Mid-term projects for

Advanced Introduction to C++, Scientific Computing and Machine Learning

Claudius Gros
2021/22

December 13, 2021

1 Formalities

The projects should be performed by groups of maximally three people, with the results being handed in as files

- A pdf-report of 5-8 pages containing an introduction to the subject and to the numerical methods used, a description of the programming effort (structure and core code snippets, but not all the code), and the presentation of the results. For the latter we expect both figures of simulated data and a statistical evaluation of any reasonable property that you choose.
- The documented code.
- Additional media files (optional) referred-to in the report. You may also generate gif files from series of images to visualize what is going on, in case your project involves time varying quantities.

Do not use libraries or external codes except for those used and presented in the lecture, if not explicitly asked by the project. We expect you to write the code simple and efficient on your own.

- The date to hand-in is: **January 31, 2022.**
- All groups present their results at the end of the term. If there is time, selected groups may present their results during the lecture hours, otherwise during the tutorials.

You are welcome to consult with the teaching assistant responsible for the project in case you need help.

1.1 Selection / Assignment

Every group should select a different project. In order to achive this, please send a mail to Oren Neumann (email on the webpage), as soon as possible. Please indicate the order of your preferences (1,2,3). In the last lecture before Christmas we will then settle eventual conflicting project selection.

2 Projects

You are welcome to propose your own project. This has to be down however well in advance.

- Contact your tutor, discuss content and workload.
- In case, submit an outline following the style of the projects below.

Your proposal will be included in this list (and reserved), if accepted. Proposals not included in this list cannot be selected.

2.1 Dynamic programming: Why non-rational behavior may be rational

(Oren Neumann)

Seemingly non-rational behavior may arise when maximizing the chance to survive in ecological models. Dynamic programming (1) is a method to evaluate optimal strategies in complex decision environments. The final goal is to show that a choice of survival strategy can violate transitivity when the available strategies are limited, see (2). Therefore, implement a class for a choice-making animal that is exposed to predation and starvation risks. The animal has a certain energy level as an internal state, being in the lowest state leads to starvation. The level of energy has an influence on the choice of the animal. Assume that its survival is certain if it is alive on the last day of winter (day 0). Then calculate its probability of survival for every previous winter day (day $-N$) using dynamic programming. Assume the animal always takes the best possible choice available.

Examine the strategies chosen at each energy level. Do they change in time or become constant? Show that the choice of strategy becomes intransitive when only two out of three strategies are available.

(1) Complex decisions made simple: a primer on stochastic dynamic programming

(2) Violations of transitivity under fitness maximization

2.2 Reinforcement learning with deep neural networks

(Oren Neumann)

Note: This is a Python project, it requires previous experience working with Python.

Reinforcement learning is a field of machine learning where the model interacts with an environment and has to learn how to maximize its reward. Q-learning (1) is a classic reinforcement learning algorithm, proven very effective over varied environments especially when guided by a deep neural network, a combination called DQN (Deep Q Network).

Use this (2) tutorial to train a PyTorch neural net to solve the OpenAI Gym (3) Cartpole game, learning to balance a pole upright on a moving cart:

- Write a fully functional training scheme and present all its parts. Train a model to solve the game, surviving for more than 195 time steps on average. Produce a sample training graph and videos of different stages of the learning process.
- Change the length of the pole and check if training becomes easier/harder for shorter or longer poles. Plot the average training time as a function of pole length.
- **Optional:** Save a model trained on the default settings and plot the score it achieves with longer/shorter pole lengths, without training on the new length. You will need to freeze the optimization process and play 100 games, averaging over their score.

(1) Q-learning

(2) DQN tutorial with PyTorch

(3) OpenAI Gym

2.3 Special function evaluation and analytical continuation

(Youjiang Xu)

This project covers two important topics in complex analysis: evaluating special functions and performing analytical continuation.

