Com S // Cpr E // Math 5250

Numerical Analysis of High-Performance Computing

Instructor: Songting Luo

Lecture 4: More Basic Python Coding

Outline

1. Good coding practices demo

Good coding practices demo

Python example: square root function

Goals:

- Develop our own version of `sqrt` function using Newton's method
- Start simple and add complexity in stages
- Illustrate some Python programming
- Illustrate use of git to track our development

Note: We will do this in

```
$ISUHPC/lectures/lecture4
```

directory so you can examine the various versions later

Python example: factorial

- We want to calculate $n!$.
- Can be done in a simple for loop.

$n!$ in a simple for loop

```
s = 1
for k in range(1,n):
    s = s*(k+1)
```

Python example: exponential function

- We want to calculate e^x.
- We will store the value of $e \approx 2.7182818284590451$.
- From x we find the nearest integer, let's call it x_0:

```
x0 = int(round(x))
```

- We then compute the Taylor series of e^x about $x = x_0$.

Taylor series for e^x about $x = x_0$

$$z = x - x_0$$

$$e^x = e^{x_0}\left(1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \dots\right)$$

Python example: natural logarithm function

- We want to calculate $\ln(x)$.

$$s = \ln(x) \quad \Longrightarrow \quad e^s = x \quad \Longrightarrow \quad f(s) = e^s - x$$

- Use Newton's method with initial guess $s = x$

Newton's method for $f(s) = e^s - x$ (x is given)

$$\begin{aligned} s^{(k+1)} &= s^{(k)} - \frac{f\left(s^{(k)}\right)}{f'\left(s^{(k)}\right)} \\ &= s^{(k)} - \frac{\exp\left(s^{(k)}\right) - x}{\exp\left(s^{(k)}\right)} \\ &= s^{(k)} - 1 + x\exp\left(-s^{(k)}\right). \end{aligned}$$

Lab assignment

- Update the functions in **demo_myfuncs.py**.
- The population $P(t)$ satisfies the *logistic growth equation*:

$$\frac{dP}{dt} = r\,P\left(1 - \frac{P}{K}\right), \tag{1}$$

where: $P(t)$ is the population at time t, with initial population $P(0) = P_0$, $(0 < P_0 < K)$. $r > 0$ is the intrinsic growth rate, $K > 0$ is the carrying capacity.
The logistic equation admits a closed-form solution:

$$P(t) = \frac{K}{1 + \left(\frac{K - P_0}{P_0}\right) e^{-rt}}. \tag{2}$$

1. Using the exact formula above, write a Python function that evaluates $P(t)$ for a list of time values.
2. Implement the forward Euler method to approximate the solution of the logistic equation.
3. Find the time when the population hits $K/2$.

For numerical experiments, you may use:

$$r = 0.5, \quad K = 100, \quad P_0 = 10, \quad t \in [0, 20].$$

- Update Git repository
- Submit both code and screenshots