High Performance Python Lab
Lecture 1

Prof. Sergey Rykovanov
Head of the AI&Supercomputing Lab
CAIT Skoltech
Term 2 / 2021

Skoltech

Course logistics

- Mondays 9:00 — 12:00 Lectures / Practical sessions
- Thursdays 9:00 — 12:00 Lectures / Practical sessions

Typically <1 hour material followed by programming lab

Activity Type	Activity weight, %
Class participation	10
Homework Assignments	30
Computer Labs	30
Final Project	30

Recommended textbook:
Micha Gorelick,
Ian Ozswald
"High Performance Python"
2nd edition (2020)

Other resources:
stackoverflow.com
medium.com
coursera.org
udacity.com
youtube.com

Your fellows and TAs

Skoltech

Course prerequisites

Prerequisites:
- Laptop with installed Python (preferably Linux or Unix-based)
- Installed ssh - client
- Basic knowledge of Python
- Great mood and smiles

Preferably:
- Knowledge of Unix-type systems (working in terminal)
- Deep knowledge of Python, knowledge of C/C++
- Jupyter notebook, numpy, scipy, matplotlib knowledge and have it installed
- Basic undergrad-level mathematics (calculus, linear algebra, ODEs and PDEs)

You will have a chance to work on one of Russia's most powerful supercomputers "Zhores"

But, you'll be fine with Google Colab as well ;)

Lectures and topics (tentative, this is a live course)

1. Introduction, logistics, crash course on Python
2. Crash course on Python continued, crash course on using supercomputers
3. Introduction to computer architecture, building parallel intuition, code profiling
4. Classification of parallel algorithms, Python multi-threading, Python multi-processing, Jupyter parallel
5. Distributed systems, MPI, mpi4py, examples
6. Advanced MPI on Python
7. GPUs, CUDA, pyCuda, cupy
8. Accelerating Python with Numba
9. Visualization in Python
10. **Cython & Pybind11**
11. PyTorch distributed & Dask (?)
12. Libraries and other languages (jax, Julia) (?)
13. Projects
14. Projects

Goals of the course

Not to be afraid of programming (in Python and overall)

Not to be afraid of supercomputers.

Be able to write parallel programs on modern multi-core CPUs and GPUs.

Learn how to use large computing infrastructure.

