Phys 256 — Midterm 2

Student Name

This exam consists of two parts:

1. Take-home assignment (AI allowed for coding only)

Deliverables:

- Code as instructed by specific problems.
- Report for specific problems.
  - aim for conciseness
  - describe what and why you implement certain things
  - choice of algorithms and tuning parameters
  - if we ask you to analyze a problem that means we want to see illustrative plots and your reasoning why certain behaviors or trends occur

2. In-class exam (AI prohibited)

- You will apply and extend prepared code to new situations.
- You will be allowed to use only your own minimization algorithms of your choice: golden section search, gradient descent, or simulated annealing.
- If you have to resort to stock library algorithms, points will be deducted, but it is ok to use them for debugging.

Problem 1:

Implement the simulated annealing minimum search algorithm.

```python
def annealing(f, x0, generateNewXfunc, Ni=1000, Tstart=100, coolingRate=0.9):
    """Simulated annealing algorithm
    f - cost or energy function that we wish to minimize
    x0 - 1D numpy array with components of the initial guess vector
        f(x0) should be able to calculate the energy at the vector x0
    generateNewXfunc - function that generates a new guess x_new from the previous x value,
        i.e. x_new = generateNewXfunc(x_previous)
    Ni - number of function iterations to run before stopping
    Tstart - starting "temperature" of the problem
    coolingRate - temperature cooling rate per iteration
                  Tnew = coolingRate * Told
    """
    pass

    # annealing function must return a 3-tuple containing:
    # x_best - estimate of the minimum location
    # f(x_best) - energy at this location
    # f_xi - 1D array of length Ni containing the energy at every iteration during the search
    #   f_xi[0] must be f(x0)
    return (x_best, f(x_best), f_xi)
```

Listing 1: Simulated annealing prototype

Submit code to the autograder as `p1.py`. Code template also available on course web site.

Report the tests for your code:

- Show the behavior of your algorithm during 1D minimization of $f(x) = x^2$, i.e. make a plot of f for every iteration.
- Add noise to the function $f(x)$ and minimize, i.e. show the behavior of your algorithm during 1D minimization of `x*x + 0.1*np.random.uniform()`
- Show the behavior of your algorithm during 2D minimization of $f(\vec{x}) = -\sin^2(r)/(.001 + r^2)$ where $\vec{x} = [x_1, x_2]$ and $r = \sqrt{x_1^2 + x_2^2}$.
- Try your algorithm on a function of your choice in 5 dimension, i.e. $\vec{x} = [x_1, x_2, \ldots, x_5]$.

Do your final solutions make sense? Justify.

Problem 2:

Setup

We are at war with viruses. Every day someone's life is taken by a deadly disease like Ebola, or something like the common cold, which might appear relatively harmless but ultimately

takes more lives in total.

For this problem you will be given access to a virus spread simulator. It will simulate the country by randomly placing water cells (which cannot be occupied) with a certain probability, then placing humans with a certain density, some of which can be vaccinated or naturally immune and thus less susceptible to the virus. Every day (turn) a virus can kill an infected human, or jump from infected person to a neighbor with a certain probability that depends on the distance between neighbors. Every day a human can recover from a virus and become immune.

The code of the simulator will be provided at the course website. It will feature an initial map with set parameters, and a map update algorithm that simulates the virus-human interaction. Make sure that you understand its logic; you are encouraged to ask about the internal workings of this simulator code during the week leading up to the midterm.

The code at the website will have some helper scripts which can visualize the situation; we advise you to use this visualizer to understand the virus dynamics.

Assignment

Do not modify the map initializer or the map update algorithm for the take-home portion! Use it as-is.

In order to fight the virus, we need to understand the virus's "mindset." Suppose that the virus's goal is to kill as many people as possible during a timespan of `Ndays=1000` on a map of size `Nrow=200` and `Ncol=100`. Your task is to find the best combination of infection probability p_i and probability p_d of an infected person dying.

```
# probability of infection per turn
p_i =
# probability of an infected person dying per turn
p_d =
```

Listing 2: the virus parameters in p2.py

Submit code to the autograder as `p2.py`. Code template also available on course web site. Also, submit any other relevant code but in separate files, no need to resubmit code provided by us.

Submit your report with a discussion of your strategy to find the best virus parameters. Provide a plot with the number of alive, infected, immune, and dead people for every turn when the virus spreads. Provide the final map of the population.

Make sure to run the optimizer several times since outcomes are probabilistic.

Does your virus use a strategy that is closer to Ebola or the common cold?

Problem 3: Bonus

We will give extra bonuses to the top 10 code submissions that find the most deadly virus. Note that this bonus is per individual submission, not per group.