Homework 3

Due date: March 5, 2020, 11:55PM.

Bibliography: Trefethen & Bau, Lectures 10-14. Demmel, 1.6

Parts 1-3 on Problem 1 replace your Test 2 Problem 1 score. Part 1 of Problem 2 replaces your Test 2 Problem 2 score.

1 Horner's scheme revisited: conditioning, stability, accuracy

1.1 Horner's scheme

Horner's algorithm for evaluation of a polynomial $p(x)=a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$ is

Input: $x, \boldsymbol{a} = (a_n, \ldots, a_0)$
$p = a_n$
for $i = n-1$ downto 0
  $p = p \cdot x + a_i$
end
return p

Define the function $f\colon \mathbb{R}^3 \to \mathbb{R}$ as a multiplication followed by an addition

$$f(p,x,a) = p \cdot x + a.$$

This definition reflects actual hardware implementation of a floating point operation (FLOP) in a central processing unit (CPU) in which p, x, a are loaded onto registers (memory locations accessed at clock speed) and $f(p,x,a)$ is evaluated in a single clock cycle. When extended to vectors as in $\boldsymbol{A}\boldsymbol{x}+\boldsymbol{y}$, the function is called is called an "axpy" and is the basis for BLAS, the basic linear algebra algorithms. Most modern CPUs are capable of vectorizing $\boldsymbol{A}\boldsymbol{x}+\boldsymbol{y}=\boldsymbol{b}$, i.e., evaluating each component of $\boldsymbol{b}$ in single clock cycles. The floating point variant of f is the function flop: $\mathbb{F}^3 \to \mathbb{F}$, and introduces an arithmetic error

$$\operatorname{flop}(p,x,a) = (p \cdot x + a)(1+\delta), |\delta| \leqslant \epsilon, \epsilon = \text{machine epsilon}.$$

1.2 Questions

1.2.1 Error propagation estimate

Question. Find a bound for $\delta p = p(x+\delta x, \boldsymbol{a}+\delta \boldsymbol{a}) - p(x, \boldsymbol{a})$ with $\delta x, \delta \boldsymbol{a}$ denoting the floating point errors introduced in Horner's algorithm.

Solution.

1.2.2 Stability of the algorithm $\tilde{p}(x)$

Question. Is Horner's algorithm forward stable?

Solution.

Question. Is Horner's algorithm backward stable?

Solution.

1.2.3 Conditioning of the problem $p(x)$

Question. Show that the relative condition number of polynomial evaluation is well approximated by

$$\kappa = \frac{|a_0| + |a_1 x| + \cdots + |a_n x^n|}{|a_0 + a_1 x + \cdots + a_n x^n|}$$

SOLUTION.

1.2.4 Conditioning of evaluation of the null polynomial

QUESTION. Is the evaluation of $z(x)=0$ well-conditioned, ill-conditioned or ill-posed?
SOLUTION.

1.2.5 Numerical experiments

QUESTION. Implement Horner's algorithm and test each of the four above estimates on Wilkinson's polynomial

$$w(x)=\prod_{i=1}^{n}(x-i)$$

for $n=10,20,30$.

SOLUTION. Mathematica can be used to generate the polynomial coefficients and export them to a mat-format file readable in Octave.

```
Mathematica
In[1]:= w[n_,x_]:=Product[(x-i),{i,n}];
        a=Table[CoefficientList[w[n,x],x],{n,1,3}];
        Grid[a]
```

-1	1		
2	-3	1	
-6	11	-6	1

```
In[2]:= dir="/home/student/courses/MATH662/homework/hw03";
        If[!DirectoryQ[dir],CreateDirectory[dir]];
        SetDirectory[dir];
```

Null

```
In[3]:= a=CoefficientList[w[10,x],x];
        Export["a10.mat",a];
        a=CoefficientList[w[20,x],x];
        Export["a20.mat",a];
        a=CoefficientList[w[30,x],x];
        Export["a30.mat",a];
```

Null

```
In[4]:= a=CoefficientList[Expand[(x-2)^9],x]
```

$\{-512,2304,-4608,5376,-4032,2016,-672,144,-18,1\}$

```
In[5]:= Export["ap2.mat",a];
```

Null

```
In[6]:= Expand[(x-2)^9]
```

$x^9-18\,x^8+144\,x^7-672\,x^6+2016\,x^5-4032\,x^4+5376\,x^3-4608\,x^2+2304\,x-512$

```
In[7]:=
```

Read the coefficient files.

```
octave: cd /home/student/courses/MATH662/homework/hw03
```

```
octave: load ap2; a=Expression1; a'
   ans =

      -512   2304  -4608   5376  -4032   2016   -672    144    -18      1
```

