ECON-GA 1025 Macroeconomic Theory I

Lecture 1

John Stachurski

Fall Semester 2018

Introduction

The first half of ECON-GA 1025 – Macroeconomic Theory I

Lecturer: John Stachurski

- Email: `john.stachurski@gmail.com`
- Office: 625 in 19 W 4th
- Office hours: 4–5pm Mondays or by appointment

TA: Fernando Cirelli

- Email: `fgc235@nyu.edu`

Contact hours

Lectures:

- Times: Monday & Wednesday, 9:30–11:30
- Location: Room 517, 19 West 4th

Recitation:

- Times: Friday 12:30–2:30
- Location: Room 517

Resources

Course notes:

- *Lectures in Quantitative Economics: Theory and Foundations* by John Stachurski and Thomas J. Sargent
- Permanently available at https://github.com/jstac/nyu_macro_fall_2018
- Will change! Don't print!
- Related: https://lectures.quantecon.org/

Supplementary reading:

- *Recursive Macroeconomic Theory* by Lars Ljungqvist and Thomas J. Sargent, MIT Press, fourth edition, 2018, chapters 1-7
    - Due out Sept 11 (according to MIT Press)
- *Recursive Methods in Dynamic Economics* by Nancy Stokey and Robert E. Lucas, Harvard University Press, 1989

Other favorites

- *Analysis for Applied Mathematics* by Ward Cheney, Springer Science, 2013
- *Introduction to Real Analysis* by Robert Bartle and Donald Sherbert, Wiley, 2011

Assessment

Assessment = assignments + exam

Assignments

- Compulsory but not graded
- Must be of reasonable quality
- Work together but submit alone!
- All assignments independently written
- Posted each Sunday, due following Friday before recitation

Exam is on Oct 22nd, more details later...

Topics

- Determinstic dynamics
- Linear stochastic models
- Markov chains
- Search problems
- LQ and discrete decision problems
- Optimal savings and consumption
- The theory of dynamic programming

Mathematics

Two strands

- Analysis
- Probability theory

We will use analysis for solving equations and optimization problems

What is/are the solution/solutions to these equations?

1. $x = ax + b$
2. $x = x + 1$
3. $x^2 = 1$

Now let x be $n \times 1$ and A be $n \times n$

When does this **vector equation** have a solution ?

$$Ax = b$$

When does this vector equation in $\mathbb{R}^n$ have a unique solution?

$$x = Ax + b$$

When does the **method of successive approximations** converge?

1. pick any $x_0 \in \mathbb{R}^n$
2. set $x_{n+1} = Ax_n + b$ for $n = 0, 1, \ldots$

Now let's make it a bit curly:

$$x = (b + (Ax)^{1/\gamma})^{\gamma}, \qquad \gamma > 0$$

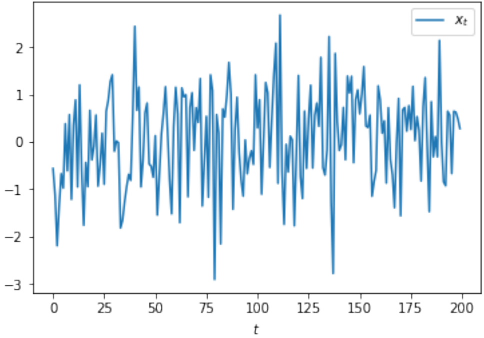
When does this have a solution?

Is it unique?

How would we compute it?

Probability

This sequence is IID



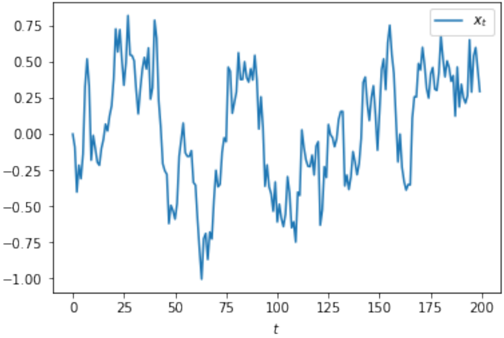
Does it follow that, for some function h, we have

$$\frac{1}{n}\sum_{t=1}^{n} h(x_t) \to \mathbb{E}\, h(x_t) \quad ? \tag{1}$$

If so this is good:

- Left hand side is data
- Right hand side is model
- Now we can compare...

