Stochastic optimization algorithms Lecture 7, 20180918

Classical optimization methods and evolutionary algorithms: Problem-solving

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

CHALMERS

Today's learning goals

- After this lecture you should be able to
  - Solve problems 2.12 and 2.13 in the book
  - Solve problems 3.2 and 3.9 in the book

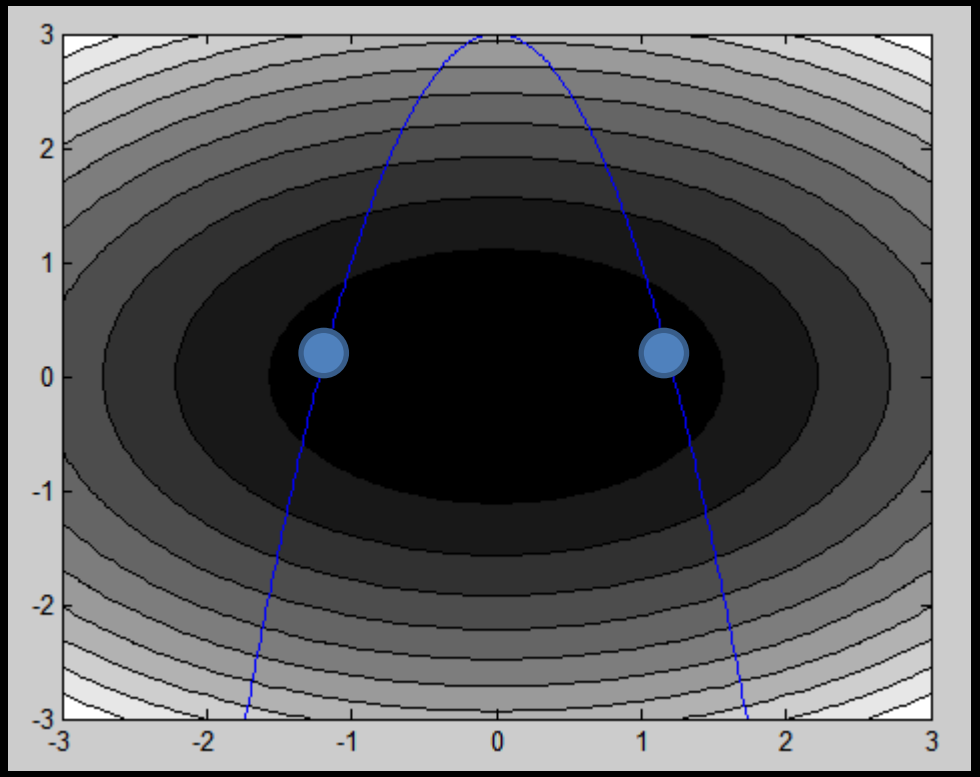
Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Problem 2.12

- Find minimum of ${x_1}^2 + 2{x_2}^2$ subject to the equality constraint $2{x_1}^2 + x_2 - 3 = 0$.
- Penalty method.
- Solution: Presented on the blackboard.
- Answer: $x_1 = \pm\frac{\sqrt{23}}{4}, x_2 = \frac{1}{8}$