Special functions are non-elementary functions with established names to remark their importance. In physics, special functions can appear as solutions to differential equations, e.g., Legendre functions solves the Maxwell equations in electrodynamics, or they can also come from integration, e.g., Bose-Einstein distribution gives physical observables in terms of Riemann zeta function. In order to numerically calculate the special functions, one must notice that each special function has various definitions, which have different efficiency and accuracy for evaluation, and some of them are applicable only in a limited region on the complex plane. Therefore, one must carefully choose the proper definition to carry out the calculation, in consideration of the domain one is interested in. In the first section of this project, the students are asked to evaluate $\Gamma(z)$, the extension of the factorial function to complex numbers, using power series, infinite product, and contour integral respectively, and to compare the performances of the methods.

In evaluating functions, it often happens that one only knows the values or definition of a function in a small region on the complex plane, and wants to evaluate the function in a broader region. This goal can be achieved by analytic continuation. The simplest case is that one knows the exact Taylor expansion of a function in a finite disk, then one could move the center of expansion to another point, get a new Taylor series order by order, which is valid in a new, larger disk. Furthermore, one could use some tricks to improve the convergence of the new series (1). However, it happens in practice that one only knows the values of the function at finite number of points, but still wants to perform analytic continuation. Although this 'numerical analytic continuation' problem is actually ill-defined mathematically, but people have come up with methods to approximate the solutions. In the second half of the project, the students are asked to perform analytic continuation of a function, whose poles locate on the real axis, from the upper plane to the lower plane, and to learn the simplified maximum entropy method (2) for solving a physics-related 'numerical analytic continuation' problem.

2.3.1 Evaluate the Gamma function

(Youjiang Xu)

Calculate $\Gamma(z)$ using the following definitions:

- Power series (not applicable when $\Re z < 0$ and $\Im z = 0$, i.e., z can't be negative numbers):

$$\ln\Gamma(z) = \left(z - \frac{1}{2}\right)\ln z - z + \frac{1}{2}\ln z + \sum_{k=1}^{n} \frac{B_{2k}}{2k(2k-1)} z^{-2k+1} + O\left(z^{-2n-1}\right),$$

  where B_k's are Bernoulli numbers defined by the following recurrence relations:

$$\begin{aligned} B_0 &= 1, \\ \sum_{k=0}^{n-1} \frac{B_k}{k!\,(n-k)!} &= 0. \end{aligned}$$

- Infinite product:

$$\Gamma^{-1}(z) = z e^{\gamma z} \prod_{n=1}^{\infty} \left\{ \left(1 + \frac{z}{n}\right) e^{-z/n} \right\},$$

  where γ is the Euler constant.

- Contour integral:

$$\Gamma(z) = \oint e^{-t} t^{z-1} dt,$$

  where the contour is given by: i. t goes from ∞ to 0 along the real axis (slightly above, with an infinitesimal positive imaginary part); ii. t circles original point by π; iii. t goes from 0 to ∞ along the real axis (slightly below, with an infinitesimal negative imaginary part).

Compare the performances of above methods, e.g., which one is most accurate given finite number of elementary arithmetic operations, which one performs better when $|z|$ is large or small, or how the accuracy is changed when the contour (in the third definition) is getting closed to the real axis.

Finally, we have some optional tasks. Students are encouraged to calculate the exponential and logarithmic functions using elementary arithmetic, to evaluate the integral using their own codes, and to implement arbitrary-precision algorithms.

2.3.2 Analytic continuation of the Green's function

(Youjiang Xu)

Consider the following function which resembles the Green's function in many-body physics:

$$f(z) = \sum_{k=-N}^{N} \frac{w_k}{z - k/N},$$

where the weights $w_k \in [0, 1)$ are summed to 1:

$$\sum_{k=-N}^{N} w_k = 1.$$

In practice, we don't know the exact values of w_k. Instead, we have some limited information about $f(z)$, e.g., some of its values on the imaginary axis, and we want to find out its values on the real axis.

Consider the following simplified scenario: the program randomly generates w_k's, which are hidden, but it gives the Taylor coefficients of $f(z)$ around $z = i\pi$

$$f(z) = \sum_{k=0}^{\infty} C_{i\pi}^{(k)} (z - i\pi)^k .$$

We know the Taylor series is only valid in the circle $|z - i\pi| < \pi$. In order to find the value of $f(\pi)$, one could change the basis of the Taylor series from $z = i\pi$ to $z = \frac{\pi}{2} + i\pi$ by calculating the coefficients $C_{\frac{\pi}{2}+i\pi}^{(k)}$'s from $C_{i\pi}^{(k)}$'s, and then calculate $C_{\pi+i\pi}^{(k)}$'s, whose corresponding Taylor series is valid at $z = \pi$. Find out the efficiency and accuracy of this naive analytic continuation. Take care of the truncation error.

