Homework 2 – 2016-05-02

NUMA01: Computational Programming with Python

Claus Führer

You should present your solution for this homework **at the latest on**

2016-05-12

This assignment has 10 tasks.

All the functions must be *properly documented* and also tested. All the functions must be *properly documented* and also tested. We recommend that you produce a report of your work with IPython-Notebook (see lecture on Monday April 18). You may work and present in groups by two. Use the upload link http://www.maths.lth.se/na/courses/NUMA01/dl/ to upload your code. Upload one file only, either a *.py-file or an IpythonNotebook file. You will get then an email later from one of the teaching assistants with a presentation time slot.

Interval analysis

Theory

Interval analysis is a branch of mathematics that deals with the theory of intervals. Instead of e.g. real numbers, the basic objects in interval analysis are closed intervals $[a, b]$.

The motivation for this is that in the real world, measurements are usually not exact, but come with some error margin. So the mass of an object, for example, might be $m = 3 \pm 10^{-3}$ kg, if the scale has a sensitivity of 1 g. This is naturally expressed as $m \in [3 - 10^{-3}, 3 + 10^{-3}]$ kg. If we also measure the acceleration of this object, $a \in [2 - 10^{-2}, 2 + 10^{-2}]$ m/s^2, then by Newton's second law $F = ma$ it follows that the force F exerted on the mass lies in the interval

$$F \in [(3 - 10^{-3})(2 - 10^{-2}), (3 + 10^{-3})(2 + 10^{-2})].$$

Note how the lower limit on the measurement error is less than the upper limit, so we cannot write $F = 6 \pm \text{err}$ without losing some information.

We will implement a class for representing intervals and performing operations on them. We consider first the basic arithmetic operations. If $\odot$ is a binary operator (e.g. $+$) then

$$[a, b] \odot [c, d] = \{z \in \mathbb{R} |\ z = x \odot y, \text{where } x \in [a, b] \text{ and } y \in [c, d]\}.$$

This can be written out explicitly as

$$[a,b] \odot [c,d] = [\min(a \odot c, a \odot d, b \odot c, b \odot d), \max(a \odot c, a \odot d, b \odot c, b \odot d)],$$

and for the four basic arithmetic operators we get the following rules:

$$\begin{aligned}
[a,b] + [c,d] &= [a+c, b+d], \\
[a,b] - [c,d] &= [a-d, b-c], \\
[a,b] \cdot [c,d] &= [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)], \\
[a,b] \,/\, [c,d] &= [\min(a/c, a/d, b/c, b/d), \max(a/c, a/d, b/c, b/d)], \quad 0 \notin [c,d].
\end{aligned}$$

One can also define functions of intervals, though one has to be careful. We just take one example, the power functions $x \mapsto x^n$, $n \geq 1$. If n is odd, this is a monotonically increasing function, so we have

$$[a,b]^n = [a^n, b^n].$$

For even $n > 0$ we have to distinguish between three cases:

$$[a,b]^n = \begin{cases} [a^n, b^n] & a \geq 0, \\ [b^n, a^n] & b < 0, \\ [0, \max(a^n, b^n)], & \text{otherwise.} \end{cases}$$

Task 1

Construct a class Interval which is initialized with two real numbers representing the left and right endpoints respectively.

Task 2

Provide methods for the four basic arithmetic operations.

Task 3

Provide a print method so that the code

```
Interval(1,2)
```

prints

```
[1, 2]
```

Task 4

Make sure that the following code works as expected and prints the values given in the comments

```
I1 = Interval(1, 2)     # [1, 2]
I2 = Interval(3, 4)     # [3, 4]
I1 + I2                 # [4, 6]
I1 - I2                 # [-3, -1]
I1 * I2                 # [3, 8]
I1 / I2                 # [0.25,0.6666666666666666]
```

Please note: Use `__truediv__` for the division.

Task 5

Extend your division function so that it raises appropriate exceptions if the dividing interval contains zero, or if the resulting interval would be infinitely large.

Task 6

A real number r is naturally identified with a degenerate interval $[r, r]$. Extend the class so that it can be initialized with only one real value, i.e.

```
Interval(1)   # [1, 1]
```

Task 7

Modify your code so that the following works

```
Interval(2,3) + 1   # [3, 4]
1 + Interval(2,3)   # [3, 4]
1.0 + Interval(2,3)   # [3.0, 4.0]
```

Task 8

Implement the `__contains__` method for checking if a real value is within the given interval.

Task 9

Implement the power function $x \mapsto x^n$ as the `__pow__` function, so that one can write

```
x = Interval(-2,2)   # [-2, 2]
x**2                 # [0, 4]
x**3                 # [-8, 8]
```

Task 10

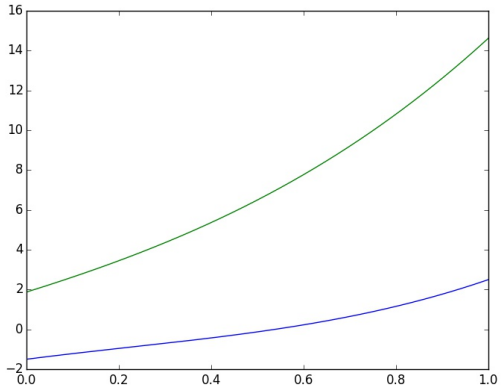
Define a list of 1000 intervals by creating a list of lower boundary values with `xl=linspace(0.,1,1000)` and upper boundaries `xu=linspace(0.,1,1000)+0.5`.

Evaluate the polynomial

$$p(x) = 3x^3 - 2x^2 + 5x - 1$$

on each interval of your list of intervals and create in such a way another list of intervals. Extract from this lists a list of lower boundaries yl and upper boundaries yu and plot both versus xl.

The result should look like this:



Good luck!