NORTH CAROLINA STATE UNIVERSITY

Homework 1 for Chapter 2

Jinwen Zhao, Chang Liu, Ling Zhang

February 24, 2016

PROBLEM 1

PROBLEM 2 PAGE 119 PART A Let $g(t)$ be the approximation to f,

$$\|f-g\|_\infty = \sup_{1\le i\le N} |f(t)-g(t)|$$
$$g(t) = c = \frac{y+1}{2}$$

Therefore, the L_∞ approximation for $f(x)$ is $g(t) = \frac{y+1}{2}$, with error $\|f-g\|_\infty = \frac{y-1}{2}$.

PROBLEM 2 PAGE 119 PART B Let $g(t)$ be the approximation to f,

$$\|f-g\|_2 = (\sum_{i=1}^{N} |f(t_i)-g(t_i)|^2)^{\frac{1}{2}}$$
$$= \sqrt{(N-1)(c-1)^2+(c-y)^2} = \sqrt{Nc^2-2(N-1+y)c+(N-1)+y^2}$$

Solve this the function $h(c) = Nc^2-2(N-1+y)c+(N-1)+y^2$ for the minimum value and we get $c = \frac{N-1+y}{N}$.

Therefore, the L_2 approximation for $f(x)$ is $g(t) = \frac{N-1+y}{N}$, with error $\|f-g\|_2 = y^2(1-\frac{1}{N}) - y(\frac{2N-2}{N}) + \frac{N-1}{N}$

PROBLEM 2 PAGE 119 PART C As $N \to \infty$, the constant in the least square approximation goes to 1. It shows the least square approximation weights less on the outliers than the infinity approximation. Request more input.

PROBLEM 5 PAGE 119 PART A Define the following $\hat{f}(x) = 1 + cx$ and $f(x) = e^x$.

$$\begin{aligned}
\|\hat{f} - f\|_2^2 = &\int_0^1 |e^x - 1 - cx|^2 dx = \int_0^1 e^{2x} - 2e^x(1+cx) + (1-cx)^2 dx \\
= &\int_0^1 e^{2x} - (2e^x + 2ce^x x) + (1 - 2cx + c^2x^2) dx \\
= &\frac{1}{3}c^2 - c + \frac{e^2}{2} - 2e + \frac{5}{2}
\end{aligned}$$

Minimize the function $h(c) = \frac{1}{3}c^2 - c + \frac{e^2}{2} - 2e + \frac{5}{2}$ and the minimum reaches at $c = \frac{3}{2}$.

PROBLEM 5 PAGE 119 PART B Solve for the general case, $\max\limits_{0 \le x \le 1} |e^x - (1+cx)|$. Let $f_c(x) = e^x - (1+cx)$ and,

$$f_c'(x) = e^x - c$$

$c = 1$. $f_1(x)$ is monotonic increasing in the interval $[0,1]$ and the minimum is $f_1(0) = 0$. Then $|e_1(x)| = f_1(x)$. The maximum is at $x = 1$ and the max error is $e - 2$. For the case $c = \frac{3}{2}$, $f_{\frac{3}{2}}(x)$ is decreasing at $[0, ln\frac{3}{2}]$ and increasing at $[ln\frac{3}{2}, 1]$. The minimum is $f_{\frac{3}{2}}(ln\frac{3}{2}) < 0$. Therefore $\max e_2(x) = |f_{\frac{3}{2}}(x)|$ is either reached at the endpoint, $\{0,1\}$, or at $x = \ln\frac{3}{2}$. Plug in and the maximum of $e_2(x)$ at the interval $[0,1]$ is $e - 2.5$ at the endpoint $x = 1$.

PROBLEM 5 PAGE 119 PART C Define the following minimization problem,

$$\min_{0 \le x \le 1} \|e^x - (1 + c_1 x + c_2 x^2)\|_2^2$$

$$\begin{aligned}
g(c_1, c_2) = &\int_0^1 (e^x - (1 + c_1 x + c_2 x^2))^2 dx \\
= &\frac{1}{3}c_1^2 + \frac{1}{2}(c_2 - 2)c_1 + \frac{1}{5}c_2^2 + \left(\frac{14}{3} - 2e\right)c_2 + \frac{1}{2}\left(5 - 4e + e^2\right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial g}{\partial c_1} &= \frac{1}{2}(c_2 - 2) + \frac{2}{3}c_1 = 0 \\
\frac{\partial g}{\partial c_2} &= \frac{1}{2}c_1 + \frac{2}{5}c_2 + \frac{14}{3} - 2e = 0
\end{aligned}$$

Solve this system of equations and get $c_1 = 164 - 60e = 0.9031$ and $c_2 = 80e - \frac{650}{3} = 0.7959$.

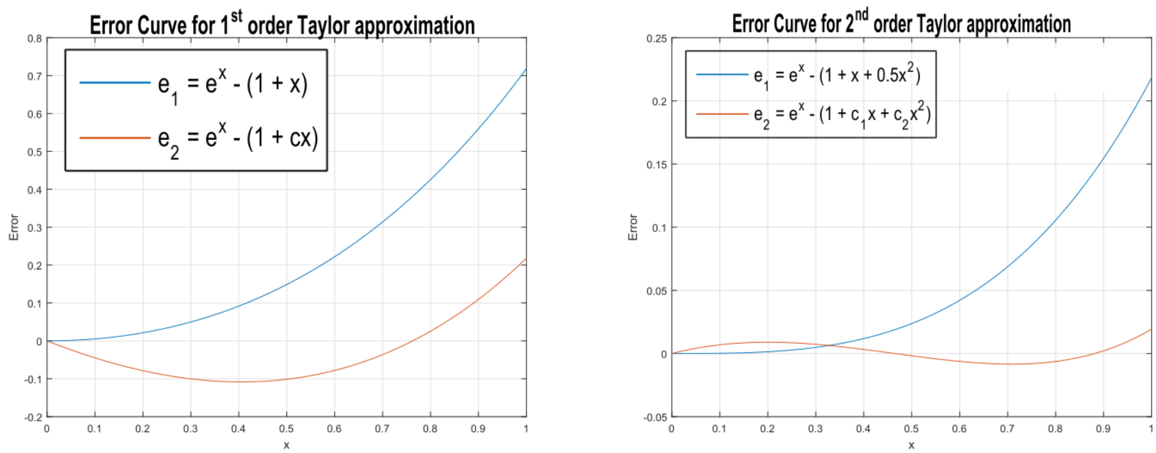

Figure 0.1: Error curves of $e_1(x)$ and $e_2(x)$

PROBLEM 33 PAGE 126 We use two different method to solve this problem.

1. Lagrange Interpolation

$$
\begin{aligned}
i = 0,\ l_0(x) &= \frac{x - x_1}{x_0 - x_1}\frac{x - x_2}{x_0 - x_2} = \frac{(x-11)(x-12)}{2} \\
i = 1,\ l_0(x) &= \frac{x - x_0}{x_1 - x_0}\frac{x - x_2}{x_1 - x_2} = -(x-10)(x-12) \\
i = 2,\ l_0(x) &= \frac{x - x_0}{x_2 - x_0}\frac{x - x_1}{x_2 - x_1} = \frac{(x-10)(x-11)}{2} \\
p(x) &= \ln(10) * l_0(x) + \ln(11) * l_1(x) + \ln(12) * l_2(x) \\
p(11.1) &= 2.406969856623995
\end{aligned}
$$

The relative error is 1.028×10^{-5}

2. Newton's Interpolation
To solve the problem in Newton's form, first let's creating the divided difference table using the given points.

x_i	$f(x)$	$f[x_i, x_{i-1}]$	$f[x_i, x_{i-1}, x_{i-2}]$
10	$\ln(10)$		
11	$\ln(11)$	$\ln(11/10)$	
12	$\ln(12)$	$\ln(12/11)$	$0.5\ln(121/120)$

Then, the interpolation polynomial can be written as

$$p(x) = \ln(10) + (x-10)\ln(11/10) + 0.5(x-10)(x-11)\ln(121/120).$$

therefore, $p(11.1) = 2.407882724933612$, the relative error for the interpolation is

$$\epsilon_{rel} = \frac{|p(11) - \ln(11.1)|}{\ln(11.1)} = 3.895 \times 10^{-4}$$

PROBLEM 36 PAGE 126 PART A We know that,

$$
\begin{aligned}
|E(x)| =& |e^x - p_n(f;x)| = |\frac{e^{\xi(x)}}{(n+1)!}\prod_{i=0}^{n}(x-x_i)| \\
=& \frac{e^{\xi(x)}}{(n+1)!}\prod_{i=0}^{n}|(x-\frac{i}{n})| \\
=& \frac{e^{\xi(x)}}{(n+1)!}\prod_{i=0}^{n}\sqrt{|(x-\frac{i}{n})(x-\frac{n-i}{n})|}
\end{aligned}
$$

We then show the hint is true by a simple maximization problem, $\forall i = 0 \cdots n$

$$\max_{0\le x\le 1}|(x-\frac{i}{n})(x-\frac{n-i}{n})|$$

Define $f(x) = (x-\frac{i}{n})(x-\frac{n-i}{n})$ and the minimum point is achieved at $x = \frac{1}{2}$. That is, $|f(x)|$ achieves maximum of $max = |(\frac{1}{2}-\frac{i}{n})(\frac{1}{2}-\frac{n-i}{n})| \le |(\frac{1}{2}-0)(\frac{1}{2}-1)| = \frac{1}{4}$ at $x = \frac{1}{2}$.

$$\max_{0\le x\le 1}|E(x)| \le \frac{e^1}{(n+1)!}\frac{1}{2^n}$$

Solve the following inequality,

$$\frac{e^1}{(n+1)!}\frac{1}{2^n} \le 10^{-6}$$

The smallest n is 7.

PROBLEM 36 PAGE 126 PART B For Taylor polynomial, the error can be written as:

$$|E(x)| = |e^x - p_n(f;x)| = |\frac{e^{\xi(x)}}{(n+1)!}x^{n+1}|$$

On the domain $[0,1]$, $\max_{0\le x\le 1}(|E(x)|) = \frac{e}{(n+1)!}$. So that only when $n \ge 10$, $\max_{0\le x\le 1}(|E(x)|) \le 1\times 10^{-6}$
According to this calculation result, we can see interpolation is a better approximation method than Taylor polynomial for the given problem. Thanks to the separation of the gird point, the interpolation approximation converge to the true function faster than Taylor series with a factor of 2^{-n}

PROBLEM 46 PAGE 128 We know that,

$$T_n(\cos\theta) = \cos(n\theta)$$

Let's $x = \cos\theta$, then we have

$$\frac{dT_n(x)}{d\theta} = \frac{dT_n(x)}{dx}\frac{dx}{d\theta}.$$

So that,

$$\begin{aligned}\frac{dT_n(x)}{dx} &= \frac{dT_n(x)}{d\theta}\Big/\frac{dx}{d\theta}\\ &= n\frac{\sin n\theta}{\sin\theta}\\ &= n\frac{\sin\left(n\cos^{-1}x\right)}{\sqrt{1-x^2}}\end{aligned}$$

So that

$$\begin{aligned}\frac{dT_n(0)}{dx} &= n\sin\left(\frac{n\pi}{2}\right)\\ &= \begin{cases}0 & \text{n is even,}\\ (-1)^{(n-1)/2} & \text{n is odd.}\end{cases}\end{aligned}$$

Programming Assignment

Problem 9 Page 137

The coding are listed in the Appendix. As the question asked for a natural spline, M relationship were used, so that the boundary condition $f^{(2)}(a) = f^{(2)}(b) = 0$ can be set as $M_0 = M_n = 0$.
The spline get from our script and the MATLAB's default spline function's results are shown in figure 0.2. As shown in the figure, for $f(x) = e^x$, $f(x) = \cos(2\pi x)$, and $f(x) = \sqrt{x}$, both our script and MATLAB's default routine did a good job. Due to the different method for treating the boundary points, the edge of the two method for $f(x) = \cos(2\pi x)$ and $f(x) = \sqrt{x}$ are slightly different.
Splines for $f(x) = \cos(20\pi x)$, on the other hand, is a very poor approximation for the original function. This is due to the selection of knots are accidentally having the same value. Increasing the number or knots will help to solve this issue.(see figure 0.3)

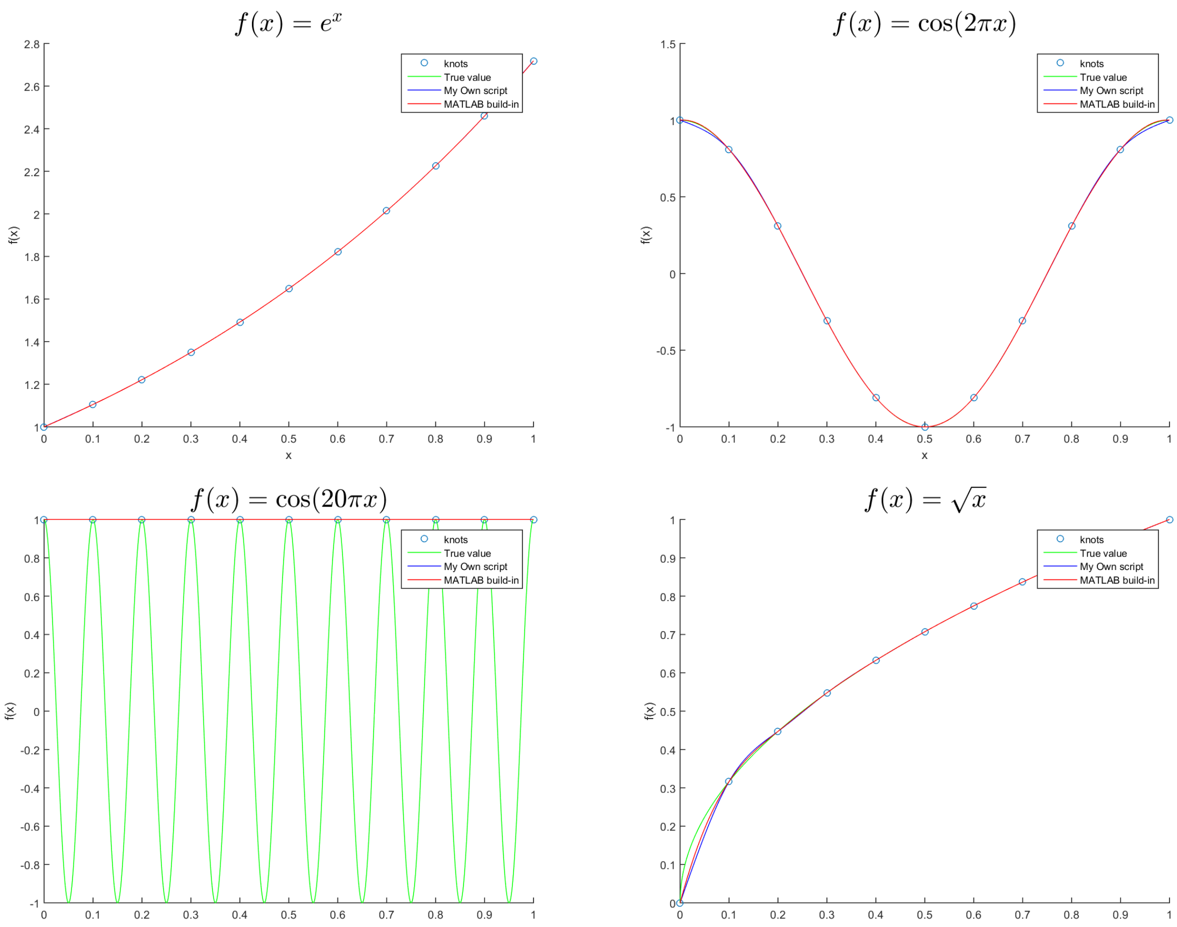

Figure 0.2: Natural spline(blue), MATLAB's default spline(red) and the true value(red) of given functions in domain $[0, 1]$. The circles are the knots.

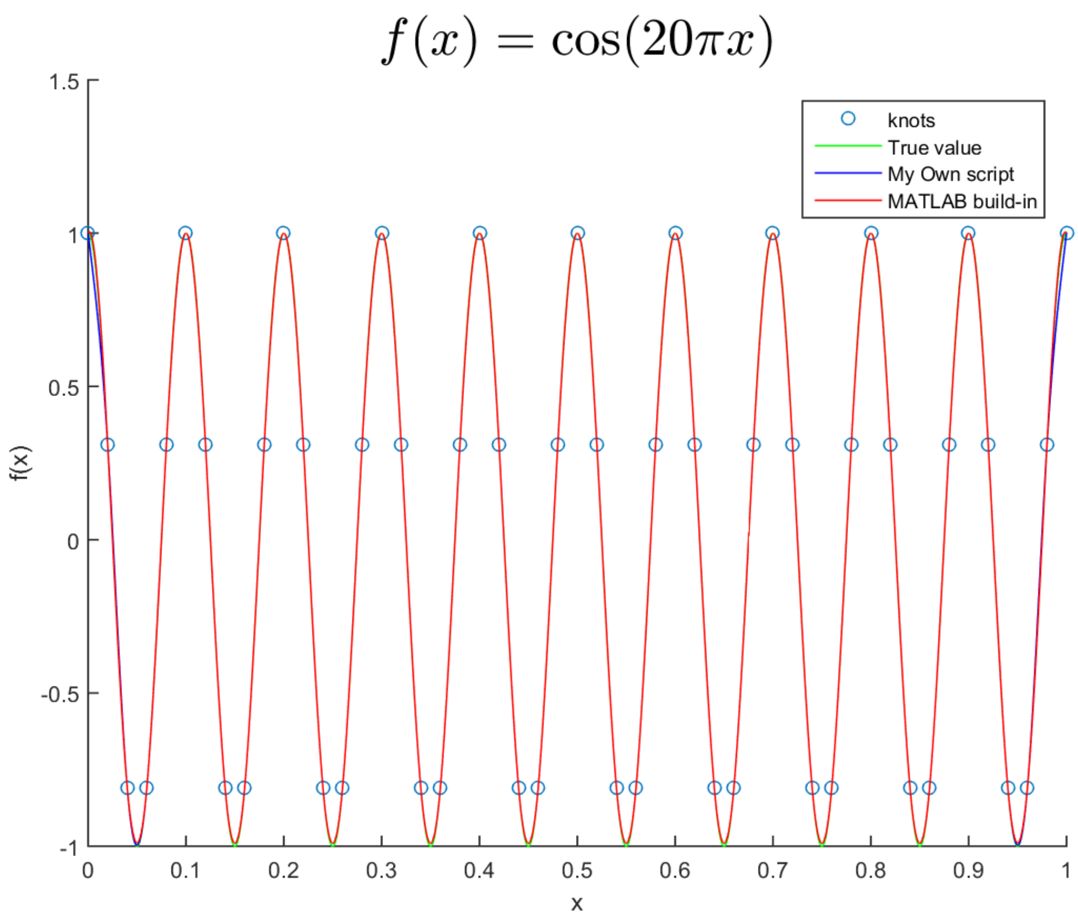

Figure 0.3: Increasing the knots by 5 times for $f(x) = \cos(20\pi x)$

APPENDIX

APPENDIX A natural_spline.m

```
function sp = natural_spline(knote,y,xx)
%%% Useage sp = natural_spline(x,y,xx)
%%% program to calculate the natural spline of a function
%%% using the sparse matrix operations of MATLAB
%%% use the same structure as MATLAB's spline function return (pp form)
%%% but also including the result for yy and xx

%% set some initial structure
sp = struct();
sp.breaks = knote;
if exist('xx','var')
    sp.xx = xx;
end
sp.pieces = length(knote) - 1;
sp.order = 4;
sp.dim = 1;

%% build the matrix to solve M
% see http://www.bb.ustc.edu.cn/jpkc/xiaoji/szjsff/jsffkj/chapt1_6_1.htm
h = knote(2:end) - knote(1:end-1);
lambda = h(2:end) ./ (h(1:end-1) + h(2:end));
mu = 1 - lambda;
d1 = h(1:end-1) + h(2:end) ;
d2 = (y(3:end) - y(2:end-1))./h(2:end) ;
d2 = d2 - (y(2:end-1) - y(1:end-2))./h(1:end-1);
d = 6 ./ d1 .* d2;
% matrix to solve
A = eye(length(lambda));
A = 2 * A + diag(lambda(1:end-1),1) + diag(mu(2:end),-1);
% as is natural spline, M0, Mn = 0
M0 = 0;
Mn = 0;
b = d;
b(1) = d(1) - mu(1)*M0;
b(end) = b(end) - mu(end)*Mn;

%% solve for M
b = b';
M = A\b;
M = [0; M; 0];


%% calculate the coeficient
coef = ones(sp.pieces,4);
for i = 1:sp.pieces
    coef(i,1) = M(i)/(6*h(i)); % for (x(i+1)-x)^3
    coef(i,2) = M(i+1)/(6*h(i)); % for (x-x(i))^3
    coef(i,3) = y(i) / h(i) - h(i) * M(i)/6; % for (x(i+1)-x)
    coef(i,4) = y(i+1) / h(i) - h(i) * M(i+1)/6; % for (x-x(i))
```

```
end
sp.coefs = coef;

%% calculate yy
if exist('xx','var')
    yy = [];
    %knote
    for xi= xx
        yy = [yy envIntp(coef,knote,xi)];
    end
    sp.yy = yy;
end

end


function y = envIntp(coef,x,xi)
%%% calculate the value of points from spline
for i=1:length(x)-1
    if (ge(xi,x(i)) && le(xi,x(i+1)))
        y = coef(i,1)*(x(i+1)-xi)^3 + coef(i,2)*(xi-x(i))^3 + ...
            coef(i,3)*(x(i+1)-xi) + coef(i,4)*(xi-x(i));
        break;
    end
end

end
```

APPENDIX B S_nat.m

```matlab
function [yy,xx,errmax] = S_nat(f,x,N)
%%% Useage yy = S_nat(f,x,N)
%%% return the yy value on selected gird points xij = xi+(j-1)/(N-1)∆_x

    %% gen y and xx list
    y = f(x(1));
    xx = [];
    xx_span = (0:(N-1)) ./ (N-1);
    for i=2:length(x)
        y = [y f(x(i))];
        xx = [xx x(i-1)+xx_span.*(x(i)-x(i-1))];
    end

    %% gen yy_true list
    yy_true = xx;
    for i=1:length(xx)
        yy_true(i) = f(xx(i));
    end

    %% do the spline
    sp = natural_spline(x,y,xx);
    yy = sp.yy;

    %% calculate error
    errmax = [];
    for i = 1:length(x)-1
        currErr = max(abs(yy((N-1)*(i-1)+1:(N-1)*i) - ...
            yy_true((N-1)*(i-1)+1:(N-1)*i)));
        errmax = [errmax currErr];

    end
end
```

APPENDIX C test_func.m

```
%%% top script to solve homework's problem
diary('result.txt')
diary on
% functions to test
listFunc = {@(x)exp(x), ...
    @(x)cos(2.*pi*x), ...
    @(x)cos(20.*pi*x), ...
    @(x)sqrt(x)};
nameFunc = {'$f(x) = e^x$', ...
    '$f(x) = \cos(2 \pi x)$', ...
    '$f(x) = \cos(20 \pi x)$', ...
    '$f(x) = \sqrt{x}$'};

x = 0:0.1:1;
N = 100;
for i = 1:length(listFunc)
    f = listFunc{i};
    y_true = f(x);
    [yy,xx,errmax] = S_nat(f,x,N);
    yy_buildin = spline(x,y_true,xx);
    yy_true = f(xx);
    % print the error pre block
    fprintf('\nFunction: %s\n',nameFunc{i});
    fprintf('ERR for each block\n');
    fprintf('block \t\t errmax\n');
    for j = 1:length(errmax)
        fprintf('%.2f,%.2f\t%6E\n',x(j),x(j+1),errmax(j));
    end

    % plot out
    fig = figure;
    hold on
    plot(x,y_true,'o')
    plot(xx,yy_true,'g')
    plot(xx,yy,'b')
    plot(xx,yy_buildin,'r')
    legend('knots','True value','My Own script','MATLAB build-in')
    title(nameFunc{i},'Interpreter','latex','fontsize',24)
    xlabel('x')
    ylabel('f(x)')
    fname = sprintf('func_%d',i);
    savefig(fname);
    print(fig,fname,'-depsc','-tiff');
    print(fig,fname,'-dpng');
    close(fig);
end

% increase the size of knots(5 times)
x = 0:0.02:1;
N = 100;
for i = 1:length(listFunc)
```

```
    f = listFunc{i};
    y_true = f(x);
    [yy,xx,errmax] = S_nat(f,x,N);
    yy_buildin = spline(x,y_true,xx);
    yy_true = f(xx);
    % print the error pre block
    fprintf('\nFunction: %s\n',nameFunc{i});
    fprintf('ERR for each block\n');
    fprintf('block \t\t errmax\n');
    for j = 1:length(errmax)
        fprintf('%.2f,%.2f\t%6E\n',x(j),x(j+1),errmax(j));
    end

    % plot out
    fig = figure;
    hold on
    plot(x,y_true,'o')
    plot(xx,yy_true,'g')
    plot(xx,yy,'b')
    plot(xx,yy_buildin,'r')
    legend('knots','True value','My Own script','MATLAB build-in')
    title(nameFunc{i},'Interpreter','latex','fontsize',24)
    xlabel('x')
    ylabel('f(x)')
    fname = sprintf('funcL_%d',i);
    savefig(fname);
    print(fig,fname,'-depsc','-tiff');
    print(fig,fname,'-dpng');
    close(fig);
end




diary off
```