Recitation 9

Recitation Instructor: Shivam Verma
Email: shivamverma@nyu.edu
Ph: 718-362-7836
Office hours: WWH 605 (2.50 - 4.50 pm, Tuesdays)

Brief Overview

- Overview from last week, including:
  - Overview of Lagrange interpolation
  - Runge's phenomenon
  - Using Chebyshev points
  - DFT and FFT, MATLAB example
  - MATLAB *interp1* example
- More about Chebyshev points
- FFT (continued)
- Recursive relation, orthogonal polynomials
- Numerical integration
- Adaptive Simpson numerical quadrature
- HW4 tips

Chebyshev nodes

Chebyshev nodes are points which are the roots of Chebyshev polynomials. They're discussed below:

$$x_i = cos(\frac{2i-1}{2n}\pi),\ i = 1, ..., n\ .$$

For these special nodes, the Lebesgue constant is finite as n increases, and therefore we do not observe Runge's phenomenon.

An example below shows how the Lebesgue constant varies for both types of nodes.

n	Λ_n for equidistant nodes	Λ_n for Chebyshev nodes
5	3.106292	2.104398
10	29.890695	2.489430
15	512.052451	2.727778
20	10986.533993	2.900825

This can be seen by:

$$Equispaced\ points:\ \Lambda_N \sim 2^N/eN\log N.$$

$$Chebyshev\ points:\ \Lambda_N \sim const\log N.$$

Chebyshev polynomials satisfy the 3-term recurrence relation:

$$T_k(x) = 2cos(x)\cdot T_{k-1}(x) - T_{k-2}(x),\ k = 2,3,...$$

One form of Chebyshev polynomials is:

$$T_k(x) = cos(k\cdot cos^{-1}(x))$$

With $T_0(x) = 1,\ T_1(x) = x.$

See Ch. 6 of [2] for more detail.

Another set of important points, sometimes also referred to as Chebyshev points, are given by:

$$x_j = cos(\tfrac{j\pi}{n}),\ j = 0,1,...,n$$

These points are the extrema of the Chebyshev polynomials.

Chebyshev points are the projection of equally-spaced points on the complex circle |z|=1 onto the interval [-1,1].

Orthogonal Polynomials & Recurrence Relations

I will use slides 4-9 from [17] for discussing orthogonal polynomials and Chebyshev recurrence relation.

FFT (Fast Fourier Transform) - continued

Main idea for FFT:

- $X_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n W_N^{kn}$ *where* $W_N = e^{-2\pi i/N}$
- Separate X_k into sum of odd and even indices terms
  - $X_k = \frac{1}{N}\sum_{r=0}^{N/2-1} x_{[2r]} W_N^{k2r} + \frac{1}{N}\sum_{r=0}^{N/2-1} x_{[2r]} W_N^{k(2r+1)}$
  - $= \frac{1}{N}\sum_{r=0}^{N/2-1} x_{[2r]} (W_N^2)^{kr} + \frac{1}{N} W_N^k \sum_{r=0}^{N/2-1} x_{[2r]} (W_N^2)^{kr}$
  - $W_N^2 = e^{-4\pi i/N} = e^{-2\pi i/(N/2)} = W_{N/2}$
  - $= \frac{1}{N}\sum_{r=0}^{N/2-1} x_{[2r]} (W_{N/2})^{kr} + \frac{1}{N} W_N^k \sum_{r=0}^{N/2-1} x_{[2r]} (W_{N/2})^{kr}$
  - $X_k = X_e[k] + W_N^k X_o[k] \Rightarrow$ *sum of* $2\ N/2$ *point DFTs* $\Rightarrow N^2/2\ +\ N$ *mults*
  - Similarly, split $N/2 \rightarrow 2\ N/4$ *point DFTs*
  - Keep splitting: $N/2,\ N/4,\ ...,\ N/2^{p-1},\ N/2^p\ =\ 1\ \Rightarrow p = log_2 N$ *times*
  - For p, we have $N^2/2^p + pN = N^2/N\ + N log_2 N = N + N log_2 N = O(N log_2 N)$

See [9] for explanation.

Numerical Integration

I will give an overview from Sec. 7.1-7.2 of textbook.

- Trapezoid rule
- Newton-Coates rule
- Simpson's rule

Adaptive Simpson

What is an adaptive quadrature? **Adaptive quadrature** is "a process in which the integral of a function is approximated using static quadrature rules on adaptively refined subintervals of the integration domain."

- Just as efficient/effective or "well behaved" integrands
- Also effective for "badly behaved" integrands for which traditional algorithms fail

The Adaptive Simpson quadrature applies the idea of adaptivity to Simpson's rule, by **dividing the interval of integration into two** and **applying adaptive Simpson's method to each subinterval in a recursive manner,** if the error exceeds a given tolerance.

I will use the handout notes and [18,19,20] for discussing this in class.

Homework 4 references

- Q1: Refer to current & previous recitation for DFT, FFT and Chebyshev points.
- Q2: Read about Clenshaw-Curtis quadrature from [12,13,14].
- Q3: Refer to handout for Adaptive Simpson quadrature theory. Refer below for helpful discussion on recursive functions and function calls in MATLAB.

Recursive function in MATLAB

A recursive function is a function which calls itself. See [15,16]. There are three main steps in writing a recursive function:

1. **Base case:** This is to specify information where recursion is not involved. This is optional. Eg. 1st two function values in fibonacci sequence.
2. **Recursive step:** This statement defines the function recursion, where the function is called with a different/changed parameter.
3. **Stopping criterion:** A statement which tells the function when to stop *recursing*! This is important, as otherwise the function will go on an infinite loop. This is typically given by an if-else statement.

Recall that we defined a function in MATLAB as follows:

```
function [out1, out2, …] = fun(param1, param2, …)
    %body
end
```

An example of a recursive *factorial* function is:

```
function [n] = fact(n)
    if n>1
        n=n*fact(n-1);    %Recursive step
    else
        n=n*1;            %stopping criterion at 1!
    end
end
```

Number of function calls in MATLAB

There are many ways to find out the number of calls that a recursive function makes. This is known as the *recursion depth*. One way is to define a return parameter which counts the number of recursive steps taken. An example for the factorial case is given below.

```
function [x,calls] = fact(x,calls)
    calls=calls+1;
    if x>1
        [y,calls]=fact(x-1,calls);    %recursive step
        x=x*y;
    else
        x=x*1;       %stopping criterion
    end
end
```

Verifying your Integration solutions using MATLAB

Once you've written the Adaptive Simpson algorithm, it would be useful to verify that you get the right solution by comparing it with MATLAB's solution to the integration.

Note: Remember to use **".^"** instead of **"^"** for element-wise exponentiation. Otherwise, you'll probably get an error, as MATLAB expects the function to be able to take a vector.

```
%Use: q = integral(fun,xmin,xmax)

%Eg. 1

func = @(x) sqrt(x)
integral(func,0,1)

%Eg. 2

func = @(x) @(x)(1- x.^2).^(3/2)
integral(func,0,1)
```

For more information, read [22].

Note: I *highly* recommend going through the handouts given by Prof. Widlund in class. I will try to cover most of the topics in the handouts, but due to lack of time, my approach may not be exhaustive or theoretical enough. You can visit me with any doubts during office hours, or via email throughout the week. I also recommend going through the references mentioned in this/previous recitation.

Helpful links

1. https://en.wikipedia.org/wiki/Interpolation
2. Deuflhard and Hohmann et al. (Numerical Analysis in Modern Scientific Computing)
3. http://pages.cs.wisc.edu/~sifakis/courses/cs412-s13/lecture_notes/CS412_12_Feb_2013.pdf
4. http://www2.lawrence.edu/fast/GREGGJ/Math420/Section_3_1.pdf
5. http://www.mathworks.com/help/matlab/ref/interp1.html
6. http://my.math.wsu.edu/help/matlab/interp1.html
7. http://butler.cc.tut.fi/~piche/numa/lecture0708.pdf
8. https://people.maths.ox.ac.uk/trefethen/ATAP/
9. https://www.youtube.com/watch?v=EsJGuI7e_ZQ
10. An interactive guide to FT - http://betterexplained.com/articles/an-interactive-guide-to-the-fourier-transform/
11. https://en.wikipedia.org/wiki/Chebyshev_polynomials
12. https://en.wikipedia.org/wiki/Clenshaw%E2%80%93Curtis_quadrature
13. http://homerreid.dyndns.org/teaching/18.330/Notes/ClenshawCurtis.pdf
14. http://www.mff.cuni.cz/veda/konference/wds/proc/pdf11/WDS11_111_m6_Novelinkova.pdf
15. https://www.cs.umd.edu/class/fall2002/cmsc214/Tutorial/recursion2.html
16. http://www.dummies.com/how-to/content/how-to-create-recursive-functions-in-matlab.html
17. http://math.nyu.edu/~stadler/num1/material/num1_interpolation
18. https://en.wikipedia.org/wiki/Adaptive_Simpson%27s_method
19. https://en.wikipedia.org/wiki/Adaptive_quadrature
20. http://www2.math.umd.edu/~mariakc/teaching/adaptive.pdf
21. http://www.math.usm.edu/lambers/mat460/fall09/lecture30.pdf
22. http://www.mathworks.com/help/matlab/ref/integral.html