PHYS358: Session 12

Group A
Optimization: Downhill Simplex and Genetic Algorithms

(1) dhsimplex: Background: In your class package, you'll find some literature on genetic algorithms. Find section 1.4 and answer the following questions about the test problem *P4*:

- Why is this a difficult problem for optimization?
- What is the location of the maximum?

(2) dhsimplex: The code: Check out `dhsimplex.py` and answer the following questions:

- What parameters is the program taking?
- What is the convergence criterion?
- Where are the cost functions defined?

(3) dhsimplex: The test: Run `dhsimplex.py` with the problem `charbonneau4` (that's P4 in Charbonneau's paper, and answer the following questions:

- How many tries do you need to get `dhsimplex.py` to converge on the correct solution?
- What are the "false" maxima it tends to find?
- What would be a good starting position for this specific problem? Try it out. You'll have to find the initialization of the first simplex in `dhsimplex.py`.

If you've reached this point, please wait and help your fellow students. Before we'll proceed with this activity, we'll have to discuss some background material.

(4) Genetic algorithms: Background: The two other programs in your class package deal with genetic algorithms. Start out with `demo_genalg.py`. The code will try to find a sentence of 115 letters. The sentence is from a wonderful poem (unfortunately in German...) about Prometheus[1] by J.W. von Goethe. To appreciate the power of "natural" selection, answer the following questions:

- The German alphabet has – as the English one does – 26 letters (not counting the infamous "Umlaute" ä, ö, ü). Since we'd like to separate the words for better legibility, we add the space character, so we have a total of 27 characters. The sentence has 115 letters. How many possible combinations are there if you attempted to find the sentence by randomly chosing characters?
- How long would that take if your computer could do 10^9 simultaneous guesses per second? What's the age of the Universe (yes, these questions are related)? If every atom in the Universe were a computer capable running 10^9 guesses per second, how long would it take (on average) to find the solution?
- How long does `demo_genalg.py` take to find the correct solution?

(5) Genetic Algorithms: The code: Check out `geneticalg.py` and answer the following questions:

- What parameters is the program taking?
- What is the convergence criterion?
- Why do we need to convert the optimization parameters to Decimal (fixed point)?

(6) Genetic Algorithms: The test: Repeat step (3) on `charbonneau4`: Does this work better or worse than `dhsimplex.py`? Why?

[1] Cover Your heavens, Zeus,
With cloud vapor
And try Your strike, as a boy
Beheading thistles,
Against oaken tree and mountain height