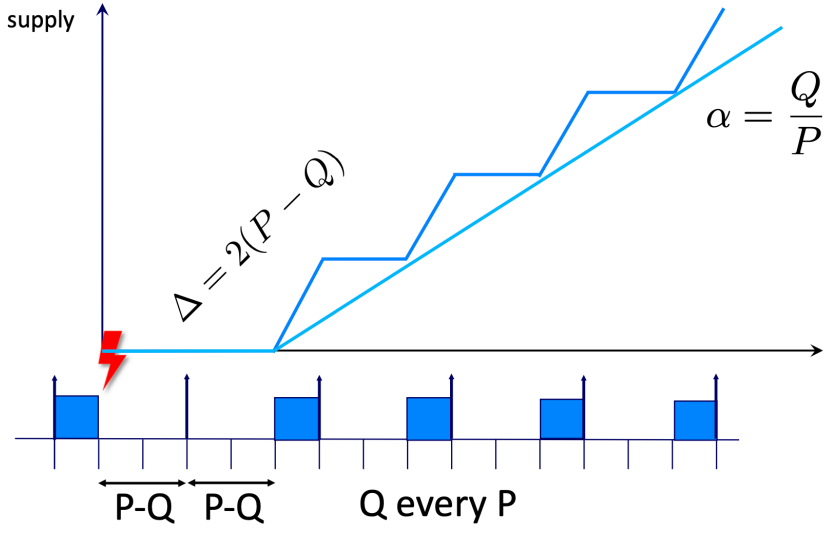
$$\text{sbf}_{\text{BDR}}(R, t) = \begin{cases} \alpha(t - \Delta) & \text{if } t \geq \Delta \\ 0 & \text{otherwise} \end{cases}$$



The Half-Half Algorithm

- We start with the budget and period (Q, P):
  - Q: Budget (resource units)
  - P: Period
  - These are given in the test case files.
  - **Question:** How would you derive these yourself?
- We want to find the **equivalent BDR** interface: (α, Δ).
- **The Half-Half Algorithm** provides the conversion:
  - α is resource utilization, i.e., $\frac{Q}{P}$.
  - Δ needs to be as low as possible, i.e., $2(P - Q)$ (as derived from the half-half algorithm).

Relationship between (Δ, α) and (Q, P)



Demand Bound Functions (DBF)

- The **Demand Bound Function (DBF)** calculates the **maximum** resource demand of a set of tasks within a given time interval.
- It's crucial for schedulability analysis.
- DBF depends on the scheduling algorithm.
  We will cover DBF for:
  - **Rate Monotonic (RM)**
  - **Earliest Deadline First (EDF)**

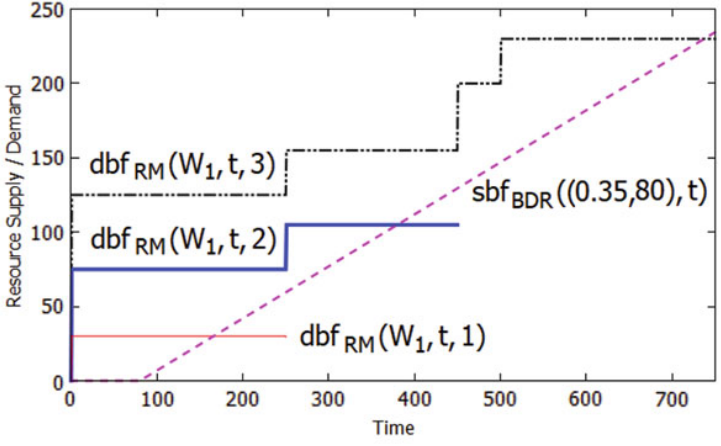
DBF for RM

- **Implicit-deadline** periodic tasks: $W = \{(C_i, T_i)\}$
- Priorities are assigned according to Rate Monotonic (RM) algorithm.
  Shorter period = Higher priority.
- DBF for a time interval t and task τ_i:

$$\mathrm{dbf}_{\mathrm{RM}}(W, t, i) = C_i + \sum_{(C_k, T_k) \in HP_W(i)} \left\lceil \frac{t}{T_k} \right\rceil \cdot C_k$$

- $HP_W(i)$: Set of tasks in W with higher priority than τ_i.
- Interpretation: The worst-case demand of task τ_i includes its own execution time plus the cumulative demand of all higher-priority tasks within the interval $[0, t]$.
- Implementation:
  - To check schedulability of task τ_i, you need to verify if there exists at least one time point $t \in [0, T_i]$ where $\mathrm{dbf}_{\mathrm{RM}}(W, t, i) \leq \mathrm{sbf}_{\mathrm{BDR}}(R, t)$.
  - **Optimization:** It's fine to start by checking every t, but think about how would you optimize this check?