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Problem 2.12

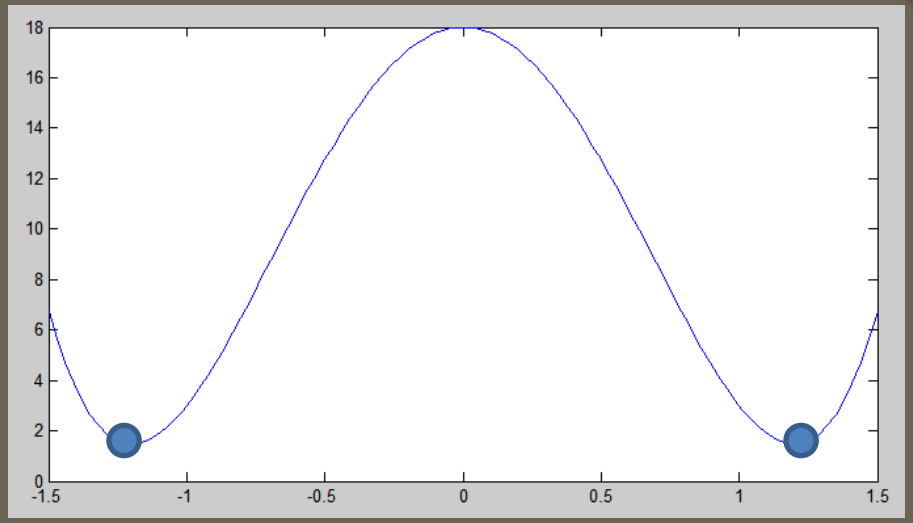


Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

CHALMERS

Problem 2.12, alternative approach

- Eliminate $x_2 \Rightarrow \mathrm{g}(x_1) = {x_1}^2 + 2(3 - 2{x_1}^2)^2$.
- Solve without penalty method (e.g. Newton-Raphson or analytically ($\mathrm{g}'(x_1) = 0 \Rightarrow \ldots$)



Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

CHALMERS

Problem 2.13

- Find the *maximum* value of $\mathrm{f}(x_1, x_2) = 2{x_1}^2 - 4x_1 + {x_2}^2 + 2x_2$ subject to the inequality constraint $2{x_1}^2 + {x_2}^2 \leq 12$.
- Analytical method:
  - Find stationary points in the interior of the set S defined by the constraint.
  - Find stationary points on the boundary ∂S.
  - Check the points one by one.
- Solution: Presented on the blackboard.
- Answer: See next slide.

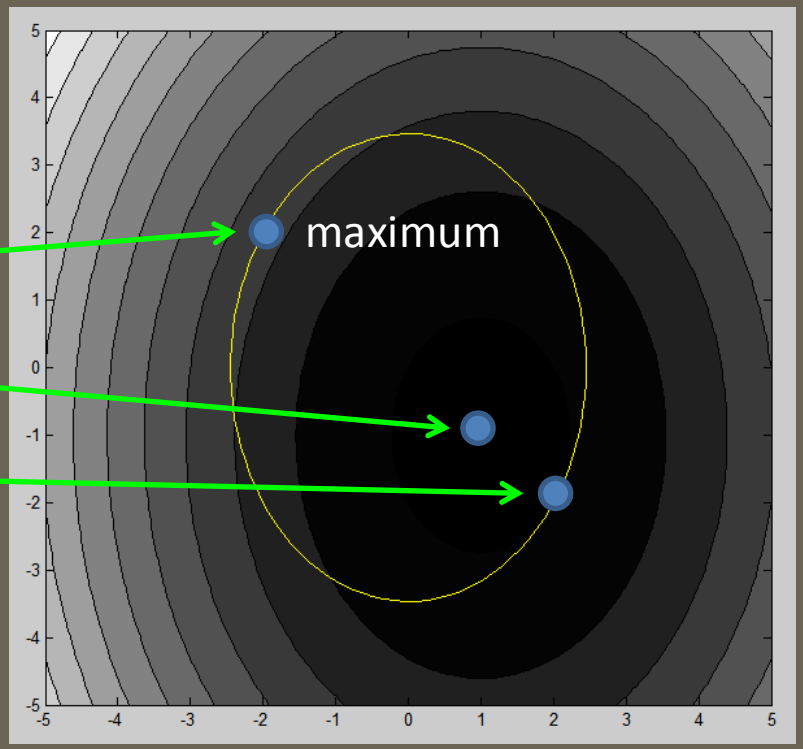
Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Problem 2.13

$(x_1, x_2)^T = (-2, 2)^T, \mathrm{f} = 24$

$(x_1, x_2)^T = (1, -1)^T, \mathrm{f} = -3$

$(x_1, x_2)^T = (2, -2)^T, \mathrm{f} = 0$



Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

CHALMERS

Today's learning goals

- After this lecture you should be able to
  - Solve problems 2.12 and 2.13 in the book
  - Solve problems 3.2 and 3.9 in the book

