Yunus Güngör No:150150701 Date:04.04.2016

BLG202E CRN:21843 Homework 2

Q1)a)

Matlab Code:

>> syms x;

>> y(x)=x^4-x-10;

>> g(x)=10/(x^3-1);

>> error=diff(g,x);

>> i=0;

>> x0=2;

>> while (abs(y(x0))>(10^-4)) && (abs(error(x0))<1)

i=i+1;

x0=g(x0);

vpa(x0)

end

>> i

i =

0

>> x0

x0 =

2

>> vpa(abs(error(x0)))

ans =

2.4489795918367346938775510204082

It does not converge, therefore we cannot find a root via this function.

b)

Matlab Code:

>> syms x;

>> y(x)=x^4-x-10;

>> g(x)=(x+10)^(1/4);

>> error=diff(g,x);

>> i=0;

>> x0=2;

>> while (abs(y(x0))>(10^-4)) && (abs(error(x0))<1)

i=i+1;

x0=g(x0);

vpa(x0)

end

ans =

1.8612097182041991978824374939665

ans =

1.8558045970397769563351377355042

ans =

1.8555931396181665093258308444551

ans =

1.8555848655790224285625663784682

>> i

i =

4

>> vpa(y(x0))

ans =

0.0000082740391440807632644659869008921

We find the root as 1.8555848655790224285625663784682 with 0.0000082740391440807632644659869008921 error

C)

Matlab Code:

>> syms x;

>> y(x)=x^4-x-10;

>> g(x)=((x+10)^(1/2))/x;

>> error=diff(g,x);

>> i=0;

>> x0=1.8;

>> while (abs(y(x0))>(10^-3)) && (abs(error(x0))<1)

x0=vpa(g(x0));

i=i+1;

x0

end

x0 =

1.9083960041464075829778723767147

x0 =

1.8082485920442550027104075764334

x0 =

1.9003544323640786564065045579104

x0 =

1.8152871776991452120891954106111

x0 =

1.893550102827184477178912268804

x0 =

1.821289367723323050635330437299

x0 =

1.887789090100565387729242577206

x0 =

1.826404942680599839549370174487

x0 =

1.8829088595893103177274313098583

x0 =

1.8307628207399336907957044371097

x0 =

1.8787729112735388411964448419786

x0 =

1.8344737436710255851241601666824

x0 =

1.8752664120437583071606940435175

x0 =

1.8376326802399104459475967063766

x0 =

1.8722926145425613797284562132397

x0 =

1.840320957363649579495492122601

x0 =

1.8697699066579860416634800584327

x0 =

1.8426081361555956111799814928141

x0 =

1.8676293702986127481257936278787

x0 =

1.8445536536588776668108324488113

x0 =

1.8658127540709327795929447388865

x0 =

1.8462082525879283560339059741272

x0 =

1.8642707842671977520251584999462

x0 =

1.8476152214875090451432997866964

x0 =

1.8629617537840428857804448570299

x0 =

1.8488114668921501076428489784575

x0 =

1.8618503403955735100310029015985

x0 =

1.8498284376939050951274821731556

x0 =

1.8609066150441301828471940257381

x0 =

1.8506929202592449264999618528583

x0 =

1.8601052081049038702054477978147

x0 =

1.8514277210512606955635894766462

x0 =

1.8594246073821116062421218114989

x0 =

1.8520522517261359912014045929836

x0 =

1.8588465662431579896041441595826

x0 =

1.852583029955686127531622348807

x0 =

1.8583556040472012875583873913716

x0 =

1.8530341076234371873264443316518

x0 =

1.8579385840681698749711091596521

x0 =

1.8534174365725034181316677680518

x0 =

1.8575843565963977464799860218786

x0 =

1.8537431807578854484477680272004

x0 =

1.857283456940742340133467353068

x0 =

1.8540199824732263880665366815702

x0 =

1.8570278497320937339086413652712

x0 =

1.8542551892763964212024989028187

x0 =

1.8568107123182705250050769306939

x0 =

1.8544550473201493740430961663084

x0 =

1.8566262511935896734816243590013

x0 =

1.8546248659924861364968972198158

x0 =

1.8564695463669226584561184257414

x0 =

1.8547691580746777531765888713256

x0 =

1.8563364193742266301126712316167

x0 =

1.8548917590216722700480965387001

x0 =

1.856223321313078439346941982043

x0 =

1.854995928448903882102125494719

x0 =

1.8561272378400888102330495298778

x0 =

1.8550844364611947063755505499763

x0 =

1.8560456085455181847337438865998

x0 =

1.8551596370742444606262280459881

x0 =

1.8559762585179248410769616322715

x0 =

1.8552235306488347687349459870576

x0 =

1.8559173402475624040018246292797

x0 =

1.8552778169749470670318164681379

x0 =

1.8558672843006768455239516996316

x0 =

1.8553239404009934306560742714906

x0 =

1.8558247574362621841961386426692

x0 =

1.8553631281965826403751902023266

x0 =

1.8557886270392300645872568705202

x0 =

1.8553964231607174776402325708223

x0 =

1.8557579309151787478237318968349

x0 =

1.8554247113367291814544701170692

x0 =

1.8557318516369014353140353357689

x0 =

1.8554487455668704945900626304591

x0 =

1.8557096947555500680336603794181

x0 =

1.8554691655100929773292473316082

x0 =

1.855690870293409571771610186919

x0 =

1.8554865146533601938182079608304

x0 =

1.8556748770234329012738641710924

x0 =

1.8555012547675204252390580159667

x0 =

1.8556612891154767183308687519528

x0 =

1.8555137781912446561825374115481

x0 =

1.8556497447926168071493165252247

x0 =

1.8555244182690849948786712504103

x0 =

1.8556399366947468978729943964602

x0 =

1.8555334582208355110586271489501

x0 =

1.8556316036923409961038226768955

x0 =

1.8555411386778091595998626227849

x0 =

1.8556245239320278719331167447431

>> vpa(y(x0))

ans =

0.000982179398310899865422275452152

>> i

i =

89

>>

D)

does not converge but, converges at step 4 with a appoximite error of and converges at step 89 with a approximate error of .

Q2)

,

, ,

, , , , , ,

, , , ,

, ,

, , ,

, ,

, , ,

, ,

, , ,

, ,

, ,

Q3)

, (u and v diagonal) (it only has elements on diagonal.)

,

( is eigen vector), therefore;

where is the singular values of and eigenvectors of .

Q4)

To get the smallest singular value k must be as large as possible. To act reasonable, getting k as 256 gets close enough to zero.

For

For

For

For

Q5)

Matlab Code:

>> load('C:\Users\yunus\Desktop\2015-2016 Bahar\BLG202\HW2\A.mat')

>> B=A\*transpose(A);

>> [V,D] = eig(B); %Get eigenvalues by eigenvalue decomposition

>> v0 = rand(500,1); %Create random v0

>> for k=1:500 %PIM Method

w=B\*v0;

v=w/norm(w);

eigValue(k)=v'\*B\*v;

v0=v;

end

>>max(eigValue) %find dominant eigenvalue

ans =

4464.70498441718

Power iteration method only found a range of eigenvalues with highest values. But eigenvalue decomposition find all the eigenvalues.