The computer as a tool to develop mathematical problem solving skills

Never Stand Still

UNSW Canberra

Tristram Alexander

School of Physical, Environmental and Mathematical Sciences

UNSW Canberra

Background: “Applied Nonlinear Dynamics” (3rd year), “Engineering Problem Solving” (1st year)

A **dedicated problem solving course** for first year engineering students. Algorithmic thinking and programming a key part of this course.

A **third-year mathematics course** aiming to give students some experience with research using a computer.

Typically around 90% lecture attendance.

Students from all over Australia. Some international. Clear career path on graduation.

What we know about good problem solvers (Whimbey)

1. **Positive Attitude**: strong belief that academic reasoning problems can be solved through careful, persistent analysis.
2. **Concern for Accuracy**: take great care to understand the facts and relationships in a problem fully and accurately. Almost compulsive in checking if understanding is correct and complete.
3. **Break the Problem into Parts**: tackle piece by piece.
4. **Avoid Guessing**: work methodically, carefully, no jumps.
5. **Active in Problem Solving**: do things to try to understand and answer difficult questions (e.g. reword, draw diagrams, 'talk to themselves').

All require the student to slow down, and have a belief that they can solve the problem by working slowly through a problem, without relying solely on intuitive leaps, and using techniques to reduce cognitive overload (Kahneman).

What aspect has the largest impact on capability as a problem solver?

1. Attitude
2. Knowledge
3. Strategy
4. Experience
5. Creativity

What aspect has the largest impact on capability as a problem solver?

1. Attitude
2. Knowledge
3. Strategy
4. Experience
5. Creativity

If students are intrinsically motivated (rather than extrinsically through chasing a reward for instance) then many challenges of the teaching and learning environment go away.

Ideally the student should:

- Have belief in success of their skills and process.
- Be motivated by learning goals (rather than performance goals).
- Be problem-finders rather than problem-followers.

Developing mental toughness: learning from Polya’s mouse

“The landlady hurried into the backyard, put the mousetrap on the ground (it was an old-fashioned trap, a cage with a trapdoor) and called to her daughter to fetch the cat. The mouse in the trap seemed to understand the gist of these proceedings; he raced frantically in his cage, threw himself violently against the bars, now on this side and then on the other, and in the last moment he succeeded in squeezing himself through and disappeared in the neighbour's field. There must have been on that side one slightly wider opening between the bars of the mousetrap . . . I silently congratulated the mouse. He solved a great problem, and gave a great example. That is the way to solve problems. We must try and try again until eventually we recognize the slight difference between the various openings on which everything depends. We must vary our trials so that we may explore all sides of the problem. Indeed, we cannot know in advance on which side is the only practicable opening where we can squeeze through.”

G. Polya “Mice and Men”

Psychological strategy (Zeitz)

Moral of Polya’s story: don’t give up easily, and don’t keep banging your head senselessly against a wall, but instead vary the attempt each time.

“most beginners give up too soon, because they lack the mental toughness attributes of confidence and concentration. It is hard to work on a problem if you don 't believe that you can solve it, and it is impossible to keep working past your ‘frustration threshold’”

P. Zeitz, “The Art and Craft of Problem Solving”

How to increase frustration threshold? (the challenge for the teacher)

- Work on problems not exercises, so your brain gets a workout and your subconscious gets used to success.
- Start with easy problems, to warm up, but then work on harder and harder problems that continually challenge and stretch you to the limit.
- As confidence rises, so too does your frustration threshold.

Toughen up, loosen up and practice

Attitude and tool use

The frustration threshold is context dependent, and is determined by student past experience in the particular context.

This has significant consequences for using tools in the classroom.

The majority of students have a low frustration threshold when using a computer (desktop or phone) for programming.

Past experience is that things “just work”, so when suddenly faced with a situation where things don’t work there is immediate frustration.

This is compounded by easy access to more familiar patterns of use which do work.

http://www.thetealbrickroad.com/wp-content/uploads/2015/10/fork-in-the-road.jpg

Implementation considerations (PC vs mobile)

Mobile device

Javascript (pros)

- Javascript is free and available on every platform
- All students have a device of some form, so useable in the lecture theatre
- Javascript is a relatively simple programming language

Javascript (cons)

- Phone has many more familiar uses, so easy to be side-tracked when using
- Requires control of a separate editor and use of two files to implement

PC

Matlab (pros)

- Standard format (all students see the same thing)
- In-built editor and filesystem
- Many in-built routines, including for visualisation

Matlab (cons)

- Requires a computer lab
- In-built routines make it easy to default to using as a “black box”

Resource load: 15 students per lecturer/tutor when engaged in device use

An overview of the problem solving process: mountaineering analogy (Zeitz)

1. **Strategy**: high level. E.g. climb easier surrounding peaks to observe target mountain from different angles.

Problem solving strategies for orientation phase: **“get hands dirty”, “make it easier”**

2. **Tactics**: middle level. E.g. if crossing a snowfield, go early in the morning.

Example problem solving tactics: **“draw a picture”, “factorise”**

3. **Tools**: lowest level. To cross snowfield, set up safety ropes, and use ice axes.

Example problem solving tools: **“completing the square”, “method of undetermined coefficients”**

The role of the computer in the problem solving process?

The computer can serve a number of purposes. It can be used to:

1. uncover patterns and generate hypotheses;
2. explore different ways of approaching a problem;
3. develop and practice algorithmic thinking;
4. confirm or explore expected results;
5. engage in inquiry-based learning on research-level problems.

Primarily, I will look at using the computer to help develop problem solving **strategy**.

It provides an avenue to **“get your hands dirty”**, and think about how to **“make it easier”.**

Goals in terms of the student outcomes

[Outcome 1] ***Be able to demonstrate successful application of self-learning skills***

Specifically, be able to:

- Identify own thinking and problem solving approach and articulate this to another.
- Self-assess progress towards achieving a given goal, and self-assess quality of completion of the goal.
- Assess others in a constructive and unambiguous way and identify the differences between own approach and that of another.
- Generate options when required, and be aware of any pitfalls or biases when doing so [Creativity].
- Analyse options when required, and identify pros and cons, differences and similarities [Analysis].
- Understand and seek to enact the mentality of a successful problem solver, and demonstrate this through relevant examples (e.g. through presentation of cases of non-obvious problems identified in daily and academic life).

Goals in terms of the student outcomes

[Outcome 2] ***Be able to successfully demonstrate a problem solving process***

Specifically, given a problem be able to:

- Identify the goals ("the unknown");
- Identify explicit and implicit assumptions;
- Represent the problem in multiple ways if required;
- Be able to represent graphically or in dot point form the background knowledge needed to approach the problem, showing the relationships between components, and be able to identify any areas which need further investigation before the problem can be solved;
- Identify different strategies that may be used to solve the problem;
- Carry through a solution process methodically and without error;
- Reflect on the validity of a solution, the generality and limitations of the method and solution, and identify any areas which could be improved.

Goals in terms of the student outcomes

[Outcome 3] ***Be able to successfully solve problems in a variety of contexts***

- Specifically, in the following areas:
- Context-free (e.g. puzzles)
- Mathematics (e.g. proof)
- Programming (e.g. numerical solution)
- Engineering (e.g. real-world problems, in the presence of constraints)
- Non-academic (e.g. decision making in life beyond class)

Goals when using a computer

- Develop confidence in using the computer to explore a problem
  - Use a computer to get hands dirty and try multiple approaches
  - Use a computer to see patterns and develop hypotheses
- Development of algorithmic thinking, and implementation on a computer
- Be able to compute and check solution of a problem, for which nature of solution is known
- Be able to investigate a problem using a variety of computational methods, and identify patterns and formulate hypotheses when the form of the solution is unknown (research)

Building confidence with “simple” computation: looking for patterns and generating hypotheses

The simplest engagement with the tool is repetitive calculation, which may be carried out by hand, on a calculator, or using a spreadsheet.

Example problems which encourage the strategies “get your hands dirty” and “make it easier”:

Solve: $$\frac{1}{1\cdot 2}+\frac{1}{2\cdot 3}+\frac{1}{3\cdot 4}+\cdots+\frac{1}{99\cdot 100}$$

Identify the pattern, come up with a conjecture for the general rule, and if you can prove your conjecture:

$$1+8=9$$

$$1+8+27=36$$

$$1+8+27+64=100$$

When “brute force” becomes inefficient: exploring multiple approaches

Brute force is often a go-to strategy for students, so problems which require them to question this approach are useful. Questions such as “when to stop calculating?” or “when to try a different approach?” then naturally emerge.

Example:

Find the smallest natural number in which all digits are 0s and 1s and which is divisible by 225.

When patterns might not be obvious: the need for multiple approaches

Word problems

Lockers in a row are numbered 1, 2, 3, . . . , 1000. At first, all the lockers are closed. A person walks by and opens every other locker, starting with locker #2. Thus lockers 2, 4, 6, . . . , 998, 1000 are open. Another person walks by, and changes the “state” (i.e., closes a locker if it is open, opens a locker if it is closed) of every third locker, starting with locker #3 . Then another person changes the state of every fourth locker, starting with #4, etc . This process continues until no more lockers can be altered. Come up with a conjecture, which lockers will be closed?

Not an obvious pattern

Deduce a formula for:

$$1^2 + 3^2 + 5^2 + \cdots + (2n-1)^2$$

Algorithms and mathematical thinking: problem solving driven by the computer

Algorithmic thinking: getting to a solution with a clear definition of the steps needed.

Challenges:

- Awareness of what is even possible (types of step?) may be absent (skills required: contextual reading, writing)
- Requires an iterative approach to solutions (a skill likely unfamiliar prior to using computers)
- Thinking plus tool use requires skill and knowledge in multiple areas
- Frustration threshold may be low and unexpected knowledge barriers may appear (for instance lack of understanding of what a file system is when required to use files; inability to understand computer error messages; inability to use help pages)

On the plus side, tackling algorithm development rewards all five aspects of a good problem solver: Positive Attitude; Concern for Accuracy; Break the Problem into Parts; Avoid Guessing; Active in Problem Solving

Introduction to algorithmic thinking through device-free problems (development of confidence)

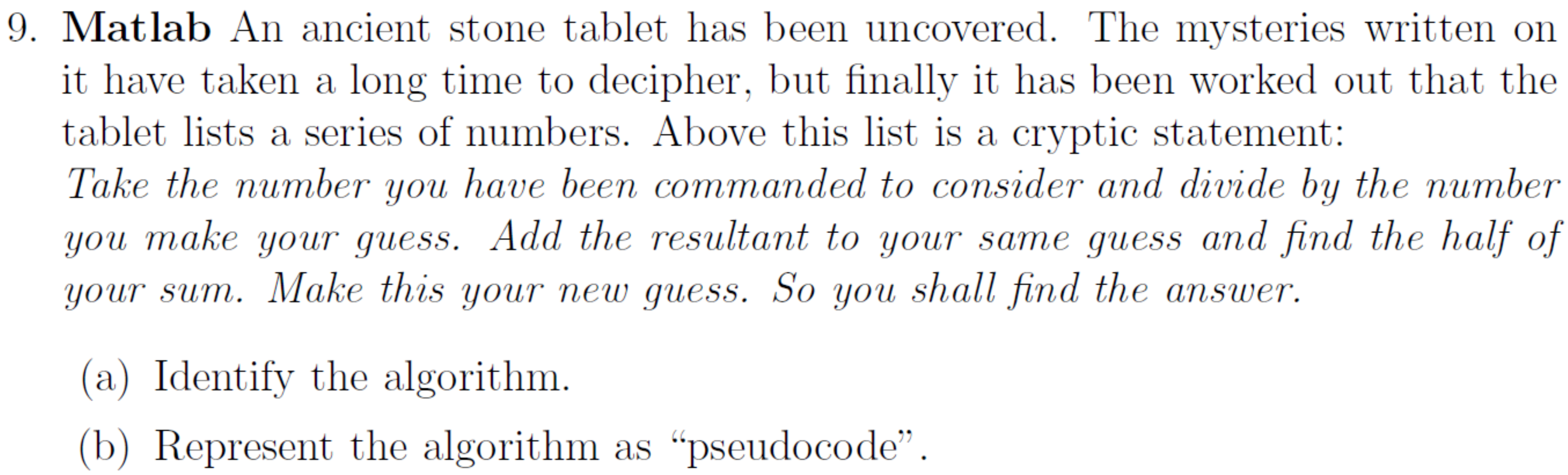
Example problems:

You have ten stacks of coins, each consisting of ten 50 cent pieces. One entire stack is counterfeit, but you do not know which one. You do know the weight of a genuine half-dollar and you are also told that each counterfeit coin weighs one gram more than it should. You may weigh the coins on a scale. What is the smallest number of weighings necessary to determine which stack is counterfeit?

Devise a general procedure so that *n* persons can cut a cake into *n* portions in such a way that everyone is satisfied that they have at least 1/*n* of the cake.

Exploration and discovery as a part of algorithm development

Example: Babylonian method for finding square roots



9. **Matlab** An ancient stone tablet has been uncovered. The mysteries written on it have taken a long time to decipher, but finally it has been worked out that the tablet lists a series of numbers. Above this list is a cryptic statement:
Take the number you have been commanded to consider and divide by the number you make your guess. Add the resultant to your same guess and find the half of your sum. Make this your new guess. So you shall find the answer.

   (a) Identify the algorithm.

   (b) Represent the algorithm as "pseudocode".

Making the leap to code

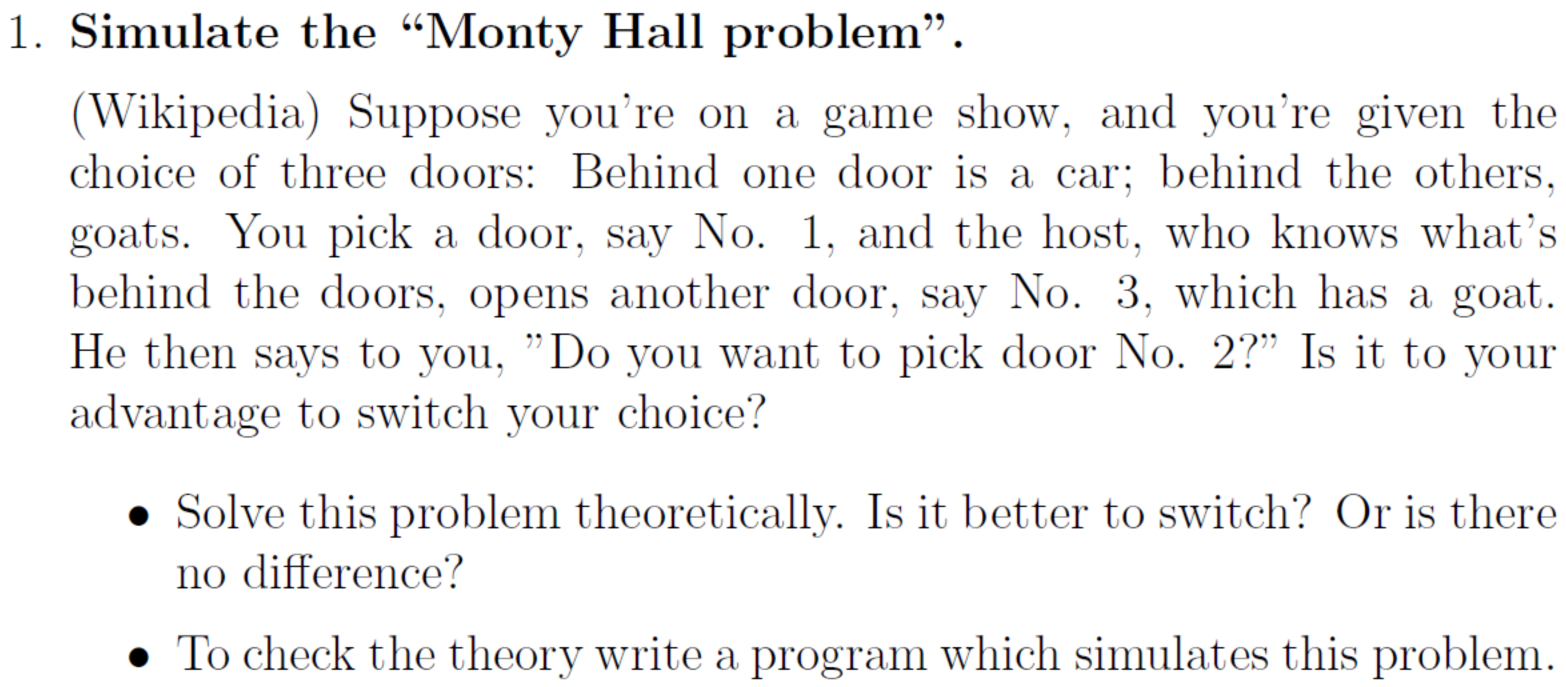
"In teaching computing we seem to have overlooked or neglected what corresponds to the reading stage in the process of learning to read and write. To put it strongly, asking people to design and write programs early on in their computing experience is like expecting that they be able to competently write essays before that have learned to read or even to write short sentences – it is expecting just too much of a lot of people. It also probably explains why many otherwise very able people just don't get started in computing." **R.G. Dromey "How to Solve it by Computer"**

Tips:

- Make execution visible.
- Start simply (E.g. "Given two variables *a* and *b*, exchange the values assigned to them")
- Build up slowly.

Where to? The power of simulation

Exploring problems in probability:



1. **Simulate the "Monty Hall problem".**

   (Wikipedia) Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?

   - Solve this problem theoretically. Is it better to switch? Or is there no difference?
   - To check the theory write a program which simulates this problem.

Provides another tool for analysis and problem exploration, particularly useful when stuck with analytical approach.

Where to? The power of simulation

Examining applied problems:

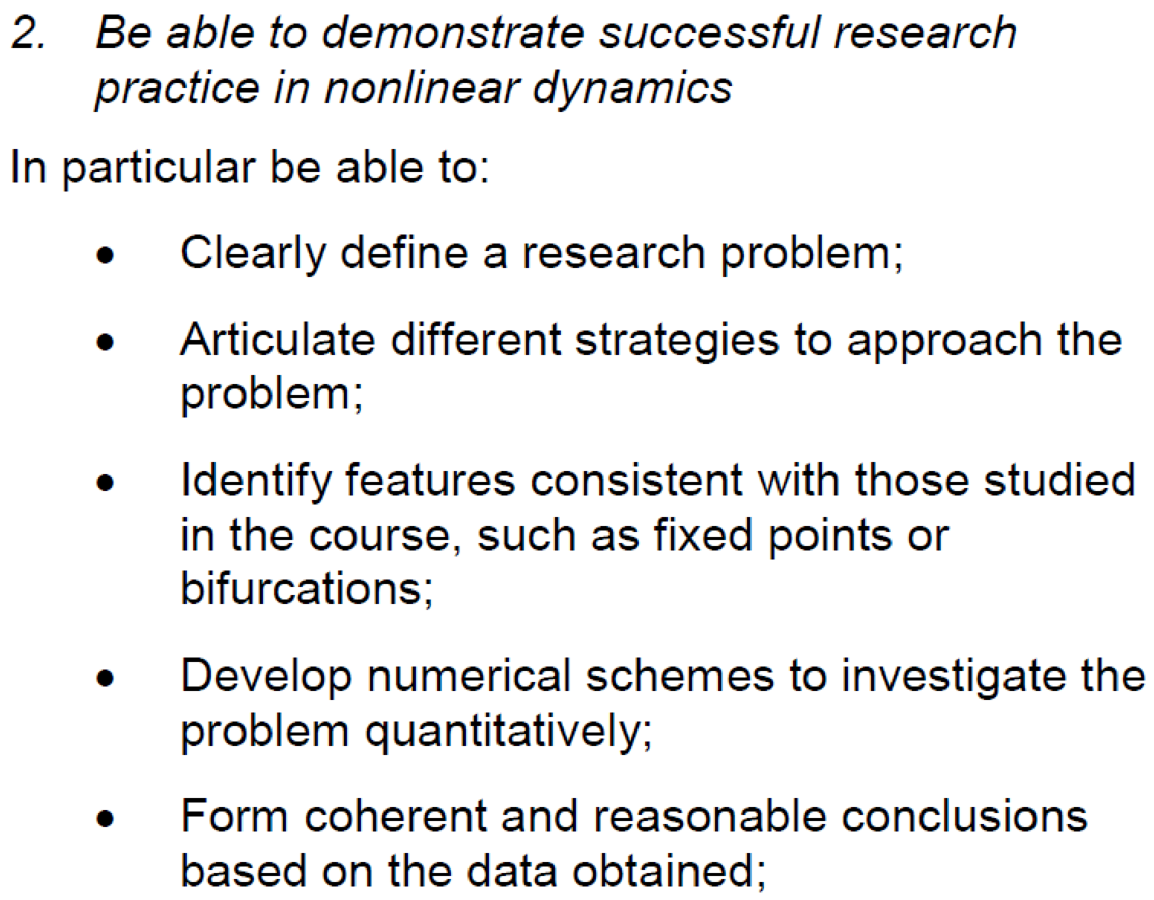
4. A beach ball is thrown upward with initial speed v_0. Assume that the drag force from the air is $F = m\alpha v$. What is the speed of the ball, v_f , when it hits the ground? (An implicit equation is sufficient.) Does the ball spend more time or less time in the air than it would if it were thrown in vacuum?

Computers and research

Enables the possibility of approaching problems for which analytical solutions do not exist

In our curriculum at least, this means problems well beyond the familiar types of systems studied in the maths degree.

Reflection is a key component of computer based research



2. *Be able to demonstrate successful research practice in nonlinear dynamics*

In particular be able to:

- Clearly define a research problem;
- Articulate different strategies to approach the problem;
- Identify features consistent with those studied in the course, such as fixed points or bifurcations;
- Develop numerical schemes to investigate the problem quantitatively;
- Form coherent and reasonable conclusions based on the data obtained;

Summary

Some uses for the computer in mathematical problem solving:

1. uncover patterns and generate hypotheses;
2. explore different ways of approaching a problem;
3. develop and practice algorithmic thinking;
4. confirm or explore expected results;
5. engage in inquiry-based learning on research-level problems.

Graduate attributes to aim for: Positive Attitude; Concern for Accuracy; Break the Problem into Parts; Avoid Guessing; Active in Problem Solving

Challenges: Low frustration threshold, many sources of temptation, reflection needs to be an integral part of the process, requires high resource load (15 students per lecturer/tutor).

Reward: Very satisfying to student if succeed.