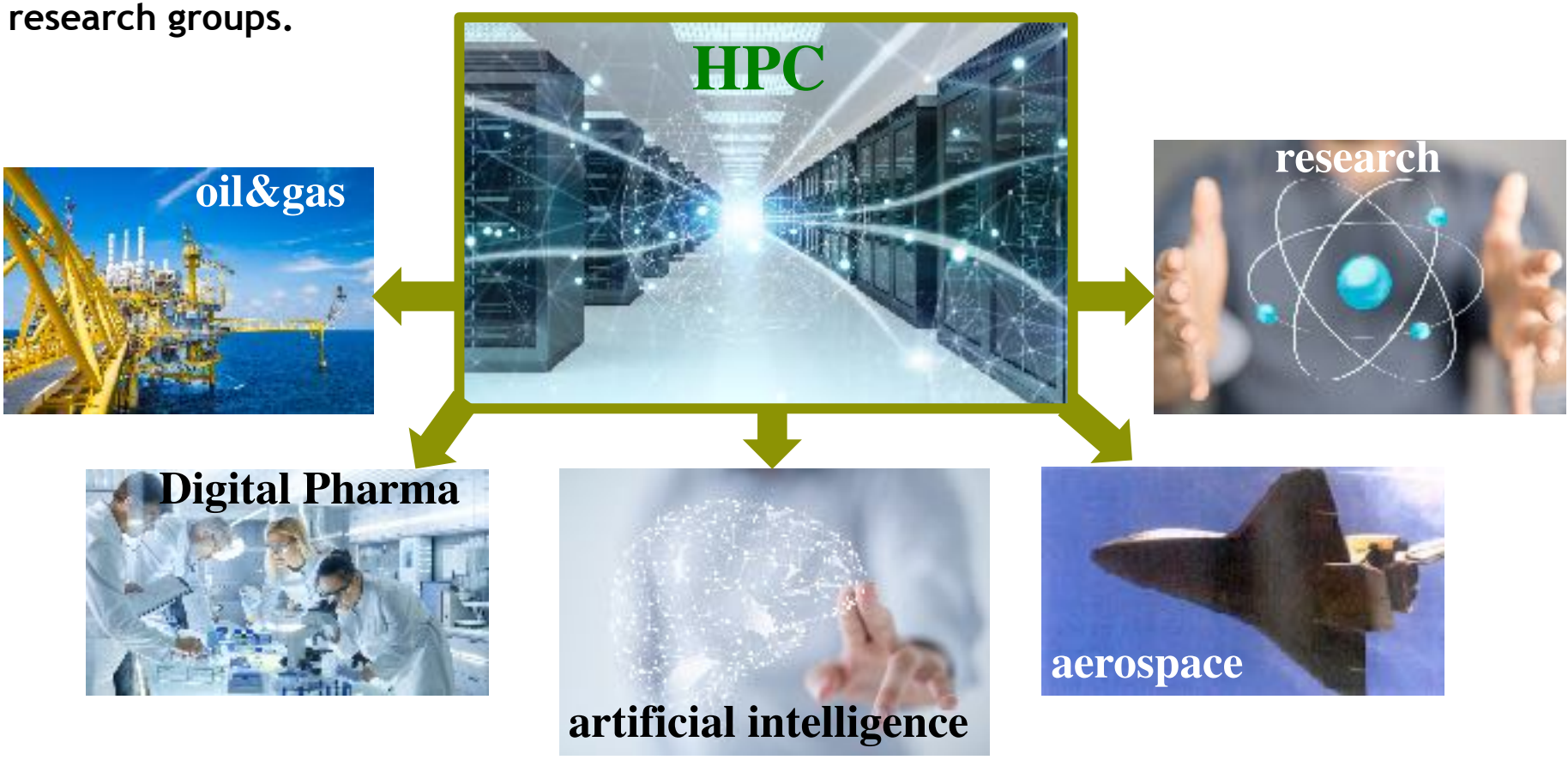
Have fun learning. (We will have various examples / assignments)

Finish with a cool project that involves Python code optimization.

General introduction: HPC in Skoltech, “Zhores”

“To out-compete is to out-compute”

HPC provides competitive advantage for industrial and business companies and research groups.



Supercomputing enables most of the projects of CDISE

Skoltech

What is HPC?

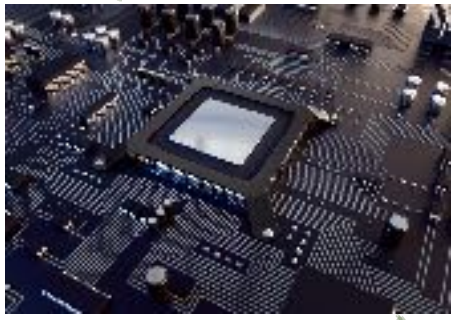
What is HPC?

Aggregation of power well beyond your typical laptop/desktop

High Performance Computing has to be holistic

Application, innovation, scientific breakthrough

Mathematical model



Computing architecture and methods

Datacenter infrastructure

Supercomputer hardware infrastructure

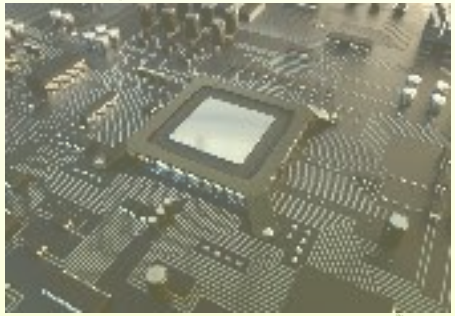
Supercomputer software infrastructure

High Performance Computing has to be holistic

Application, innovation, scientific breakthrough

High Performance Python Lab

Mathematical model



Computing architecture and methods

Datacenter infrastructure

Supercomputer hardware infrastructure

Supercomputer software infrastructure

CDISE “Zhores” supercomputer for data-driven modeling, Big Data and AI – one of the top computers in Russia

“Zhores” supercomputer with energy efficient hybrid architecture:

- 74 compute nodes
- 26 compute nodes with powerful graphic cards (each node with 4xNVidia Tesla V100, NVLink + RDMA)
- Tensor cores ideally suit **the machine learning (deep learning)** applications
- **energy efficient** – only 90 kWatt electricity consumption
- **1 Pflop/s computational power**
- **0.5 Pbytes storage system**
- **#7 most powerful supercomputer in Russia**
- was installed and is currently maintained and administered by our own small but highly professional team (HPC & Big Data Lab)

”Zhores” is an unique Russian supercomputer capable of tackling a variety of interdisciplinary problems uniting machine learning, data science and mathematical modeling. It has a lot of potential to provide scientific breakthroughs in such areas as Digital Pharma, Biomedicine, Computer Vision, Photonics, New Materials Discovery, New X-Ray and Gamma-ray sources etc.

