SPRING 2010

即時控制系統設計
Design of Real-Time Control Systems

Lecture 12
Characterizing Real-Time Systems

Feng-Li Lian
NTU-EE
Feb10 – Jun10

A Definition of Real-Time Systems:

- Any system
  where a timely response by the computer
  to external stimuli is vital
  is a real-time system

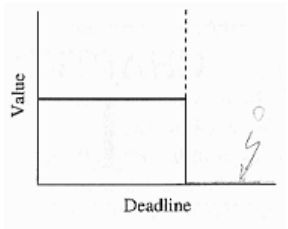
➢ Not a good definition!

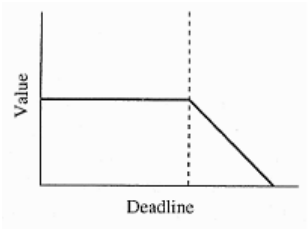
O.K. Let us see:

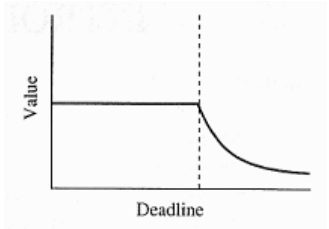
- You: What do you mean by "timely"?
- Us: It means a real-time runs tasks
  that have deadlines.
- You: By "deadlines" do you mean that
  the task must be done by then?
- Us: Not necessary.
  - Sometimes, yes: If you are controlling an aircraft by computer and you miss a sequence of deadlines
    as the aircraft comes in to land, you risk crashing the plane.
  - Sometimes, no: If you are playing a video game and the response takes a mite longer than specified,
    nothing awful will happen.

- You: What do you mean by a task being "done"?
  Is there a sharp distribution
  between when a task is "done"
  and when it is not?
- Us: Not necessary.
  - Sometimes, yes:
    If you have a banking application
    that needs to total some figures
    before it will let you draw a million dollars
    from your checking account, then yes.
  - Sometimes, no:
    If your applications needs to calculate the value of π,
    it can decide
    either to sop early & accept a less accurate value,
    or to continue calculating
    and make the estimate more and more accurate.

- You: What do you do with a real-time task
  that missed its deadline?
  Do you drop it or complete it anyway?
- Us: It depends.
  - If you are on the aircraft has crashed
    because a series of deadlines has been missed,
    neither you nor the computer is in a position to care.
  - If, on the other hand,
    you have a video-conferencing application
    that encounters a minor delay in processing a voice packet,
    you may decide not to drop that packet.
  - In any case, a task's value will drop to a certain level
    after the deadline has been missed.
  - In some cases, it will be reduced abruptly to zero;
  - In others, it will declined more gradually.
  - Figure 1.1 shows some examples.







- You: Does this make every computer
  a real-time computer by your definition?
- Us: Unfortunately, yes.
  - If you read our definition to legalistically,
    the general-purpose workstation or personal computer
    is also a real-time system:
  - If you hit the key and the computer takes an hour to echo the
    character onto the screen, you will not be very happy.
  - Everything is "real-time" in the sense of our needing the result
    within a finite time.
  - So, our definition covers all computers and
    is therefore worthless.

- You: Do you want to change your definition of real-time systems?

- Us: Yes.
  - The new definition is fuzzier, less sweeping, and not as clear-cut, but it has the inestimable virtue of ending this argument.
  - A real-time system is anything that one person considers to be a real-time system!

- This includes embedded systems that control systems like aircraft, nuclear reactors, chemical power plants, jet engines, and other objects where Something Very Bad will happen if the computer does not deliver its output in time.
- Theses are called hard real-time systems.
- There is another category called (not surprisingly) soft real-time systems, which are systems such as multimedia, where nothing catastrophic happens if some deadlines are missed, but where the performance will be degraded below what is generally considered acceptable.
- In general, a real-time system is one in which a substantial fraction of the design effort goes into making sure that task deadlines are met

Real Time:

(from the Oxford Dictionary of Computing)

- Any system in which the time at which the output is produced is significant.
- This is usually because the input corresponds to some movement in the physical world, and the output has to related to that same movement.
- The lag from input time to output time must be sufficiently small for acceptable timeliness.

Real Time Systems:

(Cooling 1991)

- Rea-time systems are those which must produce correct responses within a definite time limit.
- Should computer responses exceed these time bounds then performance degradation and/or malfunction results

- A real-time system reads inputs from the plant and sends control signals to the plant at times determined by plant operational considerations – not a times limited by the capability of the computer system

Real Time System Programs:

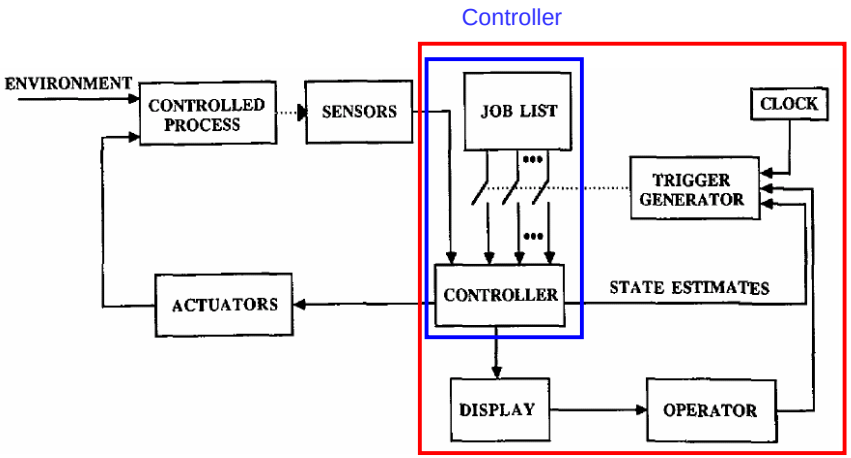
- A program for which the correctness of operation depends both on the logical results of the computation and the time at which the results are produced.

