# Introduction to Scientific Computing, PSCB57, Midterm

Professor Hanno Rein
University of Toronto, Scarborough

Monday, October 29th, 2018

Solutions

- Duration: 50 minutes.
- This midterm is worth 25% of your final grade.
- No aid sheets, books or other notes are allowed.
- A basic calculator is allowed, although not needed. No programmable calculators or any other electronic devices are allowed.
- Write your answers directly on the question sheet. If you need scrap paper raise your hand.
- The University of Toronto's Code of Behaviour on Academic Matters applies to all University of Toronto Scarborough students. The Code prohibits all forms of academic dishonesty including, but not limited to, cheating, plagiarism, and the use of unauthorized aids. Students violating the Code may be subject to penalties up to and including suspension or expulsion from the University. (1)

## Question 1

**7 Points**

Convert the following IEEE754 double floating point numbers from their binary representation to the standard decimal format:

`0011111111011000000000000000000000000000000000000000000000000000` 0.375

`1100000000100010000000000000000000000000000000000000000000000000` −9

---

Also convert the following IEEE754 double floating point number to decimal notation. Here, it is ok to give an approximate answer.

`1111111011101001000000110011011100111000101001100110100111110001` ≈ −9e305

`0000000011001100000101101100010111000101001001010011010101110101` ≈ +8e−305

---

Circle all numbers that can be approximated by a finite IEEE754 double floating point number:

$\pi \cdot 10^{-128}$ $\quad$ $1.23$ $\quad$ $7^{8312} - 1$ $\quad$ -sin(0.12) $\quad$ $2^{456}$

---

Circle all expressions that result in a division by zero when implemented using IEEE754 double floating point numbers:

$$\frac{11}{(11111 + 10^{-15}) - 11100 - 11} \qquad \frac{101}{(11 + 10^{-11}) - 10 - 1} \qquad \frac{101}{(11 + \pi^{-10} \cdot 10^{-11}) - 10 - 1}$$

---

In python 3, what are the result of the following expressions?

`1.0 // 2.0 == 0.5` False

`4.0 / 2.0 == 2.0` True

## Question 2 **5 Points**

Instead of using an integer or floating point representation, some computer programs also use what is called a fixed point number representation. Assume we have 8 bits, $b_i$, $i = 0, \ldots 7$ to represent a number $x$ in the following way:

$$x = (-1)^{b_0} \sum_{i=1}^{7} b_i 2^{i-4}.$$

What is the largest number than can be represented (an approximate answer is ok)?

15.876
___

Add the following 8 bit fixed point numbers in their binary representation, then convert the result into the normal decimal format: 01100110, 00000101.

01101011 ≙ 13.375

(binary representation) (decimal format)

Mark all statements which are true:

- [x] Negative numbers can be represented using this number representation.
- [x] Additions and subtractions are exact when using this number representation.
- [ ] Any number between 0 and $2^8$ can be represented exactly using this scheme.
- [ ] An 8-bit fixed point number uses more memory than a double precision floating point number.
- [x] Some fractions can be represented exactly using this number representation, for example $\frac{1}{8}$.

## Question 3 **4 Points**

Write down the central finite difference formulas (second order) for the first, second and third derivatives!

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

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

$$f'''(x) \approx \frac{f(x+2h) - 2f(x+h) + 2f(x-h) - f(x-2h)}{2h^3}$$

Hint: we did not discuss the equation for the third derivative in detail during the lectures. Try to construct it in an analogous way to the first two derivatives, then test it with a function where you know the answer, for example with a third order polynomial.

## Question 4

**5 Points**

Consider the follwing integral:

$$\int_0^{2\pi} \sin(x)dx.$$

What is the exact value of this integral?

0

Now, discretize the integral by splitting the interval $[0, 2\pi]$ into four sub-intervals. Then use the trapezoidal rule to calculate the value of the integral numerically.

0

Consider the integral once again, but this time over the interval $[0, \pi]$. Split this integral into four sub-intervals and use the trapezoidal rule to calculate the value of the integral numerically. Hint: $\sin\left(\frac{\pi}{4}\right) = \frac{1}{\sqrt{2}}$.

$\frac{\pi}{4}(\sqrt{2}+1)$

Explain the concept behind Simpson's rule in contrast to the trapezoidal rule! Would you expect a better approximation to the integral for the above case or not?