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

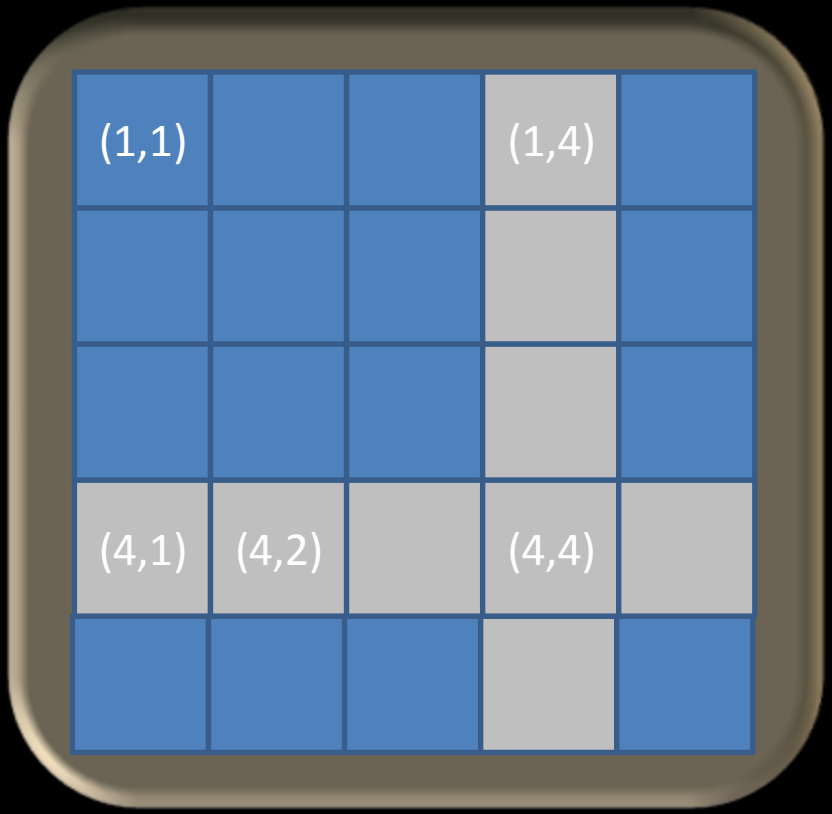
Problem 3.2

- Population of five individuals
- Fitness values: $F_1 = 5, F_2 = 7, F_3 = 8, F_4 = 10, F_5 = 15$.
- Find the probability of selecting individual 4 (in a single selection step) with
  - Roulette-wheel selection
  - Tournament selection ($P_{\text{tour}} = 0.75$)
  - Roulette-wheel selection with ranking from 1 to 10.
- Solution: Presented on the blackboard.
- Answers: (1) 0.2222, (2) 0.2400, (3), 0.2818

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Problem 3.2: Tournament selection

- 25 possible pairs, of which 9 include individual 4.
- All pairs equally likely.
- Individual 4 is the better individual in 6 of the pairs, and the worse individual in two of the pairs.
- $p_4 = \frac{1}{25}(1 + 6 * 0.75 + 2 * 0.25) = 0.24$



Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

CHALMERS

Problem 3.9

- Infinite-population GA.
- Fitness function: A function of unitation, namely $f(j) = j(m - j)$.
- Find
  - (a) The average fitness in the first generation.
  - (b) The probability distribution in the second generation.
  - (c) The average fitness in the second generation.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Problem 3.9

- Solution: Presented on the blackboard. Answers:
  - (a) $F_1 = \frac{m(m-1)}{4}$,
  - (b) $p_2(j) = 2^{2-m}\frac{j(m-j)}{m(m-1)}\binom{m}{j}$
  - (c) $F_2 = \frac{m(m-1)}{4} + \frac{1}{2}$
- For part (c), use the binomial expansion of $(a+b)^m$ and alternate between taking derivatives with respect to a and b (and then multiplying by to a or b, respectively) in order to find the required sum.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Today's learning goals