SBF and DBF for M_1 in Figure 1



DBF for EDF

- **Implicit-deadline** periodic tasks: $W = \{(C_i, T_i)\}$
- C_i: Worst-Case Execution Time (WCET) of task τ_i.
- T_i: Period (and relative deadline) of task τ_i.
- DBF for a time interval t:

$$\mathrm{dbf}_{\mathrm{EDF}}(W, t) = \sum_{(C_i, T_i) \in W} \left\lfloor \frac{t}{T_i} \right\rfloor \cdot C_i$$

- Interpretation: The sum of the execution times of all jobs of all tasks that have deadlines within the interval $[0, t]$.
- Implementation:
  - To check schedulability, you need to verify that $\mathrm{dbf}_{\mathrm{EDF}}(W, t) \leq \mathrm{sbf}_{\mathrm{BDR}}(R, t)$ for all values of t from 0 up to the hyperperiod $H = \mathrm{lcm}(T_1, T_2, \ldots, T_n)$.
  - **Optimization:** It's fine to start by checking every t, but think about how would you optimize this check?

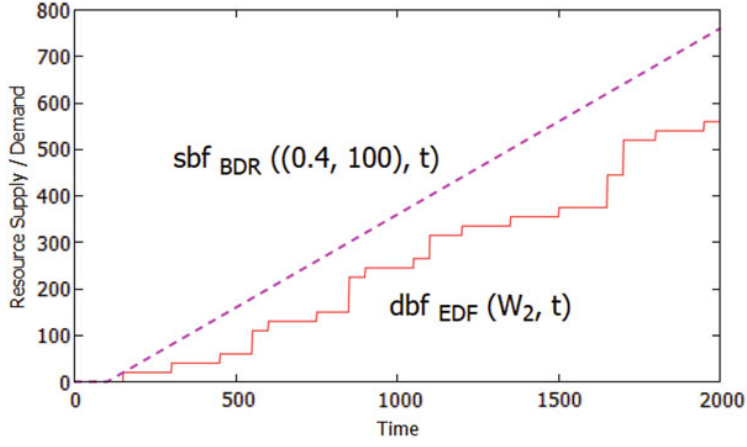
DBF for EDF (cont.)

- **Explicit-deadline** periodic tasks: $W = \{(C_i, T_i, D_i)\}$
- D_i: Relative deadline of task τ_i.
- DBF for a time interval t:

$$\text{dbf}_{\text{EDF}}(W, t) = \sum_{\tau_i \in W} \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor \cdot C_i$$

- Interpretation: The sum of execution times of all task instances with absolute deadlines within $[0, t]$, considering the release jitter.

SBF and DBF for M_2 in Figure 1



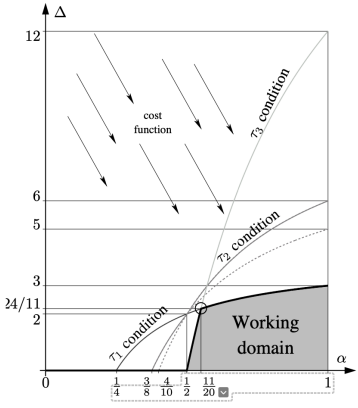
Optional Extensions

- **System Optimization:** Consider core assignment optimization, resource model parameter tuning, or priority assignment optimization.
- **Inter-task Communication:** Model, simulate and analyze communication delays between tasks.
- **Different Resource Models:** Explore using the Periodic Resource Model (PRM) or Explicit Deadline Periodic (EDP) resource model instead of BDR. Compare their effectiveness.
- **Worst-Case Response Time (WCRT) Analysis:** Develop methods to compute WCRTs for tasks in the hierarchical system. See the two papers in `Content/Project`.

Optimization Problem: Determining Component Budgets and Periods

The optimization problem involves finding ideal BDR parameters (α, Δ) while:

- Ensuring schedulability for all tasks.
- Balancing competing objectives with a cost function $c_1 \frac{\delta_{switch}}{P} + c_2\alpha$
- Lower α: Conserves processor capacity (efficiency). Lower Δ: Improves response time but may increase context switching overhead.



- The **working domain** (shaded area) shows valid (α, Δ) pairs
- Each task adds constraints (diagonal lines). The optimal point balances bandwidth and delay.

Questions

Task Periods and Hyperperiod:

- What happens if you play around with task periods, especially with respect to the resulting hyperperiod?
- How does the runtime of the schedulability analysis algorithm behave as the hyperperiod changes?

Schedulability Test Complexity:

- What do you think the complexity of the schedulability check algorithm is for each interface?
- What does the complexity depend on? (Consider the DBF and SBF calculations.)