Introduction to Information Engineering

Stephen Roberts

Lecture 1

A gentle introduction

Aims

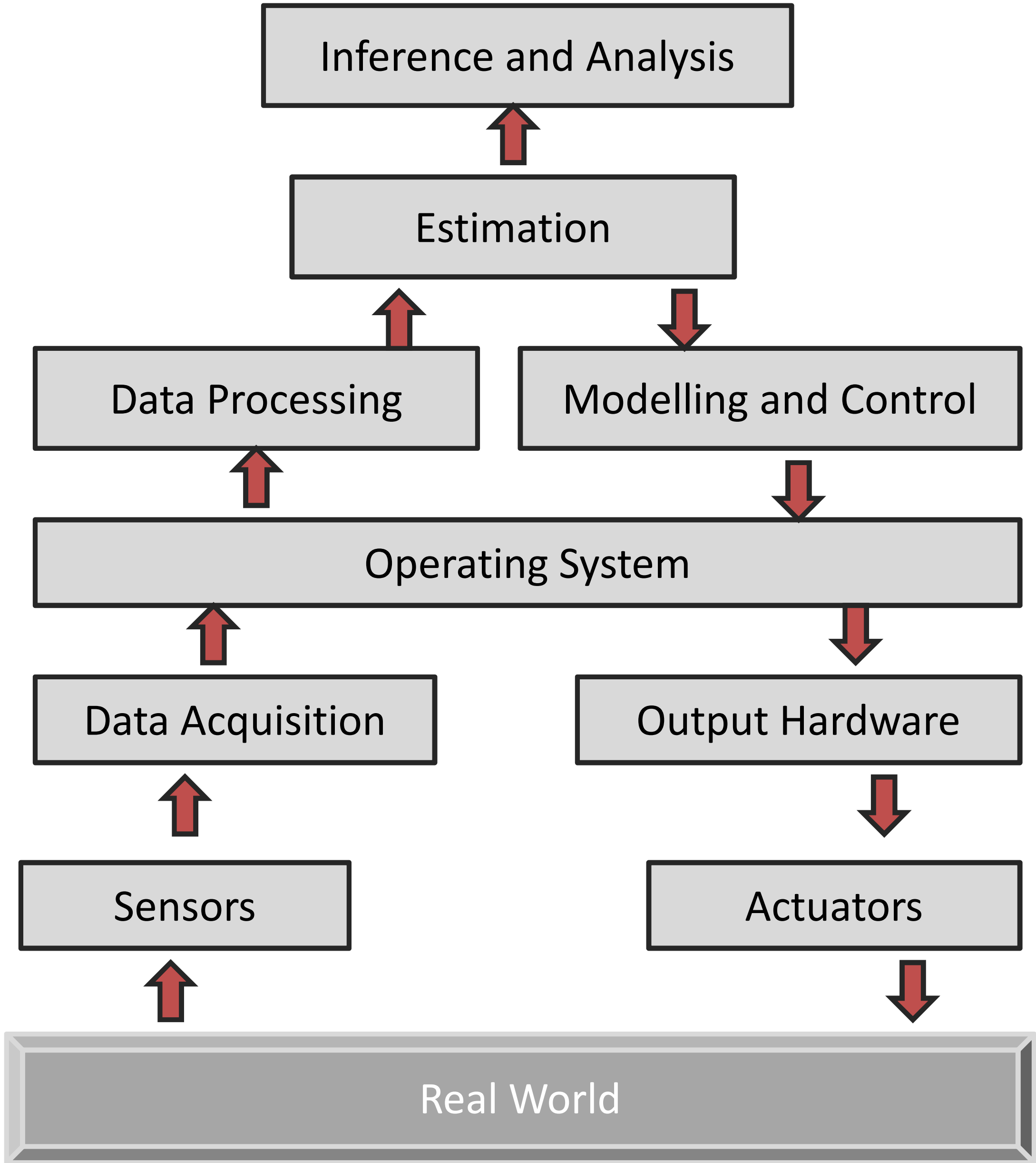
- To provide you with an overview of what B4 and information engineering in general is concerned with
- To make explicit links between information engineering and the core syllabus especially A1, A2 and A3
- To give you some sense of how central information engineering is to the engineer’s career and to our every day lives.

Course Outcomes

At the end of this 4 lecture course you should

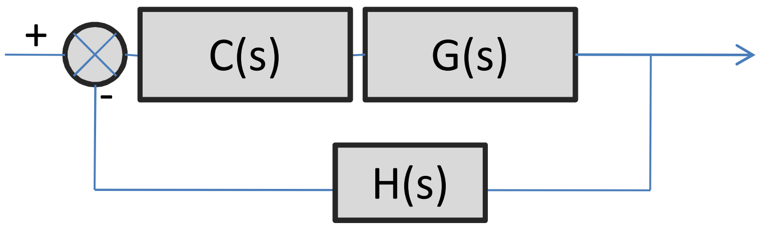
- be able to deconstruct overall data capture / analysis / control system into components and understand how they interact
- appreciate the role of the computer as a general purpose information processing tool
- understand the role of the operating system and how both sensors and actuators can be interfaced to a computer at the hardware and software level
- understand the role of probability as the mathematical tool for modelling uncertainty in sensors, and how to use Bayes rule as a means to combine sensor measurements or prior information
- understand the consequences of sampling and ZOH, and how to discretize continuous controllers
- be able to analyse the components of a fast-sampled feedback system both in isolation and in the context of the complete system

The Information Engineering Domain



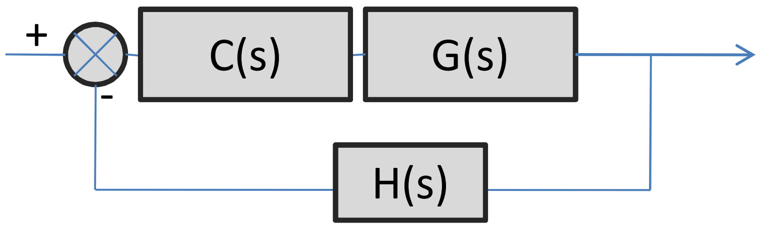
The Role of Feedback

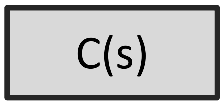
- Note the presence of a feedback loop in the previous architecture.



- The control system block diagrams you manipulate in A3 are powerful mathematical abstractions for devising control strategies for systems
- To actually instantiate/embed this control system in a real vehicle, the controller design and analysis, is only part of the story. Information engineering (inc B4) is much more than control theory.

What is C(s)



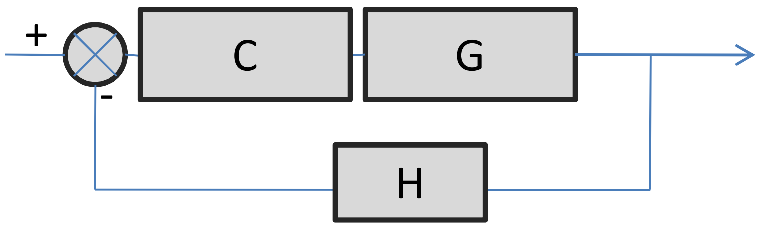


A controller that is implemented in all likelihood on a computer

Issues:

- What does the software of the controller look like?
- What speed must it run at?
- Computers are discrete devices but the world is continuous so how does one link the two?
- Does using a discrete controller have stability implications?
- What design tools are available for the discrete domain?
- Do familiar continuous domain analysis tools have discrete time duals?

What is H?



H

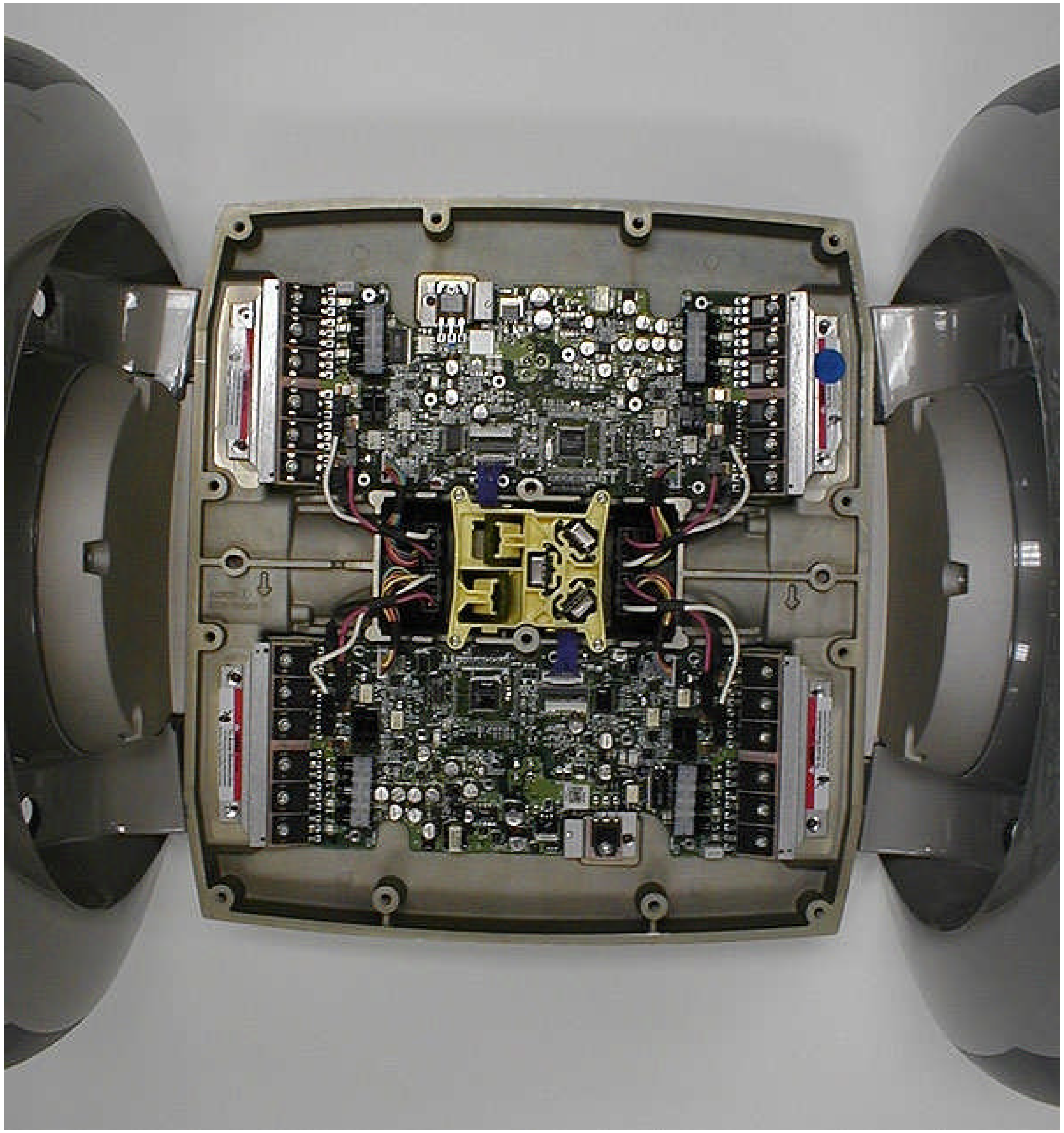
A transfer function between a sensed plant output and the quantity we wish to control

Issues:

- How does one sample the plant output ?
- How does one transmit measurements to the CPU running the controller?
- How does one guarantee that measurements will always be processed ?
- What does one do if the sensed output is not what we wish to control e.g sensing color but wanting to control flow rate?
- How does one deal with noisy sensor data?
- How does one fuse multiple measurements?

Information Systems Exemplar

- The “Segway” robot shown here is a container of many of the central concerns of the information engineer (and as it happens, electrical engineers)
- Sensing (accelerometers, gyro)
- Actuation Control (varying payload)
- Computing
  - IO from sensors
  - Output to actuators
  - Controllers in software
  - Estimation of state by processing sensor data

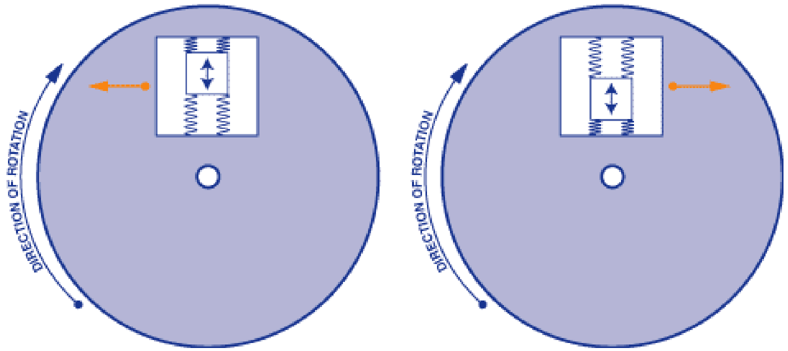


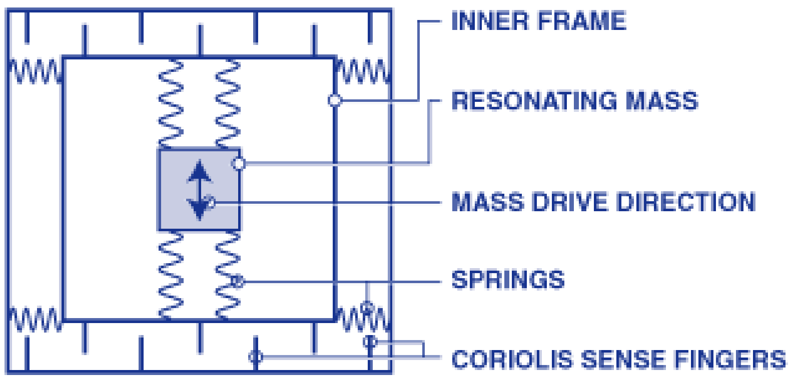
Note:

•Duplication of electronics (safety)
•Requires interfacing of sensors and motors to computation
•Requires control to be implemented on a computer
•Control laws are non trivial : to stop you have to first speed up!
•Requires interpretation of sensor data
•Requires an internal model