ZHORES
SUPERCOMPUTER
HIGH PERFORMANCE COMPUTING AND BIG DATA
SKOLTECH CDISE

Skoltech
Skolkovo Institute of Science and Technology

“Zhores” is a critical CDISE facility

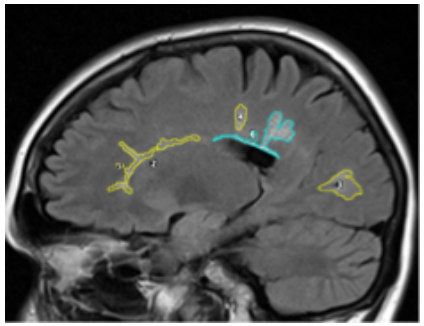
Operation officially started in Nov. 2018

Some numbers for 2020:

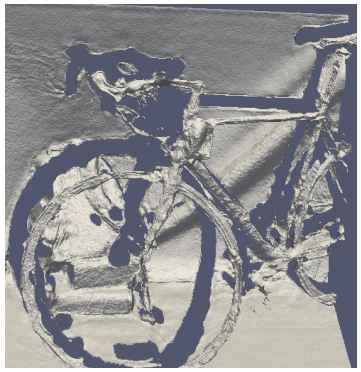
- Scientific papers citing “Zhores” – ~40 (including 4 Nature Index papers)
- Enabled research and industry grants worth ~70 Mln Rubles (~1 Mln $)
- Accomplished MSc and/or PhD theses - 5
- Media articles about “Zhores” and research >40
- Hackathons using “Zhores” - 2
- Total projects - ~100
- Total users >200

“Zhores” supercomputer:

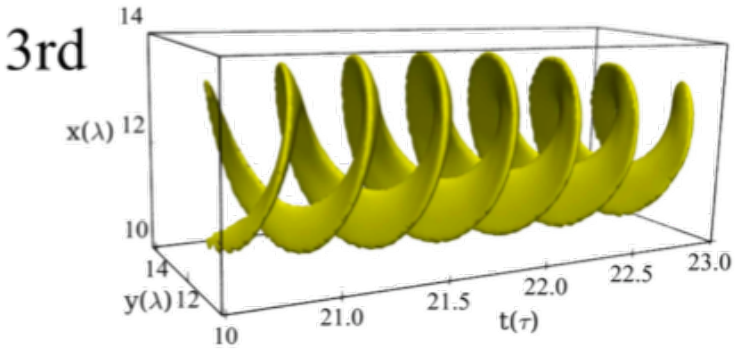
- Critical tool for CDISE researchers
- Tool for grant generation
- Dedicated HPC team:
  - helping with code optimization and all computational activities on “Zhores”
  - Tuning / modifying computing infrastracture per project
- **Upgrade planned**



Neurohackaton



DL for 3D objects



OAM in Plasma Photonics

ZHORES SUPERCOMPUTER
HIGH PERFORMANCE COMPUTING AND BIG DATA
SKOLTECH CDISE

Skoltech
Skolkovo Institute of Science and Technology

List of topics of HPC&Big Data Lab + more

- **HPC in photonics and plasma physics (Prof. Sergey Rykovanov)**
  - Nanophotonics and nanolasers
  - metamaterials & OAM splitters
  - UV light with OAM
  - Novel X-ray, gamma sources for materials (spintronics)
  - Computational plasma physics (laser physics, astrophysics, plasma photonics)
- **Neurointerfaces and inverse problems in geophysics and electromagnetic (Prof. Nikolay Koshev)**
  - Fast and efficient solvers
  - Inverse and ill-posed problems
- **Numerical and computational methods (Prof. Rykovanov, Prof. Koshev)**
  - Hybrid parallel computing (CPUs, GPUs, FPGAs…)
  - Deep code optimisation
  - Novel numerics for photonics
- **Supercomputing infrastructure research (Zacharov, Panarin, Maliutin, Shkandybin)**
  - Novel cooling techniques / immersive cooling
  - Novel energy-efficient architectures
  - Intelligent datacenter & supercomputer monitoring
  - AI & HPC on FPGAs
  - Virtualisation, Containers, Kubernetes for HPC