- After this lecture you should be able to
  - Solve problems 2.12 and 2.13 in the book
  - Solve problems 3.2 and 3.9 in the book


Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Introductory programming problem

- Not so many solutions (≈10%) were completely correct (=no “Not OK” remarks), but …
- …the results on the IPP will not affect your grades negatively. ☺
- You do not need to resubmit the IPP, regardless of the comments that you will receive.
- However, for HP1 and HP2, *significant* improvements will be required (in many cases, though not all).
- When correcting HP1 and HP2 we *will deduct points* for (some) errors of the kinds described below.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Introductory programming problem

- Typical errors:
  - Programming-related
    - Can be attributed to a lack of experience with (proper) coding, i.e. following a coding standard.
    - No disaster – just read the comments *and* the coding standard (and follow it!)
  - Not following instructions
    - This is more serious!
    - You will not lose any points now, but for HP1 and HP2 (some of) you need to make ***significant improvements***!

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Sending unnecessary files:
  - Do not include discarded test files and other unused files (e.g. files such as “PolynomialTest.m”, “Polynomial1.” etc.)
  - Remove temporary Matlab files (close the editor before compressing the folder and submitting the solutions)
  - Clean up the folder before compressing and submitting files!
  - In this case, no report was required. However, for HP1 and HP2 a report should be written, as mentioned in the problem formulations.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Not testing your code:
  - Always run your code, making sure that it works as intended, before submitting the files. Make sure also to clear the workspace (clear all) before testing your program.
  - From the solutions received, it was evident that some students had not even run their code before submitting!

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Unnecessary comments:
  - Code that follows the coding standard will normally be self-explanatory.
  - No need to add comments except in rare cases (very complex algorithms etc.).
  - For this problem (IPP), no comments are needed in the code.
  - Adding a comment header (at the top of the file), briefly describing the function can be a good idea, however.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Using non-descriptive variable names:
  - You should use descriptive variable names.
  - Avoid abbreviations. If a variable measuring (say) acceleration is needed, call it “acceleration”, not “acc”. (“acc” can signify acceleration, accuracy, accumulated value etc. etc.).
  - Check the spelling of your variable and function names (e.g. “derivative” instead of “derviative”, “individual” not “indvidual” etc.)

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Using non-descriptive variable names:
  - For variables with very limited scope (*a few lines* within a method or as part of a method with only *a few lines*), one can use shorter variable names, however, as long as there is no risk of confusion. For example, it is allowed to write…

    *v = v + a*deltaT*

  - …where *a* is acceleration and *v* is speed) *if* (NOTE!) the scope is limited. However, if *a* and *v* are both used in many places in a long method or script, it is better to call the variables *acceleration* and *velocity*, rather than *a* and *v*.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Not keeping the code clean:
  - Simplify the appearance of the code by dividing long statements into several statements (see the coding standard).

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Using function calls as inputs to a method:

```
functionPrime = Polynomial(iterates(i-1),PolynomialDifferentiation(polynomialCoefficients,1));
```

  - Not a good idea:
    - The expression becomes unnecessarily complex.
    - It is easy to make a mistake regarding the actual input to the (outer) function.
    - It is not clear what is actually returned by PolynomialDifferentiation.
  - Better:

```
primeCoefficients = PolynomialDifferentiation(polynomialCoefficients, 1);
functionPrime = Polynomial(x, primeCoefficients);
```

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Specifying numerical constants at various places in the code.
  - Instead specify constants clearly at the beginning of the method in question.
  - In particular, avoid specifying the value of a given constant at various places in a method: It is easy to make mistakes then (changing one occurrence but not the others etc.)

```
  index = index + 1;
  if (index == 1000)
    break;
  end
end
```

Instead specify a constant
MAXIMUM_NUMBER_OF_ITERATIONS = 1000
..at an appropriate place (not here).

