Assignment 4

Name: Manav Kothari
UIN: 133008008

Example 6.7.1

Inputs

The example input is for the Crank Nicolson Scheme, ie -

- alfa = 0.5
- DT = 0.05

The input example considers 8*8 Linear rectangular element.

Mesh Inputs

Parameter	Value
NX	8
NY	8
NPE (Linear rectangular element)	4
NDF	1

Domain Dimension

The computational domain is a square of side 1 unit.
X0 and Y0 are the coordinates of the 1st node of 1st element

Parameter	Value
x_length	1
y_length	1
X0	0
Y0	0

Time Simulation Parameter

Parameter	Value
DT	0.05
NTime	25

Differential Equation Parameter

ITEM = 1 (Parabolic Equation)
ITEM = 2 (Hyperbolic Equation)
Assumed equations for A11, A22, A00 are same -

A11 = A10 + A1xX + A1yY + A1uU + A1uxdUdX + A1uydUdY

Assumed equation for C and F are same -

C = C0 + CxX + CyY

Parameter	Value
ITEM	1
PDECOEFF.A11	[1 0 0 0 0 0]
PDECOEFF.A22	[1 0 0 0 0 0]
PDECOEFF.A00	[0 0 0 0 0 0]
PDECOEFF.C	[1 0 0]
PDECOEFF.F	[1 0 0]

Simulation Parameters

Considering Forward difference scheme

Parameter	Value
NONLIN	1
ITMAX	5
Epsilon	0.001
NLS	5
alfa	0.5

GAMA	0.5

GAMA is not used in this code, but it is mandatory to provide value of GAMA since it is passed in ELEMATRCS2D_Time function (where it is not used for parabolic case)

Essential Boundary Conditions

Parameter	Value
NSPV	17
ISPV	[9 1; 18 1; 27 1; 36 1; 45 1; 54 1; 63 1; 72 1;81 1; 80 1; 79 1; 78 1; 77 1; 76 1; 75 1; 74 1; 73 1]
VSPV	zeros(NSPV,1)

Natural Boundary Condition

Parameter	Value
NSSV	15
ISSV	[1 1; 2 1; 3 1; 4 1; 5 1; 6 1; 7 1; 8 1; 10 1; 19 1; 28 1; 37 1; 46 1; 55 1; 64 1]
VSSV	zeros(NSSV,1)

Additional Input

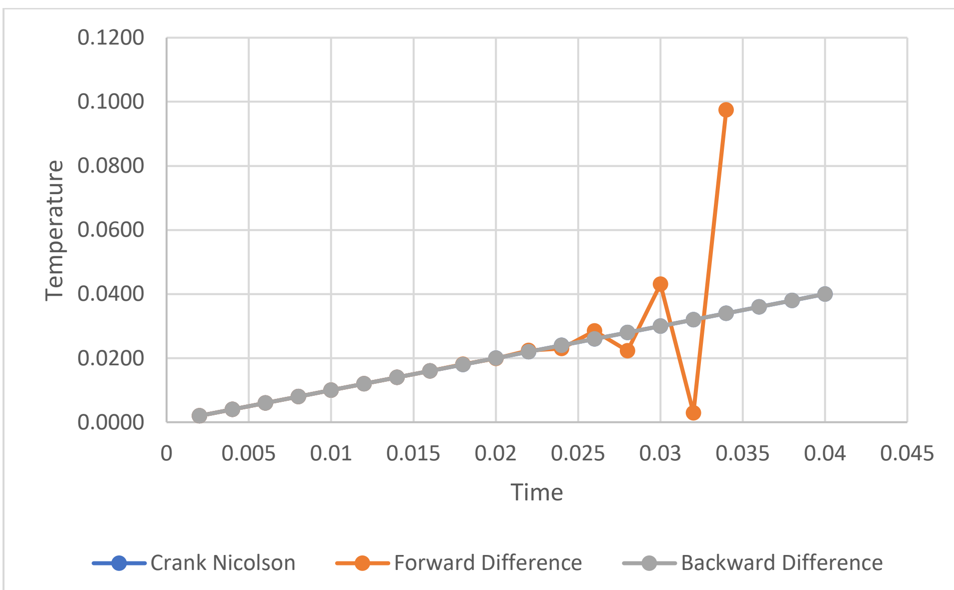
This is an optional input, it can be used if you just want to check the values of time steps that is a subset of all the time steps.
For example, if my DT = 0.001 but I want to check the values after every 0.05sec.