Info Eng. Components of the Segway

- Sensors - 5 Corriolis (interesting) gyros



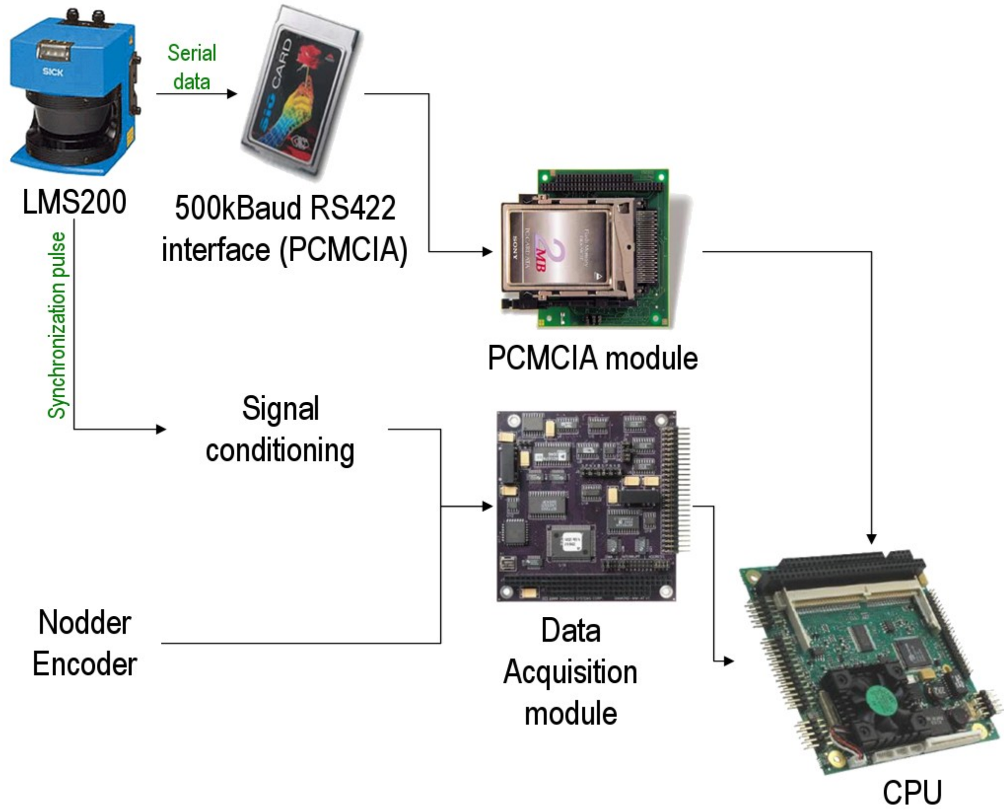


How do you combine the information from 5 noisy sensors in a principled way? (B4…)

Computation Hardware

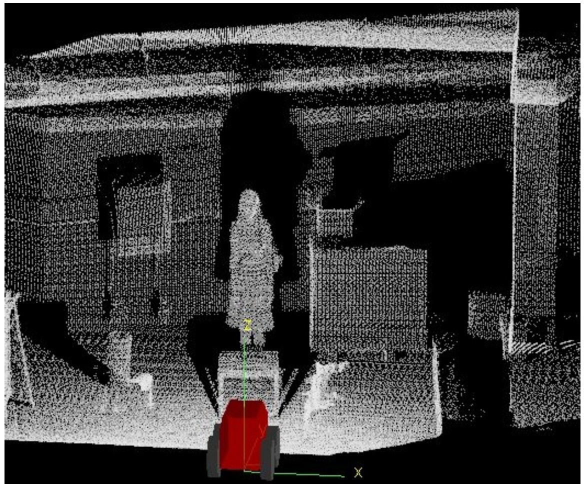
- Data Acquisition : The PIC16F87x Flash microcontrollers process sensor data from the inertial monitoring unit and communicate information to the control module.
- Control Module is a 100 MIPS Digital Signal Processor TMS320C2000 from Texas Instruments.
- Communication is via CAN and I2C bus
- Two boards acting in duplicate for safety
- Some interesting stories on redundancy here….

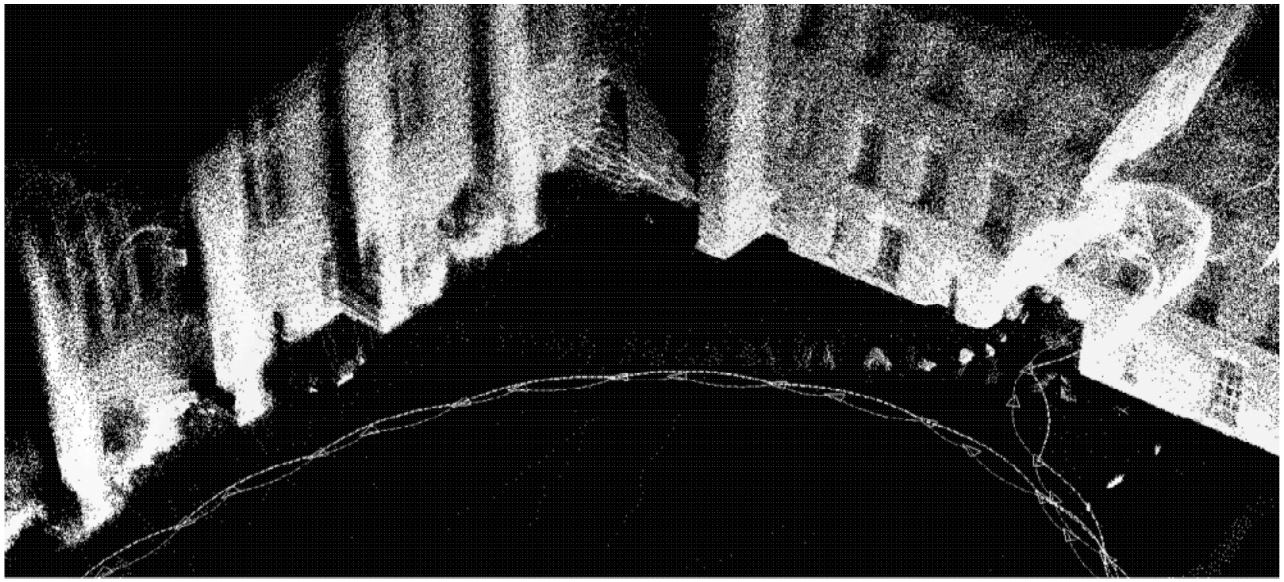
Exemplar II – a 3D laser System

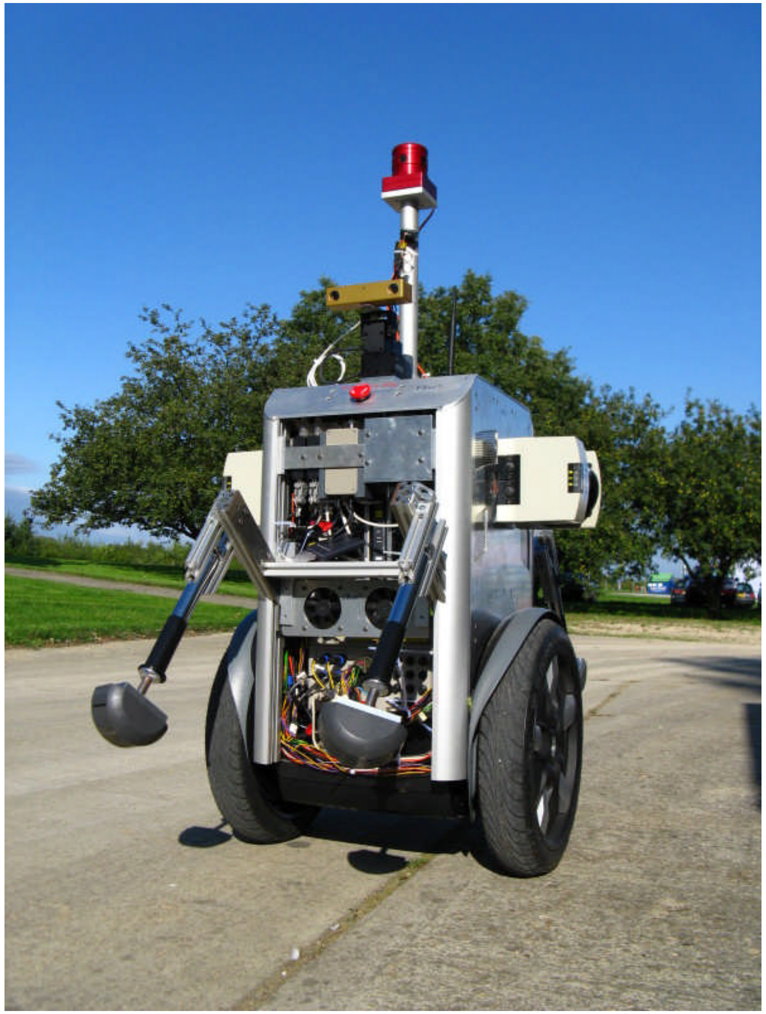


- Issues – synchronisation of disparate data streams
- <u>Estimation</u> of system latencies

3D Reconstruction



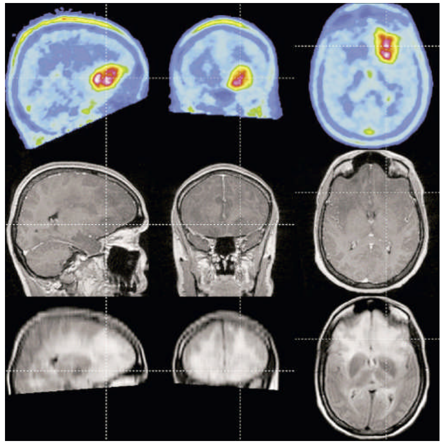




Compelling Cross Discipline Problems

Building climate control
Very uncertain Plant
Large unknown lags

Engine Management
Abnormality detection
Machine Learning



Medical imaging
Imprecise sensor data
Deformable structure
Complex 3D reconstruction
Diagnosis from measurements

And Some More

Network analysis
National Grid
Complicated non-linear coupled dynamics

Car design
Complex optimisation task
Active suspension
Traction control, slip estimation
Plant identification

Lecture II –The Role of the Computer

- IO sensor interfaces
  - Serial ports
  - Ethernet
  - PCI
  - Firewire
- Microcontrollers
  - PICs
  - embedded systems,
  - pic diagram ref segway
- OS
  - device drivers
- Processes and IPC (inter process communication)

Motivation

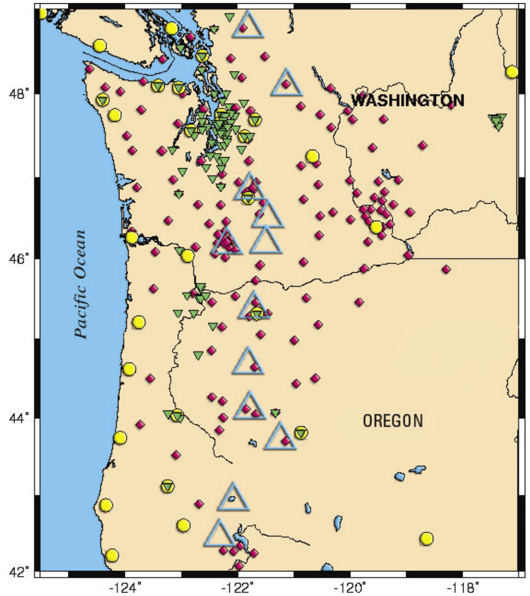
- If we are to design a complete information engineering system we may need to consider of how data is or should be marshalled
- Data transfer technology is ubiquitous and 5* Engineers should be able to say something sensible about every day equipment!

Sensor/Actuator Interfacing

- How to get data from sensor to processor? Common choices
  - Direct to bus (PCI)
  - External serial protocols RS232, firewire, USB
  - CAN bus (controller area network)
  - All need hardware/software to transport data

Vehicle control

Flight surface control and anomaly detection



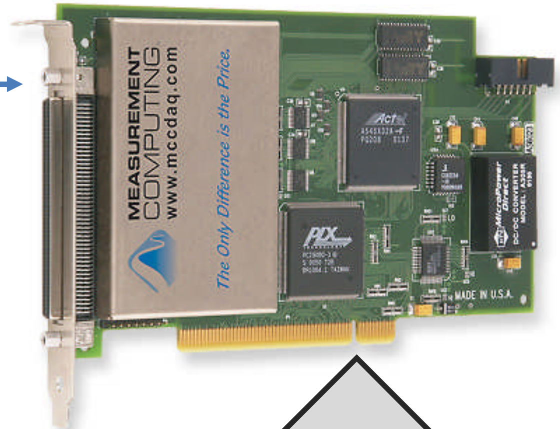
Seismic sensing networks

Straight to PC Bus

Example Engberg PCI-DAS6035
•16 channels of 16-bit A/D board
•two 12-bit analog outputs
•8 digital I/O lines, two 16-bit counter

Instrumentation Rig

Analog/digital data



Control signal

High power amplifier

PCI BUS

PCI Bus (Peripheral Component Interface Bus)
- •33Mhz Clock
- •Generally 32 bits wide (specification allows for 64 bits)
- •Allows plug and play – BIOS configures interrupts /address space
- •Allows burst mode transfers

Offline analysis and inference

Inter-Device Serial Protocols

Can be very simple to implement
 - at its simplest one wire for Tx one for Rx and one for Gnd between two devices
- varying electrical and data protocols dictate complexity and performance

Slow speed RS232, RS485, RS422 – the COM Ports on your PC, long distance, simple hardware, simple data protocol

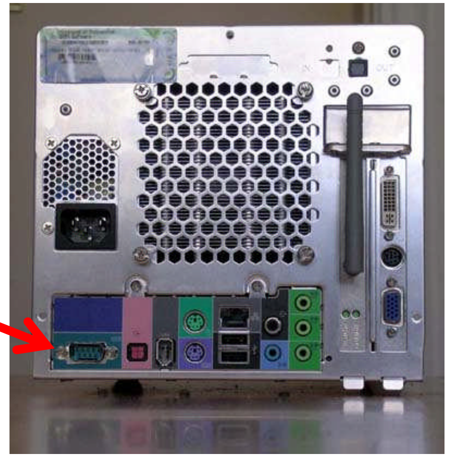
USB (universal serial bus) faster now ubiquitous, short distances, 12 Mbits/s or 480Mbits/s (USBII)