Real Time Systems:

- Are defined as those systems in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are produced
- Example:
  - Command and control systems, process control systems, flight control systems, the space shuttle avionics systems, space station, space-based defense systems

For designing a real-time system, we need:

- Specification languages & performance measures
  - that are capable of expressing timing requirements
- Means by which
  - To predict the execution times of programs (task, job, process)
  - To model the reliability of software and hardware
  - To assign tasks to processors and schedule them
    - > So that deadlines are met
  - To develop mechanisms
    - > by which the system can quickly recover from the failure of an individual component



Controller in general

- Tasks can be classified in two ways:
  - By the predictability of their arrival
    - Periodic and aperiodic tasks
  - By the consequence of their not being executed on time
    - Critical and non-critical tasks

- Periodic and aperiodic tasks:
  - Periodic tasks: Tasks that are done repetitively
    - To monitor the speed, altitude, and attitude of an aircraft
    - Tasks can be pre-scheduled
  - Aperiodic tasks: Tasks that occur only occasionally
    - When pilot wishes to execute a turn, many subtasks are set off
    - Aperiodic tasks can NOT be pre-scheduled and sufficient computing power must be held in reserve to execute them in a timely fashion
  - Sporadic tasks:
    Aperiodic tasks with a bounded inter-arrival time

- Critical and Non-critical tasks:
  - Critical tasks: The timely execution is critical
    - If deadlines are missed, catastrophes occur
      - > e.g., life-support systems, the stability control of aircraft
    - Critical tasks are often executed at a higher frequency than is absolutely necessary
    - That is, time redundancy and one successful computation every n_i iterations of critical periodic task i, which is sufficient to keep the systems alive
  - Non-critical tasks: Non-critical real-time or soft real-time
    - Do deal with time-varying data and are useless if not completed within a deadline

- Architecture Issues:
  - Processor architecture
  - Network architecture
  - Architectures for clock synchronization
  - Fault-tolerance & reliability evaluation
- Operating System Issues:
  - Task assignment & scheduling
  - Communication protocols
  - Failure management & recovery
  - Clock synchronization algorithms
- Other Issues:
  - Programming languages
  - Databases
  - Performance measures

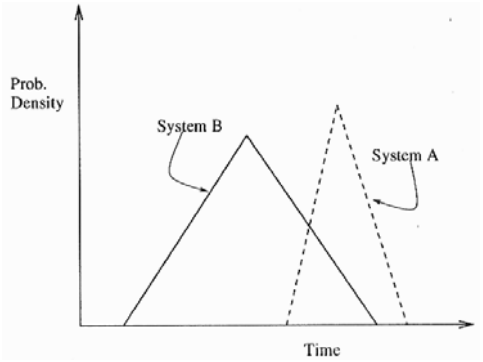
- Introduction
- Characterizing Real-Time Systems & Tasks
- Task Assignment & Scheduling
- Real-Time Programming Languages and Tools
- Real-Time Database
- Real-Time Communications
- Fault-Tolerance Techniques
- Reliability Evaluation Techniques
- Clock Synchronization

Two big questions:

- How to measure "goodness" of RTS?
  - Which performance measures are the most appropriate for real-time systems?
  - Does these have to be different from those used for general-purpose computers?
- How to estimate execution time of a program given source code & target architecture?
  - Estimate the worst-case run time of a program
  - Determine whether a real-time computer can meet task deadlines

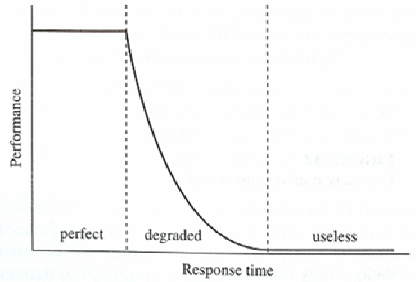
Which one is better?

- w.r.t. average execution time
- w.r.t. predictability
- What about a M + b V?
- What about (M,V)?
- How to rank systems A and B?
- etc.



Performance-Response Time Characteristic:

- The performance perceived by a user depends in a complex way on the system response time
  - e.g., a typist cannot distinguish between a delay of 5 μs and 10 μs
- As the response time becomes increasingly noticeable, the performance degrades
- Beyond a point, the performance degrades to essentially zero



Performance from the point of view of a typist

Functionality versus Speed:

- System C:
  - Has a special array-processing unit to multiply two matrices of up to 256X256 in size in four clock cycles
- System D:
  - Has a clocking frequency of 10 MHz, twice that of System C
- Both systems cost roughly the same
- Which one performs better (faster)?

Execution Time versus Code Length:

- System E:
  - On average, each instruction takes 1.2 clock cycles
- System F:
  - On average, each instruction takes 1.8 clock cycles
- When translated into machine code is twice as long in System E as it is in System F
- Which one performs better (faster)?