$(-512 \;\; 2304 \;\; -4608 \;\; 5376 \;\; -4032 \;\; 2016 \;\; -672 \;\; 144 \;\; -18 \;\; 1)$

```
octave: function p=horner(x,a)
          n=max(size(a))-1;
          p=a(n+1);
          for i=n:-1:1
            p=p*x+a(i);
          end
        end
octave: horner(2,[1,1,1])
```

7

```
octave: horner(2 * (1 + eps), a)
```

$2.8422e-12$

```
octave: horner(2., a)
```

0

```
octave: eps
```

$2.2204e-16$

```
octave: m=1000;
        kappa=ones(m,1);
        for k=1:m
            kappa(k) = horner(2*(1+k*eps),a)/(k*eps);
        end
octave: max(kappa)
```

12800

```
octave:
```

2 Reduction to tridiagonal form

2.1 Algorithm

Write an algorithm that uses Householder reflectors to reduce a matrix $A \in \mathbb{R}^{m\times m}$ to tridiagonal form.

Algorithm 1

Given $A \in \mathbb{R}^{m\times m}$
for $j = 1{:}\, m-2$
$x = A(j+1{:}\, m, j)$; $e = I_{m-j}(:, 1)$
$A(j+1{:}\, m, j) = \|x\| \cdot e$
$v = A(j+1{:}\, m, j) - x$; $v = v / \|v\|$
for $k = j+1{:}\, m$
$A(j+1{:}\, m, k) = A(j+1{:}\, m, k) - 2 \cdot v \cdot (v' \cdot A(j+1{:}\, m, k))$
end
$x = A(j, j+1{:}\, m)$; $e = e'$;
$A(j, j+1{:}\, m) = \|x\| \cdot e$;
$v = A(j, j+1{:}\, m) - x$; $v = v / \|v\|$

for $k=j+1{:}\,m$

$\quad A(k, j+1{:}\,m) = A(k, j+1{:}\,m) - 2\cdot(A(k, j+1{:}\,m)\cdot v')\cdot v$

end

end

```
octave: function H=hess(A)
          H=A; [m,n]=size(A);
          for j=1:n-2
            x=H(j+1:m,j); e=eye(m-j)(:,1);
            H(j+1:m,j)=norm(x)*e;
            v=H(j+1:m,j)-x; v=v/norm(v);
            for k=j+1:n
              H(j+1:m,k)=H(j+1:m,k)-2*v*(v'*H(j+1:m,k));
            end
          end
         end
octave: function H=trihess(A)
          H=A; [m,n]=size(A);
          for j=1:n-2
            x=H(j+1:m,j); e=eye(m-j)(:,1);
            H(j+1:m,j)=norm(x)*e;
            v=H(j+1:m,j)-x; v=v/norm(v);
            for k=j+1:n
              H(j+1:m,k)=H(j+1:m,k)-2*v*(v'*H(j+1:m,k));
            end;
            x=H(j,j+1:m); e=e';
            H(j,j+1:m)=norm(x)*e;
            v=H(j,j+1:m)-x; v=v/norm(v);
            for k=j+1:m
              H(k,j+1:m)=H(k,j+1:m)-2*(H(k,j+1:m)*v')*v;
            end
          end
        end
octave: A=rand(6); B=hess(A)
   B =

      0.78215   0.71152   0.27652   0.06195   0.07758   0.21170
      0.78566   0.80476   0.65135   0.76913   1.29802   0.73950
      0.00000   0.69271   0.67738   0.47653   0.42505   0.27126
      0.00000   0.00000   1.10421   0.25141   0.23196  -0.15548
      0.00000   0.00000   0.00000   0.06575  -0.20646   0.25765
      0.00000   0.00000   0.00000   0.00000   0.22732  -0.52527

octave: C=trihess(A)
   C =

      0.78215   0.79837   0.00000   0.00000   0.00000   0.00000
      0.78566   1.32471   1.46299   0.00000   0.00000   0.00000
      0.00000   1.08066   0.63612   0.65691   0.00000   0.00000
      0.00000   0.00000   0.23730   0.71707   0.53007   0.00000
      0.00000   0.00000   0.00000   0.39858  -0.30470  -0.24957
```

```
   0.00000   0.00000   0.00000   0.00000   0.15313   0.16872

octave:
```

2.2 Implementation

Implement the above algorithm and test on matrices of size $n = 50, 100, 250$:

1. Hilbert matrices
2. Vandermonde matrices arising from discretization of the interval $[-1, 1]$
3. Normal random matrices.