Input an array of time step that you want to check to Check_Time variable. Note - make Check_Time = 0 if you want to see values at every time step

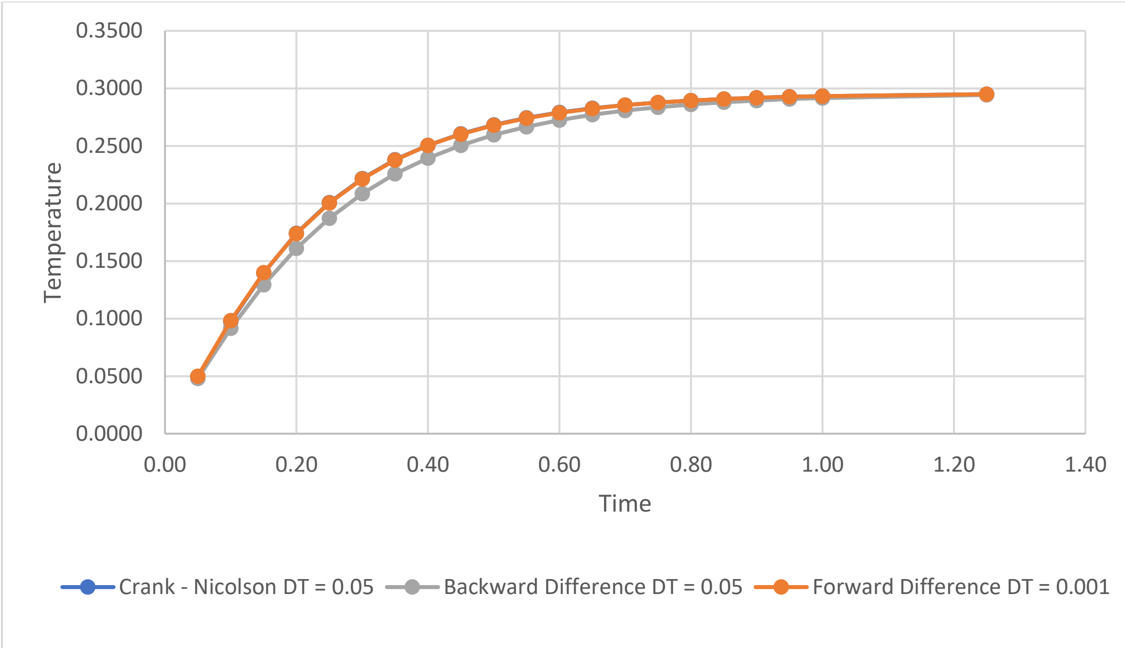
Parameter	Value
Check_Time	[0.05:0.05:1.00 1.25]