How to measure performance of RTS?

A good performance measure must:

- Represent an efficient encoding of relevant information
- Provide an objective basis for ranking of candidate controllers for a given application
- Provide objective optimization criteria for design
- Represent verifiable facts

Properties of Performance Measures:

- If a performance measure is to be comprehensive, it must do each of following:
  1. Express the benefit gained from a system, and
  2. Express the cost expended to receive this benefit
- Benefit:
  - Rewards that accrue from the system when it is functional
  - May be vector related to system states
- Cost:
  1. Arises when the computer does not function even at the lowest level of acceptability
  2. Life-cycle cost --- capital, installation, repair, running cost
  3. Design & development costs

Properties of Performance Measures:

- Performance measures must
  - Represent an efficient encoding of relevant information
    - > Complex systems have large amount of information
    - > The performance measure is congruent or a language to the application, then specifications can be written concisely and without contortion
  - Provide an objective basis for the ranking of candidate controllers for a given application
    - > Must quantify the goodness of computer systems
    - > Should permit the ranking of computers for the same application
  - Be objective optimization criteria for design
    - > Optimization criteria between complexity, reconfigurability, etc.
  - Represent verifiable facts
    - > Should hold out some prospect of being estimated reasonably accurately

Expressions of Performance Measures:

- Linear Combination:
  - A scalar function of a linear combination of measurable attributes of a computer

$$\sum_{i=1}^{N} a_i \, x_i$$

a_i : weights indicator or importance

x_i : attributes

- Performability:
  - Define a set of accomplishment levels and performance levels
  - = probability of the computer's performance made it perform at each of these accomplishment levels
- Cost Functions:
  - Focuses attention on the computer response time for the various computational tasks

Attributes of Computers:

- Reliability Attributes:
  - Interval Reliability, R(α,T):
    - > Probability that the computer continues to operate over an interval [α, α +T], under the assumption that it is operational at α
  - Strategic Reliability, SR(T):
    - > SR(T) = $\lim_{\alpha \to \infty} R(\alpha, T)$
    - > Steady-state reliability
    - > A function of the frequency with which preventive maintenance is carried out
  - Job-Related Reliability, $R_{job}(t, J)$:
    - > Probability that the computer, at time t, has enough hardware resources to complete job J satisfactorily
    - > Computational reliability is a similar measure

Attributes of Computers:

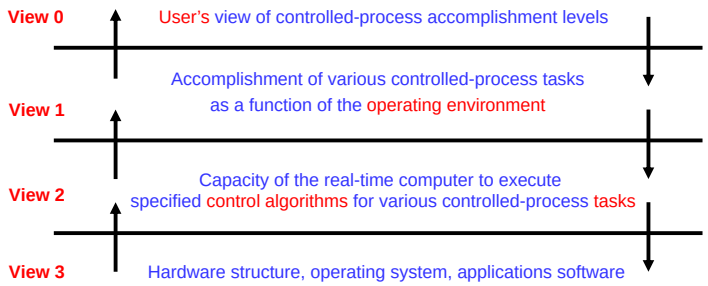
- Reliability Attributes:
  - Capacity Reliability:
    - > Probability that the system remains out of a certain set of states throughout the interval of interest
    - > Similar measures:
      - » Performance reliability, expected capacity survival time, expected capacity reduction time, mean computation before failure, pseudo-reliability (combination of relative performances)
  - Mean Time between Failure:
    - > The average time between two successive failures of the item under study
    - > Similar measures: Mission time between critical failures

- Attributes of Computers:
  - Availability Attributes:
    - Pointwise Availability, $A_p(t)$:
      - > Probability that the system will be operating within tolerance limits at time t
    - Interval Availability, $A_i(a,b)$:
      - > The expected fraction of the interval [a,b] that the system will be operating within tolerance limits
    - Performance Availability, $A_p(t)$:
      - > Similar to pseudo-reliability (combination of availability at state i)

- Attributes of Computers:
  - Maintenance Attributes:
    - Mean Time between Maintenance:
      - > The average time between two successive maintenance actions
    - Mean Maintenance Time:
      - > The average length of a maintenance job
    - Mean Ratio:
      - > The ratio of maintenance man-hours to the lifetime of the system being maintained

- Attributes of Computers:
  - Other Attributes:
    - Throughput:
      - > The average number of instructions that the system is capable of processing per unit time
    - Response Time:
      - > The time that elapses between a job commencement and termination

- Performability:
  - Given n accomplishment levels A_1, A_2, …, A_n
  - Performability is $(P(A_1), P(A_2), \ldots, P(A_n))$
  - $P(A_i)$ is probability the computer functions to allow the controlled process to reach accomplish level A_i



Qualities of Performability:

- View 0: For the user to specify
- View 1: For the control engineer
  - Who knows what control tasks need to be run and what the deadlines of such tasks are
  - List the set of environmental conditions, controlled-process performance, computer performance
- View 2: For the computer architect
  - The capacity of the computer to meet each of the demands specified in View 1
- View 3: For the computer architect
  - Focuses on the hardware, the OS, & the application software

Example: An automatic landing system of aircraft

- Landing phase:
  - Automatic landing (AL) system allows the aircraft to land even in zero-visibility weather
  - If the AL system is NOT working AND if the destination airport has low-visibility weather, the aircraft is diverted to another airport
  - If the AL feature fails during automatic landing, there is a crash
- Accomplishment levels:
  - A_0: Safe arrival at the designated destination
  - A_1: Diversion to another airport, but safe landing there
  - A_2: Crash

