Topic 0
Introduction to Numerical Solutions and Coding

Adam Hal Spencer

The University of Nottingham

Applied Computational Economics

Roadmap

The Menu for Today

1. What role does a computer play in solving economic models?
2. How do numerical solutions compare with pen and paper methods you studied with Giammario in Macro I?
3. What types of research can you use numerical solutions for?
4. What's coding?
5. What are the basics of Matlab?

Roadmap

Numerical vs Analytical Solutions

- This course will all be about model solving using a computer.
- Why do we need to use computers for such activities at all?
- Complicated models can't be solved with a pen and paper.

Numerical vs Analytical Solutions

- Take the social planner's problem for the neoclassical growth model for example.

$$\max_{\{c_t, k_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u(c_t)$$

subject to

$$\begin{aligned} c_t + i_t &= f(k_t) \\ i_t &= k_{t+1} - (1-\delta)k_t. \end{aligned}$$

Numerical vs Analytical Solutions

- How would you solve this in an exam using pen and paper?

Numerical vs Analytical Solutions

- Solution can be characterised by the Euler equation and resource constraint

$$1 = \beta \frac{u'(c_{t+1})}{u'(c_t)}[1 - \delta + f'(k_{t+1})] \quad (1)$$

$$c_t + k_{t+1} = f(k_t) + (1 - \delta)k_t. \quad (2)$$

- Where does one go from here?

Numerical vs Analytical Solutions

- There are really two options.

(1) Impose assumptions on the problem to get an analytical (pen and paper) solution.

(2) Take it to a computer and solve it numerically.

Numerical vs Analytical Solutions

- What's the tradeoff?
- Numerical solutions usually involve "less restrictive" assumptions.
- But your solution only holds for a specific set of parameters.

Analytical Solutions

- In the context of our neoclassical growth model, we can get analytical solutions under the right assumptions.
- Assume the following

$$
\begin{aligned}
u(c_t) &= \log(c_t) \\
f(k_t) &= k_t^{\alpha} \\
\delta &= 1.
\end{aligned}
$$

- We can then use the guess and verify method to find an analytical solution.

Analytical Solutions

- What is a state variable?
- What is a policy function?

Analytical Solutions

- Conjecture that $k_{t+1} = \omega y_t$ for $\omega \in [0, 1]$ where $y_t = k_t^{\alpha}$.
- Says that your investment is some fraction of final output.
- Get then from equation (2) that $c_t = (1 - \omega) y_t$.
- I.e. consumption takes the remainder of final output.

Analytical Solutions

- Recall equation (1), but now under our assumptions

$$1 = \beta \frac{c_t}{c_{t+1}} \alpha k_{t+1}^{\alpha-1}.$$

- Notice that $\alpha k_{t+1}^{\alpha-1} = \alpha y_{t+1}/k_{t+1}$.

Analytical Solutions

- Follows then that

$$
\begin{aligned}
1 &= \beta \frac{(1-\omega)y_t}{(1-\omega)y_{t+1}} \alpha \frac{y_{t+1}}{k_{t+1}} \\
\Rightarrow 1 &= \alpha\beta \frac{y_t}{k_{t+1}} \\
\Rightarrow k_{t+1} &= \alpha\beta y_t = \alpha\beta k_t^{\alpha}
\end{aligned}
$$

- This is our solution: tell me the current period capital stock k_t and then, using equation (3), I can tell you k_{t+1} through

$$k_{t+1}(k_t) = \alpha\beta k_t^{\alpha}. \tag{3}$$

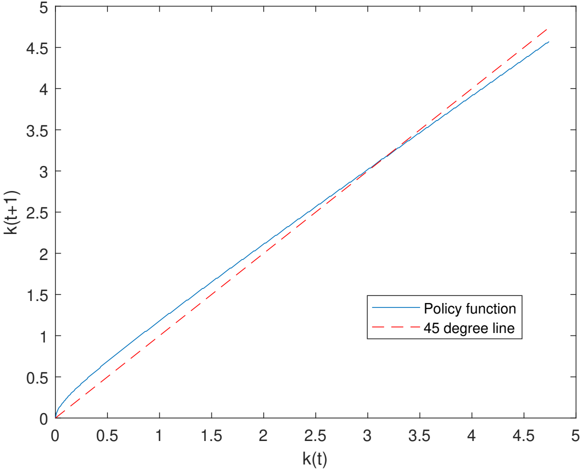
Numerical

- What can we do with a policy function?

Numerical

- What if we don't like log utility and full depreciation though?
- Say we want $u(c_t) = \frac{c_t^{1-\sigma}}{1-\sigma}$ with $f(k_t) = k_t^{\alpha}$.
- Set $\alpha = 0.33$, $\delta = 0.10$, $\sigma = 2.0$ and $\beta = 0.95$.
- Solve for the numerical equivalent of $k_{t+1}(k_t)$ in (3).

Numerical



Numerical

- Finding these numerical solutions is what this course is all about.

Roadmap

Research Questions you can Answer

- You use these methods to answer quantitative questions.
- If the government increases this tax rate by 1%, GDP will drop by X%.

