CMPT/MATH 420: Numerical Analysis

Dr. Ben Cameron

Assignment 5

Due: 11:59PM Friday December 9, 2022

Instructions: Questions 1.-3. are to be done by hand **showing all work** for full marks. You can use sage or a scientific calculator to get approximate values for numbers where appropriate. Question 4. is to be done in the A5_shell.ipynb posted on moodle.

You should upload both your .pdf file containing your written responses and the .ipynb file for Question 4.

For Questions 1.-3., assume $y(x)$ is the solution to IVP

$$\begin{aligned} y' &= -y^2 + e^x + \cos(y) \\ y(1) &= 1. \end{aligned}$$

1. (5 marks) Use Euler's Method to to approximate $y(3)$ in $n = 4$ steps.
2. (5 marks) Use Taylor Series Method of order $N = 3$ to approximate $y(3)$ in $n = 4$ steps.
3. (5 marks) Use Huen's Method to $y(3)$ in $n = 6$ steps.
4. (5 marks) In the A5_shell.ipynb, write a function called `TaylorSeriesMethod(f, x0, y0, n, N, xn)` which will use the Taylor Series Method of order `N` to approximate $y(\texttt{xn})$ in n steps where y is the solution to the IVP

$$\begin{aligned} y' &= \texttt{f}(x, y) \\ y(\texttt{x0}) &= \texttt{y0}. \end{aligned}$$

Your output should be formatted the same as for the Lab11 solution, so the call:

```
TaylorSeriesMethod(2*y+3*sin(x)+e^(x), 0, 7, 5, 1, 1)
```

should produce the output:

```
x0 = 0,         y(x0) = 7
x1 = 1/5,       y(x1) approx. 10.0000000000000
x2 = 2/5,       y(x2) approx. 14.3634821501091
x3 = 3/5,       y(x3) approx. 20.6408909550661
x4 = 4/5,       y(x4) approx. 29.6004565812077
x5 = 1,         y(x5) approx. 42.3161610539290

42.3161610539290
```

(See next page for notes and hints for the Q5)

Notes and Hints

Up to this point, you have enough information about the problem but I would like to give some hints and notes to help guide you on your solution. You will note that in the cell of A5_shell.ipynb where you are to write your function, I have included the following two lines:

```
x = var('x')
y = function('y')(x)
```

These lines make it easy to differentiate `f`, by simply calling `diff(func,x)`. However, when you do this with for example:

```
f =  2*y+3*sin(x)+e^(x)
diff(f,x)
```

the output is:

```
3*cos(x) + e^x + 2*diff(y(x), x)
```

The `diff(y(x), x)` makes evaluating `f'` at `x0` and `y0` impossible to do directly. Here are some hints that might help you solve this dilemma:

(a) You can convert any symbolic expression into strings in sage by using the python's built-in `str` function. for example

```
str(3*cos(x) + e^x + 2*diff(y(x), x))
```

returns the string

```
'3*cos(x) + e^x + 2*diff(y(x), x)'
```

(b) You can convert strings into symbolic expressions in sage by using the sage's built-in `SR` function. for example

```
SR('3*cos(x) + e^x + 2*diff(y(x), x)')
```

returns the symbolic expression

```
3*cos(x) + e^x + 2*diff(y(x), x)
```

(c) Python's built-in string `replace` function can be used to replace all instances of substrings with a difference substring in a given string. For example

```
txt = "ben is the only benevolent ben that I know"
replaced_txt = txt.replace("ben", "huy")
```

results in `replaced_txt` containing the string

```
'huy is the only huyevolent huy that I know'
```

I recommend writing a helper function or two to handle the string replacements you will need to do, but it is up to you.