Example: An automatic landing system of aircraft

- A two-tuple state description at View 0: (a_0, b_0)

$$a_0 = \begin{cases} 0 & \text{if the aircraft is not diverted} \\ 1 & \text{if the aircraft is diverted} \end{cases}$$

$$b_0 = \begin{cases} 0 & \text{if the aircraft does not crash} \\ 1 & \text{if the aircraft crashes} \end{cases}$$

- Mapping of View-0 states to accomplishment levels

Accomplishment level	Corresponding View-0 states
A_0	$(0,0)$
A_1	$(1,0)$
A_2	$(0,1),\ (1,1)$

Example: An automatic landing system of aircraft

- At View 1: A three-tuple state description (a_1, b_1, c_1)

$$a_1 = \begin{cases} 0 & \text{if the visibility is good at the designated airport} \\ 1 & \text{if the visibility is poor at the designated airport} \end{cases}$$

$$b_1 = \begin{cases} 0 & \text{if the AL feature is functional during the landing phase} \\ 1 & \text{if the AL feature fails before the landing phase begins} \\ 2 & \text{if the AL feature fails during the landing phase} \end{cases}$$

$$c_1 = \begin{cases} 0 & \text{if all the flight-critical mechanical parts work properly} \\ 1 & \text{if there is flight-critical mechanical failure} \end{cases}$$

Example: An automatic landing system of aircraft

- Mapping of View-1 states to View-0 states

View-0 states	Corresponding View-1 states
$(0,0)$	$(0,0,0),\ (0,1,0),\ (0,2,0),\ (1,0,0)$
$(0,1)$	$(0,0,1),\ (0,1,1),\ (0,2,1),\ (1,0,1),$ $(1,2,0),\ (1,2,1)$
$(1,0)$	$(1,1,0)$
$(1,1)$	$(1,1,1),$

Example: An automatic landing system of aircraft

- At View 2: A single-state description (a_2)

$$a_2 = \begin{cases} 0 & \text{if the computer has sufficient resources to run the AL job throughout the landing phase} \\ 1 & \text{if the computer does not have sufficient resources to run the AL job at any time during the landing phase} \\ 2 & \text{if the computer has sufficient resources at the beginning of the landing phase, but suffers failures which make it impossible to run the AL job some time during the landing phase} \end{cases}$$

- The weather state variable (w)

$$w = \begin{cases} 0 & \text{if the visibility at the designated ariport is good} \\ 1 & \text{otherwise} \end{cases}$$

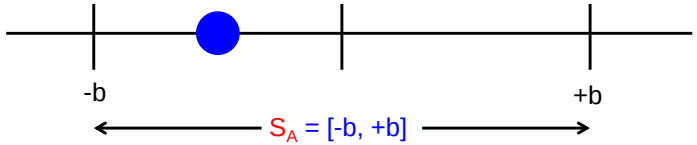
Example: An automatic landing system of aircraft

- The performability of the computer: $(P(A_0), P(A_1), P(A_2))$
  - $P(A_i)$: the probability of the computer being able to function sufficiently well to ensure that A_i is attained
  - Can be done by tracing through the mapping of the states
  - For example, A_0 is attained whenever the system is in
    - > View-0 state { (0,0) }, which happens whenever the system is in
    - > View-1 states { (0,0,0), (0,1,0), (0,2,0), (1,0,0) },
    - > Which happens whenever the states of the weather and the computer are $(w,a_2) \in$ { (0,0), (0,1), (0,2), (1,0) } and When $c_1 = 0$
  - $P(A_0) = Pr\{w=0\}\ Pr\{c_1=0\} + Pr\{w=1\}\ Pr\{b_1=0\}\ Pr\{c_1=0\}$

Cost Functions & Hard Deadlines:

- Hard deadline:
  - The time by which they must finish executing if catastrophic failure of controlled process is avoided
  - Maximum controller (computer) “think” (response) time that will allow the controlled process to be kept within allowed state space S_A
- Cost function (of response time)
  - Compare the performability of a RTS with a zero response time to a system with a given positive response time of ξ
  - i.e. $C(\xi) = P(\xi) - P(0)$

Example: A body of mass m



– State vector Σ = (x,v,a)

– The job of real-time computer controlling m is
to use thrusters that can exert a thrust of magnitude up to H
in either a + or – direction
to keep the body at a given ideal point
for as much of the time as possible

Example: A body of mass m

- Hard deadline:
  – The delay such that the controller may not be able
  to stop the body from moving out of the allowed state space
  – Other delays are assumed ZERO
- Cost function:
  – The energy expended by the system
  in getting the body back to the ideal point ASAP
  – That is, the energy difference under the controller response time
  (time delay) ξ
- Both hard deadline and cost functions are functions of
the current state of the controlled process

Example: A body of mass m

- In (0,0,0):
  – Hard deadline = ∞ & Cost function = 0
- In (x,0,0):
  – Hard deadline = ∞ & Cost function = 0
- In (x,v,0): (x>0, v>0)

$$
\begin{aligned}
(x, v, 0) \quad &\Rightarrow \quad (x+\xi v, v, 0) \\
&\Rightarrow \quad -a = -\frac{H}{m} \\
&\Rightarrow \quad \text{rest at } x_1 = x + v\xi + v^2/2a
\end{aligned}
$$