Research Questions you can Answer

- Can also use them to gauge the quantitative significance of certain channels in your model.
- E.g. stick additional friction into a standard model.
  - Estimate the baseline model with all the features in place.
  - Compare with a more standard model without the friction.
  - What differs? Maybe the counterfactual you're running gives different qualitative or quantitative results.

Research Questions you can Answer

- Positives of this type of research:
  - Can put any features you want into a model and be able to solve it.
  - Easily interpretable results.
  - Super useful for policy analysis.
  - Tons of fertile areas of application.

Research Questions you can Answer

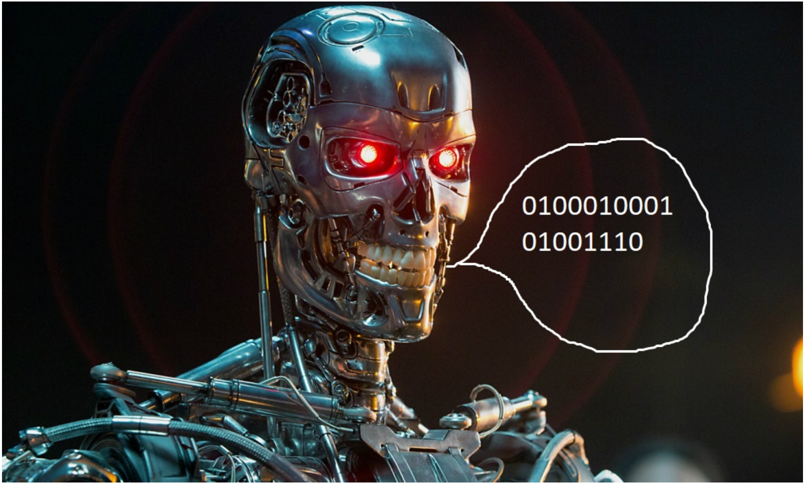
- Negatives of this type of research:
  - "Black boxy": can't do closed-form comparative statics.
  - How do you choose parameters? Derogatory term: "theory with numbers".
  - General scepticism from pure empiricists or pure theorists.
  - Solving these things is hard and time-consuming!

Roadmap

Talking to the Machines

- Human language: words, letters and numbers.
- Machine language: binary of 0s and 1s.
- How do we communicate with the machines? Through coding/programming languages.

Talking to the Machines



Languages

- Programming languages take our commands and then translate to binary for the machines to understand.
- Lots of alternatives with different pros and cons.
- Matlab is pretty close to a standard for economics.
- Or at the least, it's a good starting point.

Roadmap

Matlab

- Stands for Matrix Laboratory.
- Basically an interface built over the top of C/C++.
- Matlab always works fastest when you use lots of matrices in your code.

Very Basics

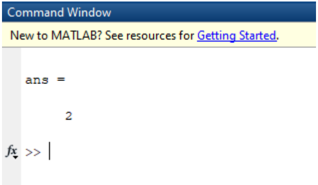
- Matlab script (code) files have .m extensions. E.g. PS1.m.
- We'll proceed in the lecture by example.

Very Basics

- Clear the memory and workspace, then crunch the sum of $1+1$

```
clear;clc;
1+1;
1+1
```

gives the output



- The ; suppresses output: i.e. the $1+1$ was only printed once when the ; didn't follow.

Arrays

- Best practice is always to declare the size of arrays (vectors) before filling them.

- Declare an array (call it A) of size 3×1 and then fill it with the numbers 1, 2 and 3.

```
A = zeros(3,1);
A(1,1) = 1;
A(2,1) = 2;
A(3,1) = 3;
A
```

gives the output

```
A =

     1
     2
     3
```

Arrays

- Declare an array (call it B) of size 1×3 and then fill it with the numbers 1, 2 and 3.

```
B = zeros(1,3);
B(1,1) = 1;
B(1,2) = 2;
B(1,3) = 3;
B
```

gives the output

```
B =

     1     2     3
```

- The same idea follows for matrices.

Arrays

- There is a difference between matrix operations and element-by-element operations.
- Declare two matrices of size 2×2. Call them C and D. Fill them both with ones.

```
C = zeros(2,2);
D = zeros(2,2);
C(:,:) = 1;
D(:,:) = 1;
```

Arrays

- Then multiply them together (in the matrix sense).

  ```
  C*D
  ```

  gives the output

```
>> C*D

ans =

     2     2
     2     2
```

Arrays

- Now multiply C and D together element-by-element

  ```
  C.*D
  ```

  gives the output

  ```
  >> C.*D

  ans =

       1     1
       1     1
  ```

  which is like crunching $C(1,1) * D(1,1)$, $C(1,2) * D(1,2)$, ... etc and storing the results in a 2×2 matrix.

For Loops

- These are known as “do loops” in other languages.
- Says to perform an operation several times, where each run is indexed by an integer.

For Loops

- Create a 3×1 array of ones called E. Loop through each element of the array and print the output from multiplying each element by its position number.

```
E = zeros(3,1);
E(:,:) = 1;
for i = 1:length(E);
    i*E(i,1)
end;
```

gives output

```
ans =

     1


ans =

     2


ans =

     3
```

If Statements

- The conditional statement.
- If *something is true* then *do this*.

