Dominic Becker
Math 4446 Project 1
3/21/2024

1a.) We have the following piecewise polynomials and conditions given for the quadratic spline.

$$S(x) = \begin{cases} S_0(x) = a_0 + b_0(x - x_0) + c_0(x - x_0)^2 & [x_0, x_1] \\ S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 & [x_i, x_{i+1}] \\ S_{i+1}(x) = a_{i+1} + b_{i+1}(x - x_{i+1}) + c_{i+1}(x - x_{i+1})^2 & [x_{i+1}, x_{i+2}] \end{cases}$$

$$S'(x) = \begin{cases} S_0'(x) = b_0 + 2c_0(x - x_0) & [x_0, x_1] \\ S_i'(x) = b_i + 2c_i(x - x_i) & [x_i, x_{i+1}] \\ S_{i+1}'(x) = b_{i+1} + 2c_{i+1}(x - x_{i+1}) & [x_{i+1}, x_{i+2}] \end{cases}$$

Conditions:

1. $S_i(x_i) = f(x_i)$ 2. $S_i(x_{i+1}) = f(x_{i+1})$ 3. $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$ 4. $S_0'(x_0) = f'(x_0)$

First, we must find a_0, b_0, and c_0. This can be done with the following equations:

$$a_0 = S_0(x_0) = f(x_0)$$

$$b_0 = S_0'(x_0) = f'(x_0)$$

$$c_0 = \frac{S_0(x_1) - a_0 - b_0(x_1 - x_0)}{(x_1 - x_0)^2} = \frac{f(x_1) - a_0 - b_0(x_1 - x_0)}{(x_1 - x_0)^2}$$

Now we must find a_{i+1}, b_{i+1}, and c_{i+1} for $\mathrm{i} = 0,1,\ldots,\mathrm{n}-2$. We get the following:

$$a_{i+1} = a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 = f(x_{i+1})$$

$$b_{i+1} = b_i + 2c_i(x_{i+1} - x_i)$$

$$c_{i+1} = \frac{S_{i+1}(x_{i+2}) - a_{i+1} - b_{i+1}(x_{i+2} - x_{i+1})}{(x_{i+2} - x_{i+1})^2}$$

Using these equations, we can get a_i, b_i, and c_i for $\mathrm{i} = 0,1,\ldots,\mathrm{n}-1$.

Dominic Becker
Math 4446 Project 1
3/21/2024
1b.


```
function [pp] = quadspline(x,y,k)
%QUADSPLINE - computes the quadratic spline with given slope at the left
% endpoint.
% -Input:
% x - the vector of x_i values
% y - the vector of f(x_i)
% k - derivative f'(x) at left endpoint
% -Output :
% pp - A piecewise polynomial structure for the quadratic spline
%% initialize coefficient matrix
n = length(x)-1;
A = zeros(n,3);
%% Calculate a_0, b_0, and c_0
A(1,3) = y(1);
A(1,2) = k;
A(1,1) = (y(2)-A(1,3)-A(1,2)*(x(2)-x(1)))/(x(2)-x(1))^2;
%% Complete the coefficient matrix for a_i, b_i, and c_i
for i = 1:n-1
    A(i+1,3) = A(i,3)+A(i,2)*(x(i+1)-x(i))+A(i,1)*(x(i+1)-x(i))^2;
    A(i+1,2) = A(i,2)+2*A(i,1)*(x(i+1)-x(i));
    A(i+1,1) = (y(i+2)-A(i+1,3)-A(i+1,2)*(x(i+2)-x(i+1)))/(x(i+2)-x(i+1))^2;
end
%% Store spline in pp format
pp = mkpp(x,A);
```

Dominic Becker
Math 4446 Project 1
3/21/2024

2a.) The order of accuracy for this quadratic spline should be three. This is because the spline uses quadratic polynomials where the error is contained in the cubic term with h^3. α is consistent with this expectation.


```
function [S] = P2a(a)
%P2a — Creates a table of numerical error and order of accuracy for k
% piecewise polynomials, each with n_k polynomials, at h_k stepsize for
% f = xe^(-x) using quadratic spline.
% —Input:
% a - number of piecewise polynomials to compute
% —Output :
% S — A table of values for k, n_k, h_k, numerical error, and order of
% accuracy of each piecewise polynomial
%% set the function to interpolate
f = @(x) x*exp(-x);
%% initialize each column of the output table
n = zeros(a,1);
k = zeros(a,1);
h = zeros(a,1);
enum = zeros(a,1);
alpha = zeros(a,1);
%% loop for each entry in every vector previously initialized
for i = 1:a
%% calculate ith entry for k, n_k, and h_k
    k(i) = i;
    n(i) = 2^(i+1);
    h(i) = 2^(1-i);
%% calculate x and f(x) vectors to use in quadspline
    X = 0:h(i):n(i);
    Y = zeros(n(i)+1,1);
    for l = 0:h(i):n(i)
        Y(l/h(i)+1) = f(l);
    end
    pp = quadspline(X,Y,1);
%% calculate numerical error and order of accuracy for each quadspline
    for m = 0:0.001:4
        if abs(f(m)-ppval(pp,m)) > enum(i)
            enum(i) = abs(f(m)-ppval(pp,m));
        end
    end
    if i ~= 1
        alpha(i) = log(enum(i)/enum(i-1))/log(h(i)/h(i-1));
    end
end
alpha(1) = "NAN";
%% create a results table for k, n_k, h_k, numerical error, and order or accuracy
R = table(k,n,h,enum,alpha,'VariableNames',{'k','n','h','error','accuracy'});
S = table(R,'VariableNames',{'Results'});
```

Dominic Becker
Math 4446 Project 1
3/21/2024

```
a = 5;
P2a(a)
```

```
ans =

  5×1 table

                       Results
    k    n       h          error         accuracy
    ________________________________________________

    1     4        1       0.057408          NaN
    2     8      0.5      0.0084533       2.7637
    3    16     0.25      0.0011163       2.9208
    4    32    0.125     0.00014239       2.9708
    5    64   0.0625      1.795e-05       2.9878
```

Published with MATLAB® R2021b

2b.) The splines would be ranked as follows:

1. Clamped cubic spline – 4^{th} order of accuracy
    - Advantages: Smallest error
    - Disadvantages: Needs $f'(x)$ at both endpoints

2. Not-a-knot spline – 4^{th} order of accuracy
    - Advantages: 4^{th} order of accuracy, does not need endpoint conditions
    - Disadvantages: Third derivatives must be continuous

3. Quadratic spline – 3^{rd} order of accuracy
    - Advantages: Does not need to compute a d_i column
    - Disadvantages: Needs $f'(x)$ at one endpoint

4. Natural cubic spline – 2^{nd} order of accuracy
    - Advantages: Simple endpoint conditions
    - Disadvantages: Low order of accuracy

Natural cubic spline does not work well for this problem since the boundary conditions are set so that $f''(0) = f''(4) = 0$. Since $f''(x) = xe^{-x} - 2e^{-x}$, $f''(0) = -2 \neq 0$, and $f''(4) = 2e^{-4} \neq 0$, the boundary conditions of the natural cubic spline inhibit its performance interpolating $f(x)$.

Dominic Becker
Math 4446 Project 1
3/21/2024


```
function [T1,T2,T3,T4] = P2b(a)
%P2b — Creates a table of numerical error and order of accuracy for k
% piecewise polynomials, each with n_k polynomials, at h_k stepsize for
% f = xe^(-x) using quadratic spline and the three cubic splines.
% —Input:
% a - number of piecewise polynomials to compute
% —Output :
% T1 — A table of values for k, n_k, h_k, numerical error, and order of
% accuracy of each piecewise polynomial using a quadratic spline
% T2 — A table of values for k, n_k, h_k, numerical error, and order of
% accuracy of each piecewise polynomial using a natural cubic spline
% T3 — A table of values for k, n_k, h_k, numerical error, and order of
% accuracy of each piecewise polynomial using a clamped cubic spline
% T4 — A table of values for k, n_k, h_k, numerical error, and order of
% accuracy of each piecewise polynomial using a not-a-knot cubic spline
%% set the function to interpolate
f = @(x) x.*exp(-x);
%% setup x and f(x) vectors for quadratic, natural, and not-a-knot splines
x = [0,1,2,3,4];
y = [0,exp(-1),2*exp(-2),3*exp(-3),4*exp(-4)];
%% setup f(x) vector with f'(0) and f'(4) for clamped spline
y2 = [1,0,exp(-1),2*exp(-2),3*exp(-3),4*exp(-4),exp(-4)-4*exp(-4)];
%% calculate the four splines and store in pp format
pp1 = quadspline(x,y,1);
pp2 = myspline(x,y,1);
pp3 = myspline(x,y2,2);
pp4 = myspline(x,y,3);
%% graph the four splines along with the original function f, and the 5 nodes
hold on
xq = 0:0.001:4;
fplot(f,[0,4],'black')
plot(xq,ppval(pp1,xq),'red');
plot(xq,ppval(pp2,xq),'green');
plot(xq,ppval(pp3,xq),'blue');
plot(xq,ppval(pp4,xq),'cyan');
scatter(x,y,'filled','black');
legend('f','quad','natural','clamped','not-a-knot','nodes');
%% create an error and accuracy table for each of the four splines
T1 = P2a(a);
T2 = mysplinetable(a,1);
T3 = mysplinetable(a,2);
T4 = mysplinetable(a,3);
```

Dominic Becker
Math 4446 Project 1
3/21/2024


```
function [S] = mysplinetable(a,b)
%P2a — Creates a table of numerical error and order of accuracy for k
% piecewise polynomials, each with n_k polynomials, at h_k stepsize for
% f = xe^(-x) using either natural, clamped, or not-a-knot cubic spline.
% —Input:
% a - number of piecewise polynomials to compute
% —Output :
% S — A table of values for k, n_k, h_k, numerical error, and order of
% accuracy of each piecewise polynomial
%% set the function to interpolate
f = @(x) x*exp(-x);
%% initialize each column of the output table
n = zeros(a,1);
k = zeros(a,1);
h = zeros(a,1);
enum = zeros(a,1);
alpha = zeros(a,1);
%% loop for each entry in every vector previously initialized
for i = 1:a
%% calculate ith entry for k, n_k, and h_k
    k(i) = i;
    n(i) = 2^(i+1);
    h(i) = 2^(1-i);
%% calculate x and f(x) vectors to use in myspline
    X = 0:h(i):n(i);
    Y = zeros(n(i)+1,1);
    for l = 0:h(i):n(i)
        Y(l/h(i)+1) = f(l);
    end
%% calculate f'(0) and f'(4) for clamped spline
    if b == 2
        Y = zeros(n(i)+3,1);
        Y(1) = 1;
        Y(n(i)/h(i)+3) = exp(-4)-4*exp(-4);
        for l = 0:h(i):n(i)
            Y(l/h(i)+2) = f(l);
        end
    end
%% use myspline to create either a natural, clamped, or not-a-knot spline
    pp = myspline(X,Y,b);
%% calculate numerical error and order of accuracy for each quadspline
    for m = 0:0.001:4
        if abs(f(m)-ppval(pp,m)) > enum(i)
            enum(i) = abs(f(m)-ppval(pp,m));
        end
    end
    if i ~= 1
        alpha(i) = log(enum(i)/enum(i-1))/log(h(i)/h(i-1));
    end
```



```
end
alpha(1) = "NAN";
%% create a results table for k, n_k, h_k, numerical error, and order or accuracy
R = table(k,n,h,enum,alpha,'VariableNames',{'k','n','h','error','accuracy'});
S = table(R,'VariableNames',{'Results'});
```

Dominic Becker
Math 4446 Project 1
3/21/2024

```
a = 5;
[quadratic,natural,clamped,notaknot] = P2b(a)
```

```
quadratic =

  5×1 table

                    Results
    k    n       h          error        accuracy
    ______________________________________________

    1     4        1       0.057408          NaN
    2     8      0.5      0.0084533       2.7637
    3    16     0.25      0.0011163       2.9208
    4    32    0.125     0.00014239       2.9708
    5    64   0.0625      1.795e-05       2.9878


natural =

  5×1 table

                    Results
    k    n       h          error        accuracy
    ______________________________________________

    1     4        1       0.080493          NaN
    2     8      0.5       0.023188       1.7955
    3    16     0.25      0.0060438       1.9399
    4    32    0.125       0.001528       1.9838
    5    64   0.0625     0.00038312       1.9957


clamped =

  5×1 table

                    Results
    k    n       h          error        accuracy
    ______________________________________________

    1     4        1       0.007331          NaN
    2     8      0.5     0.00056371        3.701
    3    16     0.25     3.8241e-05       3.8818
    4    32    0.125     2.4723e-06       3.9512
    5    64   0.0625     1.5678e-07        3.979
```

Dominic Becker
Math 4446 Project 1
3/21/2024

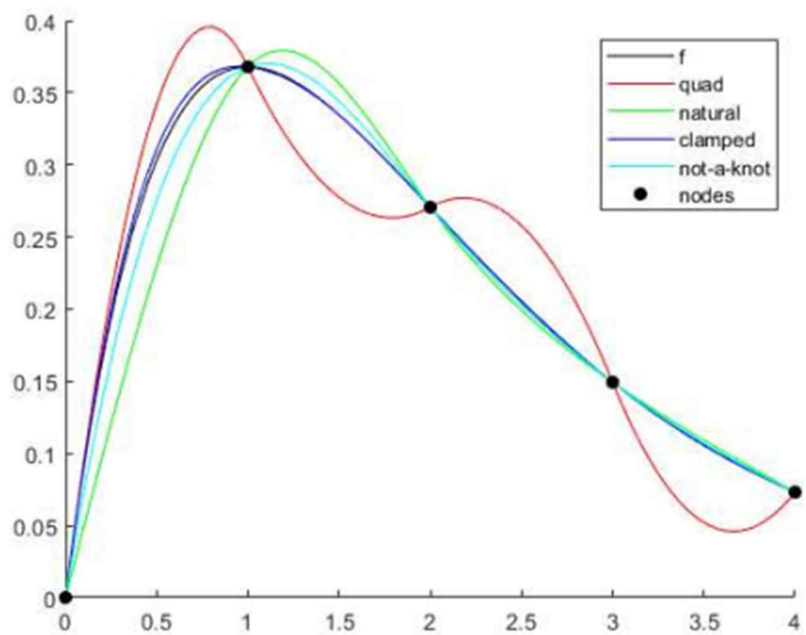
```
notaknot =

  5×1 table

                  Results
 k     n       h          error        accuracy
 ________________________________________________

 1     4         1       0.031593          NaN
```

```
 2     8       0.5       0.003714       3.0886
 3    16      0.25     0.00032003       3.5367
 4    32     0.125     2.3494e-05       3.7678
 5    64    0.0625     1.5913e-06        3.884
```



Published with MATLAB® R2021b

3a.) quadspline2 will maintain the order of accuracy of quadspline since $f'(x_0)$ was derived using the five-point forward difference formula. Since the order of accuracy for the five-point forward difference formula is greater than that of quadspline, quadspline will maintain its order of accuracy.

Dominic Becker
Math 4446 Project 1
3/21/2024
3b.)

3/21/24 12:32 AM C:\Users\becke\OneDriv...\quadspline2.m 1 of 1

```
function [pp] = quadspline2(x,y)
%QUADSPLINE2 — computes the quadratic spline with slope at the left
% endpoint approximated by finite difference.
% —Input:
% x — the vector of x_i values
% y — the vector of f(x_i)
% —Output :
% pp — A piecewise polynomial structure for the quadratic spline
%% initialize coefficient matrix
n = length(x)-1;
A = zeros(n,3);
%% Calculate a_0, b_0, and c_0 using finite difference approximation
A(1,3) = y(1);
A(1,2) = finitder(0.001);
A(1,1) = (y(2)-A(1,3)-A(1,2)*(x(2)-x(1)))/(x(2)-x(1))^2;
%% Complete the coefficient matrix for a_i, b_i, and c_i
for i = 1:n-1
    A(i+1,3) = A(i,3)+A(i,2)*(x(i+1)-x(i))+A(i,1)*(x(i+1)-x(i))^2;
    A(i+1,2) = A(i,2)+2*A(i,1)*(x(i+1)-x(i));
    A(i+1,1) = (y(i+2)-A(i+1,3)-A(i+1,2)*(x(i+2)-x(i+1)))/(x(i+2)-x(i+1))^2;
end
%% Store spline in pp format
pp = mkpp(x,A);
```

3/21/24 12:39 AM C:\Users\becke\OneDrive\D...\finitder.m 1 of 1

```
function [f0] = finitder(h)
%FINITDER — computes the slope at the left endpoint using five-point
% forward difference
% —Input:
% h — distance between x_i and x_{i+1}
% —Output :
% f0 — f'(0) when f = xe^(-x)
%% First five interpolation points of the function given in 2a
f = @(x) x.*exp(-x);
X0 = [0,h,2*h,3*h,4*h];
Y0 = f(X0);
%% approximate f'(0) with five-point forward difference formula
f0 = (-25*Y0(1)+48*Y0(2)-36*Y0(3)+16*Y0(4)-3*Y0(5))/(12*h);
```

Dominic Becker
Math 4446 Project 1
3/21/2024

3c.) Yes, the convergence order is the same.


```
function [S] = P3c(a)
%P2a — Creates a table of numerical error and order of accuracy for k
% piecewise polynomials, each with n_k polynomials, at h_k stepsize for
% f = xe^(-x) using quadratic spline with finite difference approximation.
% —Input:
% a - number of piecewise polynomials to compute
% —Output :
% S — A table of values for k, n_k, h_k, numerical error, and order of
% accuracy of each piecewise polynomial
%% set the function to interpolate
f = @(x) x*exp(-x);
%% initialize each column of the output table
n = zeros(a,1);
k = zeros(a,1);
h = zeros(a,1);
enum = zeros(a,1);
alpha = zeros(a,1);
%% loop for each entry in every vector previously initialized
for i = 1:a
%% calculate ith entry for k, n_k, and h_k
    k(i) = i;
    n(i) = 2^(i+1);
    h(i) = 2^(1-i);
%% calculate x and f(x) vectors to use in quadspline2
    X = 0:h(i):n(i);
    Y = zeros(n(i)+1,1);
    for l = 0:h(i):n(i)
        Y(l/h(i)+1) = f(l);
    end
    pp = quadspline2(X,Y);
%% calculate numerical error and order of accuracy for each quadspline
    for m = 0:0.001:4
        if abs(f(m)-ppval(pp,m)) > enum(i)
            enum(i) = abs(f(m)-ppval(pp,m));
        end
    end
    if i ~= 1
        alpha(i) = log(enum(i)/enum(i-1))/log(h(i)/h(i-1));
    end
end
alpha(1) = "NAN";
%% create a results table for k, n_k, h_k, numerical error, and order or accuracy
R = table(k,n,h,enum,alpha,'VariableNames',{'k','n','h','error','accuracy'});
S = table(R,'VariableNames',{'Results'});
```

Dominic Becker
Math 4446 Project 1
3/21/2024

```
a = 5;
P3c(a)
```

```
ans =

  5×1 table
```

Results

k	n	h	error	accuracy
1	4	1	0.057408	NaN
2	8	0.5	0.0084533	2.7637
3	16	0.25	0.0011163	2.9208
4	32	0.125	0.00014239	2.9708
5	64	0.0625	1.795e-05	2.9878

Published with MATLAB® R2021b

3d.) Aside from different finite difference approximations of $f'(x_0)$, I cannot think of any other ways to compute the quadratic spline without $f'(x_0)$. Since we need b_0, and b_0 comes from the derivative of x_0, we need the derivative of x_0 for the quadratic spline.