Firewire (a.k.a IEEE 1394, iLink) very fast, short distance
 800 Mbits/s

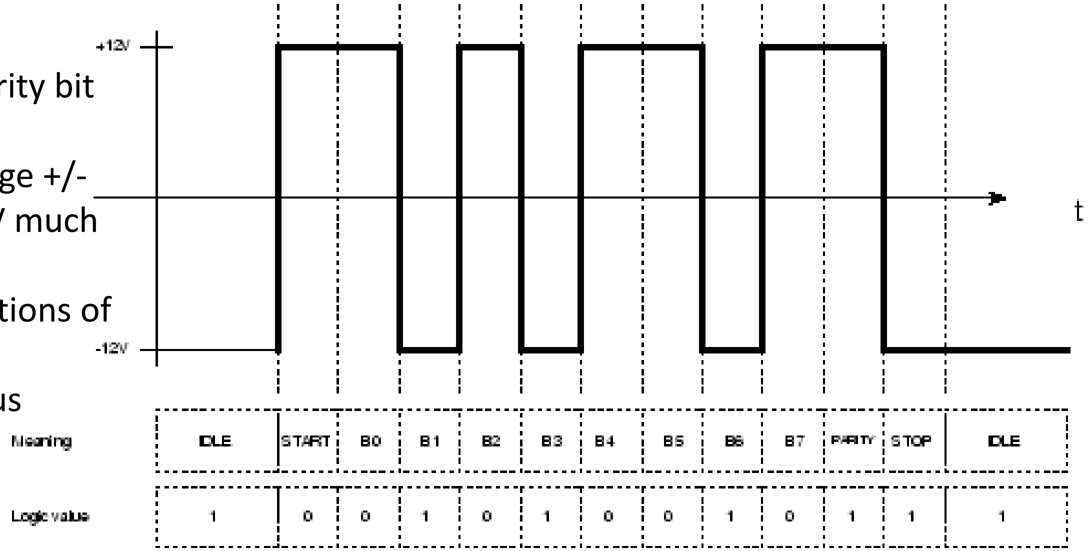
CAN bus – very robust, multiple devices, slow, an industrial favourite. Very common in cars (invented by BMW)

RS232/RS422

- •Very common found on almost every non-laptop PC ("COM ports")
- •Generally slow data rates <115kBaud. Asynchronous – no clock
- •Sends / receives data in packets serially
- •At its most basic, RS232 needs only 3 signal wires Tx/Rx and Gnd (pins 2,3 and 5 on a 9 pin connector)
- •RS422 is a differential signal – instead of raising and lowing one wire at a time TXA goes up while TXB goes down
- •Depending on Baud rate can transmit many 10s of meters
- •Data protocol described using a triplet :
    - •[Data packet size][Parity Bit][NumStopBits]
    - •8N1 and 8N2is common – 8 data bits, no parity bit with one or two stop bits.
- •Voltage levels for RS232 and RS422 are typically large +/-12V is common, but you can get away with 0 and 5V much of the time.
- •Note RS232 and RS422 are both electrical specifications of simple serial protocols
- •A special chip called a UART Universal Asynchronous Receive Transmit is used to manage the serial link and produce bytes of data from the serial stream



RS232 Transmission of the letter 'J'



This is 8E1 (even parity s.t #ones is even)

Image from www.best-microcontroller-projects.com

The Microcontroller

- Include hardware for common IO tasks
  - PWM (Pulse width modulation)
  - A2D D2A
  - Serial Ports (TTL not usually RS232 etc)
  - Digital IO
- When deployed, typically only runs one program burnt into EEPROM. (ie no OS just a while(1)…)
- On board RAM
- Self contained – little external interfacing required
- Can sometimes be programmed with high level languages like C using manufacturer’s compilers
- Very cheap (almost free!)

PIC's

- PICS (**Programmable Intelligent Computers)** are very common brand of microcontrollers and you'll find them everywhere
- Typically slow clock rates (<40MHz)
- Very cheap (from a few pence)
- Easy to program very few instructions
- small number of pins very easy to interface
- Typically little on board RAM (perhaps a few K of data space)
- Ideal for dedicated processing unit for a single device for example interpreting keyboard interaction.

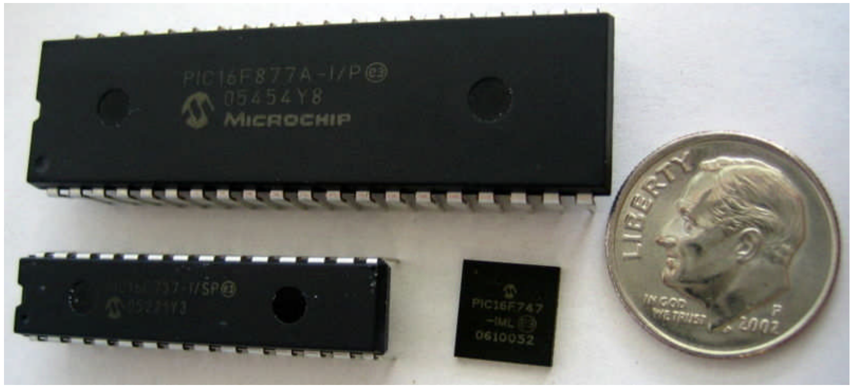
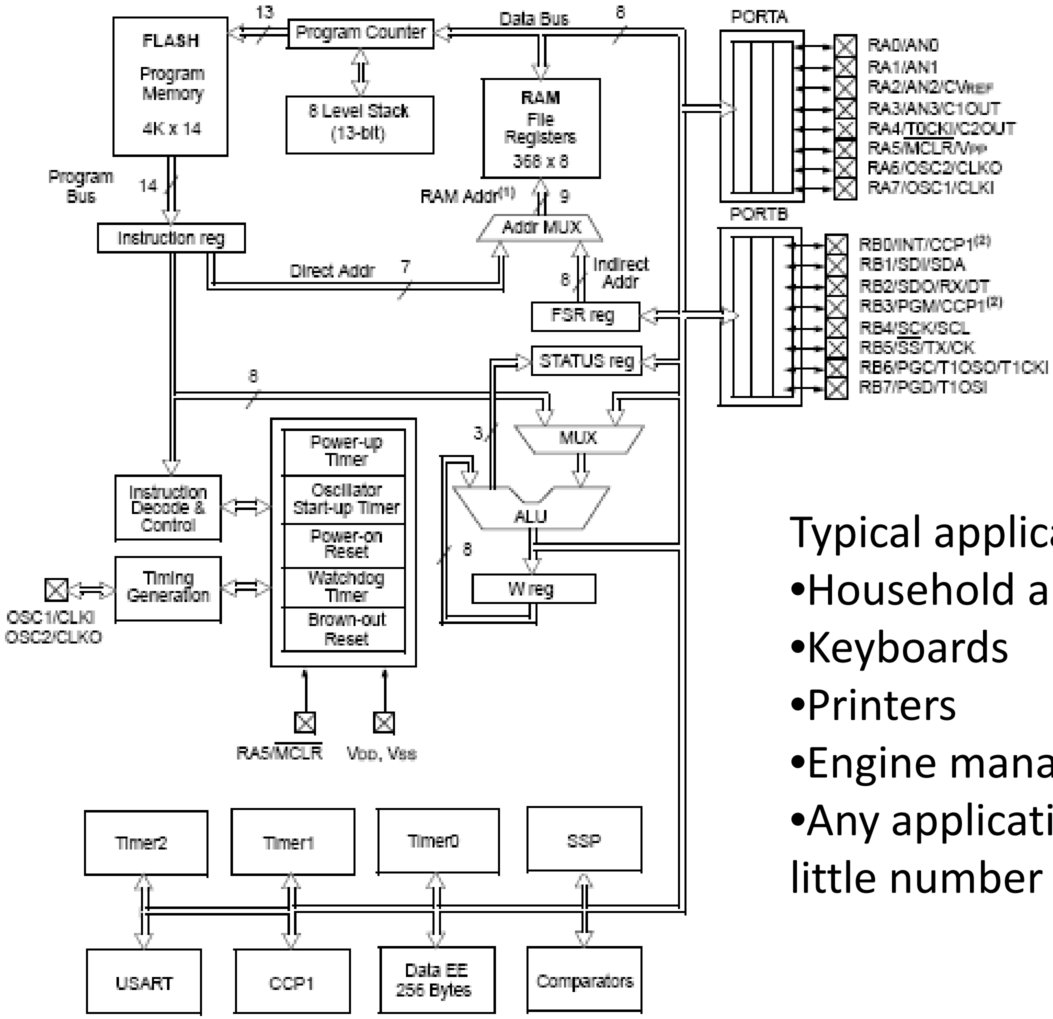
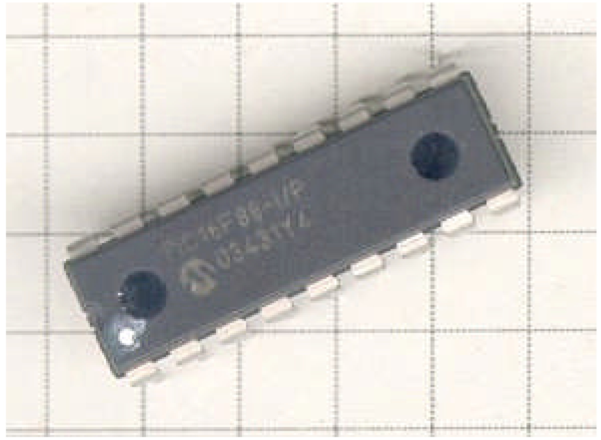


Image ex -wikepedia

Example PIC16F87 (used in Segway)





Typical applications of uControllers

- Household appliances (washing machines)
- Keyboards
- Printers
- Engine management systems
- Any application that is IO intensive but requires little number crunching

Digital Signal Processors (DSP)

- MAC (multiply and accumulate instruction) (recent PICS have MAC)
- Hardware support for looping
- Blindingly fast at common sig-processing operations
- Often not optimised for fast logic operations
- Texas Instruments have a very popular range of DSP's called the TMS320 series
- DSP's come in native integer and floating point varieties (in contrast with uController which are almost always just integer based)

Around $8 Billion market for DSPs in 2006

These chips and the algorithms they support are truly important!

•Mobile phones
•Digital TV boxes
•Satellite comms
•CD players/ MP3 players
•(Segway robots…)

The algorithms that support these applications are the domain of the information engineer.

Why is MAC so Important?

- In B4 and C4 (if you take it) you'll learn that the continuous transfer functions you are now familiar with (e.g C(s)) are in reality almost always implemented in discrete form on a computing device.
- If C(s) is a continuous function you'll soon learn how to map this to a discrete time controller C'(z) where z is the discrete time analogue of s
- The upshot of all of this is that time and time again we'll come across expressions like:

Constant "filter" coefficients

$$y_n = a_1 y_{n-1} + a_2 y_{n-2} + \ldots. + b_1 x_n + b_2 x_{n-1} + b_3 x_{n-2} \ldots..$$

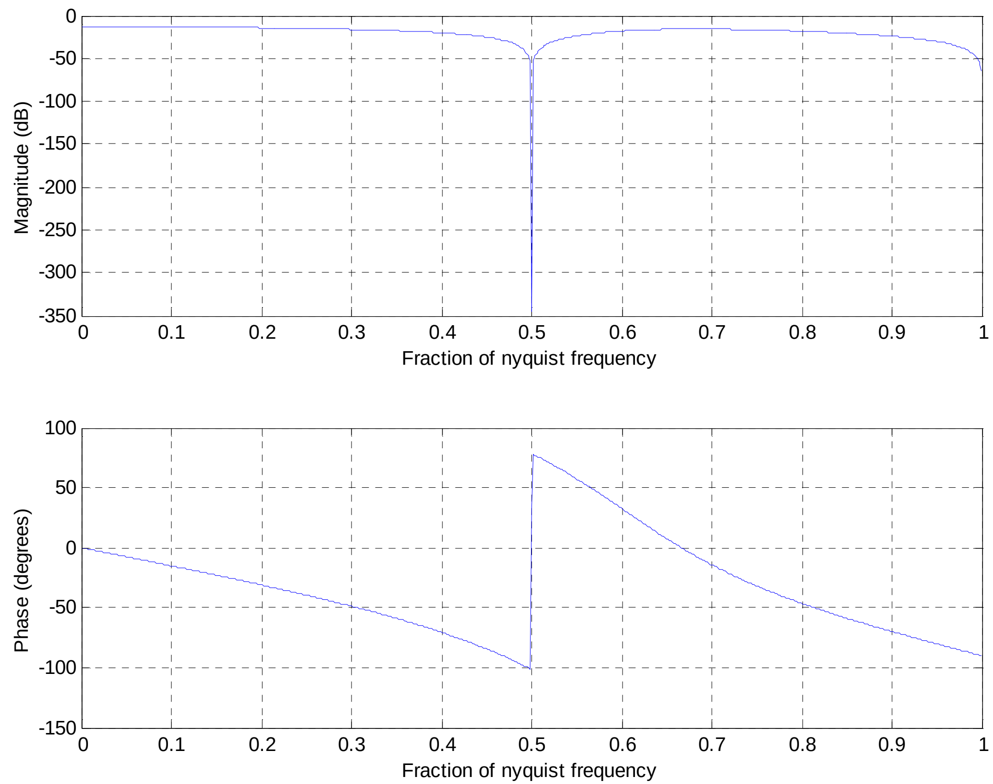
Output now | Previous outputs | Current input | Previous inputs

At each cycle a number of multiply and accumulates have to occur
Many DSP can implement the above in a single clock cycle….

Discrete Filter Design

- Digital filters are ubiquitous and many sophisticated tools exist to design fitlers with required frequency characterstics. For example a notch filter to remove contamination at a given frequency….