- Simpson's rule uses piece-wise quadratic interpolation
- Simpson's rule should give a better result.

## Question 5 **6 Points**

Use the bisection method on the function $f(x) = x^2 - 1$. Start with the interval $[0, 16]$. After 4 iterations, what is the interval in which you expect the root?

[0, 1]

How often would you have to iterate to reach machine precision?

~52 times

Write down the update rule (how to calculate $x_{n+1}$ from $x_n$) for Newton's method and the specific function $f(x)$ from above:

$$x_{n+1} = x_n - \frac{x_n^2 - 1}{2x_n}$$

Which method will converge to the true solution faster in the above case?

Newton's method

For what kind of function $g(x)$ does Newton's method converge within 1 step?

any linear function

Describe a real world situation where you might use a root finding method.

any type of optimization problem

## Question 6

**3 Points**

Below is a rather long assembler program. You'll find the syntax for the assembler commands used in the appendix (note that there are empty lines). What does this program do? Be concise and precise in your answer! Also add some comments to the program describing the various parts!

Standard map

```
SET 1. r0
SET 0. r1                       } p, θ, k
SET 3. r2

SET -30 r10
SET 0 r7
SET 1000 r8
SET 1 r9
SET 6.283185307179586 r4        -> 2pi

SET 3 r5
SET -3 r6

SIN r2 r3
MUL r0 r3 r3                    } actual standard map
ADD r3 r1 r1
ADD r2 r1 r2

IF r4 r1
JUMP r5
SUB r1 r4 r1
JUMP r6

IF r4 r2
JUMP r5
SUB r2 r4 r2
JUMP r6
                                } modulo 2p
IF r1 r7
JUMP r5
ADD r1 r4 r1
JUMP r6

IF r2 r7
JUMP r5
ADD r2 r4 r2
JUMP r6

PRINT r1                        } print p, θ
PRINT r2

SUB r8 r9 r8
IF r8 r7                        } loops 1000 times
JUMP r10
```

## Question 7

**2 Bonus Points**

Convert the assembler program from the previous question into a python 3 program! Small syntax errors will not be penalized.

(see tutorial/assignment)

# Introduction to Scientific Computing, PSCB57, Midterm

Professor Hanno Rein
University of Toronto, Scarborough

Monday, October 29th, 2018

Solutions

- Duration: 50 minutes.
- This midterm is worth 25% of your final grade.
- No aid sheets, books or other notes are allowed.
- A basic calculator is allowed, although not needed. No programmable calculators or any other electronic devices are allowed.
- Write your answers directly on the question sheet. If you need scrap paper raise your hand.
- The University of Toronto's Code of Behaviour on Academic Matters applies to all University of Toronto Scarborough students. The Code prohibits all forms of academic dishonesty including, but not limited to, cheating, plagiarism, and the use of unauthorized aids. Students violating the Code may be subject to penalties up to and including suspension or expulsion from the University. (2)

## Question 1

**7 Points**

Convert the following IEEE754 double floating point numbers from their binary representation to the standard decimal format:

`0011111111100100000000000000000000000000000000000000000000000000` 0.625

`1100000000101010000000000000000000000000000000000000000000000000` −13

---

Also convert the following IEEE754 double floating point number to decimal notation. Here, it is ok to give an approximate answer.

`0111111101110100100000011001101110011100010100110011010011110001` +9e305

`1000000100110011001100111001000100110001110001000110111110001011` −7e−303

---

Circle all numbers that can be approximated by a finite IEEE754 double floating point number:

$\pi \cdot 10^{-428}$     42.000001     $5^{-356} + 1$     $-\sqrt{2.0}$     $2^{7734}$

---

Circle all expressions that result in a division by zero when implemented using IEEE754 double floating point numbers:

$$\frac{101}{(11 + e^{-10} \cdot 10^{-11}) - 10 - 1} \qquad \frac{11}{(111 + 10^{-19}) - 100 - 11} \qquad \frac{101}{(111001 + 10^{-1}) - 111000 - 1}$$

---

In python 3, what are the result of the following expressions?