Example: A body of mass m

- If $x_1 > b$, then the controlled process fails:

$$\Rightarrow \quad x + v\xi + v^2/2a > b \qquad \Rightarrow \quad \xi > \frac{b - x - v^2/2a}{v}$$

- Hard deadline = $\quad t_d(x, v, 0) = \dfrac{b - x - v^2/2a}{v}$
- The distance of maintaining a thrust H

$$d_\xi = x_1 - (x + \xi v) + x_1$$

- Cost function:

$$
\begin{aligned}
C_{(x,v,0)}(\xi) &= H(2x_1 - x - \xi v) - H\left(x + \frac{v^2}{a}\right) \\
&= v\xi H
\end{aligned}
$$

- The fact:
  - That the controller computer and the controlled process are designed in a more or less disjoint manner only serves to compound the difficulty of achieving optimal and ultra-reliable control based upon a computer
- Performance measures:
  - To characterize the behavior of controller computers, describing precisely the goodness of the controller computer in the context of the application
- Real-time computer performance:
  - A vector $\mathbf{p} \in \mathbf{R}^p$ and a weighting vector $\mathbf{w} \in \mathbf{R}^p$
    - Made up of such traditional measure such as (convectional) reliability, throughput, survivability, availability, etc.
  - f: $\mathbf{R}^p \rightarrow \mathbf{R}$ with $f(\mathbf{p}) = \mathbf{w}^T \mathbf{p}$

- Elementary observations of performance measures:
  1. Nonzero controller response time has negative impact on the behavior of the controlled process
  2. There is a limit to how great the response time can be before the process behaves unacceptably
  3. Even if this response time is kept within the above bounds, an incremental increase tends to lead to deterioration in controlled process behavior

- Example: the final phase of aircraft flight
  - Control constraints:
    Limits on
    - The speed of touchdown (both horizontal & vertical)
    - The angle of attack α, and
    - The pitch angle θ
  - Control constraints:
    Safeguard against
    - Running out of runway,
    - Undercarriage collapse,
    - Stalling, and
    - Landing either on the aircraft nose or tail

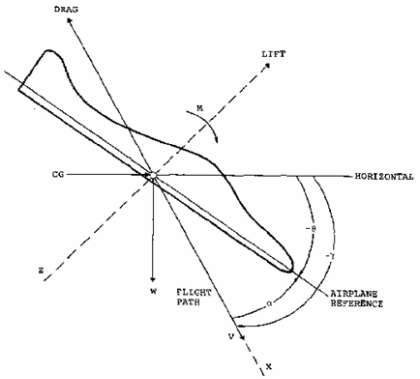


Fig. 1. Definition of aircraft angles [from (4)].

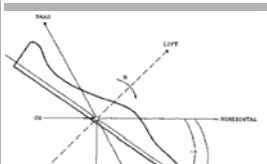
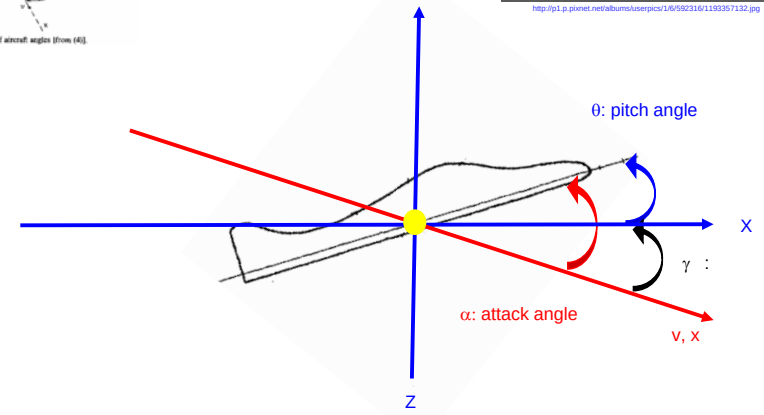
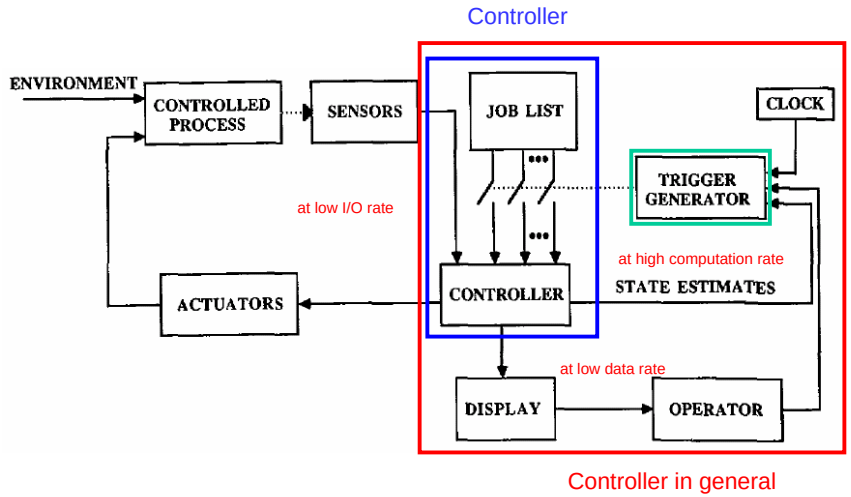