This sequence is **not** IID



Does it follow that, for some function h, we have

$$\frac{1}{n}\sum_{t=1}^{n} h(x_t) \to \mathbb{E}\, h(x_t) \quad ? \tag{2}$$

Programming

Most of the weekly assignments will require programming

Acceptable languages

- Python
- MATLAB

You should try both

Programming Background

A common classification:

- **low** level languages (assembly, C, Fortran)
- **high** level languages (Python, Ruby, Haskell)

Low level languages give us fine grained control

Example. $1 + 1$ in assembly

```
pushq   %rbp
movq    %rsp, %rbp
movl    $1, -12(%rbp)
movl    $1, -8(%rbp)
movl    -12(%rbp), %edx
movl    -8(%rbp), %eax
addl    %edx, %eax
movl    %eax, -4(%rbp)
movl    -4(%rbp), %eax
popq    %rbp
```

High level languages give us abstraction, automation, etc.

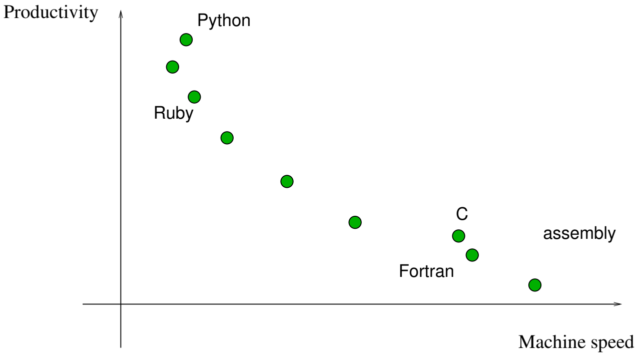
Example. Reading from a file in Python

```
data_file = open("data.txt")
for line in data_file:
    print(line.capitalize())
data_file.close()
```

Jane Street on readability:

> *There is no faster way for a trading firm to destroy itself than to deploy a piece of trading software that makes a bad decision over and over in a tight loop.*
>
> *Part of Jane Street's reaction to these technological risks was to put a very strong focus on building software that was easily understood—software that was readable.*
>
> – Yaron Minsky, Jane Street

Trade-Offs

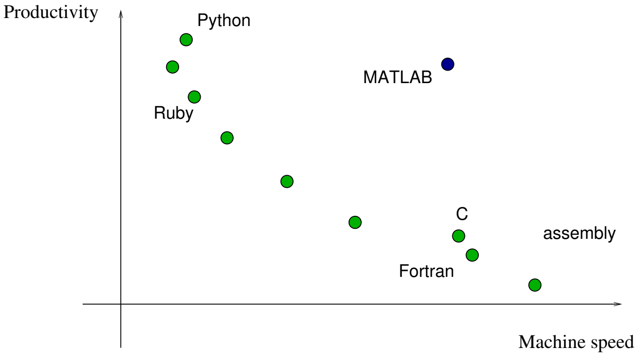

Productivity
Python
Ruby
C
assembly
Fortran
Machine speed

But what about scientific computing?

