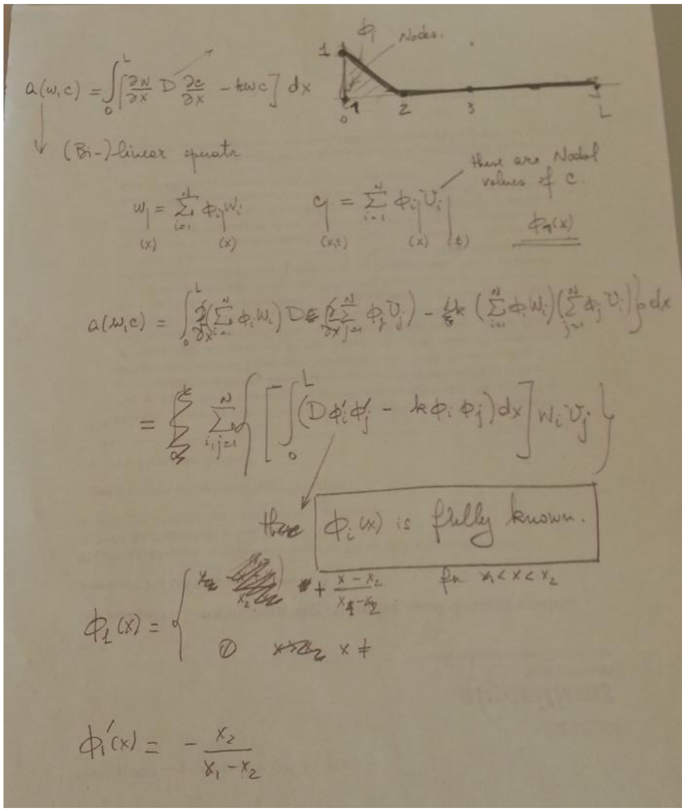
The following 3 images are work done with Dr. Costanzo on re-evaluating my discretization



$$a(w,c) = \int_0^L \left[\frac{\partial w}{\partial x} D \frac{\partial c}{\partial x} - kwc\right] dx$$

(Bi-)linear operator

$$w_{(x)} = \sum_{i=1}^{N} \phi_{i(x)} w_i \qquad c_{(x,t)} = \sum_{j=1}^{N} \phi_{j(x)} U_{j(t)}$$

these are Nodal values of c.

$\phi_1(x)$

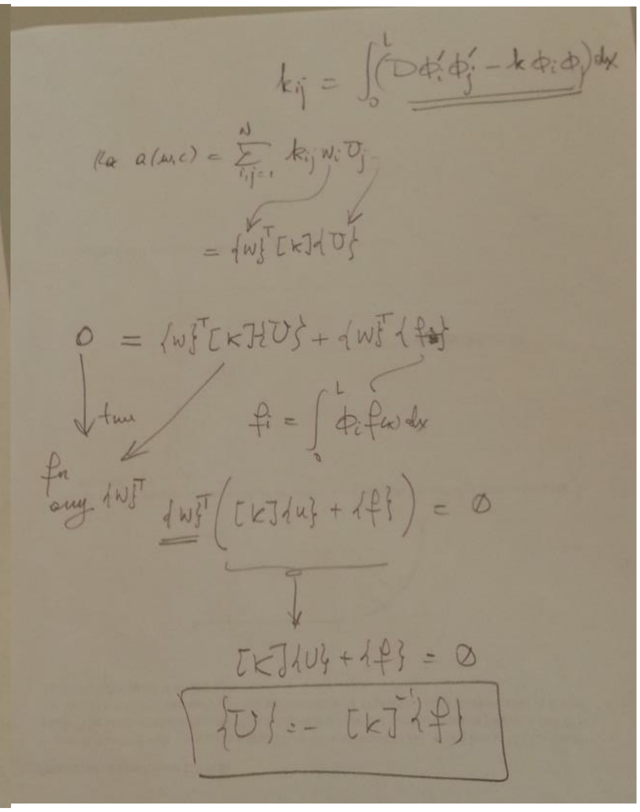
$$a(w,c) = \int_0^L \left\{ \frac{\partial}{\partial x}\left(\sum_{i=1}^N \phi_i w_i\right) D \frac{\partial}{\partial x}\left(\sum_{j=1}^N \phi_j U_j\right) - k \left(\sum_{i=1}^N \phi_i w_i\right)\left(\sum_{j=1}^N \phi_j U_j\right)\right\} dx$$

$$= \sum_{i,j=1}^{N} \left\{ \left[\int_0^L \left(D\phi_i'\phi_j' - k\phi_i\phi_j\right)dx\right] w_i U_j \right\}$$

$\boxed{\phi_i(x) \text{ is fully known.}}$

$$\phi_1(x) = \begin{cases} \dfrac{x - x_2}{x_1 - x_2} & \text{for } x_1 < x < x_2 \\ 0 & x > x_2 \end{cases}$$

$$\phi'(x) = -\frac{x_2}{x_1 - x_2}$$



$$k_{ij} = \int_0^L \left(D\phi_i'\phi_j' - k\phi_i\phi_j\right)dx$$

So $a(w,c) = \sum_{i,j=1}^{N} k_{ij} w_i U_j = \{w\}^T[K]\{U\}$

$$0 = \{w\}^T[K]\{U\} + \{w\}^T\{f\}$$

then

$$f_i = \int_0^L \phi_i f(x)\, dx$$

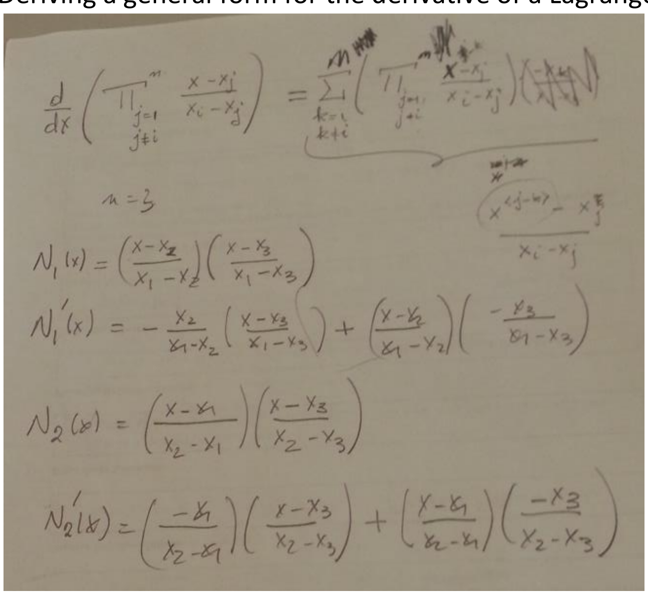
for any $\{w\}^T$

$$\{w\}^T\left([K]\{u\} + \{f\}\right) = 0$$

$$[K]\{U\} + \{f\} = 0$$

$$\boxed{\{U\} = -[K]^{-1}\{f\}}$$

Deriving a general form for the derivative of a Lagrange Polynomial



$$\frac{d}{dx}\left(\prod_{\substack{j=1\\ j\neq i}}^{m} \frac{x - x_j}{x_i - x_j}\right) = \sum_{\substack{k=1\\ k\neq i}}^{m} \left(\prod_{\substack{j=1\\ j\neq i}}^{m} \frac{x - x_j}{x_i - x_j}\right)$$

$n = 3$

$$N_1(x) = \left(\frac{x - x_2}{x_1 - x_2}\right)\left(\frac{x - x_3}{x_1 - x_3}\right)$$

$$N_1'(x) = -\frac{x_2}{x_1 - x_2}\left(\frac{x - x_3}{x_1 - x_3}\right) + \left(\frac{x - x_2}{x_1 - x_2}\right)\left(-\frac{x_3}{x_1 - x_3}\right)$$

$$N_2(x) = \left(\frac{x - x_1}{x_2 - x_1}\right)\left(\frac{x - x_3}{x_2 - x_3}\right)$$

$$N_2'(x) = \left(\frac{-x_1}{x_2 - x_1}\right)\left(\frac{x - x_3}{x_2 - x_3}\right) + \left(\frac{x - x_1}{x_2 - x_1}\right)\left(\frac{-x_3}{x_2 - x_3}\right)$$

Scan of my work on HW7 and the Generalized Lagrange Polynomial Derivative

$$\int_0^L w(x)\frac{\partial c}{\partial t}\,dx = -\int_0^L w\frac{\partial}{\partial x}\left(D(x)\frac{\partial c}{\partial x}\right)dx - \int_0^L wc\,dx + \int_0^L wf(x)\,dx$$

$$\Rightarrow -\int_0^L \frac{\partial w}{\partial x}\left(D(x)\frac{\partial c}{\partial x}\right)dx + w(x)D(x)\frac{\partial c(x)}{\partial x}\Big|_0^L$$

$$\Rightarrow w(x)D(x)\frac{\partial c}{\partial x}\Big|_0^L = -\cancel{w(0)}D(0)c'(0) + w(L)D(L)\cancel{c'(L)} = 0$$

$$\int_0^L w(x)\,\dot{c}\,dx = -\int_0^L w'\,D(x)\,c'\,dx + \int_0^L wc\,dx + \int_0^L wf(x)\,dx$$

$$\sum_j^n\sum_i^n \phi_i w_i \phi_j \dot{u}_j = \sum_{j=1}^n\sum_{i=1}^n \phi'_i w_i \phi'_j u_j D(x) - \sum_i\sum_j \phi_i w_i \phi_j u_j + \phi_i w_i \int_0^L f(x)$$

$$\phi_i\phi_j\dot{u}_j = \phi'_i\phi'_j u_j D(x) - \phi_j u_j + \int_0^L \phi_i f(x)$$

$$\int_0^L \phi_i f(x) = \int_0^L [\phi'_i\phi'_j D(x) - \phi_j\phi_i]u_j\,dx$$

$\phi_i =$

$$\sum\left(\frac{x-x_j}{x_i-x_j}\right)\left(\frac{-x_j}{x_i-x_j}\right)$$

$$\sum_{\substack{k=1\\k\neq i}}^n \left(\frac{x_k}{x_i-x_k}\right)\prod_{\substack{j=1\\j\neq i\\j\neq k}}^n\left(\frac{x-x_j}{x_i-x_j}\right)$$

$$\boxed{\sum_{\substack{k=1\\k\neq i}}^n\left(\frac{-x_k}{x_i-x_k}\right)\prod_{\substack{j=1\\j\neq k,i}}^n\left(\frac{x-x_j}{x_i-x_j}\right)} = \phi'_i$$

Matlab output for the FEM (I knew something was wrong but wasn't sure how to fix it.)

EDU>> Solver(pi/2,10,0)

```
K =

   1.0e+05 *

    0.0033   -0.0126         0         0         0         0         0         0         0         0
   -0.0126    0.0496   -0.1275         0         0         0         0         0         0         0
         0   -0.1275    0.3278   -0.5635         0         0         0         0         0         0
         0         0   -0.5635    0.9688   -1.1505         0         0         0         0         0
         0         0         0   -1.1505    1.3664   -1.1309         0         0         0         0
         0         0         0         0   -1.1309    0.9360   -0.5322         0         0         0
         0         0         0         0         0   -0.5322    0.3027   -0.1131         0         0
         0         0         0         0         0         0   -0.1131    0.0423   -0.0094         0
         0         0         0         0         0         0         0   -0.0094    0.0021   -0.0002
         0         0         0         0         0         0         0         0   -0.0002    0.0021


F =

    1.0155
   -2.8241
    6.9500
  -11.7448
   13.8190
  -11.3709
    6.4405
   -2.3995
    0.5306
    0.5306
```

U =

-0.0001 -0.0008 -0.0001 0.0000 0.0002 0.0001 -0.0000 -0.0009 -0.0012 0.0024

M =

0.4540 0.8090 0.9877 0.9511 0.7071 0.3090 -0.0016 -0.0059 -0.0089 -0.0100

As with HW6, I was unsure of how to get the theta method to work.

```
function [ ] = Solver( Length, Steps, x0 )
%SOLVER Summary of this function goes here
%   Detailed explanation goes here
dx = (Length-x0)/Steps;
X = x0+dx:dx:Length;

for i = 1:Steps
    M(i) = C_exact(x0+i*dx);
end

K = TriDiag(X)
F = F_vector(X)

U = (K\F)'
M

end
```

```
Error using Solver (line 4)
Not enough input arguments.
```

Published with MATLAB® 7.14

```
function [ C ] = C_exact( x )
%C_EXACT Summary of this function goes here
%   Detailed explanation goes here

if x > pi/3
    C = 0.01*sin(3*x);
else
    C = sin(3*x);
end

end
```

Published with MATLAB® 7.14

```
function [ F ] = F_vector( X )
%F_VECTOR Summary of this function goes here
%   Detailed explanation goes here
m = length(X);
F = zeros(m,1);

for i = 1:m-1
    F(i,1) = TwoPointRule(@f_x, i, i, X, X(i), X(i+1));
end

F(m,1) = TwoPointRule(@f_x, i, i, X, X(m-1), X(m));

end
```

Published with MATLAB® 7.14

```
function [ K ] = TriDiag( X )
%TRIDIAG Summary of this function goes here
%   Detailed explanation goes here
steps = length(X);
K = zeros(steps);

for i = 1:steps-1
    K(i,i+1) = TwoPointRule(@Coeff, i, i+1, X, X(i), X(i+1));
    K(i, i) = TwoPointRule(@Coeff, i, i, X, X(i), X(i+1));
    K(i+1,i) = TwoPointRule(@Coeff, i+1, i, X, X(i), X(i+1));
end

K(steps,steps) = K(steps-1, steps-1);
```

```
end
```

Published with MATLAB® 7.14

```
function [ Coeff ] = Coeff( x, X, i, j )
%COEFF Summary of this function goes here
%   Detailed explanation goes here
if x > pi/3
    E = 100;
else
    E = 1;
end

A = Phi_i_prime(x, X, i);
B = Phi_i_prime(x, X, j);
C = Phi_i(x, X, i);
D = Phi_i(x, X, j);
Coeff = A*B*E - C*D;
end
```

```
Error using Coeff (line 4)
Not enough input arguments.
```

Published with MATLAB® 7.14

```
function [ L ] = Phi_i( x, X_points, i )
%Phi_i is the lagrange polynomial evaluated at Xi

L = 1;
m = length(X_points);

for j = 1:m
    if j ~= i
        L = L*(x-X_points(j))/(X_points(i) - X_points(j));
    end
end

end
```

Published with MATLAB® 7.14

```
function [ Sum ] = Phi_i_prime( x, X_points, i )
%PHI_I_PRIME Summary of this function goes here
%   Detailed explanation goes here

L = 1;
Sum = 0;
m = length(X_points);

for k = 1:m
    if k ~= i
       for j = 1:m
           if j ~= k
               if j ~= i
                   L =L*(x-X_points(j))/(X_points(i)-X_points(j));
               end
           end
       end
       Sum = Sum + L*(-X_points(k))/(X_points(i)-X_points(k));
       L=1;
    end
end
```

Published with MATLAB® 7.14

```
function [ TwoPtVal ] = TwoPointRule( func, i, j, X, x1, x2 )
%TWOPOINTRULE calculates an integral value using a single application of
% the Two Point Gaussian Quadrature

%INPUT
% func -> Function handle for a function with one input argument
% x1 -> Initial value of x
% x2 -> Final value of x

% Initialize step size, weight, and the two evaluation points
h = (x2-x1)/2;
w = 1/sqrt(3);
xa = (h * -w) + h;
xb = (h*w) + h;

% Calculate Value of the function
TwoPtVal = h* (func(xa,X,i,j) + func(xb,X,i,j));

end
```

```
Error using TwoPointRule (line 11)
Not enough input arguments.
```

Published with MATLAB® 7.14

```
function [ F ] = f_x( x, X, i, j )
%F_X Summary of this function goes here
%   Detailed explanation goes here
if x > (pi/3)
    F = 9.01*sin(3*x);
else
    F = 10*sin(3*x);
end

F = F*Phi_i(x,X,i);

end
```

```
Error using f_x (line 4)
Not enough input arguments.
```

Published with MATLAB® 7.14