If Statements

- Using the E array you created, perform the same for loop as before. But for the second entry, instead of printing the entry number, print the number 100.

```
E = zeros(3,1);
E(:,:) = 1;
for i = 1:length(E);
    if (i == 2)
        100
    else
        i*E(i,1)
    end
end;
```

If Statements

- Gives output

```
ans =

     1


ans =

   100


ans =

     3
```

While Loops

- Keep repeating some action *until* some condition is satisfied.

While Loops

- Create a variable called k. Set this variable equal to zero. Keep increasing k by an increment of 1 until it reaches a value of 3. Print the output at each increment.

```
k = 0
while (k <3)
    k = k + 1
end
```

While Loops

- Yields output

```
i =

     0


i =

     1


i =

     2


i =

     3
```

Functions

- A function is a script that you can call from another, which performs specified tasks.
- It takes *input arguments* and then gives you *outputs*, to which they correspond.
- You need to follow a special syntax in writing the function script to get it to work properly.

Functions

- For a function with one input and one output, first line should read

```
function output = myfunction(input)
```

  where *output* (*input*) is the name of your output (input) and *myfunction* is the name you give the script.

- The word function must always the first in the script.

- You must also always save the function script as *myfunction*.m (where *myfunction* is whatever name you chose above).

Functions

- Write a function that takes an input then multiplies it by 100. Call this function from the main body of your script using an input of 10.
- The function script (separate from the main script) would say

```
function output = myfunction(input)
        output = input*100
end
```

where the main script calls it as follows

```
myfunction(10)
```

gives

```
>> myfunction(10)

output =

        1000


ans =

        1000
```

Roadmap

Approximations: from Continuous to Discrete

- Computers can't handle infinite dimensional objects.
- Instead we approximate the object at a finite number of points.

Approximations: from Continuous to Discrete

- Consider the function

$$f(x) = x^3, \;\; x \in [0, 1].$$

- Notice that the continuum $[0, 1]$ is uncountably infinite.

- How would we plot this object using a computer?

Approximations: from Continuous to Discrete

- "Chop-up" the interval into a discrete set of points.
- I.e. define a set X to approximate $[0, 1]$ such that

$$X \equiv \{x_0, x_1, x_2, ..., x_n\}$$

where $n \in \mathbb{N}$, $x_0 = 0$ and $x_n = 1$.

Approximations: from Continuous to Discrete

- Then evaluate the function at each point in the set X.

- I.e. define a set F such that

$$F \equiv \{f(x_1), f(x_2), ..., f(x_n)\}.$$

- Then simply plot the set F against the set X.

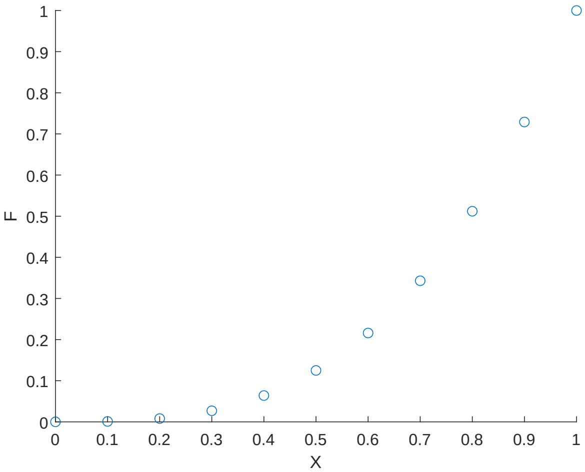
- Let's set $n = 11$ for the $f(x) = x^3$, $x \in [0, 1]$ example.

Approximations: from Continuous to Discrete

```
x_ub = 1;
x_lb = 0;
x_inc = 1/10;
x_vec = [x_lb:x_inc:x_ub];
y_vec = x_vec.^3;
scatter(x_vec,y_vec);
```

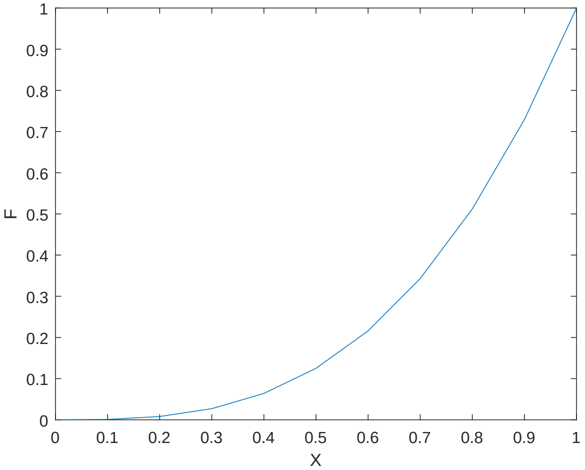
Approximations: from Continuous to Discrete

- With $n = 11$ and then using the *scatter* command.



Approximations: from Continuous to Discrete

- The *plot* command joins the dots.



Roadmap

Takeaways

- That's more-or-less the basics for us to get started with Matlab.
- Should have all the background to do PS0 now.