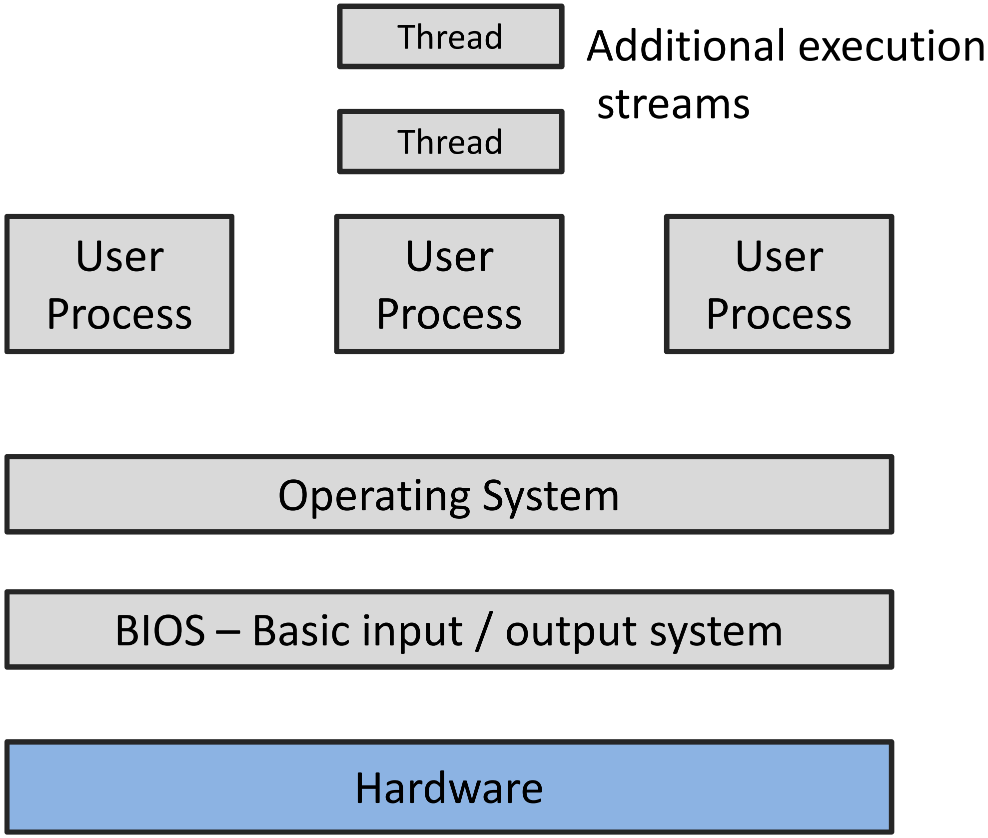
More on this later in B4



Micro Processors

- Close approximation to what you'll find in your PC – general purpose computation devices
- No onboard IO like serial ports A2D etc
- Often large word size
- Little speed optimised hardware although recent x86's have made in-roads
- Covered in A2

From Hardware to main()



Multiple Running programs written by users/3rd parties

Complicated program which abstracts hardware and provides process control

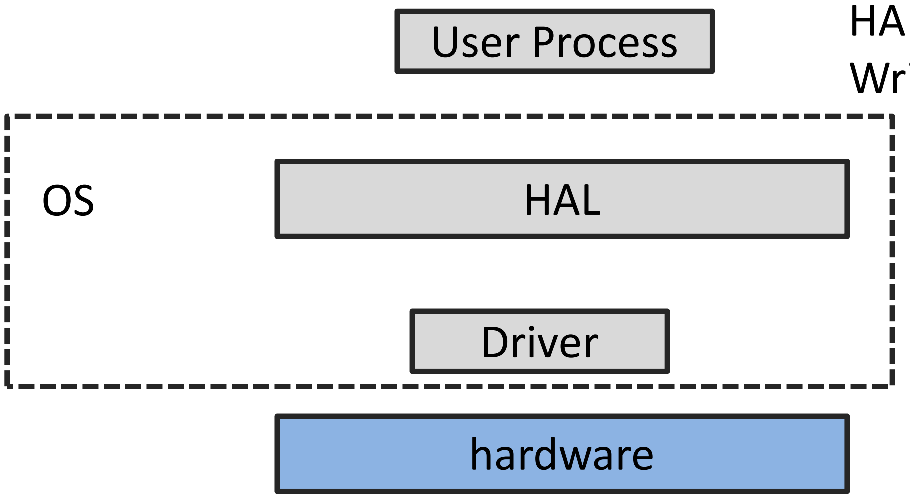
“firmware” which glues motherboard together, sets up interrupts and eventually loads OS

Hardisk, keyboard, graphics card

Play with WinXP Profiler….

The Role of the Operating System

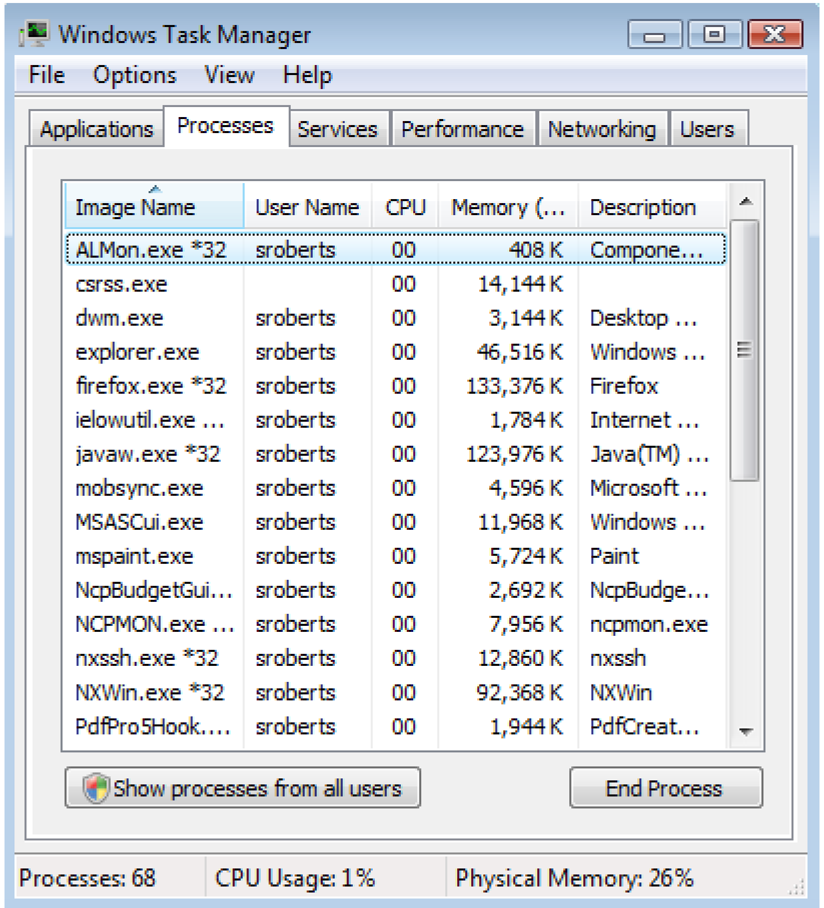
- Provides a hardware abstraction layer (HAL)
  - Provides an application programmers interface (API) for all kinds of hardware – e.g all keyboards look the same to programmers, all files on disk can be accessed in the same way (not a function of manufacturer)
  - Vendors of hardware write drivers which plug into one side of the HAL API and the writers of processes use the HAL API



HAL API is a set of function like WriteFile(), ReadFile, GetMouse()

Interlude: Units of Execution - Processes

- A process is a fundamental concept to computing.
- It represents a single instance of a running computer program – a sequence of serially executing instructions.
- A process is allocated memory which is not (generally) seen by other processes
- The times at which processes are run are scheduled by the operating system



Interlude: Units of Execution - Threads

- Threads are independent threads of execution within a single process.
- Thread scheduling by the OS gives the appearance of concurrent execution
- All threads within a given process can see (read and write) the same memory – that owned by the process.
- For example a process might have a user interface thread (drawing, handling button presses) a computation thread and a sensor IO thread.
- Operating systems provide system calls that start new threads from thread[0] (the thread started by the OS when a process is started.)

The Role of the Operating System

- Provides a mechanism for scheduling / interleaving the execution of processes
  - Gives the appearance of concurrent process execution on a serial processor
  - Manages the context switching between processes. (switching relevant data in and out of processor registers)
  - Running processes “see” uninterrupted execution and need not (usually) be written to yield execution to siblings.

The Role of the Operating System

- Provides Memory Management
  - Running processes can request allocation of memory at run time
  - The physical memory is abstracted away from running processes
  - Memory may be a combination of physical RAM and disk space

Handling Interrupts

- More often than not interrupts are intercepted by the OS and mapped to calls into a relevant device driver
- For example a UART may raise an interrupt when its Rx buffer is 50% full:
  - The interrupt calls a function in the serial port driver.
  - The driver extracts data from the hardware and places it a software buffer(array) provided by the OS
  - Processes granted access to the serial port read from this abstracted serial port when “reading” from the serial port.

Interprocess Communication

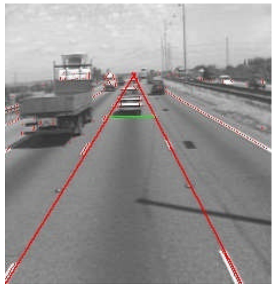


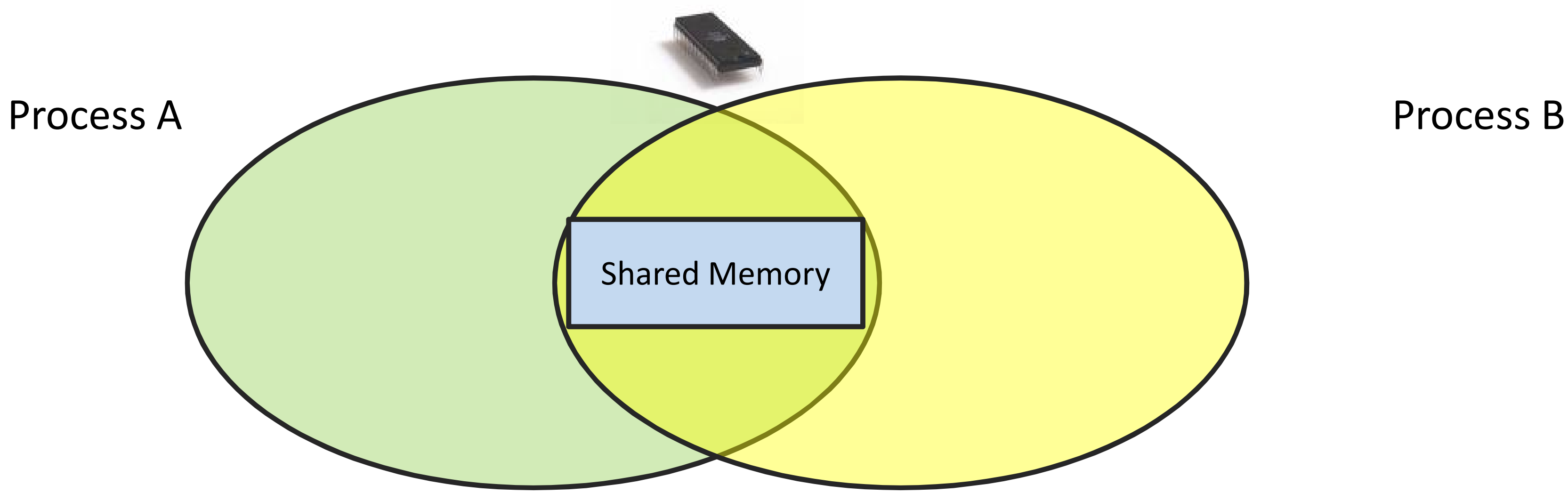
Image Grabbing → Communication mechanism → Image Understanding

Producer

Consumer

We need to consider how data could be shared between producer and consumer

Shared Memory



•Processes can make special system calls to the operating system which return a chunk of memory that can be shared between processes.

•The OS also provides a mechanism by which a process can ask to have already allocated shared memory inserted into its own address space (Process B needs to be able to ask to see the shared memory segment already created by Process A)

Q: What happens if the producer writes as the consumer tries to read?

The need for Synchronisation

write

Shared Memory

read

write

The consumer was reading as the producer executed the second write. The result is corrupted data.

We need someway to synchronise the processes to protect resources

Binary Semaphores

```
Init(Semaphore S)
{
 s = 1
}
```

Initialise a semaphore (which is a signal between threads or processes) to the number of times a protected resource can be shared (1)

```
P(Semaphore S)
{
 WaitFor(s>0);
 s=s-1;
}
```

Call when access to the resource is required.This blocks (halts) execution until completion.When s>0 is detected next line must complete before thread is rescheduled – it must be "atomic" (functionality provided by OS)

```
V(Semaphore S)
{
     s=s+1;
}
```

Call when finished with resource

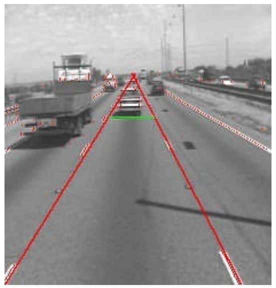
Binary Semaphore Example

Process A (Producer)


```
while(1)
{
  I = CaptureImage()
  P(s) //wait for access
     WriteImage(I)
  V(s)//release
}
```

Read and writes are to shared memory

Process B (Consumer)

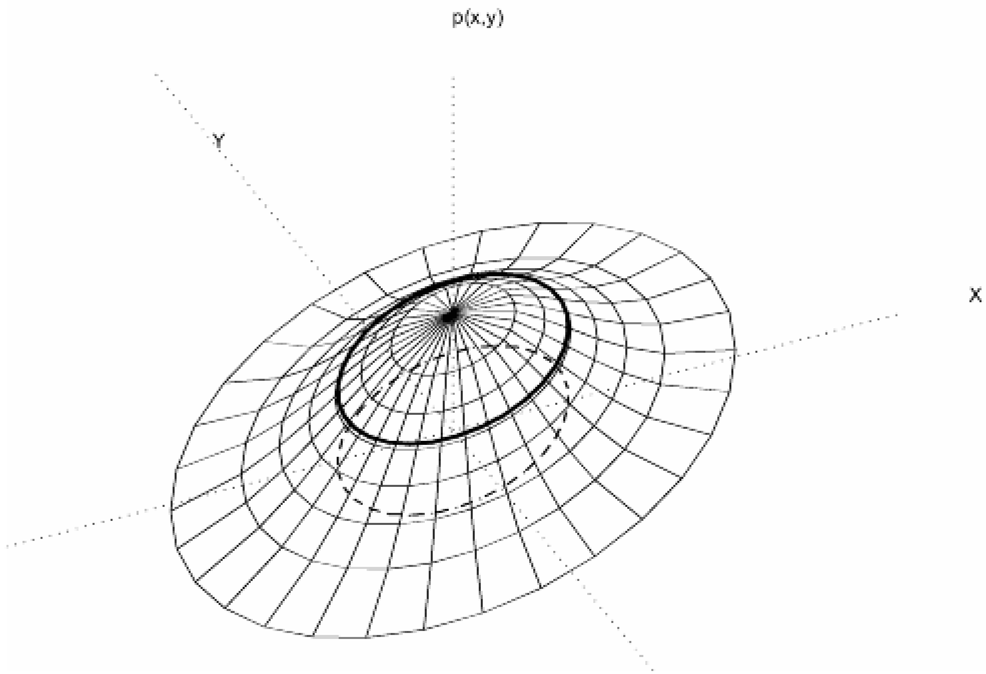


```
while(1)
{
  P(s) //wait for access
     I = ReadImage()
  V(s)//release
  ProcessImage(I)
}
```