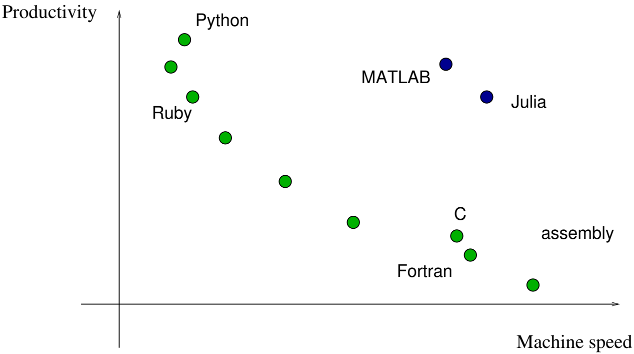
Requirements

- Productive — easy to read, write, debug, explore
- Fast computations

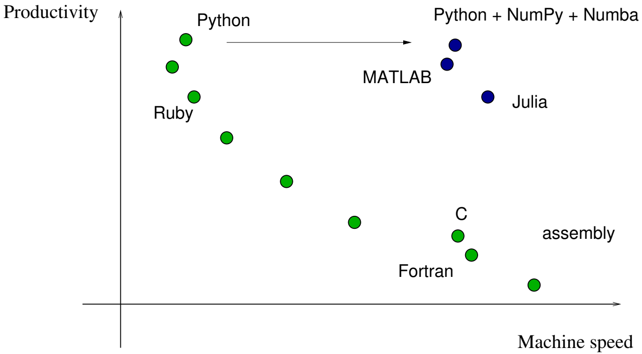
Trade-Offs


Productivity
Python
MATLAB
Ruby
C
assembly
Fortran
Machine speed

Trade-Offs


Productivity
Python
MATLAB
Julia
Ruby
C
assembly
Fortran
Machine speed

Trade-Offs

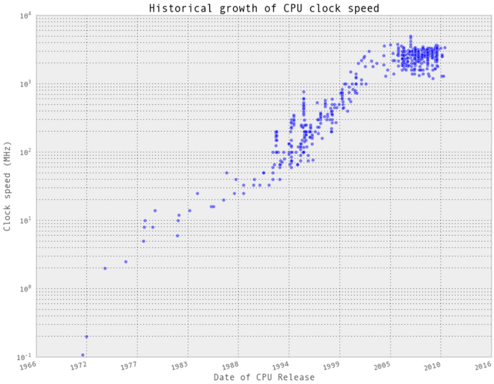

Productivity
Python
Python + NumPy + Numba
MATLAB
Julia
Ruby
C
assembly
Fortran
Machine speed

Key Takeaways

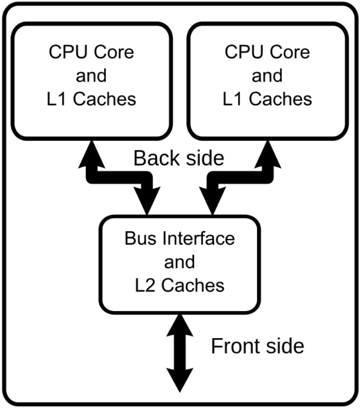
- Don't write in C / C++ / Fortran, no matter what your professor says
- JIT compilation is changing scientific computing
- Same with parallelization
- New algorithms, new techniques — and opportunities

Programming Background — Hardware

CPU frequency (clock speed) growth is slowing



Chip makers have responded by developing multi-core processors



Source: Wikipedia

Exploiting multiple cores / threads is nontrivial

Sometimes we need to redesign algorithms

Sometimes we can use tools that automate exploitation of multiple cores

Why Recursive Methods?

This course is an introduction to **recursive** methods for economic analysis

Example.

- Recursive Macroeconomic Theory by LL and TJS
- Recursive Methods in Dynamic Economics by NS and REL

Recursive methods are used to solve high dimensional optimization and equilibrium problems

Breaks the problem down into smaller steps

Helps tackle the **curse of dimensionality**

Example. A typical problem from undergraduate choice theory:

Choose consumption at time 0 and 1 to solve

$$u(c_0) + \beta u(c_1) \tag{3}$$

subject to

$$c_1 \leqslant R(y_0 - c_0) \tag{4}$$

If u is concave, strictly increasing and differentiable, then the unique solution is found by taking the c_0 that satisfies

$$u'(c_0) = \beta R u'(R(y_0 - c_0)) \tag{5}$$

In general, undergraduate style optimization problems are relatively easy

- All functions are differentiable
- Few choice variables (low dimensional)
- Concave (for max) or convex (for min)
- First order / tangency conditions relatively simple

But PhD macro / PhD research problems are harder...

Possibilities:

- High dimensions
- Can't take derivatives
- No analytical solution for FOCs
- Neither concave nor convex — local maxima and minima

Many interesting research problems have these features

Example. A typical problem from graduate macroeconomic theory:

Choose consumption at time $t = 0, 1, \ldots$ to solve

$$\mathbb{E} \sum_{t=0}^{\infty} \beta^t u(c_t), \tag{6}$$

subject to

$$w_{t+1} = R_t(w_t - c_t) + y_t \tag{7}$$

An infinite dimensional problem because we must choose $c_0, c_1, \ldots$

And stochastic!

Can Computers Save Us?