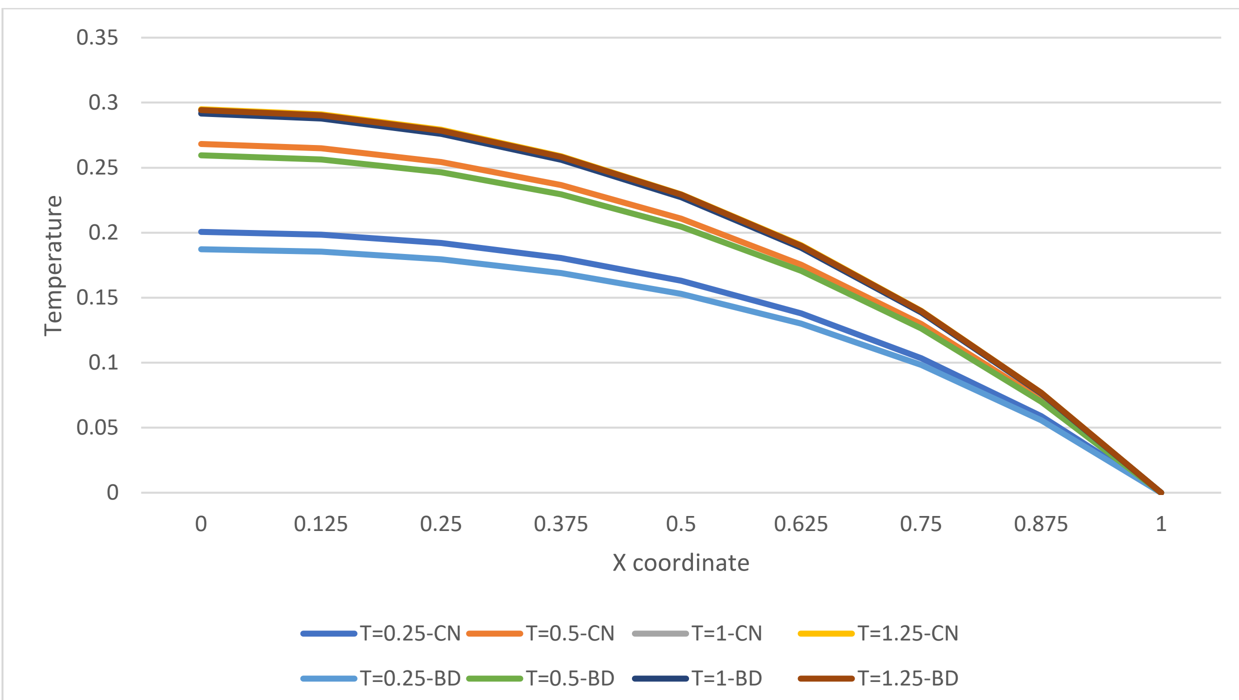
Result Table

Time (t)	8*8 Linear			4*4 Quadratic (9noded)		
	Crank - Nicolson DT = 0.05	Backward Difference DT = 0.05	Forward Difference DT = 0.001	Crank - Nicolson DT = 0.05	Backward Difference DT = 0.05	Forward Difference DT = 0.001
0.05	0.0497	0.0480	0.0500	0.0496	0.0479	0.0500
0.10	0.0975	0.0916	0.0983	0.0971	0.0913	0.0979
0.15	0.1398	0.1294	0.1400	0.1390	0.1288	0.1393
0.20	0.1740	0.1611	0.1737	0.1730	0.1604	0.1728
0.25	0.2006	0.1873	0.2004	0.1996	0.1864	0.1994
0.30	0.2215	0.2085	0.2213	0.2205	0.2075	0.2202
0.35	0.2379	0.2257	0.2376	0.2368	0.2247	0.2365
0.40	0.2506	0.2395	0.2503	0.2495	0.2385	0.2493
0.45	0.2605	0.2506	0.2603	0.2594	0.2496	0.2592
0.50	0.2682	0.2595	0.2680	0.2672	0.2585	0.2670
0.55	0.2743	0.2667	0.2741	0.2732	0.2656	0.2731
0.60	0.2790	0.2724	0.2788	0.2779	0.2714	0.2778
0.65	0.2826	0.2770	0.2825	0.2816	0.2760	0.2815
0.70	0.2855	0.2807	0.2854	0.2845	0.2797	0.2844
0.75	0.2877	0.2837	0.2876	0.2867	0.2827	0.2866
0.80	0.2894	0.2860	0.2894	0.2885	0.2850	0.2884
0.85	0.2908	0.2879	0.2907	0.2898	0.2869	0.2898
0.90	0.2919	0.2894	0.2918	0.2909	0.2885	0.2909
0.95	0.2927	0.2907	0.2926	0.2917	0.2897	0.2917
1.00	0.2933	0.2916	0.2933	0.2924	0.2907	0.2923
1.25	0.2949	0.2943	0.2949	0.2940	0.2934	0.2940



The blue line for Crank Nicolson is behind the grey line of Backward difference, that why it is difficult to see. This holds for almost all of the graphs





Example 6.7.3

Inputs

The example input is for the Constant-average acceleration method, ie -

- alfa = 0.5
- GAMA = 0.5
- DT = 0.1

For this question, I considered ONLY 8*8 Linear Rectangular Element

Mesh Inputs

Parameter	Value
NX	8
NY	8
NPE (Linear rectangular element)	4
NDF	1

Domain Dimension

The domain is a square of side 2 unit.
X0 and Y0 are the coordinates of the 1st node of 1st element

Parameter	Value
x_length	1
y_length	1
X0	0
Y0	0

Time Simulation Parameter

Parameter	Value
DT	0.1
NTime	32

Differential Equation Parameter

ITEM = 1 (Parabolic Equation)