•Between P & S privacy is guaranteed.
•OS needs to provide Semaphore functionality and a mechanism to allow both processes to share the semaphore S

Lecture III The role of probability Theory

- Sensor models
- The Role of Bayes' rule
  - Recursive estimation
  - Tracking
  - Plant models
  - Filtering



Revision of Probability

Product Rule $p(a,b) = p(a|b)p(b)$ $\quad p(a,b,c) = p(a|b,c)\underbrace{p(b|c)p(b)}_{p(b,c)}$

$$p(a|b) = \frac{p(a,b)}{p(b)}$$

Marginalisation

$$p(a) = \int_{-\infty}^{\infty} p(a,b)db = \int_{-\infty}^{\infty} p(a|b)p(b)db$$

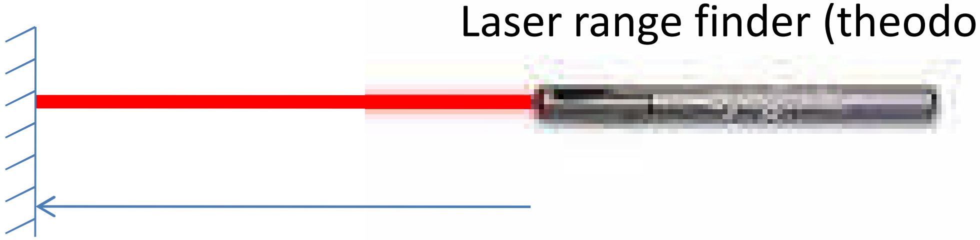
If a & b are continuous

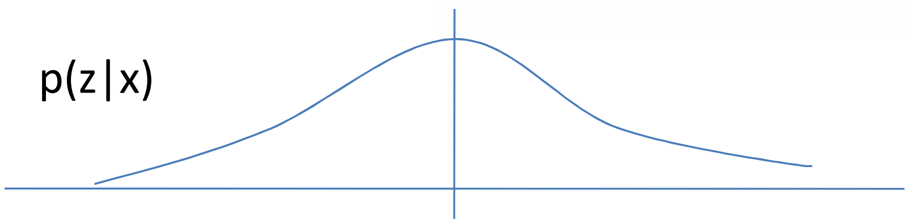
$p(a) = \sum_{s\in\mathcal{B}} \mathrm{p(a,b{=}s)}$ If a & b are discrete

If you can remember and use these two rules then so much is within your reach….(including exams!)

Probabilistic Models

- We can think of sensor measurements, z, as samples from a conditional distribution (conditioned on the state of the world, x)





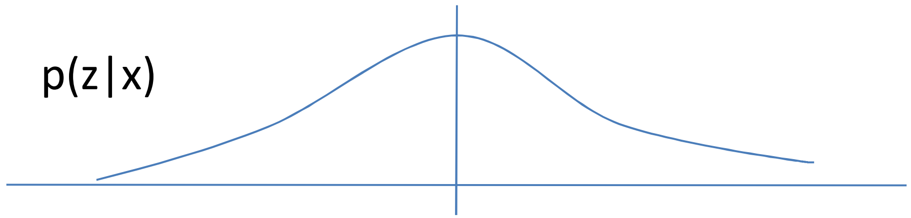
Sensor Models Cont – Gaussian Noise

$$z = x + n$$

$$n \sim \mathcal{N}(0, \sigma^2)$$

$$z \sim \mathcal{N}(x, \sigma^2)$$

$$p(z|x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{(z-x)^2}{2\sigma^2}$$

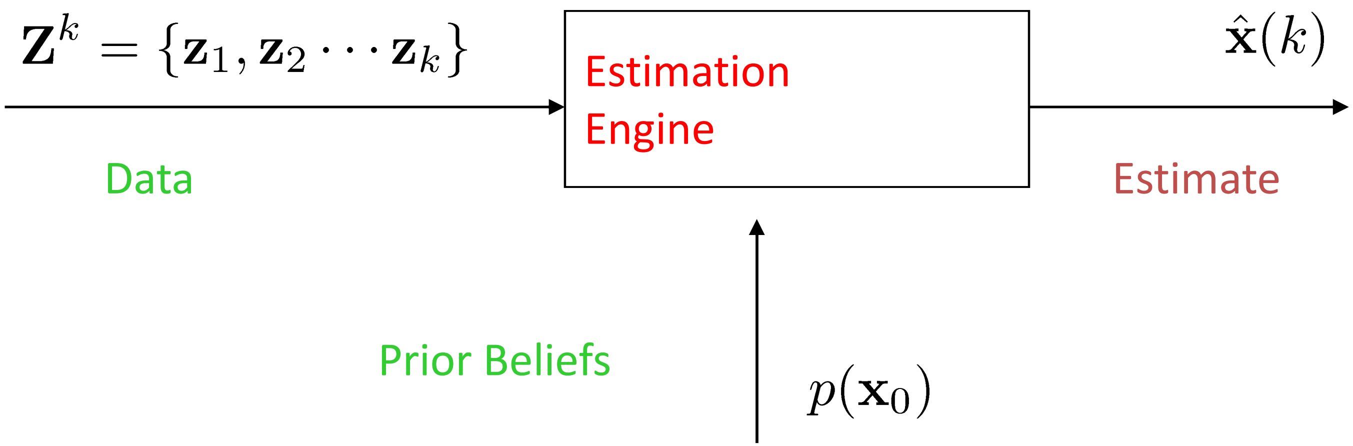


Here we have elected to model noise as samples from a Gaussian which is a very common practice

p(z|x) explains the measurement in terms of the underlying state

Estimating x from p(z|x)

"Estimation is the process by which we infer the value of a quantity of interest, $\mathbf{x}$*, by processing data that is in some way dependent on* $\mathbf{x}$ *."*



Maximum Likelihood

$$\mathcal{L} \triangleq p(\mathbf{z}|\mathbf{x}) \qquad\qquad p(\mathbf{z}|\mathbf{x}) = \frac{1}{C} e^{-\frac{1}{2}(\mathbf{z}-\mathbf{x})^T \mathbf{P}^{-1} (\mathbf{z}-\mathbf{x})}$$

N.B Multivariate Gaussian Understood?

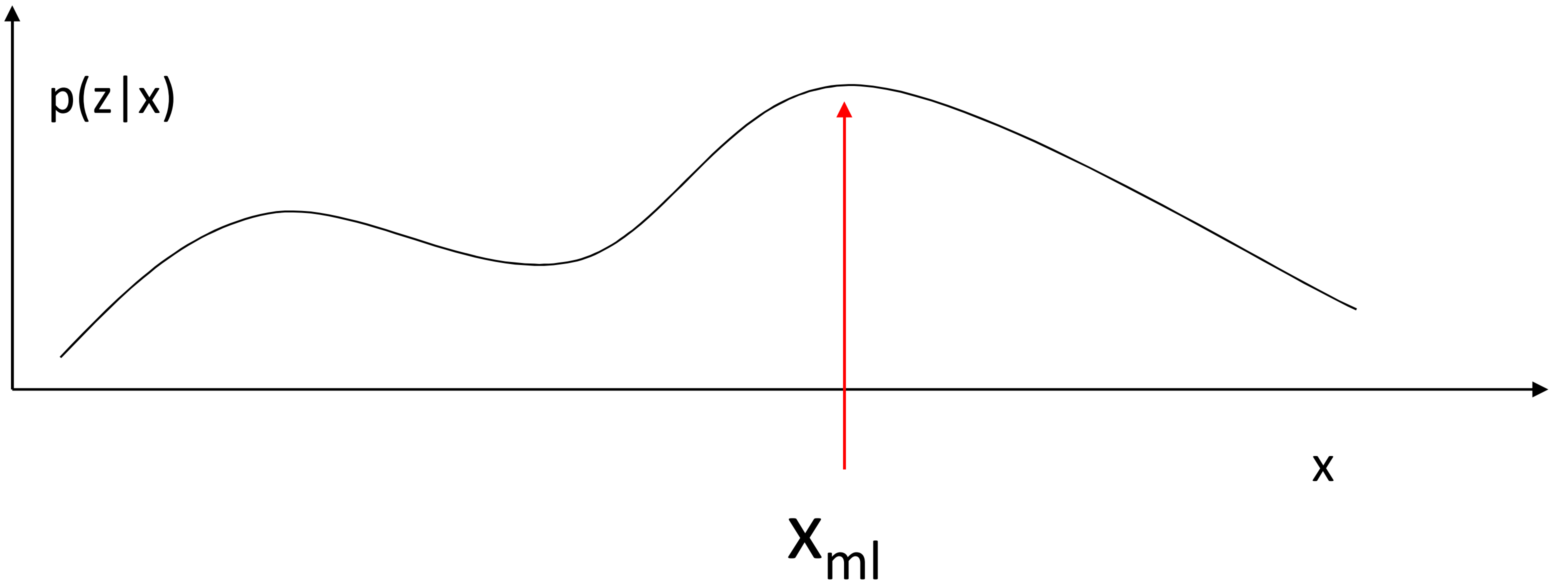
Given an observation $\mathbf{z}$ and a likelihood function $p(\mathbf{z}|\mathbf{x})$, the **maximum likelihood estimator - ML** finds the value of $\mathbf{x}$ which maximises the likelihood function $\mathcal{L} \triangleq p(\mathbf{z}|\mathbf{x})$.

$$\hat{\mathbf{x}}_{m.l} = \arg\max_{\mathbf{x}} p(\mathbf{z}|\mathbf{x}) \tag{1}$$

Find a value of x(state) that best explains z (data)

Maximum Likelihood

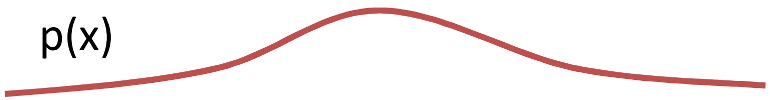
We are given a value for z and view $p(z|x) = f(x,z)$ as a function of x



ML does not incorporate prior knowledge

Incorporating Prior Knowledge

What if we knew something abut the state of the world before we took the measurement – could we incorporate that information?

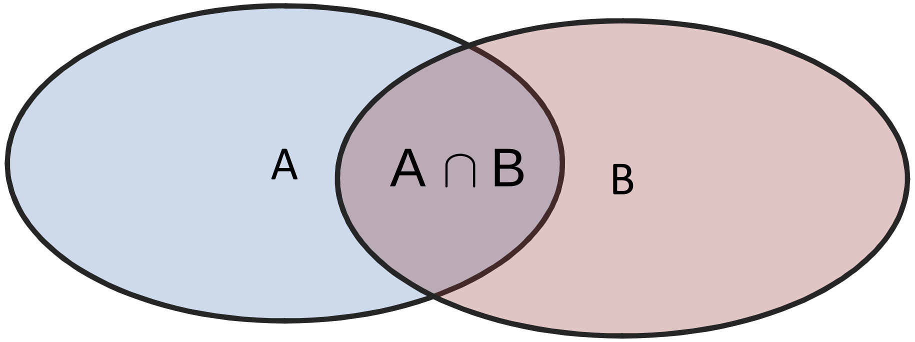


We can use a probability distribution over x to capture our prior belief in the value of x

So how can we combine p(z|x) and p(x) to yield p(x|z) ?

