非线性方程的数值解第四次作业

一、问题（方程组）：

设映射 F： $R^n \to R^n$ 定义为

$$F_1(x) = x_1 - e^{\cos\left(\frac{1}{n+1}(x_1+x_2)\right)}$$

$$F_i(x) = x_i - e^{\cos\left(\frac{1}{n+1}(x_{i-1}+x_i+x_{i+1})\right)}$$

$$F_n(x) = x_n - e^{\cos\left(\frac{1}{n+1}(x_{n-1}+x_n)\right)}$$

二、数值实验

实验：用一点割线法、两点割线法和改进 n 步迭代法，求解上述非线性方程组的解。

用 Matlab2018b 进行一点割线法、两点割线法和改进 n 步迭代法的编程，方程的维数为 100，初始值为 x0=[0,⋯,0] [1,⋯,1] [20,⋯,20]三种，精度 $\varepsilon = 10^{-6}$， h=0.5。

停止准则为： $\| F(x^k) \| < \varepsilon$

三、实验程序

(onepoint.m) (twopoint.m) (hundpoint.m) (sess_run4.m)

File: onepoint.m （一点割线法）

```
function [x,iter]=onepoint(f,x0,err)
cost=[];
x=x0;
dim=length(x0);%维度 itermax=10; %迭代次数
h=0.5;%随便选的差分距离
v=h*ones(1,dim);   %构造对角线
H=diag(v);   %构造一个只有对角线 v 的 H,维度（n，n）
for iter=1:itermax
    xh=zeros(dim);
    J=[];   %初始化 J 矩阵
    fx=f(x);   %计算出 F（x0）
    %size(fx)
    e=zeros(dim,1);%初始我的 ei=[0,0,0,0……，  0]'
    for i = 1:dim
        e(i)=1;%构造出 ei
        xh(:,i)=x+((norm(fx,2))*H*e);    %更新 x+h
        e(i)=0;%变回初始情况
    end
    fxh=f(xh);%计算出 F（xh）
    J=(fxh-fx)/(h*norm(fx,2));%割线法求出 J
    %size(fx)
    %size(J)
    x=x-J\fx   ;%更新 x
    k=(norm(f(x),2));
    cost=[cost,k];%误差
    if k<=err
       break
    end
end
```

File: twopoint.m （两点割线法）

```
function [x,iter]=twopoint(f,x0,err)
cost=[]; xk=x0;
dim=length(x0);%维度 alpha=0.5;%随便选的差分距离
h=alpha*ones(1,dim);  %构造对角线
x=x0; itermax=100; %迭代次数
for iter=1:itermax
    H=diag(h);  %更新 H
    xh=zeros(dim);%初始化 x+h
    xk=x;   %存储 x（k-1）
    J=[];  %初始化 J 矩阵
    fx=f(xk);  %计算出 F（x0）
    e=zeros(dim,1);%初始我的 ei=[0,0,0,0……, 0]'
    for i = 1:dim
        e(i)=1;%构造出 ei
        xh(:,i)=xk+(H*e);   %更新 x+h
        e(i)=0;%变回初始情况
    end
    fxh=f(xh);%计算出 F（xh）
    J=(fxh-fx)/H;%割线法求出 J
    x=xk-J\fx  ; %更新 x
    x1=x;
    x2=xk;
    h=(x1-x2)';  %更新 h
    k=(norm(f(x),2));
    cost=[cost,k];%误差列表
    if k<=err
       break
    end
end
```

File: hundpoint.m （改进 n 步迭代法）

```
function [x,iter]=hundpoint(f,x0,err)
cost=[];itermax=10; %迭代次数
x=x0;dim=length(x0);%维度
h0=0.5;%随便选的差分距离   v=h0*ones(1,dim);  %构造对角线
H=diag(v);  %构造一个只有对角线 v 的 H,维度（n，n）
xh=zeros(dim);J=[];  %初始化 J 矩阵
fx=f(x);  %计算出 F（x0）
e=zeros(dim,1);%初始我的 ei=[0,0,0,0……， 0]'
for i = 1:dim
    e(i)=1;%构造出 ei
    xh(:,i)=x+((norm(fx,2))*H*e);   %更新 x+h
    e(i)=0;%变回初始情况
end
fxh=f(xh);%计算出 F（xh）
J=(fxh-fx)/(h0*norm(fx,2));%割线法求出 J
for iter=1:itermax
    x=x-J\f(x);
    costtemp=(norm(f(x),2));
    cost=[cost,costtemp];%误差
    if costtemp<err
        break
    end
    k=mod(iter,dim);
    if k==0
        k=dim;
    end
    h0=h0*norm(f(x),2);
    x1=x;
```

```
        x1(k)=x1(k)+h0;
        J(:,k)=(f(x1)-f(x))/h0;
    end
```

File:sess_run4.m(主程序)

```
clc;clear;
dim=100; %维度

A=eye(dim);
P=diag(ones(1,dim))+diag(ones(1,dim-1),1)+diag(ones(1,dim-1),-1);
B=P/(dim+1);%三对角矩阵

f=@(x)A*x-exp(cos(B*x));%定义函数

x0=1*ones(dim,1);%intialize
err=1e-6;
format shorte
iterlist=[];

[x,iter]=onepoint(f,x0,err);
[x,iter]=twopoint(f,x0,err);
[x,iter]=hundpoint(f,x0,err);
%toc
x=x';
format shorte
iterlist=[iterlist,iter];
%end
```

四、数值实验

数值实验结果：

	一点割线法迭代法		两点割线法迭代法		改进 n 步迭代法	
初始值	迭代次数	$\|F(x^k)\|$	迭代次数	$\|F(x^k)\|$	迭代次数	$\|F(x^k)\|$
[0, ⋯, 0]	1	5.8529e-02	1	8.2267e-02	1	5.8529e-02
	2	2.7117e-07	2	8.7796e-05	2	6.1283e-05
			3	2.8117e-10	3	6.4473e-08
[1, ⋯, 1]	1	2.3358e-02	1	3.1333e-02	1	2.3358e-02
	2	4.3191e-08	2	2.1032e-05	2	1.5283e-05
			3	2.5695e-11	3	1.0021e-08
[20, ⋯, 20]	1	2.6608e+00	1	2.3180e+00	⋯3	2.7204e-03
	2	2.1912e-03	2	1.5038e-02	4	8.7536e-05
	3	1.4988e-09	3	1.3962e-06	5	2.8170e-06
			4	8.1975e-13	6	9.0734e-08

实验结果：在收敛速度上，一点割线法法最快，两点割线法法次之，改进 n 步迭代法最慢。而两点割线法法因为使用了之前两点的信息，比一点割线法计算的函数值要更少，而改进 n 步迭代法主要是为了克服计算的稳定性而提出的算法。