A more advanced way to do analytic continuation is to use conformal mappings. For example, the following transformation is called a linear fractional transformation:

$$w = \frac{\alpha z + \beta}{\gamma z + \delta}.$$

Find a proper set of $(\alpha, \beta, \gamma, \delta)$ to obtain a new Taylor series in terms of w at $w = 0$ to evaluate $f(\pi)$.

Last but not least, in a more realistic scenario, we only know the values of $f(\omega_k := (2k + 1)\, i\pi)$ for $k = 0, 1, \ldots, M < N$ and we want to calculate w_k's. This problem is ill-defined because there are more $(2N + 1)$ variables than (M) equations (In application, the problem is ill-defined even if M is large, because the values of $f(\omega_k)$'s are subject to errors, while w_k's are sensitive to those errors). To acquire a reasonable solution, one must set another constraint that the solution, denoted as $\hat{w}_k$'s, should satisfy. Here comes the principle of maximum entropy. If we interpret w_k's as probabilities, the $\hat{w}_k$'s we find should maximize the entropy

$$S(\hat{w}) = -\sum_{k=-N}^{N} \hat{w}_k \log \hat{w}_k,$$

subject to the constraint

$$f(\omega; w) = f(\omega; \hat{w}).$$

So the problem can be stated in terms of Lagrange multiplier

$$L(\hat{w}; \lambda) = S(\hat{w}) - \lambda_\omega \left(f(\omega; \hat{w}) - f(\omega; w) \right).$$

Solve this problem in the following cases:

- w_k's are purely random like before

- We randomly pick up three k's whose corresponding w_k's are $1/3$, while the other w_k's are 0
- We make $w_k = A(m)\left(1 - k^2/m\right)$ if $k^2/m < 1$ or 0 if $k^2/m \geq 1$. Here m is a random positive number and $A(m)$ is the normalization constant.

In which case the maximum entropy method performs better?

(1) Use conformal mapping to do analytic continuation

(2) Bayesian inference and the analytic continuation of imaginary-time quantum Monte Carlo data

2.4 How do the traffic signals work?

(Youjiang Xu)

In this project, we study the role the traffic signals play by simulating the traffic network in real time. First, students are asked to realize a simple traffic network described as the basic goal and to study the effect of traffic signals. Then, the students are encouraged to implement the features described in the optional goals into the system to make it more realistic.

- Basic goal: The simplest traffic network consists of $N \times N$ blocks and the road map forms a square lattice with $(N+1) \times (N+1)$ vertices or traffic junctions. On each street connecting two vertices, there are U units which randomly generate vehicles during the real-time simulation. When a vehicle is generated, it randomly chooses a unit in the system as its destination, and find a shortest path to move towards the destination. The roads have uniform capacity (maximal number of vehicles per unit length) and speed limit. If there is only one car moving freely on the map, the time for travel would simply be the distance divided by the speed limit. However, the existence of the traffic junctions can increase the travel time if the car stops at some of them. The averaged ratio between the realistic travel time and the ideal travel time measures the performance of the traffic system. In a traffic system without traffic signals, vehicles cross the junctions according to the stop signs: first come first go. Replacing the stop signs with the traffic signals often improve the performance of the traffic network. Design different strategies for the traffic signals, e.g., they change randomly, periodically, or adaptively according to the traffic flow (each traffic signal knows how many cars are stopped here in its junction). Compare the performance of the different strategies. Are traffic signals always better than the stop signs? Be sure to consider the acceleration and deceleration of the vehicles, which makes the distance between cars vary with the speed (See 4.4 in (1)).
- Optional goals:
  - Introduce major roads and minor roads to the map. The roads have different capacity and speed limit. For the junctions with stop signs, cars from minor roads yield the right of way to cars from major roads.
  - With the major roads and minor roads, the shortest path might not be the fastest path. Now the cars choose the fastest path assuming it is the only car on the road and it doesn't hit the red lights.
  - The path-finding strategy above doesn't make sense when the roads are filled with cars, e.g., there is a traffic jam. Now the cars examine the current traffic condition and guess the realistic the fastest path.
  - Road map has more complicated geometry, e.g., including a ring road or one-way traffic.
  - Inhomogeneous vehicle generation. Divide the system into areas with different population density. High-density areas have more vehicles generated, and are more likely to be the destination. Is it true that the stop signs are better in the low-density area?
  - Suppose the number of vehicles in the traffic system varies with time. Observe the formation and dissolution of traffic jams.
  - For highly capable students, try to use reinforcement learning (2) to find the optimal strategy of the traffic signals.
  - You can add as many features as you like to make the model more realistic. The major goal, however, should always be optimizing the traffic signals.

(1) Self Organization and Pattern Formation

(2) Wikipedia: Reinforcement learning

2.5 (Pseudo-)Spectral methods for ODE and PDE solution

(Carlo Musolino)

Spectral methods are a class of numerical schemes which can be very useful in the solution of Boundary Value ODEs (Ordinary Differential Equations) as well as Elliptic PDEs (Partial Differential Equations) (1). Their main feature is that the solution and the source terms are represented by means of a functional basis, so that the equations can converted to a linear system for the spectral coefficients. This class of numerical schemes is known to converge very rapidly to the correct solution for smooth problems.

Your initial task will be to implement a spectral solver based on Chebyshev Polynomials (2) for boundary value ODEs, and verify that it converges spectrally (not quite exponentially!) to the correct solution (3). This will require you to implement (you can do this with classes and OOP or traditional Functional programming) a code that will transform a "function" from its physical representation (values on points on a discrete grid) to its spectral (collocation space) representation (i.e. the coefficients of its polynomial decomposition) using Gauss quadratures, and solve the resulting linear system making sure that boundary conditions are satisfied to machine precision (chapter 1 of (3)).

[Optional-Advanced] Using a FFT (7) library such as GSL (8) or FFTW (4) implement a pseudo-spectral solver (5) for the nonlinear Burgers equation (see eq. 1) (6) and study the solution for varying viscosity ν.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{1}$$

If you want to complete the second part of the project plan in advance and make sure that your classes/function interfaces are compatible with your FFT library of choice!

(1) Spectral Methos

(2) Intro to Chebyshev polynomial basis

(3) Spectral Methods and Numerical Relativity. An excellent introductory review.

(4) The FFTW library

(5) Pseudo-Spectral methods

(6) Burger's equation

(7) Fast Fourier Transform

(8) Fast Fourier Transform With the GNU Scientific Library

2.6 Expression templates for tensor equations [Advanced]

(Carlo Musolino)

This project will have you implement a simple framework in C++ for dealing with tensorial operations such as sum, contraction and multiplication. In order to ensure that the operations are performed efficiently at runtime and to make the writing of source code which utilises your framework as error-proof as possible, you will develop this code by making use of expression templates (3).

These were one of the first applications to template meta-programming (4) and feature the use of an expression tree to map equations to recursively evaluated expressions. This recursive unfolding can be performed at compile time with zero overhead or impact at run-time letting the compiler generate efficient code for simple expressions. For example, an expression such as:

```
Vec x,y,z;
// ... [initialize your Vecs to something meaningful]
auto v = x + y + z;
```

Would require the instantiation of two temporary `Vec` objects if the `Vec` class doesn't employ meta-programming techniques to ensure that the appropriate expressions are substituted in at compile time and only explicitly evaluated when necessary. This avoids significant memory and run-time overheads.