ITEM = 2 (Hyperbolic Equation)
Assumed equations for A11, A22, A00 are same -

A11 = A10 + A1xX + A1yY + A1uU + A1uxdUdX + A1uydUdY

Assumed equation for C and F are same -

C = C0 + CxX + CyY

Parameter	Value
ITEM	2
PDECOEFF.A11	[1 0 0 0 0 0] (Linear) [1 0 0 0 0.2 0.2] (Non-Linear)
PDECOEFF.A22	[1 0 0 0 0 0] (Linear) [1 0 0 0 0.2 0.2] (Non-Linear)
PDECOEFF.A00	[0 0 0 0 0 0]
PDECOEFF.C	[1 0 0]
PDECOEFF.F	[1 0 0]

Simulation Parameters

NLS is not required for this assignment

Parameter	Value
NONLIN	1
ITMAX	5
Epsilon	0.001
NLS	5
alfa	0.5
GAMA	0.5

Essential Boundary Conditions

Parameter	Value
NSPV	17

ISPV	[9 1; 18 1; 27 1; 36 1; 45 1; 54 1; 63 1; 72 1;81 1; 80 1; 79 1; 78 1; 77 1; 76 1; 75 1; 74 1; 73 1]
VSPV	zeros(NSPV,1)

Natural Boundary Condition

Parameter	Value
NSSV	15
ISSV	[1 1; 2 1; 3 1; 4 1; 5 1; 6 1; 7 1; 8 1; 10 1; 19 1; 28 1; 37 1; 46 1; 55 1; 64 1]
VSSV	zeros(NSSV,1)

