PHYS 410-Computational Physics- Homework 2

Ali Izadi Rad
Date: Oct 2016

Part I

Here, we want to Write a code for a 7-point centered formula for differentiating a function at the point x, using evaluations of the function at x and x n h, for n = 1, 2, 3.

In this order, we need to find the weight associate to each term in linear combination by Lagrange interpolation. Let's to start by definition. We would like to interpolate the function by some few sampled point, generally:

$$f(x) \simeq \sum_{i=1}^{N} f(x_* + i \times h) L_i(x) \tag{1}$$

for this problem, we have seven points and thus:

$$f(x) \simeq \sum_{i=-3}^{3} f(x_* + i \times h) L_i(x) \tag{2}$$

We already know how does L_i's look like:

$$L_i(x) = \prod_{k=1, k\neq i}^{N} \frac{x - x_k}{x_j - x_k} \tag{3}$$

Now, if take a derivative from interpolated function at point $x = x_*$, we will have:

$$f'(x) \simeq \sum_{i=-3}^{i=3} f(x_* + ih) L'_j(x) = \sum_{i=-3}^{i=3} f(x_* + ih) \sum_{k=0}^{k=6} \frac{\prod_{m=0, m\neq k}^{6} (x_* - x_m)}{\prod_{k=0, k\neq i}^{6} (x_i - x_k)} \tag{4}$$

Now we need to calculate each term by hand. the final answer is as a following:

$$f'(x_*) \simeq \frac{-f(x_* - 3h) + 9f(x_* - 2h) - 45f(x_* - h) + 45f(x_* + h) - 9f(x_* + 2h) + f(x_* + 3h)}{60h} \tag{5}$$

Now, we want to derive the optimal values of h, which minimizes the combination of discretization and round off errors. Similar to method that we used in Tutorial 5, at first we observe that:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{1}{3!} h^2 f'''(x) + O(h^3) \tag{6}$$

Then, we need to do this calculation for other step size ($2h$ and $3h$) and see how does the leading terms behaves. Mathematica helps us to find the series:

```
In[88]:= - (Normal[Series[f[(-3 h) t + x0], {t, 0, 9}]] /. t → 1) +
    9 * (Normal[Series[f[(-2 h) t + x0], {t, 0, 9}]] /. t → 1) +
    -45 * (Normal[Series[f[(-h) t + x0], {t, 0, 9}]] /. t → 1) +
    45 * (Normal[Series[f[(h) t + x0], {t, 0, 9}]] /. t → 1) +
    -9 * (Normal[Series[f[(2 h) t + x0], {t, 0, 9}]] /. t → 1) +
    (Normal[Series[f[(3 h) t + x0], {t, 0, 9}]] /. t → 1) // Simplify
```

$$\text{Out[88]=}\ 60\, h\, f'[x0] + \frac{3}{7} h^7 f^{(7)}[x0] + \frac{1}{12} h^9 f^{(9)}[x0]$$

Figure 1: Series expansion

This means that discritization error is $\frac{3}{7\times 60} h^6 f^{(7)}(x_*)$.

because of limited precision in computer and also in MATLAB, for every floating point variable we should consider some error. We show it as:

$$\bar{f}(x) = f(x) + e(h), \qquad e(x) \leq \epsilon \simeq 2.2 \times 10^{-16} \tag{7}$$

Now if we called approximated function in eq.5 as D, then by using triangle formulawe have:

$$|D(h) - \bar{D}(h)| = |\frac{\sum c_i f_{ih}}{60h} - \frac{\sum c_i f_i h + c_i \times e_i(h)}{60h}| \leq \frac{110\epsilon}{60h} \tag{8}$$

Now we can say that

$$E(h) = |f'(x_*) - \bar{D}(h)| \leq \frac{3}{7 \times 60} h^6 f^{(7)}(x_*) + \frac{110\epsilon}{60h} := ah^7 + \frac{b}{h} \tag{9}$$

Similar to case we had in Tutorial, E(h) has minimum at some specific point which can be find by taking derivative:

$$h_{opt} = (\frac{b}{7a})^{1/8} \simeq (\frac{8 * 10^{-16}}{f^{(7)}(x_*)})^{1/8} \tag{10}$$

Now we want to apply this method for function $f(x) = Sin(x^2), 0 \leq x \leq 1$. the 7th derivite of this function given by $1680x \sin\left(x^2\right) - 128x^7 \cos\left(x^2\right) - 1344x^5 \sin\left(x^2\right) + 3360x^3 \cos\left(x^2\right)$ which varies between 0 to 2000 in our domain. We select , $x = 0.9$ as reference because it has biggest error,and we get $h = 0.005$.

Now We do MATLAB programing. We know the exact value of derivative for our function given by $2xcos(x^2)$, therefore we can compare the approximated method that we discussed above with exact answer.

Matlab Code

```
function part1

X=linspace(0,3,10^3);  %This is the graph spacing.
Y=2.*(X).*cos(X.^2);   %The exact derivative
h=0.01; %The ideal h value we computed above
Z=(-sin((X-3*h).^2)+9*sin((X-2*h).^2)-45*sin((X-h).^2)+ 45*sin((X+h).^2)-9*sin((
    X+2*h).^2)+sin((X+3*h).^2))/(60*h);
%The above is the approximation with the correct weights and h from above.
plot(X,Y,X,Z)
title('Exact Result vs. 7-Point Centered Derivative')
legend('Exact Derivative','7-PointDerivative')
xlabel('x')
ylabel('f prime')
```

As we can see the result of this two methods are not distinguishable from each other in fig.2 that shows that such a good approximation it is. Thus using Lagrange weights has very good impact on result and it's useful to use from this method.

1 Part II

We want to write a code for Romberg integration of a function f(x) over an interval [a, b], using the composite trapezoid formula (with error of order h^2, where $h = (b - a)$ applied to sub-intervals of length $h \cdot 2^{-m}$ and then the goal is eliminating all errors to order h^6. Using

$$h_m = \frac{b-a}{2^m} \tag{11}$$

The Romberg methods is given by:

$$\begin{aligned}
T_{1,1} &= h_1(f(a) + f(b)) \\
T_{n,1} &= \frac{1}{2} T_{n-1,1} + h_n \sum_{k=1}^{2^n - 1} f(a + (2k-1)h_n) \\
T_{n,m} &= T_{n,m-1} + \frac{1}{4^m - 1}[T_{n,m-1} - T_{n-1,m-1}]
\end{aligned} \tag{12}$$

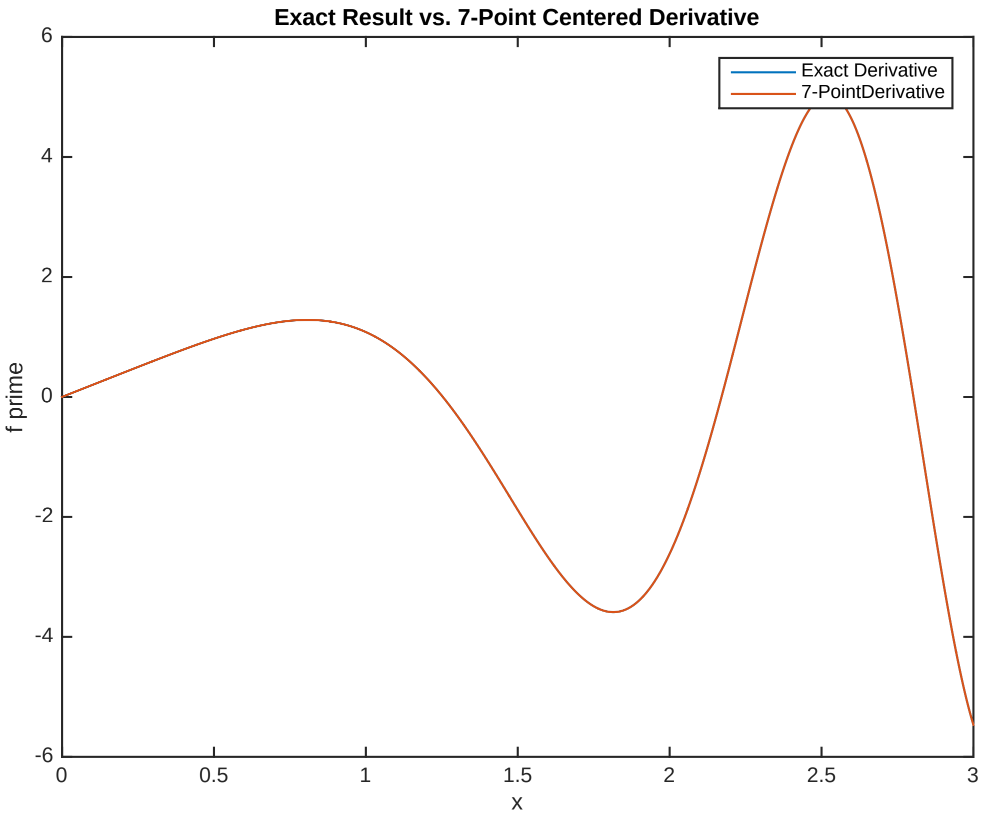

Figure 2: Exact Derivative and 7th order approximated method

Based on previous knowldege it seems the error of $T_{n,m}$ of order $O(h_n^{2m+2})$. Let's to see how does it work. Applying the composite trapezoid rule for an arbitrary number of n subintervals, we have the following exact expression for the integral I:

$$I = \frac{h_m}{2}[f(a) + 2\sum_{j=1}^{2^m-1} f(x_j) + f(b)] + \sum_{i=1}^{\infty} \alpha_i h_m^{2i} \tag{13}$$

where $x_j = a + jh_m$ and α is infinte collections of constants. It's importnat to notice that α only depends on fucntion and it's derivitivesm not the number of interval and so on. Then we can say:

$$T_{1,1} = I + \alpha_1 h^2 + \mathcal{O}(h^4) \tag{14}$$

$$T_{2,1} = I + \alpha_1 \frac{h^2}{4} + \mathcal{O}(h^4) \tag{15}$$

because:

$$T_{1,1} = \frac{b-a}{2}[f(a) + f(b)] \tag{16}$$

and

$$T_{2,1} = \frac{b-a}{4}[f(a) + f(\frac{b+a}{2}) + f(b)] \tag{17}$$

this indicates that we can eliminate the error and get the following approximation for integral:

$$T_{2,2} = \frac{4T_{2,1} - T_{1,1}}{3} \tag{18}$$

Where $I = T_{2,2} + O(h^4)$. We can iterate this method and go further:

$$T_{2,2} = I + \alpha_2 h^4 + O(h^6) \tag{19}$$

$$T_{3,2} = I + \alpha_2 \frac{h^4}{16} + O(h^6) \tag{20}$$

then by taking the linear combination of these two we have:

$$T_{3,3} = \frac{16T_{3,2} - T_{2,2}}{15} \tag{21}$$

and then $I = T_{3,3} + O(h^6)$. We can generilze this method and say for integrating $f(x)$ over domain $[a, b]$ by j subdivide region we propose:

$$T_{j,1} = \frac{b-a}{2j}(f(a) + 2\sum_{i=1}^{2^{j-1}-1} f(a + i(b - a/j)) + f(b)) \tag{22}$$

This is the Composite Trapezoid Rule. We let j = 1,...,J. We now carry out the Richardson Extrapolation step, for each of these j. We write,

$$T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} + T_{j-1,k-1}}{4^{k-1} - 1}, \qquad k = 2, 3, ..., J \tag{23}$$

Essentially, what we're doing is filling out a matrix with entries $T_{j,k}$ that only live on the diagonal and below. Notice then that the element is the very bottom right corner $T_{J,J}$ is precisely the approximation to our integral that is accurate to order 2J. Given the above theory, it is incredibly straightforward to code:

Matlab Code

```
function integral=RombergIntegrate(J,a,b)
%This computes integral of sin(x) over [a,b] correct to order 2J F=@(x)sin(x);

F=@(x)sin(x);
h=(b-a);
T=zeros(J,J);
for j=1:J
    S=zeros(1,j);
    for l=1:j
```

```
            S(l)=S(l)+feval(F,a+l.*(h/j));
end
Sp=sum(S);
 %The above for loop simply computes the sum in equation (16) above.
     T(j,1)=T(j,1)+(h/(2*j))*(feval(F,a)+2*Sp+feval(F,b));
     %The above expression is our Composite Trapezoid step.
for k=2:j T(j,k)=T(j,k)+T(j,k-1)+(T(j,k-1)-T(j-1,k-1))./(4^(k-1)-1);
     %This is the Richardson Extrapolation Step
end
end
integral=T(J,J);
%Like we note above T(J,J) is precisely the approximation of I to desired order.
```

Let us now apply the above procedure. To get an expression for the integral which is accurate to sixth order, we let J = 3. Thus, we call the function RomberIntegrate(3,0,1) which outputs 0.6877. This isn't terribly impressive but we note that RomberIntegrate(400,0,1) outputs 0.4618 which is much closer to the actual answer of 0.45970. This hints that our above prescription does indeed converge to the answer as $J \to \infty$, it's just that when J = 3, there is still a relatively large error.