3rd Specialty Conference on Engineering Mechanics and Materials

$3^{ième}$ Conférence spécialisée sur le Génie Mécanique et Matériaux

Montréal, Québec

May 29 to June 1, 2013 / *29 mai au 1 juin 2013*

Efficient debugging method development for the inclusion of hysteresis models in finite-element software S-Frame

A.B.M. Rafiqul Haque, Siriwut Sasibut and M. Shahria Alam
School of Engineering, The University of British Columbia, Kelowna, BC, Canada and S-Frame software, Richmond, BC, Canada

Abstract: This paper deals with the common problems faced during the implementation of complex nonlinear load-deformation hysteresis such as Bilinear Kinematic and Takeda models in finite element (FE) modeling software. This paper addresses this common issues and device some efficient methods in resolving them. This paper also shows some very simple and efficient debugging techniques which can be used for similar type of projects as well as for diverse software development projects. This debugging technique utilizes excel spreadsheet to efficiently update and view the variable values without the need for built in debugging tools available in the current integrated development environments. This involves pinpointing the errors in code which can drastically cut down the debugging time. This will help the future developers meet their deadlines much quickly and efficiently, which will result in economic benefits for the organization.

1. Introduction

High intensity seismic load can impose extreme load on a structure which can put the structural elements under severe stress and strain. These stresses and strains can often go beyond linear range of response which might create residual deformation and thus cause changes in the load deformation behavior for the next cycles of loading. After several such cycles of loading the structural members get severely damaged and lose most of their strengths. This type of strength deterioration can result in partial of total collapse of the structure. This is of paramount importance that the behavior of all residential, commercial or transportation structures under seismic or wind loading condition are accurately predicted before the design. This type of prediction requires extreme computational time and can only be done with the help of digital computers. Computers can run simulation software which can very accurately predict the behavior of a structure under seismic or wind loading condition. A well-executed simulation can help the structural designers and engineers make rational decision during design process. A simulation can be classified detailed if it can predict the behavior of all the structural elements present in the model under a large range of load or deformation demand. Most of the commercial finite element (FE) analysis software can analyze a structure with a large number of interconnecting members under various loading conditions. All of these software are capable of dealing with constant material stress-strain responses also known as an elastic response.

Capturing the actual structural behavior under seismic loading is particularly tricky and cannot be done with elastic structural analysis only. Material or structural element level non-linearity is a very important consideration for such loading conditions. For accurate prediction of the behavior of a structure under such loading condition it is very important to consider material or element level nonlinearity. For this reason, it is necessary to implement nonlinear material or element behavior in a finite element software. In this study, a well-known finite element analysis and design software (S-Frame 2012) has been developed to understand the complex non-linear behavior of reinforced concrete and steel structures under such loading conditions.

This project dealt with the implementation of equivalent spring model of some well-established load-deformation/moment-rotation hysteresis in S-Frame software such as Bilinear Kinematic, Bilinear Isotropic, Simplified Bilinear Takeda (Otani 1974), and Bilinear Symmetric rules etc.

A lumped plasticity based link element has been designed and implemented in the S-frame solver for the modeling of these hysteresis springs. This link element is inserted at the location on the frame where it is expected to receive the highest stress and strain during the analysis. Usually this location is near the support of a continuous beam/column and at the middle of a simply supported beam.

2. Development process

This project is divided in three major parts: development of the hysteresis module, Development of the link beam in S-frame, and integration of the hysteresis module with the link beam. This paper will describe the first part, development of the hysteresis module, its debugging and implementation. A step by step development and implementation process is shown in Figure 1 and discussed in detail in this section.

Step 1: The implementation process starts with the selection of a hysteresis model. A hysteresis model is chosen from a list of widely used models, such as: Bilinear kinematic, Bilinear isotropic, Bilinear symmetric, Bilinear asymmetric, Pivot (Dowell et al. 1999), and Simplified bilinear takeda model (Otani 1974) etc.

Step 2: Extensive literature survey is carried out to document all the rules associated with the selected hysteresis model. Hysteresis rules are written down as a step-by-step procedure.

Step 3: Algorithm is developed for code writing purpose.

Step 4: A worksheet is designed in excel for data input and output from the visual basic macro to be used in the next step. A system is implemented to track all the variables used in the macro at each step of the analysis. Furthermore, code segments are given “code location id’s”, which were also tracked at each iteration and printed on the excel worksheet. The “code location id” was used for tracking the exact location of bug when output has errors.