Skoltech

List of topics of HPC&Big Data Lab + more

- **HPC in photonics and plasma physics (Prof. Sergey Rykovanov)**
  - Nanophotonics and nanolasers
  - metamaterials & OAM splitters
  - UV light with OAM
  - Novel X-ray, gamma sources for materials (spintronics)
  - Computational p
- **Neurointerfaces and**
  - Fast and efficie
  - Inverse and ill-
- **Numerical and comp**
  - Hybrid parallel
  - Deep code opti
  - Novel numerics for photonics
- **Supercomputing infrastructure research (Zacharov, Panarin, Maliutin, Shkandybin)**
  - Novel cooling techniques / immersive cooling
  - Novel energy-efficient architectures
  - Intelligent datacenter & supercomputer monitoring
  - AI & HPC on FPGAs
  - Virtualisation, Containers, Kubernetes for HPC

ev)

If interested in doing research in HPC&Big Data Lab, please contact:
s.rykovanov@skoltech.ru
r.zagidullin@skoltech.ru
n.koshev@skoltech.ru (neurointerfaces)

Survey and “Zhores” account (2-5 minutes)

Please fill out survey:

https://bit.ly/35tSKUR

You will need an account on “Zhores”, please fill in the form:

https://bit.ly/3ksvXir

Skoltech
Skolkovo Institute of Science and Technology

Python: history, philosophy, use cases

What is Python?

What is Python? Executive Summary

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

https://www.python.org/doc/essays/blurb/

What is Python?

interpreted	*no need for a compiling stage*
object-oriented	*programming paradigm that uses objects (complex data structures with methods)*
high level	*abstraction from the way machine interprets and executes*
dynamic semantics	*can change meaning on-the-fly*
built-in	*core language (not external)*
large collection of libraries	*you can do almost anything*

Development history

- started over the Christmas break in 1989 by Guido van Rossum
- developed in the early 1990s
- name comes from **Monty Python's Flying Circus**
- Guido is the Benevolent Dictator for Life (BDFL), meaning that he continues to oversee Python's development

Development history

- Open-source development from the start (BSD license now)
- Relies on large community input and 3rd party add-on software
- Version 2.0 (2000), 2.6 (2008), 2.7 (2010)
- Version 3.X (2008) is not backward compatible, but 2.7 code is “easily” migrated to 3.X
- Latest stable version 3.10.0 (October 2021)

Alternatives

C, C++, Fortran

Pros: great performance, legacy scientific computing codes
Cons: syntax is not optimised (at least not yet) for causal programming, no interacting facilities, difficult visualisation, text processing, etc

Mathematica, Maple, Matlab, IDL

Pros: interactive, great visualisation, extensive libraries
Cons: cost a lot, proprietary, unpleasant for large-scale programs and non-mathematical tasks

Why Python?

- Free (BSD license), highly portable (Linux, OSX, Windows…).
- Interactive interpreter provided.
- Extremely readable syntax ("executable pseudo-code").
- Simple: non-professional programmers can use it effectively
  - great documentation
  - total abstraction of memory management
- Object-oriented model, but not mandatory.
- Reach built-in types: lists, sets, dictionaries, strings,…
- Very comprehensive standard library.
- Standard libraries for Matlab-like arrays (NumPy).
- Scientific algorithms in SciPy.
- Easy to wrap existing C,C++ and Fortran codes.

Why Python?