Fig. 1. Definition of aircraft angles [from (4)].


http://www.boeing.com/companyoffices/gallery/images/commercial/747400f-k57816.jpg

http://p1.p.pixnet.net/albums/userpics/1/6/592316/1193357132.jpg





- Trigger generator:
  1. Time-Generated Trigger:
     - Generated at regular intervals
     - The corresponding controller job being initiated at regular intervals
     - Open-loop triggers
  2. State-Generated Trigger:
     - Generated whenever the system is in a particular set of states
     - Closed-loop triggers
     - If time is to be regarded as an implicit state variable, the time-generated trigger is a special case of the state-generated trigger
  3. Operator-Generated Trigger:
     - Operator overrides the automatic systems, generating and canceling triggers at will

- The mission lifetime of a civilian aircraft:
  1. Takeoff/cruise until VHF omnirange (VOR)/distance measuring equipment (DME) is out of range
  2. Cruise until VOR/DME is in range again
  3. Cruise until landing is to be initiated
  4. Landing
     - Takes 20 sec
     - The control of the aircraft elevator deflection during landing
     - Sensors:
       - Altitude, descent rate, pitch angle, pitch angle rate: every 60 ms
     - Controller:
       - Time-generated triggered controller computer: every 60 ms
       - Controller response time: 20 ms

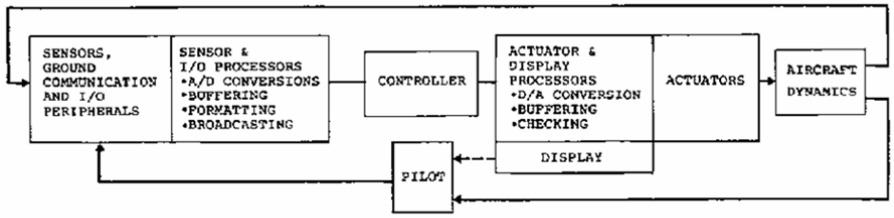


Fig. 3. Aircraft control system schematic.

- Performance Measures related to:
  - Allowed or admissible state-space, and
    - Every critical process must operate within a state-space circumscribed by given constraints
  - Dynamic failure
    - Leaving this allowed state-space constitutes dynamic failure
    - Occur as a result of the controller not responding faster enough
  - Hard deadlines
    - If controller takes longer than hard deadline to formulate the control, dynamic failure becomes possible

- Performance Measures:
  - Cost function $C_\alpha(\xi)$
    associated with controller response time ξ
    for controller job α

$$C_\alpha(\xi) = \begin{cases} g_\alpha(\xi) & \text{if } 0 < \xi \le \tau_{d\alpha} \\ \infty & \text{if } \xi > \tau_{d\alpha} \end{cases}$$

    - $g_\alpha(\cdot)$: a suitable continuous monotonically non-decreasing func.
    - $\tau_{d\alpha}$: the hard deadline associated the job α

- Hard Deadlines:
  - State x(t):
    - The state of the controlled process at time t
  - State Transitions:
    - ϕ: $T \times T \times X \times U \to X$
      - > $T \subset R$: the time region
      - > $X \subset R^n$: the state-space
      - > $U \subset R^l$: the input space
      - > $\Omega \subset U$: the admissible input space
      - > $X_A \subset X$: the allowed state-space

$$\Rightarrow \quad x(t_1) = \phi(t_1, t_0, x(t_0), u)$$

- Hard Deadlines:
  - Unconditional hard deadline:

$$\tau_{d\alpha}\left(x(t_0)\right) \equiv \inf_{u\in\Omega} \sup\{\, \tau \mid \phi(t_0+\tau, t_0, x(t_0), u) \in X_A \}$$

    - For every point in the state-space and for each critical job, we have a corresponding hard deadline
    - If the closed-form solutions are not available, the unconditional hard deadlines are impossible to obtain
  - Conditional hard deadline

$$\tau_{d\alpha|w,\sigma}\left(x(t_0)\right) \equiv \inf_{u\in w} \sup\{\, \tau \mid \phi(t_0+\tau, t_0, x(t_0), u) \in \sigma \}$$

    - $\omega \subset \Omega$, $\sigma \subset X_A$, $x(t_0) \in \sigma$

- Allowed State-Space:
  - S_i, i = 0, 1, ⋯, s: disjoint state-subsets of X_A with $X_A = U^s_{i=1} S_i$
  - J: a controller job
  - The projection:

    $$(J, X_A) \rightarrow ((T_0, S_0), (T_1, S_1), \cdots, (T_s, S_s))$$

    where T_i is the controller task generated by executing J in S_i
  - X_A^1:
    - the set of states that the system must reside in if catastrophic failure is not to occur immediately
    - An aircraft flies upside down!
  - X_A^2:
    - the set of acceptable states given the terminal constraints
  - The allowed state-space: $X_A \equiv X_A^1 \cap X_A^2$

- The Controlled Process: An aircraft in the phase of landing

x_1 : the pitch angle rate
x_2 : the pitch angle
x_3 : the altitude rate
x_4 : the altitude
m_1 : the elevator deflection
ξ : the controller response time

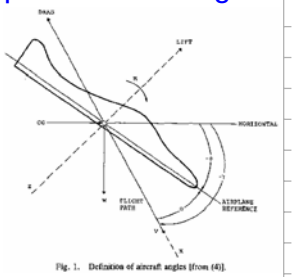


Fig. 1. Definition of aircraft angles [from (4)].

$$\dot{x}_1(t) = b_{11}x_1(t) + b_{12}x_2(t) + b_{13}x_3(t) + c_{11}m_1(t,\xi)$$
$$\dot{x}_2(t) = x_1(t)$$
$$\dot{x}_3(t) = b_{32}x_2(t) + b_{33}x_3(t)$$
$$\dot{x}_4(t) = x_3(t)$$

b_{11}	-0.600
b_{12}	-0.760
b_{13}	0.003
b_{32}	102.4
b_{33}	-0.400
c_{11}	-2.374

- The Controlled Process: An aircraft in the phase of landing
  - It takes about 20 sec
  - Initially, the aircraft
    - Altitude: 100 feet
    - Horizontal speed: 256 ft/s (assumed constant over the entire interval)
    - Rate of descent: 20 ft/s initially
    - Pitch angle: 2° (constant)
  - Constraints:
    - Motion of elevator: Between -35° and 15°
    - Pitch angle: Between 0° and 10°, to avoid landing on the nosewheel or on the tail
    - Angle of attack: less than 18°, to avoid stalling
    - Vertical speed: less than 2 ft/s, then undercarriage can withstand the force of landing

- The Controlled Process: An aircraft in the phase of landing
  - Desired altitude trajectory (feet):

$$h_d(t) = \begin{cases} 100e^{-t/5} & 0 \le t \le 15 \\ 20 - t & 15 \le t \le 20 \end{cases}$$

  - Desired rate of ascent (ft/s):

$$\dot{h}_d(t) = \begin{cases} -20e^{-t/5} & 0 \le t \le 15 \\ -1 & 15 \le t \le 20 \end{cases}$$

  - Desired pitch angle: 2°
  - Desired pitch angle rate: 0 deg/sec

- The Controlled Process: An aircraft in the phase of landing
  - Control Law for the elevator deflection:

$$\begin{aligned} m_1(t,\xi) &= w_s^2 K_s T_s \Big[k_1(t-\xi) \\ &\quad -k_{11}(t-\xi)x_1(t-\xi) - k_{12}(t-\xi)x_2(t-\xi) \\ &\quad -k_{13}(t-\xi)x_3(t-\xi) - k_{14}(t-\xi)x_4(t-\xi) \Big] \end{aligned}$$

K_s	-0.95 s^{-1}
T_s	2.5 s
w_s	1 rad/s^{-1}
k_{ij}	constant feedback paramters

- The Controlled Process: An aircraft in the phase of landing
  - Performance Index:

$$\Theta(\xi) = \int_{t_0}^{t_f} e_m(t,\xi)dt$$

$$\begin{aligned} e_m(t,\xi) &= \phi_h(t)[h_d(t) - x_4(t)]^2 + \phi_{\dot{h}}(t)[\dot{h}_d(t) - x_3(t)]^2 \\ &+ \phi_\theta(t)[x_{2d}(t) - x_2(t)]^2 + \phi_{\dot{\theta}}(t)[x_{1d}(t) - x_1(t)]^2 \\ &+ [m_1(t,\xi)]^2 \end{aligned}$$

$$\begin{aligned} \phi_h(t) &= \phi_4(t) + \phi_{4,t_f}\delta(20-t) \\ \phi_{\dot{h}}(t) &= \phi_3(t) + \phi_{3,t_f}\delta(20-t) \\ \phi_\theta(t) &= \phi_{2,t_f}\delta(20-t) \\ \phi_{\dot{\theta}}(t) &= \phi_1(t) \end{aligned}$$

$\phi_1(t)$	99.0
$\phi_{2,t_f}(t)$	20.0
$\phi_3(t)$ $(0 \le t < 15)$	0.0
$\phi_3(t)$ $(15 \le t \le 20)$	0.0001
$\phi_{3,t_f}(t)$	1.0
$\phi_4(t)$	0.00005
$\phi_{4,t_f}(t)$	0.001

Elevator deflection

stuck at -35^o for 60 ms 8 s

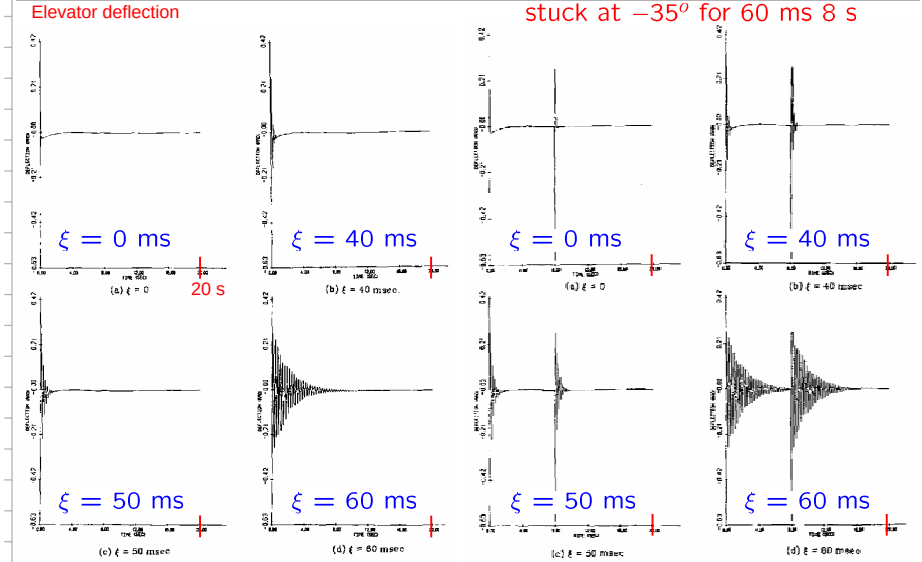
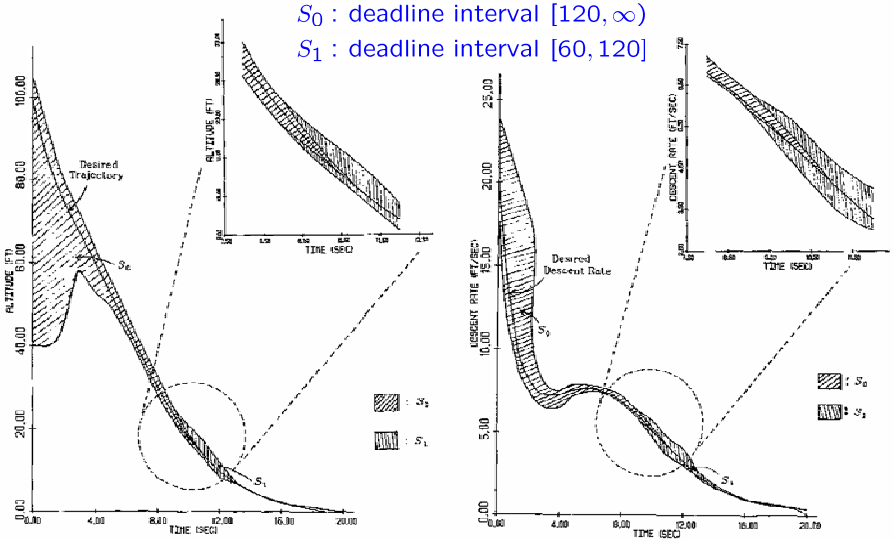


Fig. 4. Elevator deflection.

Fig. 5. Elevator deflection with abnormality.

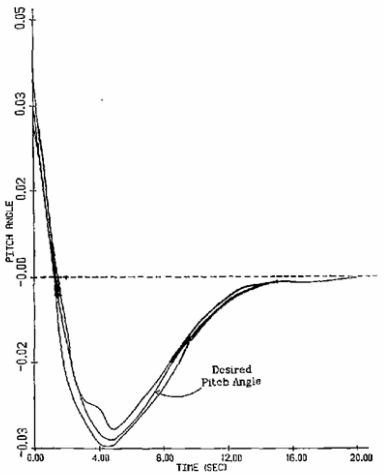
S_0 : deadline interval $[120, \infty)$

S_1 : deadline interval $[60, 120]$

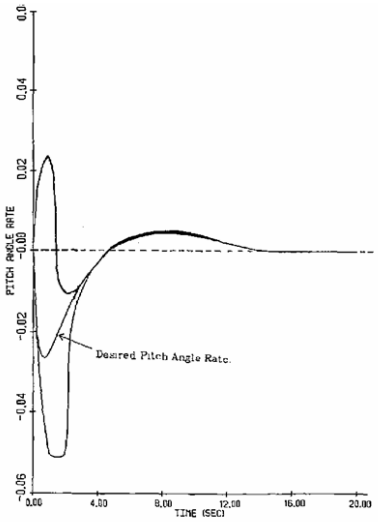


Allowed state-space: Altitude

Allowed state-space: Descent rate



Allowed state-space: Pitch angle



Allowed state-space: Pitch angle rate

- The Controlled Process: An aircraft in the phase of landing
  - Finite Cost Function:

$$C_\alpha(\xi) = \begin{cases} g_\alpha(\xi) & \text{if } 0 < \xi \le \tau_{d\alpha} \\ \infty & \text{if } \xi > \tau_{d\alpha} \end{cases}$$

$$g(\xi) = \Psi(\xi) - \Psi(0)$$

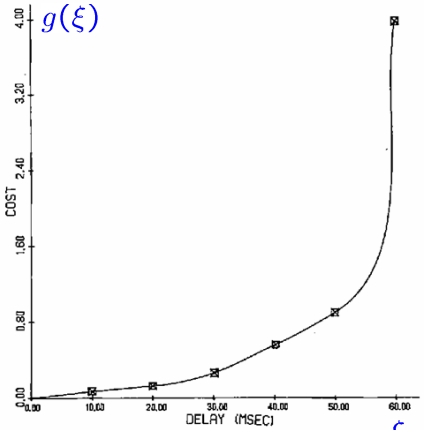


Fig. 7. Finite cost function.

- Estimating Task Execution Times:
  - Depending on:
    - Source code:
      - > Carefully tuned and optimized codes take less time to execute
    - Compiler:
      - > Non-unique mapping of source to object code
    - Machine architecture:
      - > Processors, memory, I/O devices, registers, cache, etc.
    - Operating system:
      - > Task scheduling & memory management

- Task Times:
  - Response time:
    - Time between task released to actual delivered
  - Queue time:
    - At buffer
  - End-to-end delay:
    - Delay of applications
  - Path/execution delay:
  - etc.

- Need an Ideal Tool: