how U doin'?

$$\text{🐋🐋}^{🐋} = \int_{🎃}^{🎅} 🐑 \, d🍀$$

Contents

1　Problems

问题：

$$-\frac{\mathrm{d}}{\mathrm{d}x}(p(x)\frac{\mathrm{d}u}{\mathrm{d}x}) + q(x)u \;=\; f(x), \quad x \in (a,b), \qquad (1)$$

$$u(a) \;=\; 0, \quad u'(b) = 0. \qquad (2)$$

其中 $p(x) \in C^1(\bar{I}), p(x) \geq p_{\min} > 0, q \in C(\bar{I}), q(x) \geq 0, f \in H^0(I)$,
$\bar{I} = [a,b]$. 等式两边乘以 v 在 I 上积分，并运用分部积分可得

$$a(u,v) \;=\; (f,v),$$

其中，

$$a(u,v) = \int_a^b \left(p\frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x} + q\,u\,v \right) \mathrm{d}x, \quad (f,v) = \int_a^b f v \,\mathrm{d}x.$$

> 对两点边值问题 (1)-(2)，在等距网格上（区间个数为 N 个），给出其相应的一次有限元程序。
> 要求：
>
> 1. 要求网格点个数 N 及区间 $[a,b]$ 在程序中可以变化；
> 2. 要求程序中可以很容易更换函数 p, q 和 f；
> 3. 先运行 $[0,1]$ 之间，$p = 1, q = 0, f = \pi^2 sin(\pi x), N = 10$ 的情形，输出所有结点的值；

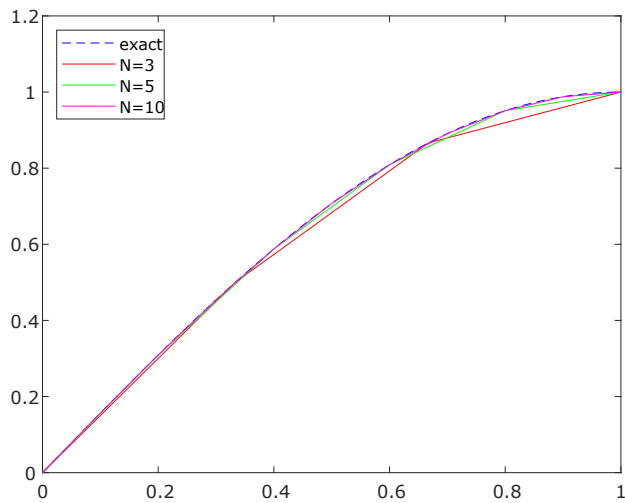*This article was typeset by Mark Taylor using the LATEX document processing system.

2   Theory

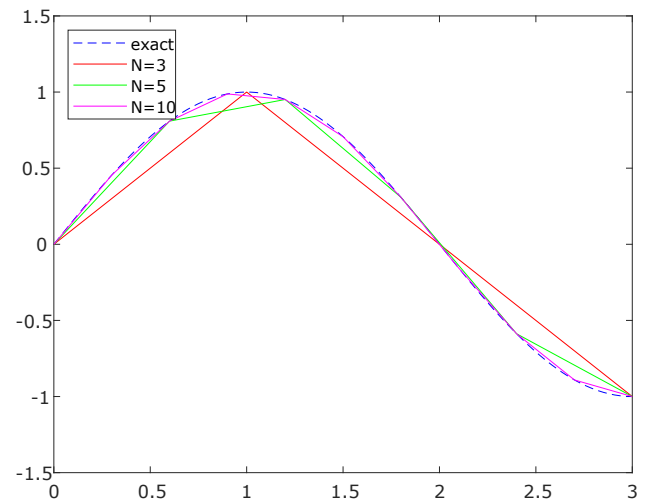See reference.mlx in the *src* folder.

3   Solutions

🔑 🔑 🔑

See graphs of those two test problems as follows. View code here🖥
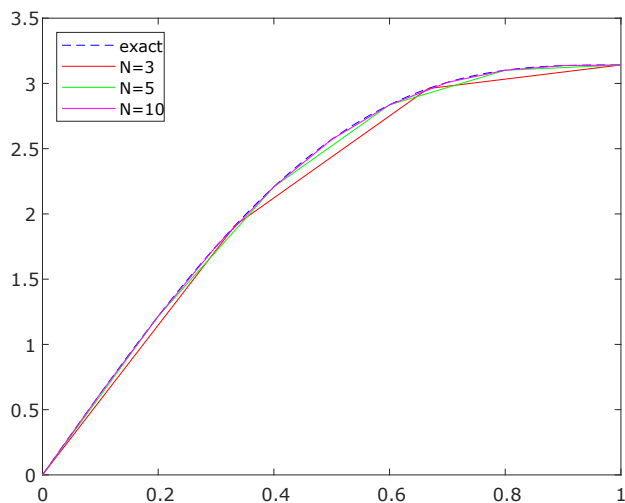

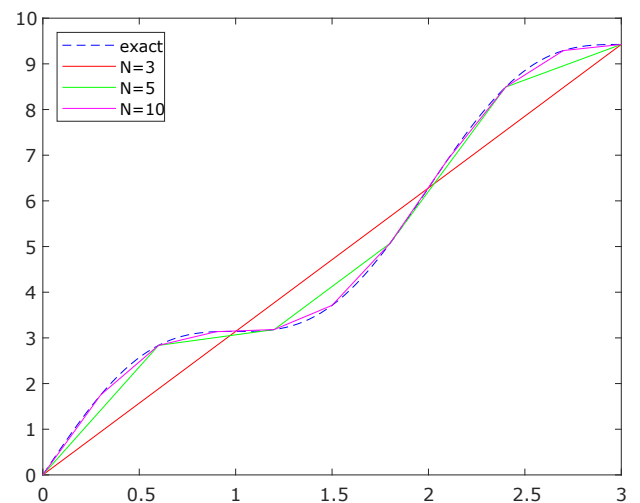(a) Fitting for problem 1 on $[0, 1]$ (b) Fitting for problem 1 on $[0, 3]$

Figure 1: BVP Test 1

(a) Fitting for problem 2 on $[0, 1]$ (b) Fitting for problem 2 on $[0, 3]$

Figure 2: BVP Test 2

4   Code

4.1   main function

```matlab
% ================================================================
% Solve a linear two-point boundary-value problem
%                  -(p(x)u'(x))'+q(x)u(x)=f(x),     a<=x<=b,
% with initial conditions
%                            u(a)=0, u'(b)=0.
% ================================================================

% ****************************************************************
% Test 1
% f=@(x)pi^2/4*sin(pi/2*x);
% u=@(x)sin(pi/2*x);

% Test 2
f=@(x)pi^2*sin(pi*x);
u=@(x)sin(pi*x)+pi*x;
% ****************************************************************

p=@(x)x.^0; % p(x)=1
q=@(x)x.*0; % p(x)=0
a=0;
% b=1;
b=3;
N=[3,5,10];

% Since the dimensions are different, we can't
% simply use a matrix for storage. List?
uc1=PLRR(f,p,q,a,b,N(1));
uc2=PLRR(f,p,q,a,b,N(2));
uc3=PLRR(f,p,q,a,b,N(3));

% output coef vector of approx solution in command window
disp(uc3)

% plot graphs of exact solution & approximate solution
t1=linspace(a,b,N(1)+1);
t2=linspace(a,b,N(2)+1);
t3=linspace(a,b,N(3)+1);
tt=linspace(a,b,100+1);

% approx solutions
u1=PLRR_intpol(uc1,a,b);
u2=PLRR_intpol(uc2,a,b);
u3=PLRR_intpol(uc3,a,b);

figure
plot(tt,u(tt),'--b',t1,u1,'r',...
    t2,u2,'g',t3,u3,'m')
legend('exact',sprintf('N=%d',N(1)),sprintf('N=%d',N(2))...
    ,sprintf('N=%d',N(3)),'Location','northwest')
```

Listing 1: Main function for solving BVP of ODEs.

4.2   PLRR

This is the exact method we used for solving this Boundary Value Problems (BVP) of ODEs.

```matlab
% Piecewise Linear Rayleigh-Ritz method
% ================================================================
% Solve a linear two-point boundary-value problem
%                  -(p(x)u'(x))'+q(x)u(x)=f(x),     a<=x<=b,
% with initial conditions
%                            u(a)=0, u'(b)=0.
% ================================================================
```

```matlab
8  % See the theory part at <reference.mlx>
9  function uc=PLRR(f,p,q,a,b,N)
10 % INPUT:
11 %   f: f(x)
12 %   p: p(x); optional, by default p(x)=1
13 %   q: q(x); optional, by default q(x)=0
14 %   a, b: interval [a,b]; optional, by default [a,b]=[0,1]
15 %   N: # of evenly spaced intervals; optional, by default N=10
16 % OUTPUT:
17 %   uc: coef vector of approx solution
18
19 % =============================================================
20 % *****  NOTE that f, p, q must be element-wise functions  *****
21 % e.g. f(x)=@(x)x.^2+1 (NOT x^2+1)
22 % =============================================================
23
24 % set default args
25 if nargin<6
26     N=10;
27     if nargin<5
28         b=1;
29         if nargin<4
30             a=0;
31             if nargin<3
32                 q=@(x)x.*0;
33                 if nargin<2
34                     p=@(x)x.^0;
35                     if nargin<1
36                         error('Error! Function f is not set.')
37                     end
38                 end
39             end
40         end
41     end
42 end
43 if b<a
44     tmp=a; a=b; b=tmp;
45 end
46
47 h=(b-a)/N;  % step length. no check for N, hhh
48 A=zeros(N); % coef matrix, symmetric tridiagonal
49 c=zeros(N,1); % right-hand side (RHS) vector
50 xx=linspace(a,b,N+1);
51
52 % determine tridiagonal elements of A, and right-side vector c
53 for i=1:N-1
54     % Or use MATLAB func <integral> instead of <Gaussquad>.
55     % --> <Ctrl + F> --> Replace All. Then go to PLRR_test
56     % script file & run it again, it shall get same graph.
57     A(i,i)=Gaussquad(p,xx(i),xx(i+2))+...
58         Gaussquad(@(x)(x-xx(i)).^2.*q(x),xx(i),xx(i+1))+...
59         Gaussquad(@(x)(xx(i+2)-x).^2.*q(x),xx(i),xx(i+1));
60     A(i,i+1)=-Gaussquad(p,xx(i+1),xx(i+2))+...
61         Gaussquad(@(x)(x-xx(i+1)).*(xx(i+2)-x).*q(x),xx(i+1),xx(i+2));
62
63     c(i)=Gaussquad(@(x)(x-xx(i)).*f(x),xx(i),xx(i+1))+...
64         Gaussquad(@(x)(xx(i+2)-x).*f(x),xx(i+1),xx(i+2));
65 end
66 A(N,N)=Gaussquad(p,xx(N),xx(N+1))+...
67         Gaussquad(@(x)(x-xx(N)).^2.*q(x),xx(N),xx(N+1));
68 c(N)=Gaussquad(@(x)(x-xx(N)).*f(x),xx(N),xx(N+1));
69
70 A=A+diag(diag(A,1),-1); % let A_{i,i-1}=A_{i-1,i}
71 A=A/h^2;
72 c=c/h;
73
74 % uc=A\c; % coef vector of approx solution
```

```matlab
75 % or use user-defined function <GauEli>
76 uc=GauEli(A,c);
77
78 end
```

<div align="center">Listing 2: Piecewise Linear Rayleigh-Ritz method</div>

4.3   PLRR_intpol

This is the function used for interpolation after obtaining those coefficients.

```matlab
1  % % Interpolation for PLRR
2  % function u=PLRR_intpol(uc,xq,a,b)
3  % N=length(uc);    % No. of coefs of approx solution
4  % m=length(xq);    % No. of query points
5  % u=zeros(m,1);    % approximations at these n query points
6  % xx=linspace(a,b,N+1);
7  % h=(b-a)/N;
8  % for i=1:N
9  %     for j=1:m
10 %         if xq(j)>=xx(i) && xq(j)<=xx(i+1)
11 %             if i==1
12 %                 u(j)=uc(1)*(xq(j)-xx(1))/h;
13 %                 continue
14 %             end
15 %             u(j)=uc(i-1)*(-(xq(j)-xx(i+1))/h)+uc(i)*(xq(j)-xx(i))/h;
16 %         end
17 %     end
18 % end
19 % end
20
21 % Interpolation for PLRR only at those N+1 evenly spaced points
22 function u=PLRR_intpol(uc,a,b)
23 N=length(uc);       % No. of coefs of approx solution
24 u=zeros(N+1,1);     % approximations at these N+1 evenly spaced points
25 xx=linspace(a,b,N+1);
26 h=(b-a)/N;
27 % u(0)=0;
28 for i=2:N
29     u(i)=uc(i-1)*(-(xx(i)-xx(i+1))/h);
30 end
31 u(N+1)=uc(i)*(xx(N+1)-xx(N))/h;
32
33 end
```

<div align="center">Listing 3: Interpolation for PLRR</div>

4.4   Gaussquad

This function is a usr-defined function that servers the purpose of numerical integration, which can be simply identically substituted by MATLAB build-in function *integral*.

```matlab
1  % Gaussian quadrature
2  function T = Gaussquad(f,a,b,n)
3  if nargin<4
4      n=8;
5  end
6  % Gaussian quarature points & weights on [-1,1]
7  if n == 2
8      w = [1,1];                          % weights
9      p = [-1/sqrt(3),1/sqrt(3)];    % points
10 elseif n == 4
11     w = [0.3478548451,0.3478548451,0.6521451549,0.6521451549];
12     p = [0.8611363116,-0.8611363116,0.3399810436,-0.3399810436];
13 elseif n == 8
```

```
14      w = [0.1012285363,0.1012285363,0.2223810345,0.2223810345,0.3137066459,...
15            0.3137066459,0.3626837834,0.3626837834];
16      p = [0.9602898565,-0.9602898565,0.7966664774,-0.7966664774,0.5255324099,...
17        -0.5255324099,0.1834346425,-0.1834346425];
18 else
19      error('n must be either 2 or 4 or 8')
20 end
21
22 % take a linear transform: x = t*(b-a)/2 + (a+b)/2, where t belongs to [-1,1]
23 % int(f,[a,b]) = (b-a)/2 * int(f(t*(b-a)/2 + (a+b)/2), [-1,1])
24 w = 0.5*(b-a)*w;
25 p = 0.5*(b-a)*p+0.5*(a+b);
26
27 T = w*f(p).';
28 end
```

Listing 4: Gaussian quadrature

4.5   GauEli

```
1 % Gaussian Elimination: solve linear systems
2 % of algebraic equations of the form Ax=b.
3 % 10170437 Mark Taylor
4
5 function [x,U] = GauEli(A, b)
6
7 [m,n]=size(A);
8 if m ~= n
9      error('A must be a square matrix!')
10 end
11
12 U=[A,b];          % Augemented matrix of A
13
14 for j=1:n
15      k=maxIndex(U(:,j),j,n);
16      if abs(U(k,j))>eps
17          if k~=j
18              temp=U(j,j:n+1);
19              U(j,j:n+1)=U(k,j:n+1);
20              U(k,j:n+1)=temp;
21          end
22      else
23          x='The system has infinitely many solutions!';
24          U=NaN;
25          return;
26      end
27
28      % Perform Gauss elimination.
29      for i=j+1:n
30          if abs(U(i,j))>eps
31              t=U(i,j)/U(j,j);
32              U(i,j:n+1)=U(i,j:n+1)-t*U(j,j:n+1);
33          end
34      end
35 end
36
37 x=zeros(n,1);
38 % Sovle upper diagonal linear system, i.e. U(1:n,1:n)x=U(:,n+1).
39 x(n)=U(n,n+1)/U(n,n);
40 for i=n-1:-1:1
41      x(i)=(U(i,n+1)-U(i,i+1:n)*x(i+1:n))/U(i,i);
42 end
43
44 end
```

Listing 5: Gaussian Elimination

5   Appendix

Hello from the Beatles.😃