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Incorrect inputs to a method (function), or inputs in the wrong order (in the interface)

> A function `NewtonRaphsonStep` (in the file `NewtonRaphsonStep.m`) which carries out a Newton-Raphson iteration step as in Algorithm 2.3 in the course book (p. 22), taking three inputs, x_j, $f'(x_j)$, and $f''(x_j)$, (in that order), and returning x_{j+1}.

  - There should be 3 inputs, not 2 or 4!
  - As specified, the inputs to this function should be x_j and the evaluated derivatives, *not their coefficients*.
  - The order of the inputs *should be as stated*: Absolutely no reason to modify the order!

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Incorrect number of inputs to a method (function), or inputs in the wrong order (in the interface)

> A function `Polynomial` (in the file `Polynomial.m`) which takes x and the $n+1$ coefficients of an n^{th}-degree polynomial $a_0, a_1, \ldots, a_n$ as input (in that order), and returns the value $f(x) = a_0 + a_1 x + \ldots + a_n x^n$.

  - This means

```
function Value = Polynomial(x, polynomialCoefficients)
```

  - Not

```
function Value = Polynomial(polynomialCoefficients, x)
```

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Lack of error handling

> Note that `PolynomialDifferentiation` should be able to handle *any* non-negative integer values of n and k (including the cases $n = 0$, $k = 0$, $k > n$ etc.), and that the overall program should be able to handle any non-negative value of n. If the iterates cannot be computed (for example, if $n < 2$ or $f''(x_j) = 0$ for some other reason), the program should give a suitable error message (in the form of a text string printed to the screen, clearly describing the error, for example "*the degree of the polynomial must be 2 or larger*") and

- It is good to know how to handle errors, but explicit error-handling will not be required for HP1 and HP2.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Incorrect output from a method (function)

A function `PolynomialDifferentiation` (in the file `PolynomialDifferentiation.m`) which takes, in order, the $n+1$ coefficients of an n^{th}-degree polynomial $a_0, a_1, \ldots, a_n$ and the order k of the derivative as input and returns the $n+1-k$ coefficients of the k^{th} derivative of the polynomial.

Example: if the input polynomial is $1+2x+3x^2$ and $k=1$, the input coefficents are $(1,2,3)$, and the output coefficients should be $(2,6)$ If, instead, $k=2$, the output coefficients should instead be (6) (a vector with one element). If $k \geq 3$, an empty vector should be returned.

  - You should, obviously, check that your function does return n+1-k coefficients (or an empty vector, where applicable!)

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Other comments:
  - PolynomialDifferentiation should return an empty vector when k > n, not when k > 2.
  - If k = 0, the method should just return the unchanged coefficients.
  - Instead of “while(1) and break”, it is better practice to use “while(condition == true)”.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Examples of errors for the IPP

- Other comments:
  - The derivatives' polynomial coefficients do not change: No need to compute them repeatedly inside the while loop.
  - To choose an appropriate range for plotting you could check the minimum and maximum values of the iterates.
  - You should check that the *value of the second derivative* is not zero in the main loop and handle the resulting error if it is.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

IPP summary

- Make sure to read and follow instructions carefully.
- Read the coding standard and the checklist!
- However, having said that, note that you are not required to do things that are not asked for. For example:
  - In the IPP, explicit error-handling was requested: In HP1 and HP2, it is not specifically requested.
- Most important: When in doubt: ***ASK!***

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Home problem 1

- Note: Your file name *should* be of the form

  LastName_FirstName_HP1.zip

- All submitted files (mail to me, not the assistants), *including the (PDF) report*, should be contained in the .zip (or .7z) file.
- Penalties for delays: See the course memo!
  - (No penalty for a delay of less than 6 hours)
- To minimize the number of unnecessary e-mails, check that you have *attached the zip file!*
- *Read* the checklist (on the web page) before submitting.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde