Numerical Methods (CS 357)

# Worksheet

## Part 1. Objectives

- Be able to apply Lagrange multipliers for constrained optimization
- Understand the working principle of floating point
- Know what is meant by a 'denormal' and the 'implied 1'.
- Understand the reason why catastrophic cancellation occurs.

## Part 2. Floating point "machine epsilon"

For a (binary) floating point system of the form $(s_1.s_2s_3)_2 \cdot 2^p$ that has an exponent range from $-128$ to 127 and that uses three bits to store the significand $s$, what is the difference between 1 and the smallest representable number greater than one?

(A) $2^{-3}$

(B) $2^{-4}$

(C) $2^{-1}$

(D) $2^{-2}$

## Part 3. Floating point: exact representation

For a (binary) floating point system of the form $(s_1.s_2s_3)_2 \cdot 2^p$ that has an exponent range from $-128$ to 127 and that uses three bits to store the significand $s$, which of the following sets of numbers can be represented accuratedly, i.e. without rounding?

(A) The integers 1 through 10

(B) The integers 1 through 5

(C) $2^{200}$

(D) $1/3$

## Part 4. Finite Differences vs. Floating Point

In this problem, you're given a function `f` and its derivative `df` as a function. For a large number of different values of the grid spacing $h$, you will use second-order centered finite differences to compute an approximation of the derivative:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

For each of the point counts given in `n_values`, compute the finite difference approximation to $f'$ everywhere except at the two endpoints. Compute the relative error in the $\infty$-norm and plot the result, using the starter code given.

What do you observe?

INPUT:

- `f`, a (reasonably wiggly) function.
- `df`, the derivative of `f`.
- `n_values`, a list of point counts to try. For each entry $n$ in this list, compute the second order finite differences on the grind between $[0, 1]$ with $n$ equispaced points.

OUTPUT:

- `a`, `b`, the final ends of your bracket.

```
import numpy as np
import numpy.linalg as la
import matplotlib.pyplot as pt

h_values = []
rel_err_values = []
for n in n_values:
    x = np.linspace(0, 1, n).astype(np.float32)
    h = x[1] - x[0]
    h_values.append(h)

    # Evaluate 2nd centered order finite differences of f at x.
    # Compute error against df at x in the infinity norm.



    rel_error = la.norm(error, np.inf) / la.norm(df(x), np.inf)
    rel_err_values.append(rel_error)

# ------------ plotting code below, no need to change
rel_err_values = np.array(rel_err_values)
pt.xlabel(r"$h$")
pt.ylabel(r"Rel. Error")
pt.loglog(h_values, rel_err_values)
```