Bayes' Rule

A "joint distribution"



$$p(a|b) = \frac{p(a,b)}{p(b)}$$

A "conditional distribution"

A "marginal distribution"

$$\begin{aligned} p(a,b) &= p(a|b)p(b) \\ &= p(b|a)p(a) \end{aligned}$$

$$p(b|a)p(a) = p(a|b)p(b)$$

$$\boxed{p(b|a) = \frac{p(a|b)p(b)}{p(a)}}$$

Should we bother with Bayes?

$$p(b|a) = \frac{p(a|b)p(b)}{p(a)}$$

Yes, you should be. Bayes' rule lies at the very heart of swathes of information engineering:

- Medical imaging
- Tracking
- Estimation
- Sensor processing signal recovery
- Machine learning

Bayes' Rule lets you invert conditionals expressing p(a|b) in terms of p(b|a)

Consider Our Laser Example

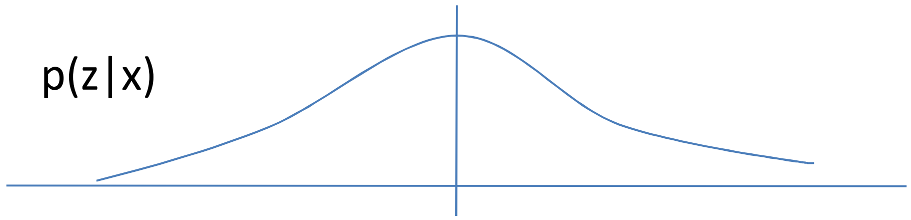
$$z = x + n$$

$$n \sim \mathcal{N}(0, \sigma_z^2)$$

$$z \sim \mathcal{N}(x, \sigma_z^2)$$

$$p(z|x) = \frac{1}{\sqrt{2\pi\sigma_z^2}} \exp -\frac{(z-x)^2}{2\sigma_z^2}$$

$$p(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp -\frac{(x-\mu)^2}{2\sigma_x^2}$$



Apply Bayes' Rule

$$p(\mathbf{x}) = C_1 \exp\{-\frac{(\mathbf{x}-\mu)^2}{2\sigma_x^2}\}$$

$$p(\mathbf{z}|\mathbf{x}) = C_2 \exp\{-\frac{(\mathbf{z}-\mathbf{x})^2}{2\sigma_z^2}\}$$

$$\begin{aligned} p(\mathbf{x}|\mathbf{z}) &= \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z})} \\ &= C(\mathbf{z}) \times p(\mathbf{z}|\mathbf{x}) \times p(\mathbf{x}) \\ &= C(\mathbf{z}) \underbrace{\exp\{-\frac{(\mathbf{x}-\mu)^2}{2\sigma_x^2} - \frac{(\mathbf{z}-\mathbf{x})^2}{2\sigma_z^2}\}}_{\text{Maximise this}} \end{aligned}$$

The x which maximises p(x|z) is called the "maximum *a posteriori*" estimate

Maximum A Posteriori Estimation

$$\underbrace{p(\mathbf{x}|\mathbf{z})}_{\text{posterior}} = \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z})}$$

Note denominator is not
a function of x – it is a normaliser

$$\propto \underbrace{p(\mathbf{z}|\mathbf{x})}_{\text{Likelihood}} \times \underbrace{p(\mathbf{x})}_{\text{prior}}$$

Given an observation $\mathbf{z}$,a likelihood function $p(\mathbf{z}|\mathbf{x})$ and a prior distribution on $\mathbf{x}$, $p(\mathbf{x})$, the **maximum a posteriori estimator - MAP** finds the value of $\mathbf{x}$ which maximises the posterior distribution $p(\mathbf{x}|\mathbf{z})$

$$\hat{\mathbf{x}}_{map} = \arg\max_{\mathbf{x}} p(\mathbf{z}|\mathbf{x})p(\mathbf{x}) \tag{1}$$

M.A.P does incorporate prior knowledge

Example Cont...

$$\underbrace{\exp\{-\frac{(\mathbf{x}-\mu)^2}{2\sigma_x^2} - \frac{(\mathbf{z}-\mathbf{x})^2}{2\sigma_z^2}\}}_{\text{Maximised when:}}$$

Mean

$$\frac{(x-\alpha)^2}{\beta^2} = -\frac{(\mathbf{x}-\mu)^2}{2\sigma_x^2} - \frac{(\mathbf{z}-\mathbf{x})^2}{2\sigma_z^2} = 0$$

Variance

$$\alpha = \frac{\sigma_z^2\mu + \sigma_p^2\mathbf{z}}{\sigma_z^2 + \sigma_x^2} \qquad \beta^2 = \frac{\sigma_z^2\sigma_x^2}{\sigma_z^2 + \sigma_x^2}$$

$$\boxed{\text{note that } \beta < \sigma_x}$$

How does the mean change?

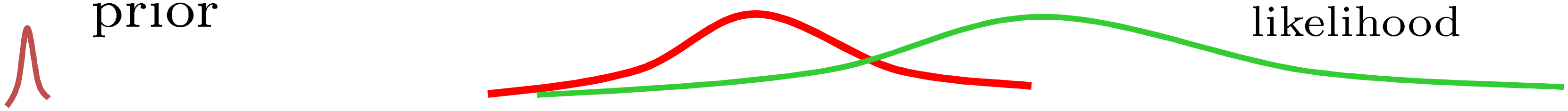
$$\hat{\mathbf{x}}_{map} = \mu + \frac{\sigma_x^2}{\sigma_x^2 + \sigma_z^2} \times (\mathbf{z} - \mu)$$

Old (prior) mean

Difference between measurement and prior

Visually...

$$\underbrace{p(\mathbf{x})}_{\text{prior}} \qquad \overbrace{p(\mathbf{x}|\mathbf{z})}^{\text{posterior (estimate)}} \qquad \underbrace{p(\mathbf{z}|\mathbf{x})}_{\text{likelihood}}$$



$$\hat{\mathbf{x}}_{map} = \mu_p + \frac{\sigma_x^2}{\sigma_x^2 + \sigma_z^2} \times (\mathbf{z} - \mu) \qquad \sigma_{map}^2 = \frac{\sigma_z^2 \sigma_x^2}{\sigma_z^2 + \sigma_x^2}$$

Remember - the variance of posterior is smaller than the prior – why? because the measurement adds information. This notion will be formalised later in the course

Discrete Time Recursive Bayesian Estimation

Subscript is time

$$\mathbf{Z}^k = \{\mathbf{z}_1, \mathbf{z}_2 \cdots \mathbf{z}_k\}$$ Sequence of data (measurements)

We want the conditional distribution $p(x|Z^k)$

State at time k (think position)

Sequence of measurements up until time k (think - list of ranges)

Question: Can we iteratively calculate this – ie every time a new measurement comes in update our estimate? – (Answer :yes, see next slide)

$$p(x|Z^k) = f\left[p(x|Z^{k-1}), p(z_k|x)\right]$$

posterior

prior

measurement

We are looking for a distribution over state at time k given all measurements up until time k

Recursive (online) Bayes'

$$p(x|Z^k) = \frac{p(Z^k|x)p(x)}{p(Z^k)}$$

Bayes' Rule

write $p(Z^k)$ as $p(z_k, Z^{k-1})$

$$= \frac{p(z_k, Z^{k-1}|x)p(x)}{p(z, Z^{k-1})}$$

use product rule to expand $p(z_k, Z^{k-1}|x)$

$$= \frac{p(Z^{k-1}|x, z)p(z|x)p(x)}{p(z, Z^{k-1})}$$

note that first $k-1$ measurements are independent of k^{th}

$$= \frac{p(Z^{k-1}|x)p(z|x)p(x)}{p(z, Z^{k-1})}$$

collapse $p(Z^{k-1}|x)p(x)$ into $p(x, Z^{k-1})$

$$= \frac{p(Z^{k-1}, x)p(z|x)}{p(z, Z^{k-1})}$$

divide by $p(Z^{k-1})$

$$= \frac{p(x|Z^{k-1})p(z|x)}{p(z|Z^{k-1})}$$

Key Result

$$p(x|Z^k) = \frac{p(z_k|x)p(x|Z^{k-1})}{p(z_k|Z^{k-1})}$$

$$\underbrace{p(x|Z^k)}_{\text{Estimate}} \propto \underbrace{p(z_k|x)}_{\text{Likelihood}} \underbrace{p(x|Z^{k-1})}_{\text{Last Estimate}}$$

At time k

Explains data at time k
as function of x at time k

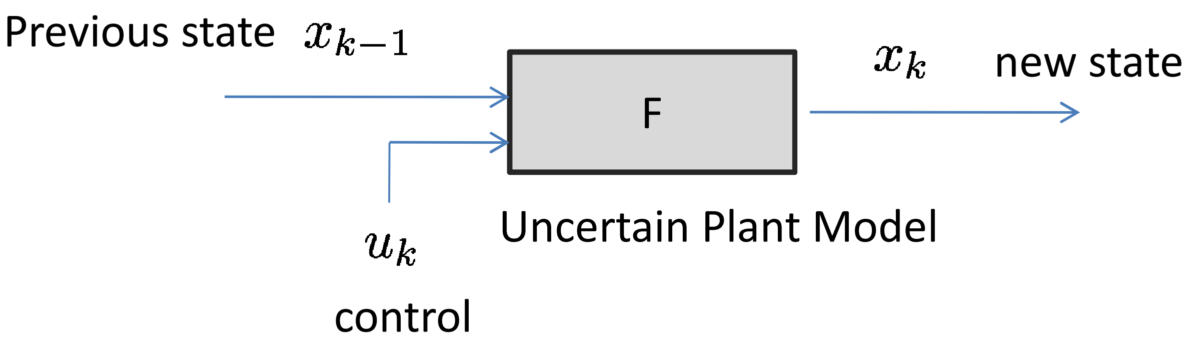
At time k-1

Incorporating Plant Models

We should have used a k subscript on x to indicate that we are referring to x at time k

$$p(x_k|Z^k) = \frac{p(z_k|x_k)p(x_k|Z^{k-1})}{p(z_k|Z^{k-1})}$$

Now the last term on the numerator looks like a prediction.



Incorporating Plant Uncertainty

$$
\begin{aligned}
p(x_k|Z^{k-1}, u_k) &= \int_{-\infty}^{\infty} p(x_k, x_{k-1}|Z^{k-1}, u_k) dx_{k-1} \\
&= \int_{-\infty}^{\infty} p(x_k|x_{k-1}, Z^{k-1}, u_k) p(x_{k-1}|u_k, Z^{k-1}) dx_{k-1} \\
&= \int_{-\infty}^{\infty} \underbrace{p(x_k|x_{k-1}, u_k)}_{\text{Probabilistic plant model}} \underbrace{p(x_{k-1}|Z^{k-1})}_{\text{Last estimate}} dx_{k-1}
\end{aligned}
$$

Here we have used the assumptions

- that given the state at k-1 and control at time k, the state at time k is independent of the observations
- The state at time k-1 is independent of the control at time k (which is in the future)

Applications

- The previous few slides have indicated the existence of a probabilistic framework which can handle uncertainty in measurements and plant models

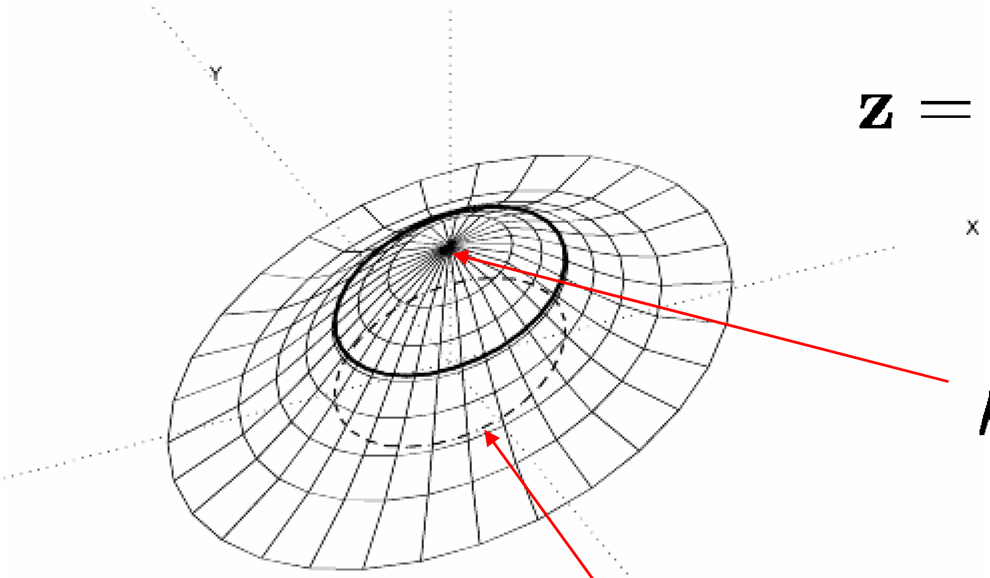
- Note that at no point were we restricted by the form of the p.d.f's or what the physical interpretation of x,z or u might be.
    - x: rate of inflation, z: the price of a car, u: intervention from the world bank
    - x: strain on a beam, z: measured voltage
    - x: car velocity, z police radar time of flight
    - x: sheet metal thickness, z:X-Ray energy, u: roller pressure
    - x: tumour state, z: PET scan, u:motion of patients head during scan

Probabilistic methods are a natural way to handle uncertainty in measurement and state evolution. The techniques they give rise find application across all domains of engineering

The Role of the Gaussian

It is common to find that the functional form of the pdfs in the previous slides is that of a Gaussian. Of course we may have distribution over a vector (for example position and velocity). In which case we shall be dealing with the multidimensional Normal distribution.

$$p(\mathbf{z}) = \frac{1}{(2\pi)^{\dim(z)/2} \mid \Sigma \mid^{1/2}} \exp\{-\frac{1}{2}(\mathbf{z}-\mu)^T \Sigma^{-1} (\mathbf{z}-\mu)\}$$



$$\mathbf{z} = \begin{bmatrix} x \\ y \end{bmatrix}$$

A “<u>joint</u>” distribution over x and y

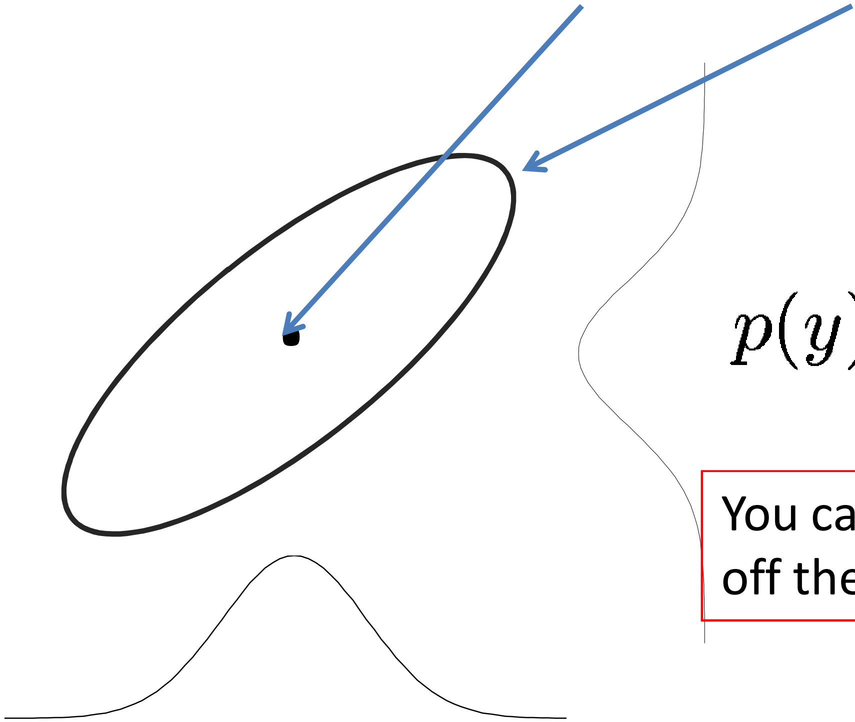
$$\mu = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}$$

