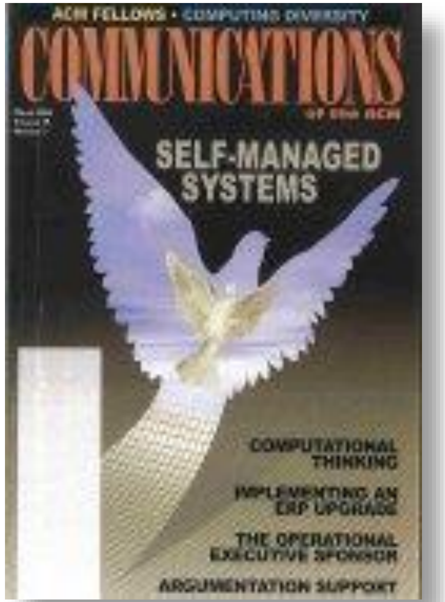
Using **Computational Thinking** to foster learning curiosity

Brad Horton
Engineer
MathWorks

Computational Thinking

2006



Viewpoint | Jeannette M. Wing

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

COMMUNICATIONS OF THE ACM March 2006/Vol. 49, No. 3 33

"Computational Thinking is the **thought processes** involved in **formulating problems and their solutions** … in a form that can be effectively **carried out by an information-processing agent**."

- Cuny, Snyder, Wing

Characteristics of Computational Thinking:

Decomposition

Break 1 complex problem into a collection of smaller/simpler problems

Abstraction

Mathematical modelling
- Symbolic representation
- Block diagrams

Algorithms + Automation

Formulating solution as a series of steps

Transforming between Modelling paradigms

Simulation

What happens when ?

Characteristics of Computational Thinking:

How does **MATLAB** support Computational Thinking ?

Decomposition

Break 1 complex problem into a collection of smaller/simpler problems

Centralize
- Narration
- Rationale
- Implementation

Abstraction

Mathematical modelling
- Symbolic representation
- Block diagrams

Algorithms + Automation

Formulating solution as a series of steps

Transforming between Modelling paradigms

Makes it easy to do this

Simulation

What happens when ?

Characteristics of Computational Thinking:

How does this foster **curiosity** ?

Decomposition

Break 1 complex problem into a collection of smaller/simpler problems

Centralize:
- Narration
- Rationale
- Implementation

Abstraction

Mathematical modelling
- Symbolic representation
- Block diagrams

Tedium is reduced.

Spend more time thinking about the core science.

There is a pathway from small to big problems

Algorithms
+
Automation

Formulating solution as a series of steps

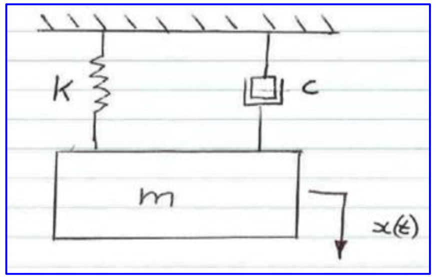
Transforming between Modelling paradigms

Makes it easy to do this

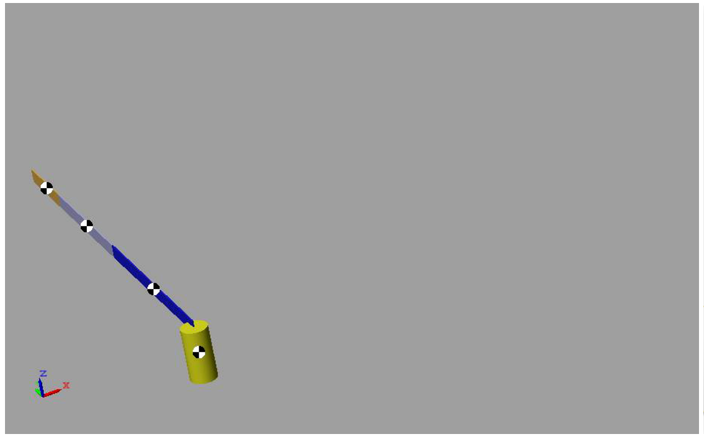
Simulation

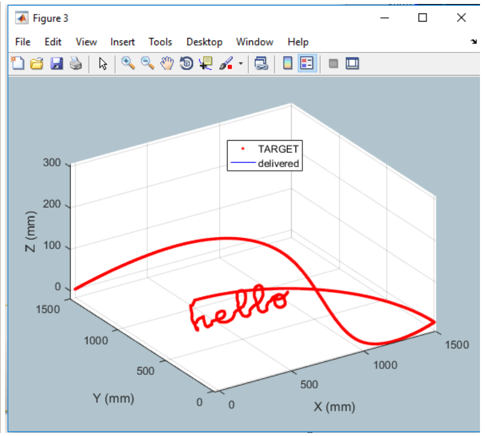
What happens when ?

Today's case study:



Solution pathway





From **this**

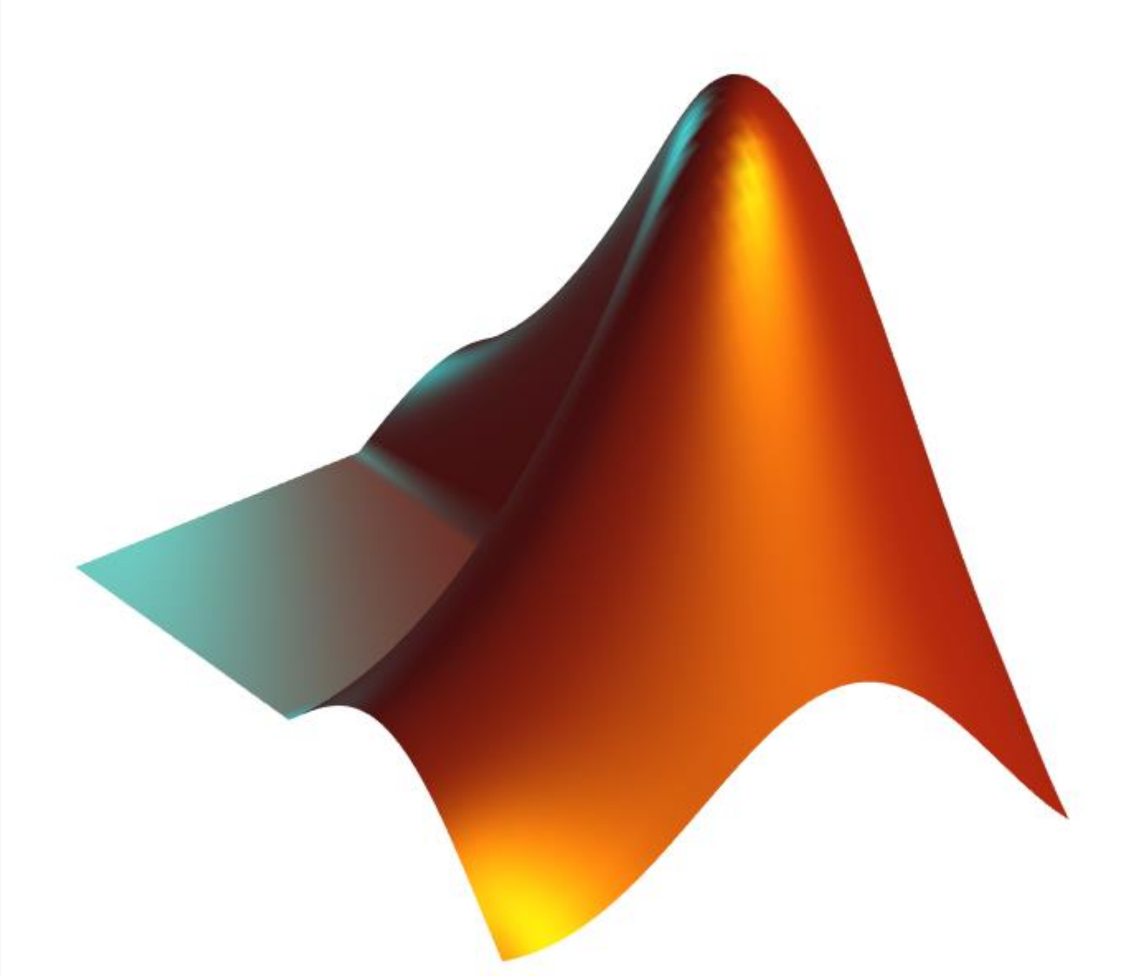
To **this**

Motivate me.

Decomposition | **Abstraction (Model Building)** | **Algorithms + Automation** | **Simulation**

Computational Thinking

Demo these concepts



Using Computational Thinking and **MATLAB** to foster learning curiosity

Centralization of thought process

Tedium busters

Modelling Choices

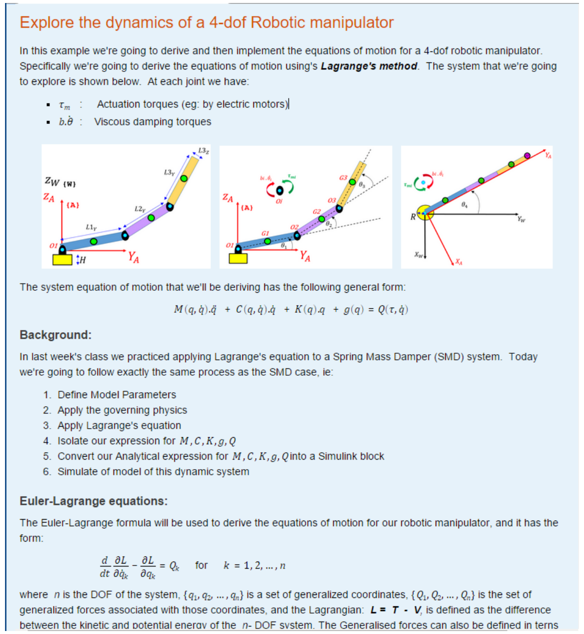
MATLAB
Live scripts

```
>> diff()

>> matlabFunctionBlock()
```

our_EOM(t) =

$$m \frac{\partial^2}{\partial t^2} x(t) + k\, x(t) = F - b \frac{\partial}{\partial t} x(t)$$



Explore the dynamics of a 4-dof Robotic manipulator

In this example we're going to derive and then implement the equations of motion for a 4-dof robotic manipulator. Specifically we're going to derive the equations of motion using's ***Lagrange's method***. The system that we're going to explore is shown below. At each joint we have:

- τ_m : Actuation torques (eg: by electric motors)
- $b.\dot{\theta}$: Viscous damping torques

The system equation of motion that we'll be deriving has the following general form:

$$M(q,\dot{q}).\ddot{q} + C(q,\dot{q}).\dot{q} + K(q).q + g(q) = Q(\tau,\dot{q})$$

Background:

In last week's class we practiced applying Lagrange's equation to a Spring Mass Damper (SMD) system. Today we're going to follow exactly the same process as the SMD case, ie:

1. Define Model Parameters
2. Apply the governing physics
3. Apply Lagrange's equation
4. Isolate our expression for M, C, K, g, Q
5. Convert our Analytical expression for M, C, K, g, Q into a Simulink block
6. Simulate of model of this dynamic system

Euler-Lagrange equations:

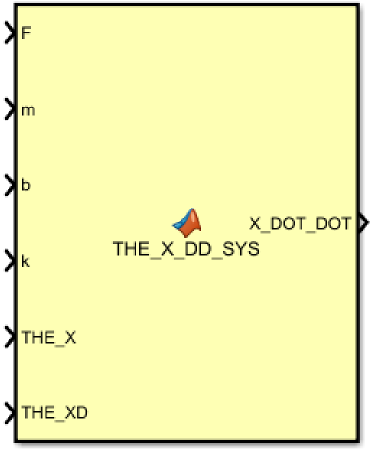
The Euler-Lagrange formula will be used to derive the equations of motion for our robotic manipulator, and it has the form:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k \quad \text{for} \quad k = 1, 2, \ldots, n$$

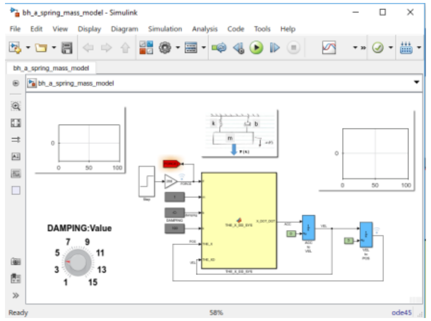
where n is the DOF of the system, $\{q_1, q_2, \ldots, q_n\}$ is a set of generalized coordinates, $\{Q_1, Q_2, \ldots, Q_n\}$ is the set of generalized forces associated with those coordinates, and the Lagrangian: $\boldsymbol{L = T - V}$, is defined as the difference between the kinetic and potential energy of the n- DOF system. The Generalised forces can also be defined in terns

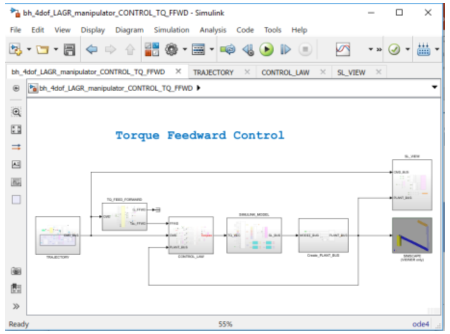
$$g(t) = \sin(z(t))^2$$

$$dg\_dt(t) = 2\cos(z(t))\sin(z(t))\frac{\partial}{\partial t} z(t)$$



THE_X_DD_SYS





Student's desires:

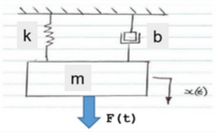
- How does what I already know:
  - Extend to NEW things
  - **Scale from simple to complex things**
- I do NOT want to do boring things

Professor's desires:

- I do want my students to:
  - focus on the science/engineering
  - Think, explore, build

Explore the dynamics of a 1-dof Spring Mass Damper

In this example we're going to derive and then implement the equations of motion for 1-dof Spring Mass Damper system. Specifically we're going to derive the equations of motion using's ***Lagrange's method***. The system that we're going to explore is shown below.



Background:

From our year 1 class in physics and mechanics, we derived using ***Newton's 2nd law***, the equation of motion for the dynamics of a Spring Mass damper system. Recall that it had the following form:

$$m.\ddot{x} \; + \; b.\dot{x} \; + \; k.x = F(t)$$

Today we'll use the ***Lagrangian approach*** to derive the same equations of motion for our spring mass damper. We're going to break this problem down into the following 6 steps:

1. Define Model Parameters
2. Apply the governing physics
3. Apply Lagrange's equation
4. Isolate our expression for $\ddot{x}(t)$
5. Convert our Analytical expression for $\ddot{x}$ into a Simulink block
6. Simulate of model of this dynamic system

Euler-Lagrange equations:

Recall our earlier class where we derived and summarised the fundamental Lagrangian equations that allow us to derive system equations of motion:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k \qquad \text{where} \qquad Q_k = \sum_{i=1}^{Nf_{nc}}\left(\vec{F_i}.\frac{\partial \vec{v_i}}{\partial \dot{q}_k}\right) + \sum_{j=1}^{N\tau_{nc}}\left(\vec{\tau_j}.\frac{\partial \vec{\omega_j}}{\partial \dot{q}_k}\right)$$

where:

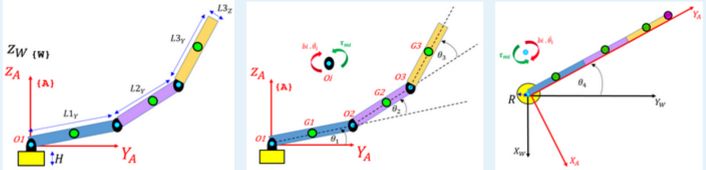
- L : is the system Lagrangian, ie: L = KE - PE
- q_k : is the k^{th} generalised co-ordinate
- Q_k : is the generalised force associated with the k^{th} generalised co-ordinate q_k
- Nf_{nc} : is the number of active NON conservative forces
- $N\tau_{nc}$: is the number of active NON conservative TORQUES

Solution pathway

Explore the dynamics of a 4-dof Robotic manipulator

In this example we're going to derive and then implement the equations of motion for a 4-dof robotic manipulator. Specifically we're going to derive the equations of motion using's ***Lagrange's method***. The system that we're going to explore is shown below. At each joint we have:

- τ_m : Actuation torques (eg: by electric motors)
- $b.\dot{\theta}$: Viscous damping torques



The system equation of motion that we'll be deriving has the following general form:

$$M(q,\dot{q}).\ddot{q} \; + \; C(q,\dot{q}).\dot{q} \; + \; K(q).q \; + \; g(q) = Q(\tau,\dot{q})$$

Background:

In last week's class we practiced applying Lagrange's equation to a Spring Mass Damper (SMD) system. Today we're going to follow exactly the same process as the SMD case, ie:

1. Define Model Parameters
2. Apply the governing physics
3. Apply Lagrange's equation
4. Isolate our expression for M, C, K, g, Q
5. Convert our Analytical expression for M, C, K, g, Q into a Simulink block
6. Simulate of model of this dynamic system

Euler-Lagrange equations:

The Euler-Lagrange formula will be used to derive the equations of motion for our robotic manipulator, and it has the form:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = Q_k \qquad \text{for} \qquad k = 1, 2, \ldots, n$$

where n is the DOF of the system, $\{q_1, q_2, \ldots, q_n\}$ is a set of generalized coordinates, $\{Q_1, Q_2, \ldots, Q_n\}$ is the set of generalized forces associated with those coordinates, and the Lagrangian: ***L = T - V***, is defined as the difference between the kinetic and potential energy of the n- DOF system. The Generalised forces can also be defined in terns

How is Computational Thinking Introduced ?

Computational Thinking

Do students just “pick up” computational thinking?

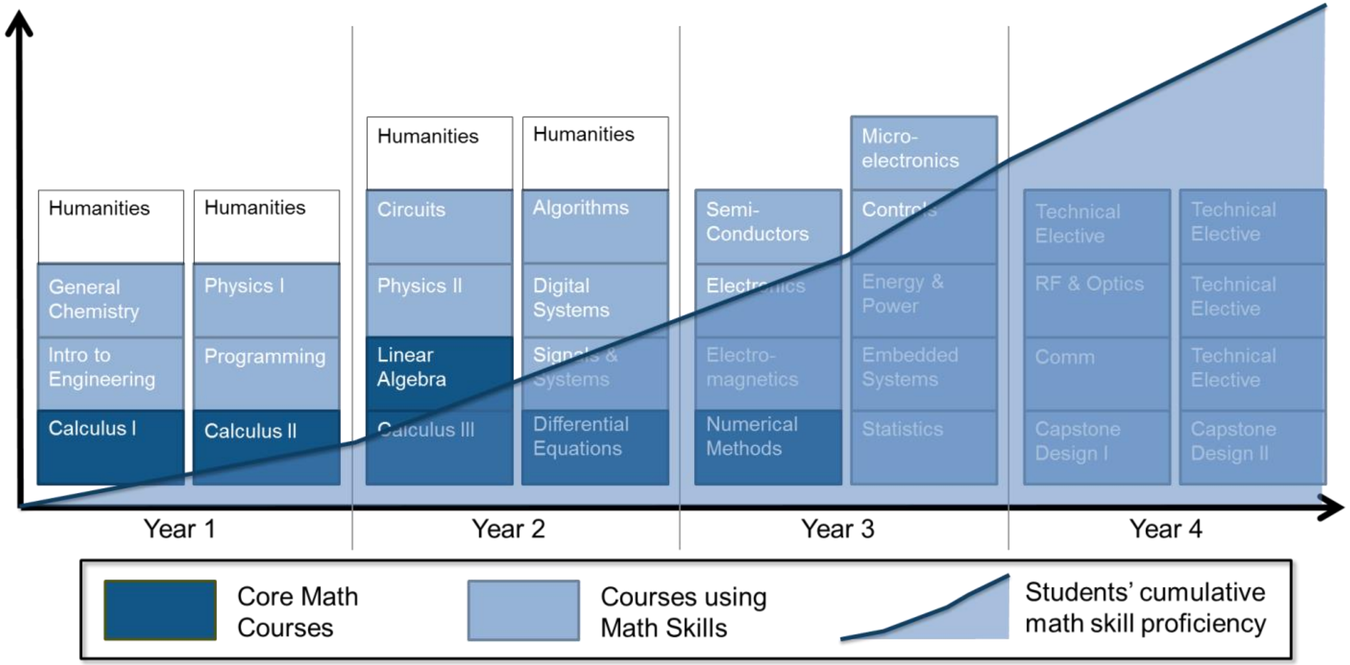
VS

VS

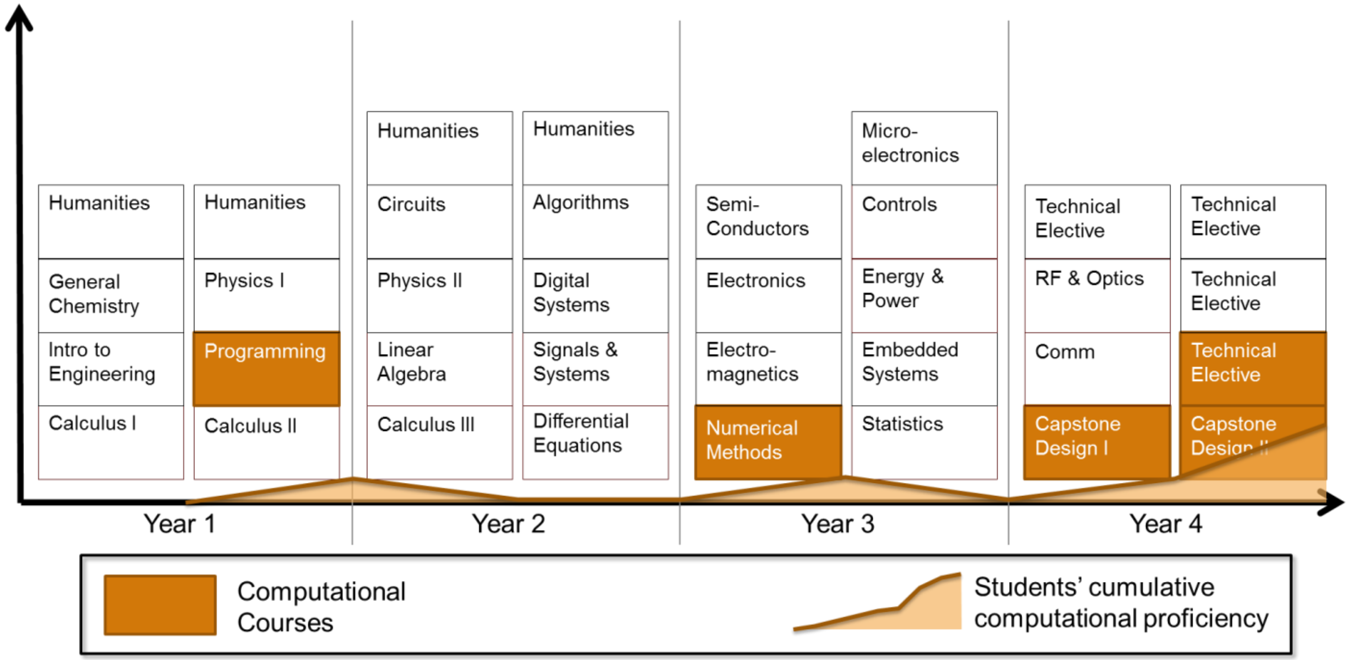
Math Skills

Isn’t math taught systematically and reinforced throughout the curriculum?

How Math is introduced in the curriculum



How is Computational Thinking introduced?



Fostering a Curiosity to Learn:

Decomposition

Abstraction (Model Building)

Algorithms + Automation

Simulation

Computational Thinking

MATLAB Technical Computing Environment

- There is a pathway from simple to complex problems
- Tedium is reduced.
- Spend more time thinking about the core science.