PURDUE UNIVERSITY · CS 31400
NUMERICAL METHODS

HOMEWORK
David F. Gleich
October 28, 2016

Please answer the following questions in complete sentences in submit the solution on Blackboard November 7, 2016 by 5pm.

Homework 5

Problem 1 (20 points)

1. Chapter 8, Problem 1, Parts (a), (b) (5 points each)
2. (5 points) Use the setup from chapter 8, Problem 1. Suppose we present the data from part (b) as the number of milliseconds from 1900. (This involves multiplying each "year" by `365*24*60*60*1000`.) Evaluate the quadratic Lagrange interpolant directly, and via the Barycentric formulation at 50 uniformly spaced points between (and including) the interpolation end points (e.g. 1900 and 1940 translated into milliseconds). Report on any differences. (Hint: there should be something weird ...)
3. (5 points) **Note, there are much better ways of evaluating the barycentric Lagrange polynomial.** Here is one that is based on a Matlab code by one of the authors who recognized the importance of Barycentric interpolation.

```
function barylag(x,y,xx)
    # direct port of
    # http://www.mathworks.com/matlabcentral/fileexchange/...
    #    4478-barycentric-lagrange-interpolating-polynomials-...
    #    and-lebesgue-constant/content/barylag.m
    # to Julia. Not that better implmentations in Julia are possible.
    M = length(x)
    N = length(xx)
    @assert M == length(y)
    X = repmat(x,1,M)
    W = repmat(1./prod(X-X'+eye(M),1),N,1)
    xdist=repmat(xx,1,M)-repmat(x',N,1)
    fixi,fixj = findnz(xdist.==0)
    H=W./xdist
    p=vec((H*y)./sum(H,2))
    p[fixi] = y[fixj]
    return p
end
```

   Use this code to interpolate the data in part (b) instead, using the values of the data in milliseconds. Report on how this code addressed the problem. (That is, you must explain why this code was able to fix the issue you found in part 2.)

Problem 2 (20 points)

1. Chapter 8, Problem 5.

2. Implement a Julia function that computes the following:

```
"""
`chebspace`
===========

A Chebyshev analog of linspace for polynomial interpolation

* `chebspace(a,b)` generates an array of 100 Chebyshev points
between `a` and `b`
* `chebspace(a,b,n)` generates an array of `n` points between a, b
and for `n=1`, this returns b.
function chebspace(a,b,n)
# fill this in!
end
function chebspace(a,b)
    return chebspace(a,b,100)
end
```

(Hint, see equation 8.15)

Problem 3 (10 points)

Write a paragraph or two (100-200 words) about what you learned about polynomials and polynomial interpolation. You should also think about answering the following questions:

1. Do you think you'd ever use polynomial interpolation?
2. If so, how?
3. If not, why not?

Be honest, but remember to be thoughtful.

Problem 4 (10 points)

1. Download and install ApproxFun. Then type `using ApproxFun` to get it added to your Julia instance. Also type `using Plots` to get the plotting interaface.

2. Run

   ```
   f = Fun(x->sign(x))
   plot(f)
   ```

   How does this relate to the Weistrauss approximation theorem?

3. Try using ApproxFun to integrate the most complicated function (in one dimension) that you can think of! Can you stump ApproxFun?

4. One of the advantages of Chebfun is we can get computer representations of functions (as polynomials) that have no analytic form what-so-ever. Consider the following code which builds a polynomial approximation of

$$\rho(t) = \max |\lambda_i \left(t \begin{bmatrix} 1 & 2 & 0 \\ 0 & 2 & 1 \\ 1 & 0 & 2 \end{bmatrix} + (1-t) \begin{bmatrix} 1 & 1 & 0 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{bmatrix} \right) |$$

   here λ_i is just the ith eigenvalue. So this is looking at the largest magnitude eigenvalue. (Example from Chebfun: http://www.chebfun.org/examples/approx/NoisyNonsmooth.html)

```
A = [1 2 0; 0 2 1; 1 0 2.0]
B = [1 1 0; 1 -1 1; -1 1 1.0]
f = Fun(t -> maximum(abs(eigvals(t*A + (1-t)*B))),[0,1])
plot(f)
```

Run this code and show your result!

5. (Fun bonus question, worth no points.) This requires Matlab. Download and install the Matlab package `chebfun` Run `chebsnake('equi')` (Remember this when you are tempted to use equally spaced points for polynomial interpolation!)

Problem 5 (40 points)

1. (10 points) Chapter 9, Exercise 1,
2. (10 points) Chapter 9, Exercise 2.
3. (10 points) Chapter 9, Exercise 3.
4. (10 points) Chapter 9, Exercise 5.