Amazingly Scalable

interactive experimentation
build small, self-contained scripts or million-lines projects
from occasional to full-time use
really can do anything you want, with impressive simplicity

Performance, if you need it

As an interpreted language, Python is slow.
But…

High Performance Python Lab

Why Python?

http://www.quora.com/Python-programming-language-1/Which-Internet-companies-use-Python

Skoltech

Boot-camp today and tomorrow

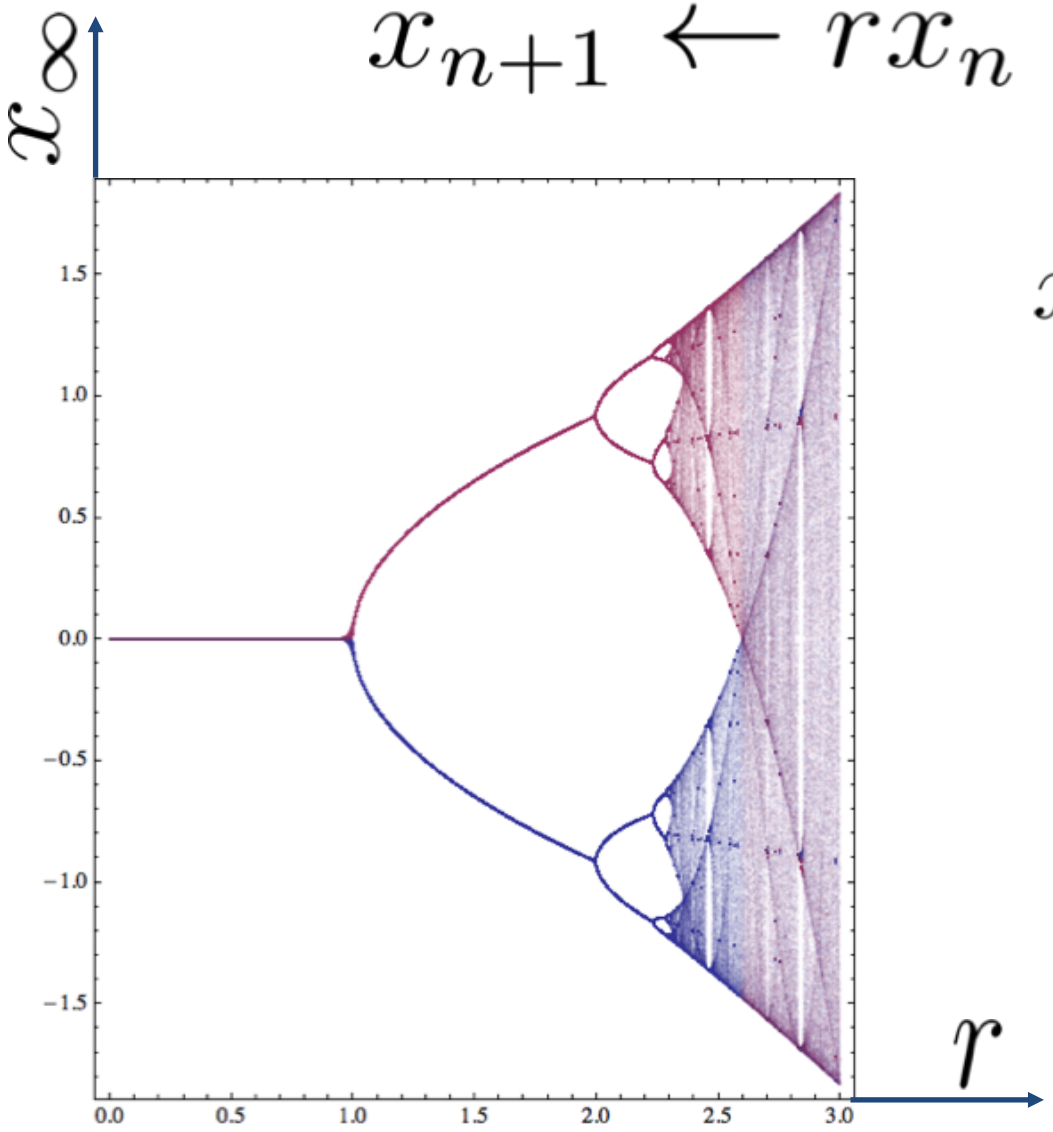
- Visualization tools (matplotlib)
- Let us practice:
  - Naive Matmul + bootcamp for beginners
  - Bifurcation map (Logistic map)
  - Julia set
  - Schelling model (cellular automata)
  - Spectrogram (wavelet for additional points)