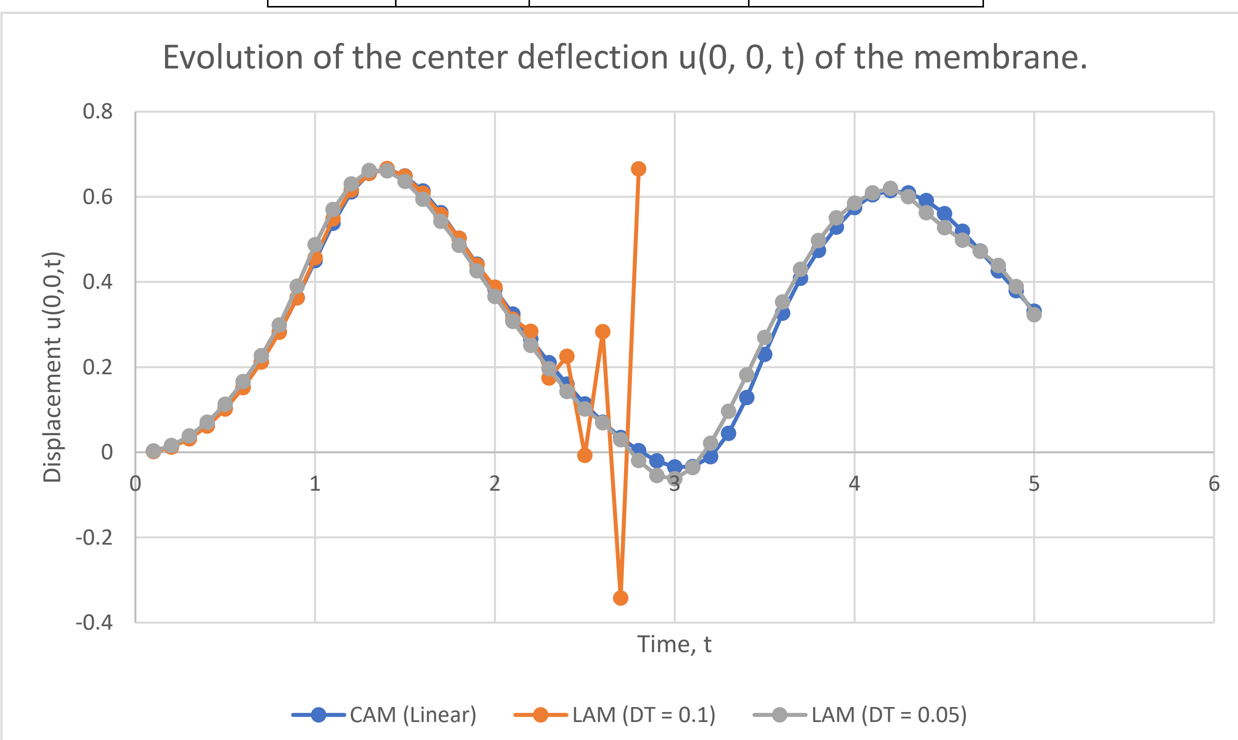
Additional Input

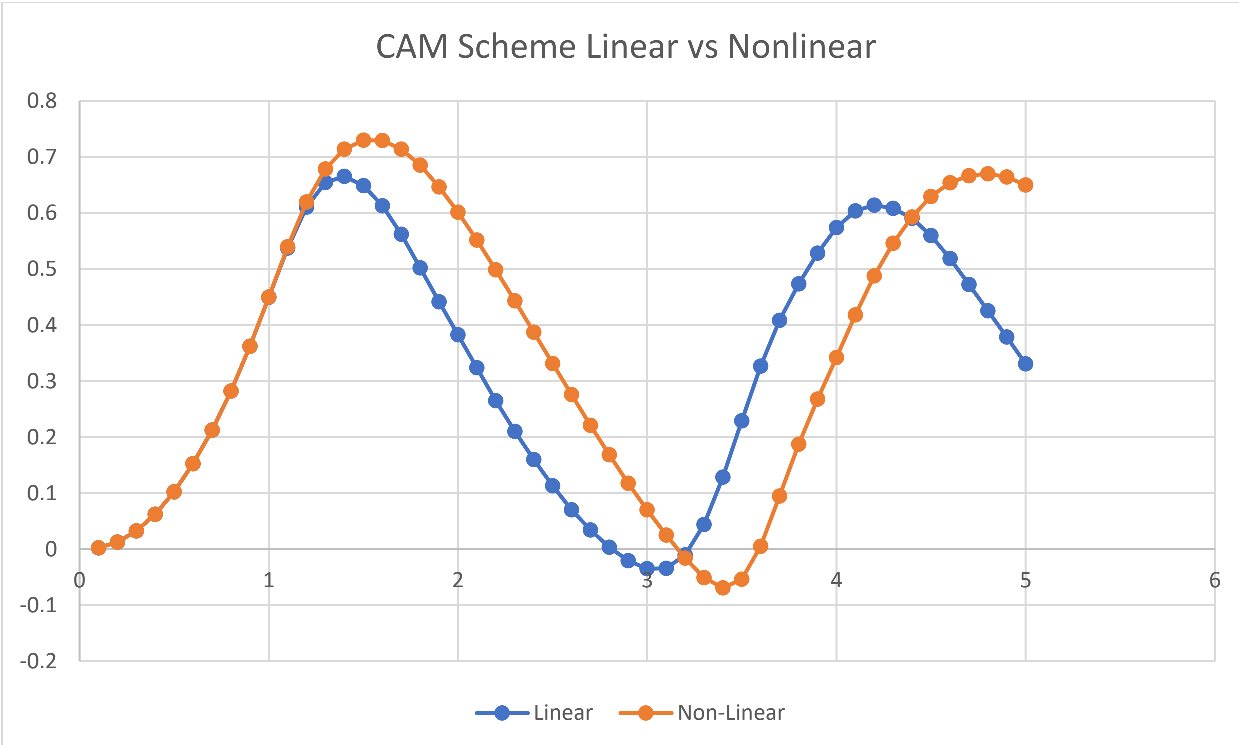
Parameter	Value
Check_Time	0.05:0.05:1.25

Results

Time	CAM	LAM (DT = 0.1)	LAM (DT = 0.05)
0.10	0.00250	0.00167	0.00292
0.20	0.01250	0.01167	0.01542
0.30	0.03250	0.03167	0.03792
0.40	0.06250	0.06167	0.07040
0.50	0.10250	0.10166	0.11293
0.60	0.15250	0.15168	0.16574
0.70	0.21250	0.21167	0.22690
0.80	0.28252	0.28124	0.29892
0.90	0.36239	0.36261	0.38962
1.00	0.45005	0.45652	0.48759
1.10	0.53780	0.54815	0.57007
1.20	0.61100	0.61605	0.63006
1.30	0.65500	0.65464	0.66192
1.40	0.66558	0.66580	0.66078

1.50	0.64920	0.64816	0.63590
1.60	0.61334	0.60793	0.59394
1.70	0.56235	0.55777	0.54241
1.80	0.50251	0.50230	0.48578
1.90	0.44190	0.43966	0.42599
2.00	0.38326	0.38715	0.36605
2.10	0.32430	0.31220	0.30689
2.20	0.26551	0.28407	0.25110
2.30	0.21050	0.17396	0.19591
2.40	0.16010	0.22499	0.14267
2.50	0.11311	-0.00777	0.10136
2.60	0.07064	0.28332	0.06927
2.70	0.03435	-0.34218	0.02892
2.80	0.00383	0.66530	-0.01963
2.90	0.02043	-1.18155	-0.05526
3.00	0.03481	1.94539	-0.06249
3.10	0.03388	-3.50582	-0.03580
3.20	0.01021	5.93986	0.02070





CAM Scheme Linear vs Nonlinear
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
0
1
2
3
4
5
6
Linear
Non-Linear

```
%% Non-Linear FEM - Assignment 4
% Pseudo Code

Input Variable

Call MESH2DR function to generate mesh, ie, define NOD and GLXY↙
matrices

initialize following array matrices
% GPU - Current Time Step and Current Iteration solution
% GLU - Current Time Step and Previous Iteration solution
% GLP - Previous Time Step solution
% GLV - Velocity matrix
% GLA - Acceleration matrix


if (ITEM == 1) % if dealing with parabolic equation
    if NSSV > 0 % ie, there are natural BCs
        VSSV = VSSV*DT; %
    end
end

Define A1, A2, A3, A4, A5, A6, A7, A8

for NT = 1:NTime % Start Time loop
    initiate iter variable
    while (iter < ITERMAX) && (convergence == 0) % Start↙
iterative loop
        initiate GLK and GLF

        for I = 1:NEM % Iterating for every element
            Defining ELU and ELU0

            if ITEM == 2 % Hyperbolic equation
                Defining 1st and 2nd derivative of ELU wrt time↙
```

```
(ELU1 and ELU2)
            else % Parabolic equation
                initializing ELU1 and ELU2
            end

            Defining ELXY % ie, defining element coords

            Calling ELEMATRCS2D_Time function to calculate ELK↙
and ELF

            Assembling GLF and GLK matrices
        end

        Calling BNDRYUNSYM_Time to apply Natural and Essential↙
BCs

        Calculating solution for the current iteration

        increasing iter counter

        if iter == 1 %If for the first iteration
            if NONLIN == 2 % for Newton's iteration only
                VSPV(:) = 0; % After applying the EBCs once, we↙
no longer have to apply the EBC in Newton's iteration
            end
        end

        Calculating Error

        Checking if error is less than epsilon

        Calculating Velocity and Acceleration arrays for next↙
time step

        Updating Previous time step solution (GLP)
```

```
    end
end

% -------------------------------------------------- %

%                      ELEMATRCS2D_Time
% Pseudo Code for ELEMATRCS2D_Time

Defining all the constants of the Differential equation

Initializing ELK, ELK0, ELM, ELF, Tangent matrix (if NONLIN ==↙
2)

Calling Gaus_int function to get the Gauss Points and Gauss↙
Weights

for I = 1:NGPF
    for J = 1:NGPF % Starting the loop for integral calculation
        %Calling INTERPLN2D function for Shape functions (SFL),↙
global derivatives of the shape function, and Jacobian

        Initialize x,y,u,dux,duy
        for I = 1:NPE
            Calculate x, y, u, dux, duy
        end

        Evaluate the functions of Differential equation -
        Fxy, Axx, Ayy, A00

        if ITEM > 0 % For time dependent DE
            evalute C function
            % C is C0 for Parabolic
            % C is C1 for Hyperbolic
        end
```

```
        if ITEM > 0 % For time dependent DT
            %Calculate u, dux, duy, Axx, Ayy, for previous time↙
step value
        end

        Define ELK, ELF, ELM
        % ELM is C matric for Parabolic DE
        % ELM is M matrix for Hyperbolic DE

        if NONLIN > 1 % For Newton's Iteration
            Calculate tangent matrix
        end

        if ITEM == 1 % Parabolic Equation
            Calculate K-Hat and F-Hat
            Equate K-Hat as ELK and F-Hat as ELF
        elseif ITEM >1 % Hyperbolic Equation
            Calculate K-Hat and F-Hat
            Equate K-Hat as ELK and F-Hat as ELF
        end
    end
end
```