SI 211: Numerical Analysis
Homework 1

Fu Min   72677006

September 29, 2018

Problem 1. We want to evaluate the function

$$f(x) = \frac{sin(10^4 x)}{x}$$

for different values of x.

(a) Evaluate the above function at $x = \pi$ by using Matlab, Julia, C++ or any other programming language of your choice. How big is the numerical approximation error? Can you explain why you observe this error?

Solution.

```
def problem1_f (x):
    F = sp.sin (10.0e4 * x)/x
    return F
x1 = np.pi
print ("F($\pi$)=",problem1_f(x1))
```

At $x = \pi$, the exact solution is f(π)=0, while the solution by program is $F(\pi) = -1.08100119082900e - 11$. So the absolute rounding error is $e = |f - F| = 1.08100119082900e - 11$. The order of magnitude the numerical approximation error is 10^{-11}.

We get the derivative of $f(x)$ is

$$f^{'}(x) = \frac{10^4 * x * cos(10^4 x) - sin(10^4 x)}{x^2}$$

At $x = \pi$, the condition number is c = $f^{'}(\pi) = \frac{10^4}{\pi}$. So we can get that the order of magnitude the numerical approximation error is 10^{-11}.

(b)Evaluate the above function at $x = 10^{-10}$. How big is the numerical evaluation error?

At $x = 10^{-10}$, the condition number is c = $f^{'}(10^{-10}) \approx 10^{14}$. So the numerical error is around

$$\begin{aligned} error &= c * eps \\ &= f^{'}(10^{-10}) * 2 * 10^{-16} \\ &\approx 10^{-2} \end{aligned}$$

Problem 2. Numeric differentiation based on central differences:

(a) Implement a function (for example in Python, Julia, or Matlab) that uses numeric differentiation based on central differences,

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

Here, the inputs of the differentiation routine are the scalar function f that we want to differentiate, the point x at which the derivative should be evaluated, and the finite perturbation $h > 0$. Use the syntax

diff(f,x,h) = ...

Solution.

```
def diff(f,x,h):
    f_hat = (f(x + h) - f(x - h)) / (2.0 * h)
    error = np.abs(f_hat - f(x))
    return error
```

(b)Evaluate the derivative of the function $f(x) = exp(x)$ at $x = 0$ using the above routine diff. Plot the numerical differentiation error in dependence on $h \in [10^{-15}, 10^{-1}]$ and interpret the result. Use logarithmic scales on both axis!

Solution.

Code :

```
f = np.exp
x = 0.
h = np.array([10.0**(-n) for n in range(1, 15)])
error = diff(f,x,h)
fig = plt.figure()
axes = fig.add_subplot(1, 1, 1)
axes.loglog(h, error, 's-', label="Centered")
axes.legend(loc=3)
axes.set_xlabel("h")
axes.set_ylabel("Absolute Error")
plt.show()
```

The resulte:

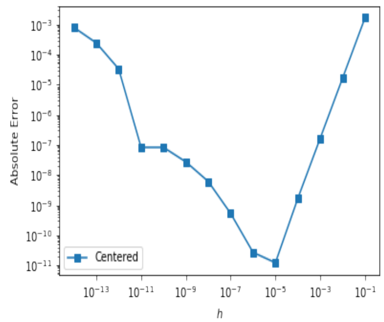

Figure 1: numerical differentiation error VS h

To the central differences, the mathematical approximation error is

$$\left|\frac{f(x+h)-f(x-h)}{2h}-\frac{\partial f}{\partial x}(x)\right| \leqslant \mathbf{O}(h^2)$$

The numerical error is still in the order of

$$\frac{eps}{h} = \mathbf{O}(\frac{eps}{h})$$

At $x = 0$, the condition number is c $= f^{'}(0) = 1$. So the f is well conditioned. We can choose $h \approx \sqrt[3]{eps} \approx 10^{-5}$. It still verifies the minmun in figure 1.

problem 3. In order to evaluate the factorable function $f(x) = sin(cos(x)) * cos(x)^2$ we write an evaluation algorithm of the form

$$\begin{aligned} a_0 &= x \\ a_1 &= cos(x) \\ a_2 &= a_1 * a_1 \\ a_3 &= sin(a_1) \\ a_4 &= a_2 * a_3 \\ f(x) &= a_4 \end{aligned}$$

What is the corresponding algorithm for evaluating the derivative of $f(x)$ using the forward mode of algorithmic differentiation (AD)? What is the order of magnitude of the numerical error that is associated with evaluating the derivative of f at $x = 0$ using this AD code?

Solution.

The AD is:

$$
\begin{aligned}
b_0 &= 1 \\
b_1 &= -six(x) \\
b_2 &= b_1 * a_1 + a_1 * b_1 \\
b_3 &= cos(a_1) * b_1 \\
b_4 &= b_2 * a_3 + a_2 * b_3 \\
f^{'}(x) &= b_4
\end{aligned}
$$

The code :

```
def AD(x):
  a0 = x
  a1 = sp.cos(x)
  a2 = a1 * a1
  a3 = sp.sin(a1)
  a4 = a2 * a3
  b0 = 1
  b1 = -1* sp.sin(x)
  b2 = b1 * a1 + a1 *b1
  b3 = sp.cos(a1) * b1
  b4 = b2 * a3 + a2*b3
  return a4, b4
f, diff_f = AD(0.)
```

From the above program, we can get that $f^{'}(0) = 0$. So the order of magnitude of the numerical error is 0.