Visualization tools

- Matplotlib:
  - matplotlib.pyplot
  - matplotlib.animation
  - matplotlib.cm
- Some other options:
  - seaborn
  - pyqtgraph

Task 1. Bifurcation map

$$x_{n+1} \leftarrow r x_n (1 - x_n)$$



$$x_0 \leftarrow \mathrm{rand}(0, 1)$$
$$r \leftarrow \mathrm{const}$$

Task 1. Bifurcation map

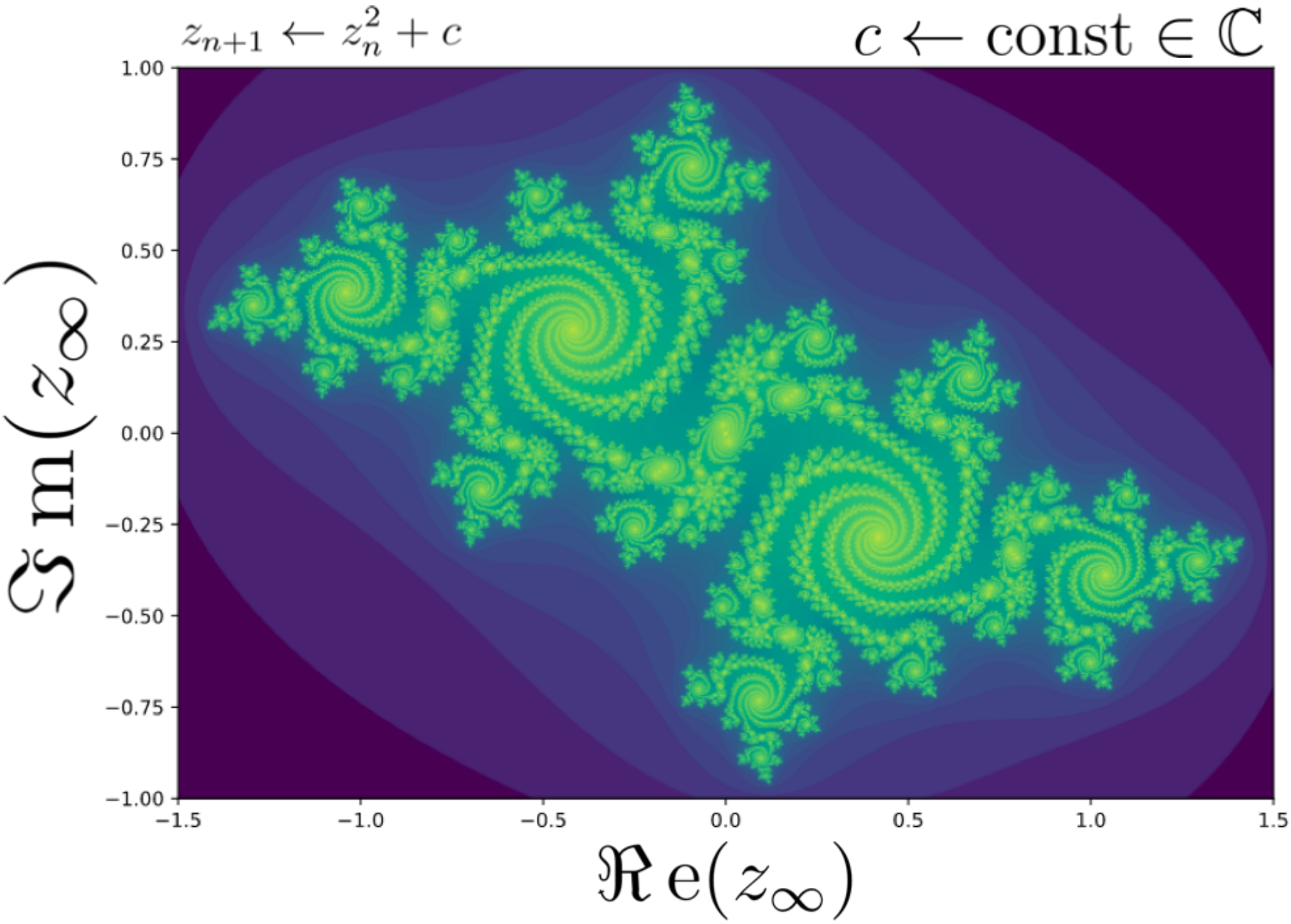
Step-by-step guide:

- implement the map, plot the evolution of x
- play around with values of r, see the change of evolution
- then create a linspace of r's, for every r save last "m" values of x after first "n" values (can be m=200, x=200), play around with values
- Get the bifurcation map
- vizualize the evolution (play around)

$$x_{n+1} \leftarrow r x_n (1 - x_n)$$
$$x_0 \leftarrow \mathrm{rand}(0, 1)$$
$$r \leftarrow \mathrm{const}$$

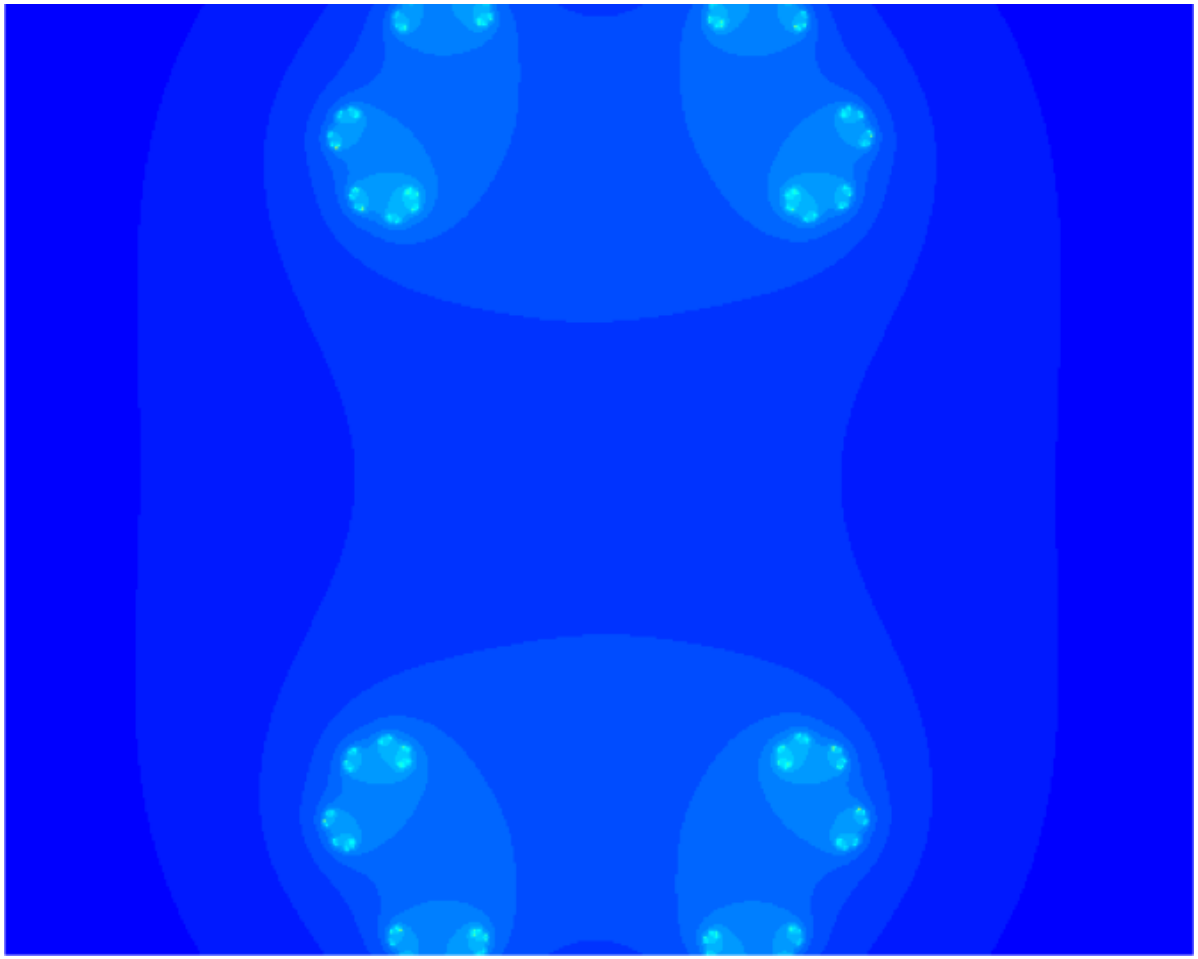
if you are interested - describe real physical systems that exhibit bifurcation

Task 2. Julia set



Task 2. Julia set – 2

$c \leftarrow 0.7885\,\mathrm{e}^{\mathbf{i}a}$ $a \leftarrow \mathrm{range}(0, 2\pi)$



Task 2. Julia set

Step-by-step guide:

$$z_{n+1} \leftarrow z_n^2 + c$$

- fix a value of C (can be 0)
- implement the map, plot the evolution of z (Re(z), Im(z))
- play around with values of z0, see the change of evolution
- for a given z0 if the sequence converges use black color, if it exponentially diverges use white color, if it starts jumping between “n” values use different colors
- plot the Julia set
- C=0 - Mandelbrot set

Question?

Julia set is self-similar (fractal)

Mandelbrot set is self-similar (fractal)

What is B in:

Benua B. Mandelbrot?

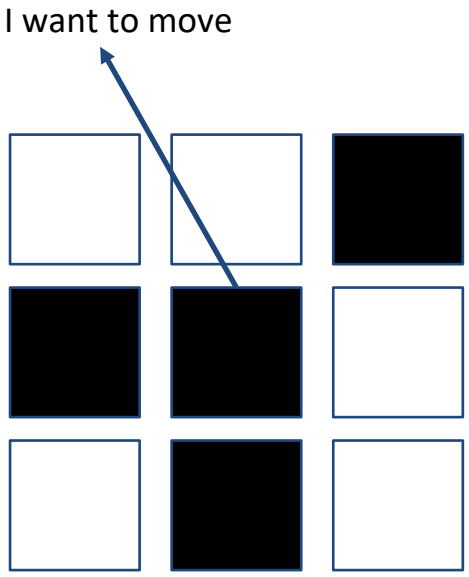
Task 3. Schelling's Model

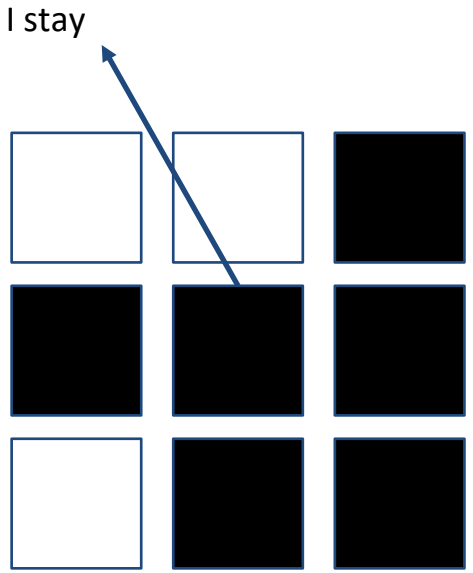
1) Suppose there are two types of agents: X and O. Two populations of the two agent types are initially placed into random locations of a neighborhood represented by a grid. After placing all the agents in the grid, each cell is either occupied by an agent or is empty.
2) Now we must determine if each agent is satisfied with its current location. A satisfied agent is one that is surrounded by at least t percent of agents that are like itself. This threshold t is one that will apply to all agents in the model.
3) When an agent is not satisfied, it can be moved to any vacant location in the grid. Any algorithm can be used to choose this new location. For example, a randomly selected cell may be chosen, or the agent could move to the nearest available location.
4) All dissatisfied agents must be moved in the same *round*. After the round is complete, a new round begins, and dissatisfied agents are once again moved to new locations in the grid.

Task 3. Schelling's Model

Segregation model (black and white households)

- map is large (i.e. 100x100 with periodic boundaries)
- each household on a map has 8 neighbours
- the agent decides to randomly move if less than R (you can change that) neighbours are of the same color





Task 3. Schelling's Model

Segregation model (black and white households)

- map is large (i.e. 100x100 with periodic boundaries)
- each household on a map has 8 neighbours
- the agent decides to randomly move if less than R*8 (you can change that) neighbours are of the same color

Steps:
Randomize the map with half white/half black
define the value of R (0, ⅛, 2/8, ⅜, 4/8, ⅝,6/8,⅞, 1)
start the game
on each step for each cell find out if the cell wants to move - this cell is now on the market (considered free for moving in)
after you finished with the whole map - cells that want to move can move into free cells
repeat the whole procedure