`4.0 / 2.0 == 2.0` True

`1.0 // 2.0 == 0.5` False

---

## Question 2

**5 Points**

Instead of using an integer or floating point representation, some computer programs also use what is called a fixed point number representation. Assume we have 8 bits, $b_i$, $i = 0, \ldots 7$ to represent a number $x$ in the following way:

$$x = (-1)^{b_0} \sum_{i=1}^{7} b_i 2^{i-4}.$$

What is the largest number than can be represented (an approximate answer is ok)?

15.876

Add the following 8 bit fixed point numbers in their binary representation, then convert the result into the normal decimal format: 00110110, 01000011.

0111 1001 (binary representation) $\triangleq$ 15.125 (decimal format)

Mark all statements which are false:

- [x] Any number between 0 and $2^8$ can be represented exactly using this scheme.
- [ ] Additions and subtractions are exact when using this number representation.
- [ ] Negative numbers can be represented using this number representation.
- [ ] Some fractions can be represented using this number representation, for example $\frac{1}{8}$.
- [x] An 8-bit fixed point number uses more memory than a double precision floating point number.

## Question 3

**4 Points**

Write down the central finite difference formulas (second order) for the first, second and third derivatives!

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

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

$$f'''(x) \approx \frac{f(x+2h) - 2f(x+h) + 2f(x-h) - f(x-2h)}{2h^3}$$

Hint: we did not discuss the equation for the third derivative in detail during the lectures. Try to construct it in an analogous way to the first two derivatives, then test it with a function where you know the answer, for example with a third order polynomial.

## Question 4

**5 Points**

Consider the follwing integral:

$$\int_0^{2\pi} \cos(x)dx.$$

What is the exact value of this integral?

0

Now, discretize the integral by splitting the interval $[0, 2\pi]$ into four sub-intervals. Then use the trapezoidal rule to calculate the value of the integral numerically.

0

Consider the integral once again, but this time over the interval $[-\pi/2, \pi/2]$. Split this integral into four sub-intervals and use the trapezoidal rule to calculate the value of the integral numerically. Hint: $\cos\left(\frac{\pi}{4}\right) = \frac{1}{\sqrt{2}}$.

$\frac{\pi}{4} \cdot (\sqrt{2}+1)$

Explain the concept behind Simpson's rule in contrast to the trapezoidal rule! Would you expect a better approximation to the integral for the above case or not?

- uses piece-wise quadratic interpolation
- Simpson's rule should be better.

## Question 5

**6 Points**

Use the bisection method on the function $f(x) = x^4 - 1$. Start with the interval $[0, 16]$. After 4 iterations, what is the interval in which you expect the root?

[0, 1]

How often would you have to iterate to reach machine precision?

52

Write down the update rule (how to calculate $x_{n+1}$ from $x_n$) for Newton's method and the specific function $f(x)$ from above:

$x_{n+1} = x_n - \frac{x_n^4 - 1}{4x_n^3}$

Which method will converge to the true solution slower in the above case?

bisection method

For what kind of function $g(x)$ does Newton's method converge within 1 step?

any linear function

Describe a real world situation where you might use a root finding method.

any type of optimization problem

## Question 6

**3 Points**

Below is a rather long assembler program. You'll find the syntax for the assembler commands used in the appendix (note that there are empty lines). What does this program do? Be concise and precise in your answer! Also add some comments to the program describing the various parts!

standard map

```
SET 0.8 r0                      } set Θ, p, K
SET 3. r1
SET 2. r2

SET -30 r10
SET 0 r7
SET 2000 r8
SET 1 r9
SET 6.283185307179586 r4      ⇐ 2pi

SET 3 r5
SET -3 r6

SIN r2 r3                       } actual standard map
MUL r0 r3 r3
ADD r3 r1 r1
ADD r2 r1 r2

IF r4 r1                        } mod 2pi on p and theta
JUMP r5
SUB r1 r4 r1
JUMP r6

IF r4 r2
JUMP r5
SUB r2 r4 r2
JUMP r6

IF r1 r7
JUMP r5
ADD r1 r4 r1
JUMP r6

IF r2 r7
JUMP r5
ADD r2 r4 r2
JUMP r6

PRINT r1                        } print p, Θ
PRINT r2

SUB r8 r9 r8                    } loop 2000 times
IF r8 r7
JUMP r10
```

## Question 7

**2 Bonus Points**

Convert the assembler program from the previous question into a python 3 program! Small syntax errors will not be penalized.

see assignment