Expressing equations governing physical systems can be a cumbersome and error prone endeavour. In the field of general relativity large systems of partial differential equations govern the evolution of exotic objects such as black holes. With the advent of gravitational wave detections from compact binary

mergers, the need for accurate gravitational wave forms requires the numerical evaluation of large tensorial quantities (see (1), (2) and (3)), such as

$$R^{\lambda}{}_{\sigma\mu\nu} = \partial_\mu \Gamma^{\lambda}{}_{\nu\sigma} - \partial_\nu \Gamma^{\lambda}{}_{\mu\sigma} + \Gamma^{\lambda}{}_{\mu\kappa}\Gamma^{\kappa}{}_{\nu\sigma} - \Gamma^{\lambda}{}_{\nu\kappa}\Gamma^{\kappa}{}_{\mu\sigma}.$$

This expression contains many implied summations of the same indices that need to be efficiently mapped to modern programming languages. While classical approaches consisted in using computer algebra systems such as Maple and Mathematica, it is equally possible to implement them directly in C++ by means of *expression templates*

Your task will be to implement a simple expression template framework that can perform contractions, scalar multiplications and additions. The expressions should keep track of the rank and dimensions of the tensor, e.g. ensure that $C_{ij} = A^k_i B_{kj}$ is correctly treated as a rank 2 tensor, and should refuse compilation if invalid expressions are entered, e.g. $C_{ij} + B_i$ would be illegal. For this purpose you might want to look into type traits (5) (6), or (if you're very brave) into the newly introduced concepts (c++20) (7).

Using this framework implement several expression, shorter and longer ones, with also explicitly written out loops for the short expression to evaluate performance. If you need some inspiration for lengthy but physically meaningful applications check out the first pages of (5).

(1) General Relativity

(2) Introduction to Ricci Calculus

(3) Introduction to Expression Templates

(4) Template MetaProgramming

(5) A primer on type traits

(6) cppreference for std type traits

(7) cppreference for concepts and constraints in c++20

(5) Complicated expressions from numerical relativity

2.7 The 2D Ising model

(Arijit Dutta)

Introduction: The Ising model is a classical model of ferromagnetism which was invented by the physicist Wilhelm Lenz (1920), who gave it as a problem to his student Ernst Ising (1). Since then the model and its variations has served as landmarks for developments in several fields of physics. The simplest version comprises of binary variables called *Ising spins* ($S = \pm 1$) which are arranged on a lattice. The energy of the system is given by:

$$E = -\frac{J_x}{2}\sum_i (S_i S_{i+\hat{x}} + S_i S_{i-\hat{x}}) - \frac{J_y}{2}\sum_i (S_i S_{i+\hat{y}} + S_i S_{i-\hat{y}}) \tag{2}$$

where $J_x, J_y > 0$. i labels sites on the lattice, $\hat{x}$ and $\hat{y}$ denote unit vectors along x and y directions respectively and the lattice constant $a = 1$.

Steps for implementation:

- Instantiate a two-dimensional array of size $L \times L$. This array holds the spin configurations (1 or -1) on the lattice. Start with a random configuration which corresponds to a high temperature state.
- Anneal the system by updating the spins at randomly chosen sites according to the Metropolis-Hastings algorithm (2) and gradually lower the temperature T after performing several Monte Carlo sweeps. Use periodic boundary conditions for next neighbors at the borders of the lattice.
- Calculate the spontaneous magnetisation, specific heat and magnetic susceptibility at each temperature for different values of L. Benchmark your results for the transition temperature in the isotropic limit $J_x = J_y = J$ against the exact solution given by Onsager(1).

- Create a domain wall and study its dynamics at a fixed T. Create a movie by stacking together configurations at the end of each Monte Carlo sweep. Next, run the simulation at a high $T \geq 4J$ and suddenly quench to a low $T \leq 0.25J$. Visualise the configurations to learn what happens. What happens when $J_x \ll T \ll J_y$?
- (Optional 1) calculate the autocorrelation function.
- (Optional 2) Learn the ordered and disordered phases (3).

(1) Wikipedia entry: Ising model

(2) Monte Carlo Simulations in Statistical Physics - From Basic Principles to Advanced Applications

(3) Machine learning phases of matter.

2.8 Kramers' escape and stochastic resonance

(Arijit Dutta)

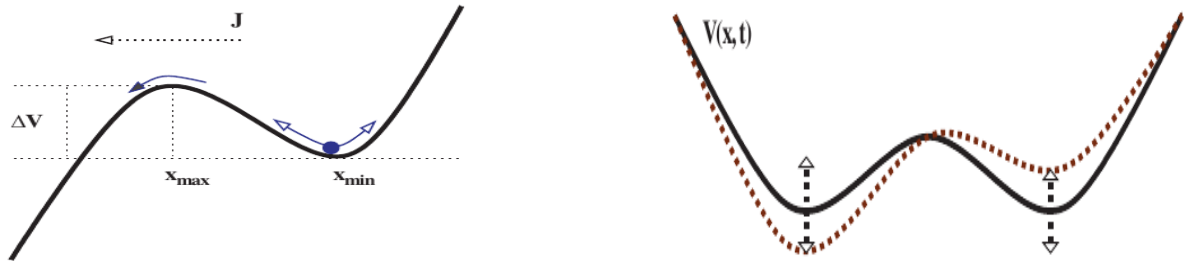

Figure 1: *(Left) Kramers' escape problem*: A particle stuck in a local minimum of a potential landscape can only escape due to noise driven dynamics (1). *(Right) Stochastic resonance*: A particle in a noisy double well potential is driven by an additional small periodic force which maintains two minima at all times. The driving frequency has interesting consequences on the dynamics of the particle. Both figures taken from (2).

Many natural systems are found to be stuck in a metastable state in which the kinetic energy available to the system is not sufficient to overcome a potential barrier that prevents it from reaching a lower energy state. In this scenario noise can come to aid and, in a rare event, allow the system to cross the barrier. Examples include chemical kinetics, flow of glasses, evolutionary kinetics, etc.

The Kramers' escape problem (1) is a simplified formulation of this. Consider a particle of mass m at position x and velocity v moving in a potential $V(x)$, as shown in Fig. 1 (left), having a friction coefficient γm and being kicked by a random force ξ. The equation of motion for the particle is,

$$\frac{dx}{dt} = v, \qquad m\frac{dv}{dt} = m\gamma v - \frac{dV}{dx} + \xi \tag{3}$$

and the random force is characterised by its first and second moments,

$$\langle \xi(t) \rangle = 0, \qquad \langle \xi(t)\xi(t') \rangle = Q\delta(t - t') \tag{4}$$

If the particle starts at the local minimum with initial speed $\sqrt{2T/m}$, what is the time after which it will be able to cross the barrier?

Steps for implementation:

- Use a suitable pseudo-random number generator to generate Gaussian noise. For the first part use $V(x) = -x + x^3$ and $Q = 2\gamma mT$, where T is the temperature.
- Start from a high temperature overdamped limit. For a given noise realisation update the position and velocity of the particle. Save these in a file. Tabulate the first passage time $\Delta\tau$ as a function of T and γ. What happens as you go to low $T < \Delta V = V(x_{max}) - V(x_{min})$?

- Plot the logarithm of escape rate $r_K = 1/\Delta\tau$ as a function of $1/T$ for different values of γ. For a few low T values, plot r_k vs γ. Is there an optimal γ which leads to maximum escape rate at a given temperature ?

Now consider the scenario where the potential has two minima Fig. 1 (right), and there is an additional weak periodic force such that two minima are maintained at all times. How does the particle's trajectory depend on the frequency of driving?

Steps for implementation:

- For the second part take $V(x) = -\frac{1}{2}x^2 + \frac{1}{4}x^4 + A\cos(\Omega t)x$. Choose A such that the potential retains two minima at all times. Plot the trajectory $x(t)$.
- Calculate the running ensemble average of the trajectory $\langle x(t)\rangle$ and check whether the following relations hold:
  - $\langle x(t)\rangle = 0$, when $A = 0$.
  - $\langle x(t)\rangle = \bar{x}\cos(\Omega t - \bar{\phi})$, when $A \neq 0$. Find $\bar{x}$ and $\bar{\phi}$ numerically. You can try fitting a cosine or perform a Fast Fourier Transform (FFT)(3)
- Plot $\bar{x}$ vs Q for different values of A. Is there a resonance ?

(1) "Brownian motion in a field of force and the diffusion model of chemical reactions" (orginal paper)

(2) Lecture notes by Prof. Gros "Dissipation, Noise and Adaptive Systems"

(3) FFTW library.