Covariance $\mathbf{z}^T \Sigma^{-1} \mathbf{z} = 1$

The structure of Σ

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mu, \Sigma)$$

$$= \mathcal{N}(\begin{bmatrix} x \\ y \end{bmatrix} | \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 \end{bmatrix})$$



$$p(y) = \mathcal{N}(y|\mu_y, \sigma_{yy}^2)$$

You can read the marginal distribution variance off the diagonals of the covariance of the joint

$$p(x) = \mathcal{N}(x|\mu_x, \sigma_{xx}^2)$$

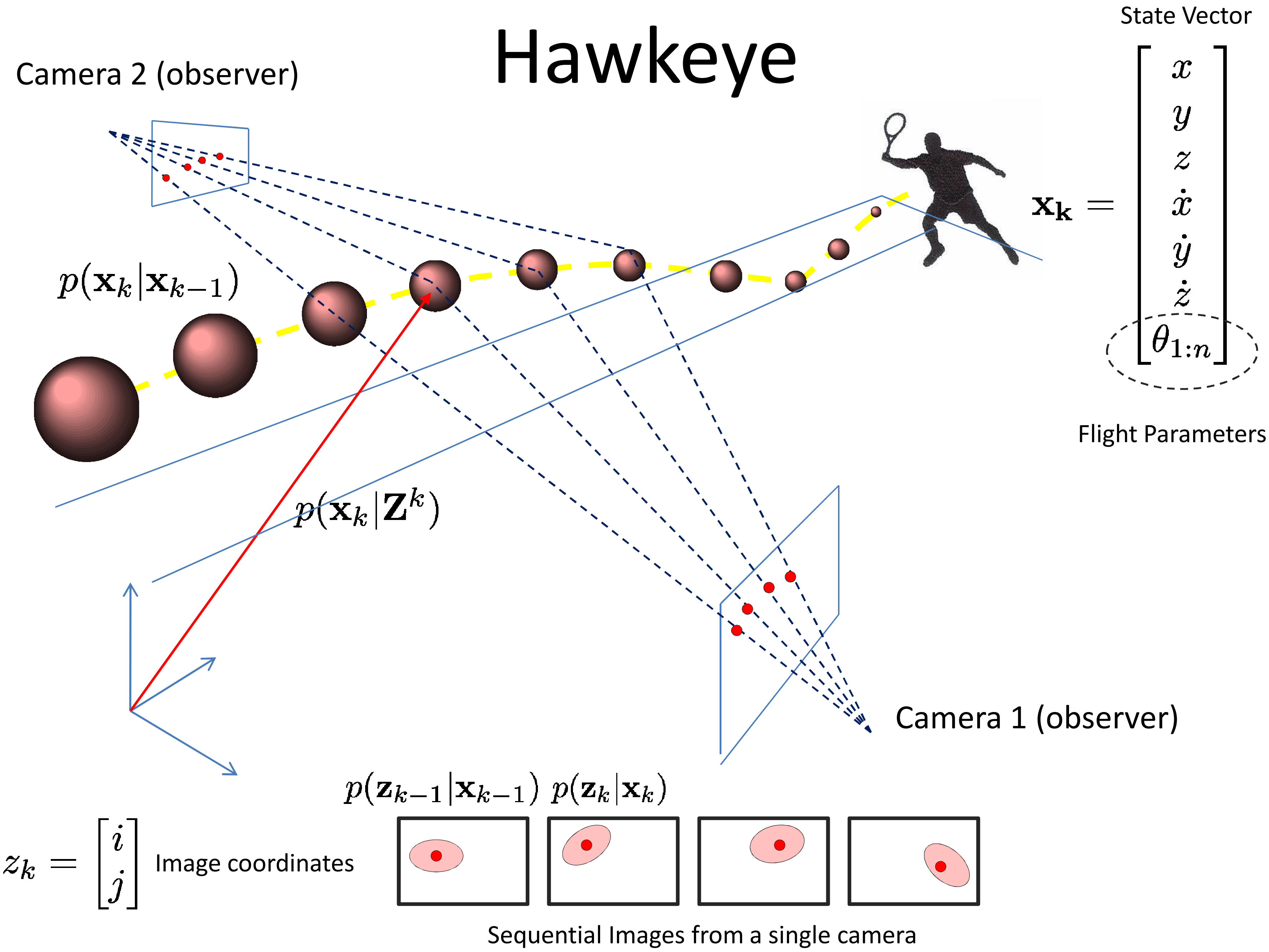
The Gaussian is a common functional form

If Gaussians are used in the pdfs of the recursive Bayes formulation and in the equations we derived for propagating plant uncertainty one ends up with something called a Kalman Filter (covered in detail in C4)

The Kalman filter is a very common tool in estimation applications. For example

- in car navigation systems
- Hawkeye
- Economic models.
- Hospital delivery systems
- Port automation

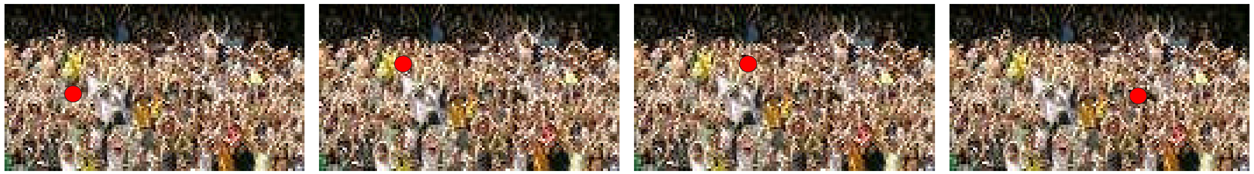
Hawkeye



Camera 2 (observer)
State Vector
$\mathbf{x_k} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \theta_{1:n} \end{bmatrix}$
Flight Parameters
$p(\mathbf{x}_k | \mathbf{x}_{k-1})$
$p(\mathbf{x}_k | \mathbf{Z}^k)$
Camera 1 (observer)
$p(\mathbf{z}_{k-1} | \mathbf{x}_{k-1})$ $p(\mathbf{z}_k | \mathbf{x}_k)$
$z_k = \begin{bmatrix} i \\ j \end{bmatrix}$ Image coordinates
Sequential Images from a single camera

Extracting the Observations…

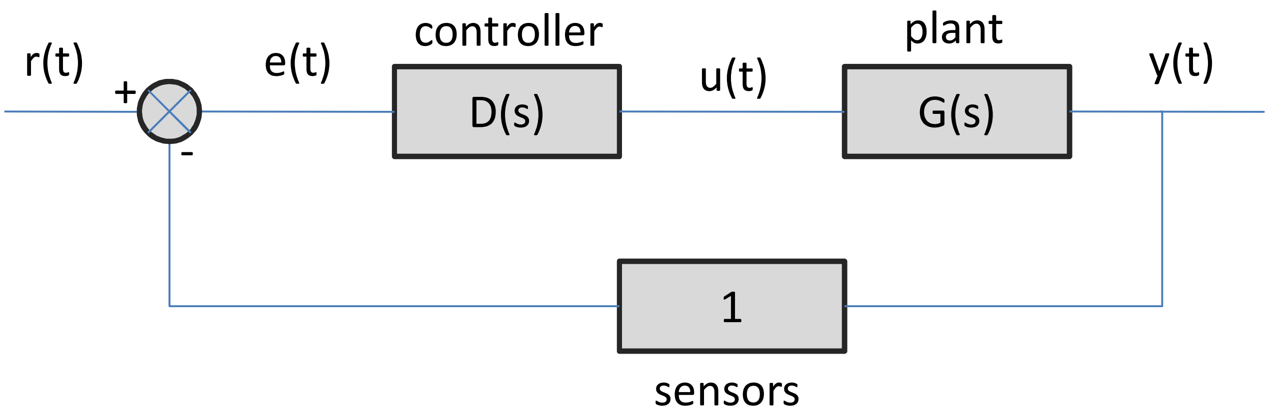
Is a hard information engineering problem in itself!



Again it turns out that solutions to this problem are underpinned by probability theory! More of this kind of problem (and solutions) in B4 and C4

Lecture IV Computer Based Feedback Control and Actuation

- You are already familiar with continuous time control systems

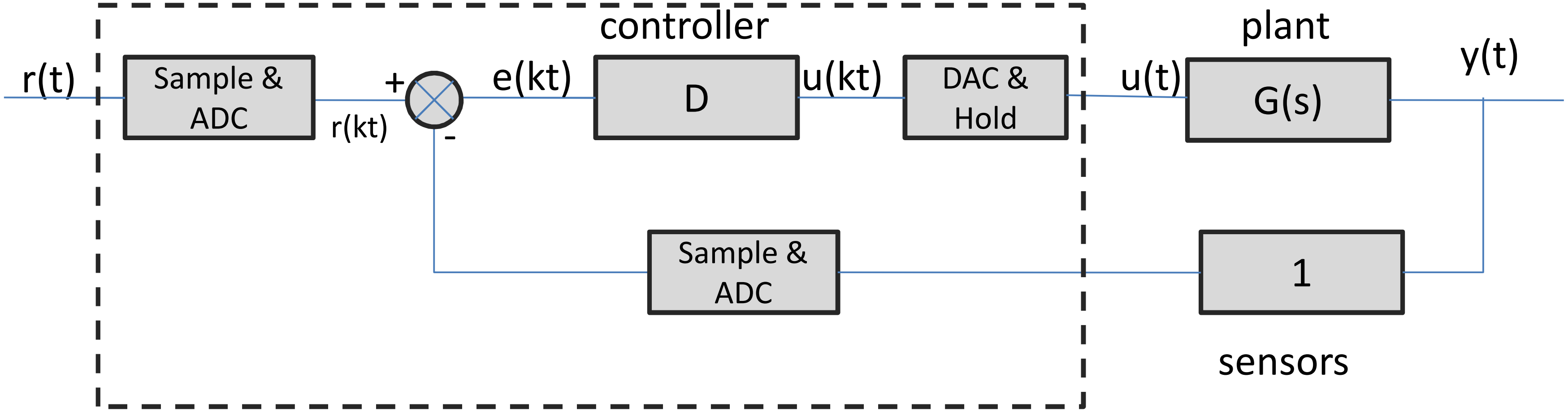


$$Y(s) = D(s)G(s)(R(s) - Y(s))$$

$$Y(s) = \frac{D(s)G(s)}{1 + D(s)G(s)} R(s)$$

In Practice: Using A Digital Computer

- We implement the controller in software running on a digital conputer
- We need to convert twixt digital and analog…

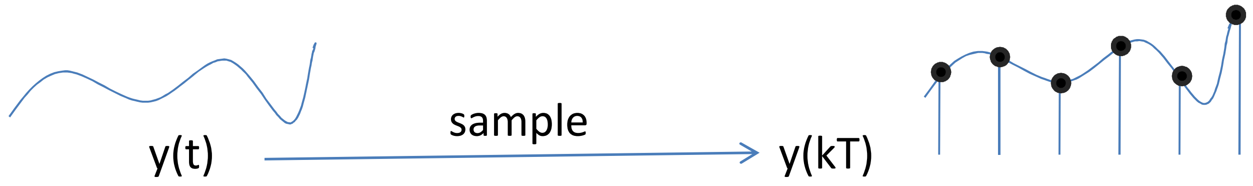


Digital, sampled system

- The signals e(kT) u(kT) and r(kT) are "discrete"
- T is the sample period,k is an integer
- A discrete signal is constant over the sample period

Notation: for a discrete signal y, constant T: $y(kT) = y(k) = y_k$

Discrete Signals



- The quantity y(kT) is a discrete sampled signal
- T is the sample period (assumed constant)
- k is an integer
- A discrete signal is representative of the continuous signal over the sample period
- Think of y(kT) as a number. Its precision is dictated by the precision of the sampling hardware (e.g 8 or 16 bits)

How do we implement a discrete time controller?

Imagine we have been given the desired controller transfer function D(s) , how might we construct a discrete time version?

Writing D(s) as a quotient of two polynomials in s

$$D(s) = \frac{B(s)}{A(s)} \quad Y(s) = D(s)E(s)$$

$$A(s)Y(s) = B(s)E(s)$$

$$Y(s)\sum_{i=0}^{n} a_i s^i = E(s)\sum_{i=0}^{m} b_i s^i$$

$$a_n \frac{d^n y}{dy^n} + a_{n-1}\frac{d^{n-1}y}{dy^{n-1}} + \ldots + a_0 y = b_m \frac{d^m e}{de^m} + b_{m-1}\frac{d^{m-1}e}{de^{m-1}} + \ldots + b_0 e$$

$$a_n \frac{d^n y}{dy^n} + a_{n-1}\frac{d^{n-1}y}{dy^{n-1}} + \ldots + y = b_m \frac{d^m e}{de^m} + b_{m-1}\frac{d^{m-1}e}{de^{m-1}} + \ldots + b_0 e$$

Last step renormalises by dividing both sides by a_o

Approximating the derivative operator

- You know from A1 that

$$\left.\frac{dx}{dt}\right|_{k+1} \approx \frac{x_{k+1}-x_k}{T}$$

$$\left.\frac{d^2x}{dt^2}\right|_{k+1} \approx \frac{x_{k+1}-2x_k+x_{k-1}}{2T}$$

- So we could substitute these discrete derivative approximations into

$$a_n\frac{d^ny}{dy^n}+a_{n-1}\frac{d^{n-1}y}{dy^{n-1}}+\ldots+y=b_m\frac{d^me}{de^m}+b_{m-1}\frac{d^{m-1}e}{de^{m-1}}+\ldots+b_0e$$

Example

$$D(s) = K\frac{s+a}{s+b} \qquad U(s) = D(s)E(s)$$

$$\frac{du(t)}{dt} + bu(t) = K\frac{de(t)}{dt} + Kae(t)$$

$$\frac{u_{k+1} - u_k}{T} + bu_k = K\frac{e_{k+1} - e_k}{T} + Kae(t)$$

$$u_{k+1} = \alpha_1 u_k + \beta_1 e_{k+1} + \beta_0 e_k$$

$$\boxed{u_k = \alpha_1 u_{k-1} + \beta_1 e_k + \beta_0 e_{k-1}}$$

where

$$\alpha_1 = (1 - bT)$$
$$\beta_1 = K$$
$$\beta_0 = K(aT - 1)$$

control at time k is function of previous control and previous and current input

Example Continued

$$u_k = \alpha_1 u_{k-1} + \beta_1 e_k + \beta_0 e_{k-1}$$

Note how easy this is to implement.
In general requires variables to be stored across iterations

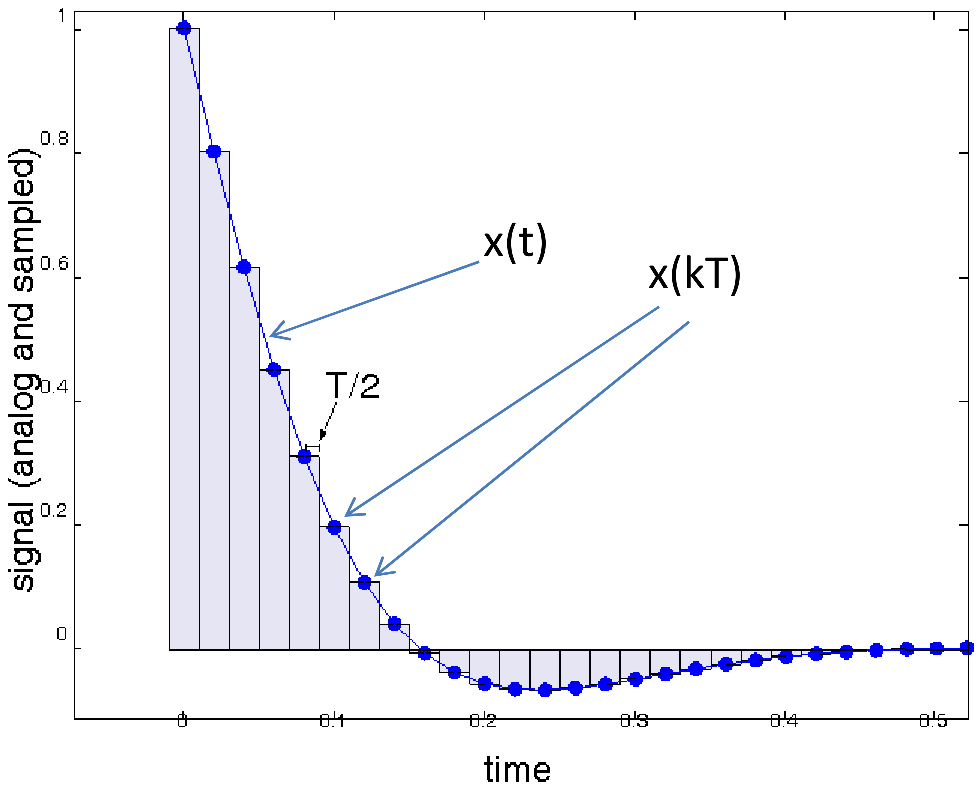
where

$$\alpha_1 = (1 - bT)$$
$$\beta_1 = K$$
$$\beta_0 = K(aT - 1)$$

Note how the constants are dependent on sample time. If we keep sample time constant computation is simplified even further.

Are we free to choose any T (even if we keep it constant)?
We can imagine that the answer is no – why?

The Effect of Sample and Hold

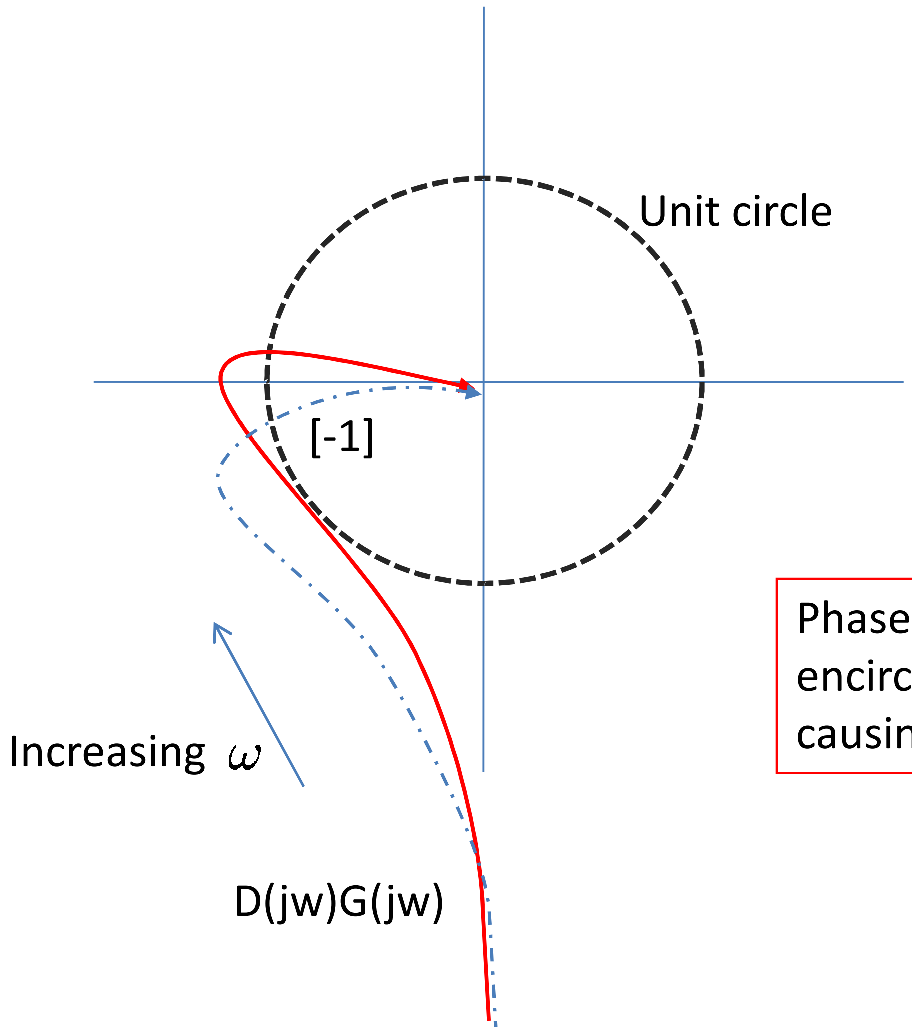


Zero order hold introduces a lag of T/2

$$\mathcal{F}\left[x(t-\tfrac{T}{2})\right] \rightarrow e^{-j\omega\frac{T}{2}}X(\omega)$$

$\boxed{\text{Or equivalently a phase lag of } \frac{\omega T}{2}}$

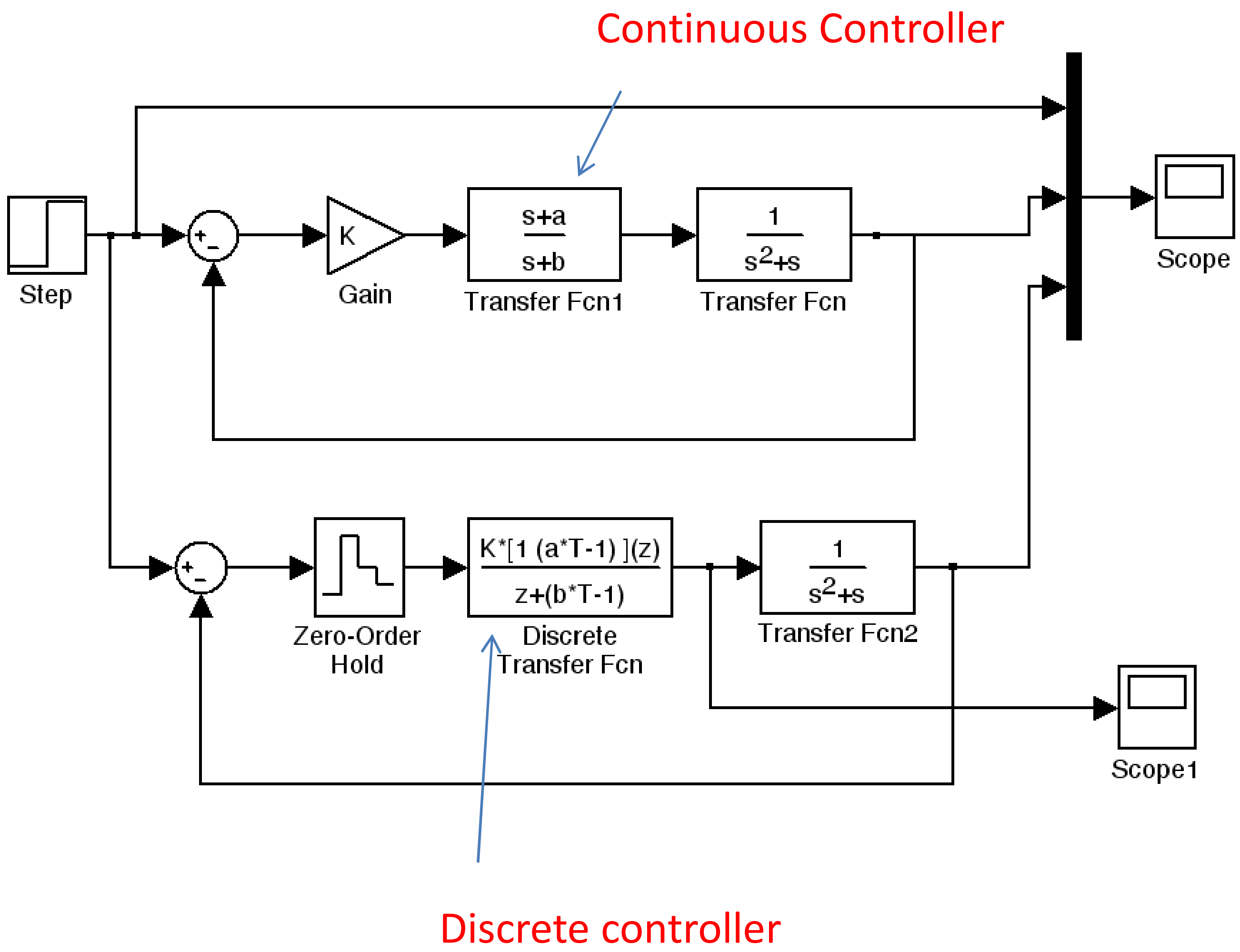
Impact of sample on Closed Loop Stability



$$\mathcal{F}\left[x(t-\tfrac{T}{2})\right] \rightarrow e^{-j\omega\frac{T}{2}}X(\omega)$$

Phase lag can rotate Nyquist diagram to encircle [-1,0]
causing instability (from A3)

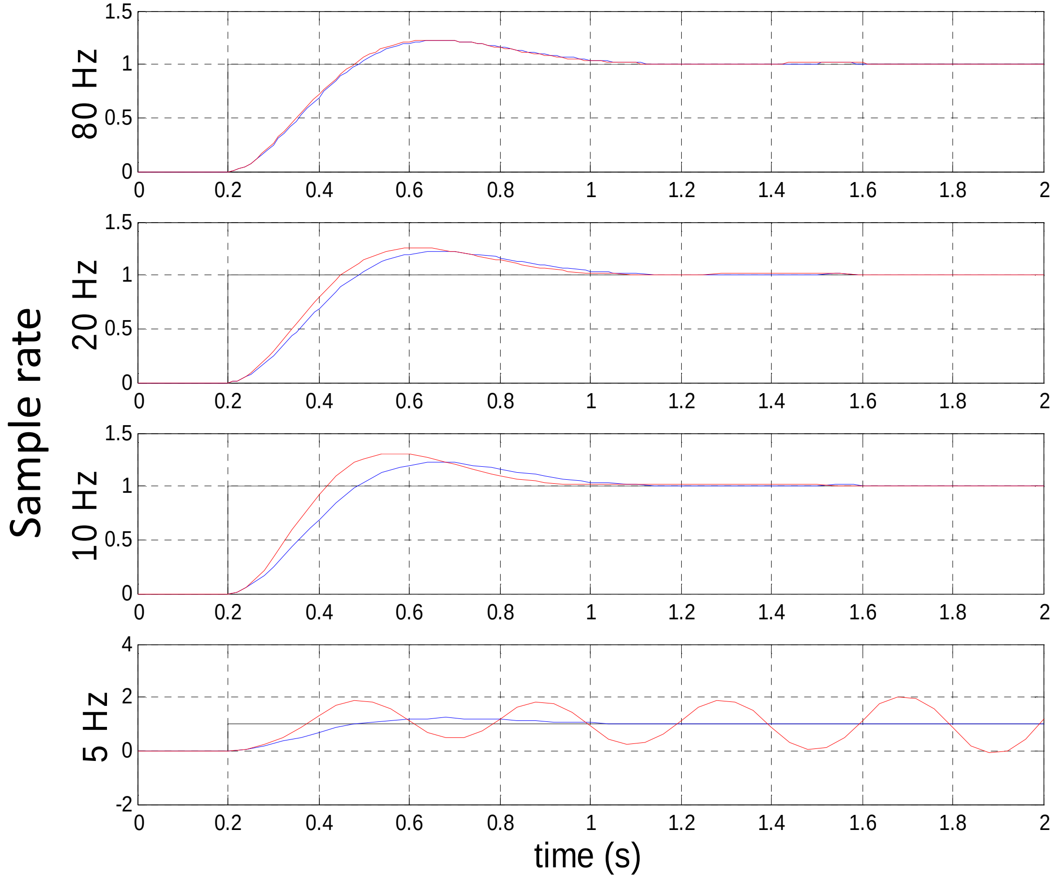
Example Using Simulink



Simulation Results

Blue: continuous system response

Red: sampled system response



Good performance

instability

As sample rate falls performance degrades – dynamics of plant dictate sample rate and ultimately speed of controller iteration

Summary

- If you sample fast enough a digital controller can be a fine approximation to continuous system.
- General rule of thumb is to sample at more than 30 times plant band width.
- If you can’t sample this fast then you need to know more information engineering and come to the Computer Controlled Systems Lectures….

Course Conclusion

- This was a *very* brief tour over just some of the areas that concern and interest information engineers and the domains that information engineering has a role to play
- In places we have given a few samples of the kind of mathematics you shall see more of in B4 and if you get hooked C4A and C4B
- Hopefully you'll now be aware that B4 is not just the "control paper" although it does contain a wholesome amount of that important information engineering topic.
- Hopefully you'll have had your interest piqued and have the sense that if you are going to a financier or an engineer that at some point needs to process data (so that's pretty much all of them) then information engineering has a great deal to offer you! After all, it brought you Google…