Numerical Experiment B

Xinzhi Li

Student ID: **2022211084**

School of Physics Science and Technology, ShanghaiTech University, Shanghai 201210, China

Email address: lixzh2022@shanghaitech.edu.cn

December 5, 2022

I. INTRODUCION

Interpolation and ***Iteration*** are the most common numerical methods. The former is a type of estimation, a method of constructing new data points based on the range of a discrete set of known data points in the mathematical field of numerical analysis. The latter is the repetition of a process in order to generate a sequence of outcomes which can produce approximate numerical solutions to some certain problems. In engineering and science, one often has a number of data points, obtained by sampling or experimentation, which represent the values of a function for a limited number of values of the independent variable. It is often required to interpolate; that is, estimate the value of that function for an intermediate value of the independent variable. We will use some interpolation methods to approximate functions and use ***Newton's iteration*** to solve nonlinear equations.

II. PROBLEM

1. Use ***Lagrange interpolating polynomial*** to approximate the function

$$f(x) = \frac{1}{1+25x^2} \qquad x \in [-1,1] \tag{1}$$

   and observe the Runge phenomenon.

2. The population of China from 1988 to 1996 is given by:

year	1988	1989	1990	1991	1992	1993	1994	1995	1996
population($\times 10^8$)	11.10	11.27	11.43	11.58	11.72	11.85	11.99	12.11	12.24

Table 1: The population from 1988 to 1996

   According to the data above and predict the population in 2018, 2019. Use ***Malthusian growth model***,

$$N(t) = e^{a+bt} \tag{2}$$

   where $N(t)$ is the number of population. Take logarithm,

$$\ln N(t) = y(t) = a + bt \tag{3}$$

3. Given nonlinear equations

$$F(x) = \begin{cases} 3x_1 - \cos(x_2x_3) - \dfrac{1}{2} = 0 \\ x_1^2 - 81(x_2+0.1) + \sin x_3 + 1.06 = 0 \\ e^{-x_1x_2} + 20x_3 + \dfrac{1}{3}(10\pi - 3) = 0 \end{cases} \tag{4}$$

   Use ***Newton's iteration*** to solve the equations with $\epsilon = 10^{-5}$. Calculate the iterations.

III. NUMERICAL RESULTS

1. We take n nodes:$-1 \le x_1 < x_2 \cdots < x_n \le 1$, the lagrange interpolating polynomial is given by

$$l_k(x) = \prod_{j \neq k} \frac{x - x_j}{x_k - x_j} \tag{5}$$

Then the approximated function is

$$L = \sum_{k=1}^{n} l_k(x) f(x_k) \tag{6}$$

The results are shown in Figure 1

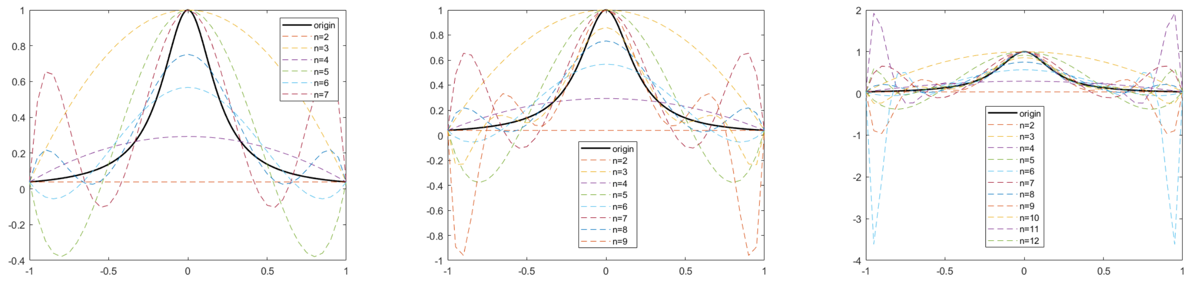
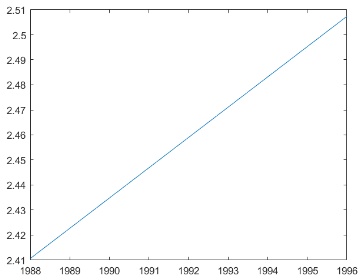


Figure 1: Lagrange interpolating polynomial

2. We use least square to fix the curve, the resulting function is shown in Figure 2



N_2018	16.0123
N_2019	16.2071

Figure 2:

3. The Newton's iteration is given by

$$x^{(k+1)} = x^{(k)} - [\nabla F(x^{(k)})]^{-1} F(x^{(k)}), \qquad k = 0, 1, 2, \cdots \tag{7}$$

where

$$\nabla F = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ & & \cdots & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \tag{8}$$

We start with $x^{(0)} = (0, 0, 0)^T$. Then the total iteration is 3, and the solution is

$$x = \begin{pmatrix} 0.4996 \\ -0.0900 \\ -0.5259 \end{pmatrix} \tag{9}$$

IV. REMARK

We have observed the Runge phenomenon, that is, with the increasing nodes and the dimension of the polynomials, it becomes more accurate, but after $n = 7$, the result performs a large deviation at the either end. The best approximation depends on the interval that you want.

According to the data in 2018 and 2019, the population is 1.405×10^9 and 1.410×10^9. It perhaps due to the lifestyle is totally changed for the young generation and the marriage rate decreases.

The Newton's iteration is so fast, since it is a quadratic approach algorithm.

V. CODE

```
%-----------------------------plot origin function----------------------------%
a=-1:0.001:1;
b=1./(1+25.*a.^2);
plot(a,b,'k','LineWidth',1.5);
hold on
%----------------Lagrange interpolating polynomial----------------%
n=[1,2,3,4,5,6,7]; %nodes%
for i = n
    x=-1:2/i:1;
    y=1./(1+25.*x.^2);
    x_2=-1:0.05:1;
    lagrange(x,y,x_2);
end
legend('origin','n=2','n=3','n=4','n=5','n=6','n=7')
%----------------------------constructe polynomial-----------------------------%
function L = lagrange(x,y,x_2)
a=x_2;
L=zeros(1,length(a));

for i = 1:length(a)
    l = ones(1,length(x));
    for k = 1:length(x)
        for j = 1:length(x)
            if j ~= k
                l(k)=l(k)*(a(i)-x(j))/(x(k)-x(j));
            end
        end
        L(i)=L(i)+l(k)*y(k);
    end
end

plot(a,L,'--');
end
```

```
%----------------------------population prediction-------------------------%
t=1988:1996;
N=[11.10,11.27,11.43,11.58,11.72,11.85,11.99,12.11,12.24];
y=log(N);
%-----------------------------------Least Square-----------------------------%
a=length(t);
b=sum(t);
c=sum(t.^2);


d=sum(y);

e=sum(t.*y);

A=eye(2);
```

```
A(1,1)=a;
A(1,2)=b;
A(2,1)=b;
A(2,2)=c;

f=zeros(2,1);
f(1,1)=d;
f(2,1)=e;

x=A\f;
%----------------plot--------------------%
y_1=zeros(1,length(t));
for i = 1:length(t)
    y_1(i)=x(1,1)+x(2,1)*t(i);
end
plot(t,y_1);

%--------------prediction-----------------%

N_2018=exp(x(1,1)+x(2,1)*2018);
N_2019=exp(x(1,1)+x(2,1)*2019);
```

```
%-----------------------Newton iteration----------------%
x_0=zeros(3,1);
f=-Vect(x_0);
F=mat(x_0);
dx=F\f;
x_1=x_0+dx;
n=0;
while norm(x_1-x_0)>=10^(-5)
    Dx=mat(x_1)\(-Vect(x_1));
    t=x_1;
    x_1=Dx+t;
    x_0=t;
    n=n+1;
end

function F = mat(x)

F=ones(3,3);
F(1,1)=3;
F(1,2)=x(3)*sin(x(2)*x(3));
F(1,3)=x(2)*sin(x(2)*x(3));
F(2,1)=2*x(1);
F(2,2)=-81;
F(2,3)=-cos(x(3));
F(3,1)=-x(2)*exp(-x(1)*x(2));
F(3,2)=-x(1)*exp(-x(1)*x(2));
F(3,3)=20;

end

function f = Vect(x)

f=ones(3,1);
f(1)=3*x(1)-cos(x(2)*x(3))-0.5;
f(2)=x(1)^2-81*(x(2)+0.1)+sin(x(3))+1.06;
f(3)=exp(-x(1)*x(2))+20*x(3)+(10*pi-3)/3;

end
```