For any function we can always try brute force optimization

Here's an example for the following function

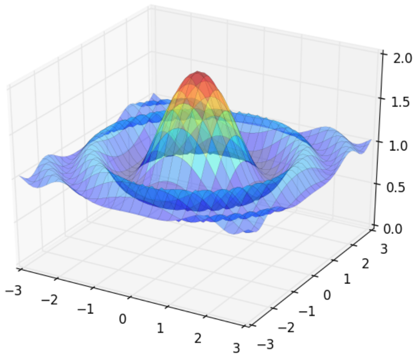


Figure: The function to maximize

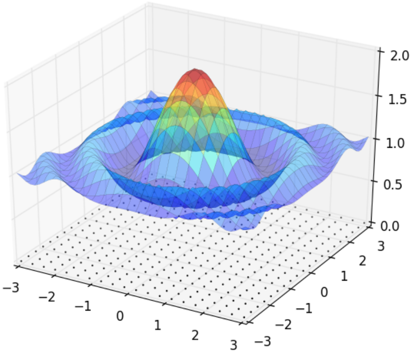


Figure: Grid of points to evaluate the function at

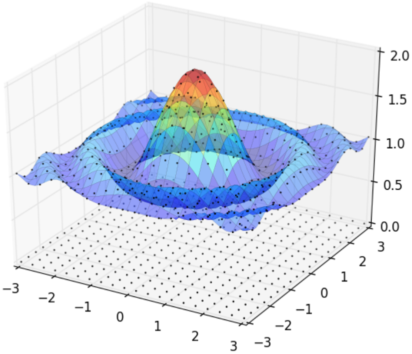


Figure: Evaluations

Grid size $= 20 \times 20 = 400$

Outcomes

- Number of function evaluations $= 400$
- Time taken $=$ almost zero
- Maximal value recorded $= 1.951$
- True maximum $= 2$

Not bad and we can easily do better

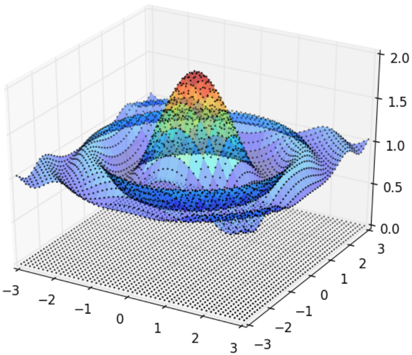


Figure: $50^2 = 2500$ evaluations

- Number of function evaluations $= 50^2$
- Time taken $= 400\ \mu$s
- Maximal value recorded $= 1.992$
- True maximum $= 2$

So why even study optimization?

The problem is mainly with larger numbers of choice variables

- 3 vars: $\max_{x_1,x_2,x_3} f(x_1, x_2, x_3)$
- 4 vars: $\max_{x_1,x_2,x_3,x_4} f(x_1, x_2, x_3, x_4)$
- $\dots$

If we have 50 grid points per variable and

- 2 variables then evaluations $= 50^2 = 2500$
- 3 variables then evaluations $= 50^3 = 125,000$
- 4 variables then evaluations $= 50^4 = 6,250,000$
- 5 variables then evaluations $= 50^5 = 312,500,000$
- $\dots$

Example. Recent study: Optimal placement of drinks across vending machines in Tokyo

Approximate dimensions of problem:

- Number of choices for each variable $= 2$
- Number of choice variables $= 1000$

Hence number of possibilities $= 2^{1000}$

How big is that?

```
In [10]: 2**1000
Out[10]:
1071508607186267320948425049060001810561404811 70
5533607443750388370351051124936122493198378815 69
5858127594672917553146825187145285692314043598 45
7757469857480393456777482423098542107460506237 11
4187795418215304647498358194126739876755916554 39
4607706291457119647768654216766042983165262438 68
37205668069376
```

Let's say my machine can evaluate about 1 billion possibilities per second

How long would that take?

```
In [16]: (2**1000 / 10**9) / 31556926  # In years
Out[16]:
339547840365144349278007955863635707280678989995
899349462539661933596146571733926965255861364854
060286985707326991591901311029244639453805988092
045933072657455119924381235072941549332310199388
301571394569707026437986448403352049168514244509
939816790601568621661265174170019913588941596
```

What about high performance computing?

- more powerful hardware
- faster CPUs
- GPUs
- vector processors
- cloud computing
- massively parallel supercomputers
- ...

Let's say speed up is 10^{12} (wildly optimistic)

```
In [19]: (2**1000 / 10**(9 + 12)) / 31556926
Out[19]:
339547840365144349278007955863635707280678989958
993494625396619335961465717339269652558613648540б
028698570732699159190131102924463945380598809204 5
```

For comparison:

```
In [20]: 5 * 10**9 # Expected lifespan of sun
Out[20]: 5000000000
```

Message: There are serious limits to computation

What's required is clever analysis

Exploit what information we have

- without information (oracle) we're stuck
- with information / structure we can do clever things

Getting Started with Python

See https://lectures.quantecon.org/py/getting_started.html

Jupyter notebooks

- How to use
- Markdown
- LaTeX
- Getting help

Homework

Go to `https://lectures.quantecon.org/py/index.html`

Study the following lectures **before** Friday's recitation

- Setting up your Python Environment
- An Introductory Example
- Python Essentials
- OOP
- NumPy
- Matplotlib
- SciPy

Bring your laptop on Friday