Figure 2 shows a screenshot of the excel spreadsheet utilized for the debugging of bilinear takeda (Otani 1974) hysteresis model. In this figure, column “D” has the input rotation values used for generating the output moment values (Column E). The chart at the bottom left corner shows the calculated hysteresis response from the Visual Basic (VB) macro. Cell A1-C10 holds some important input and output parameters for the hysteresis model, e.g. Yield moment, Initial Stiffness, Post elastic stiffness, and energy (work done) calculation etc. Column “F” to “N” show the printed values of the program variables at each iteration of the analysis.

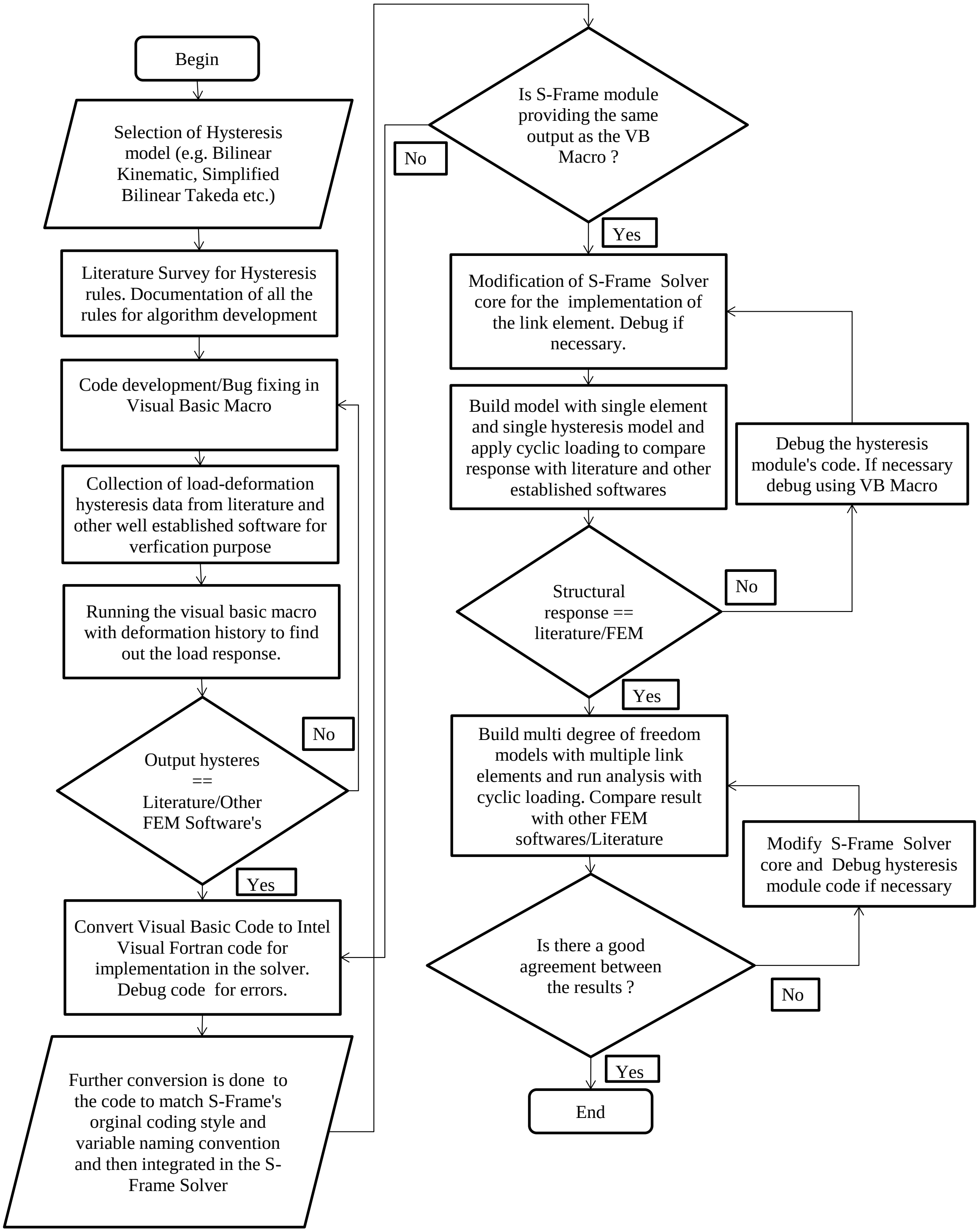
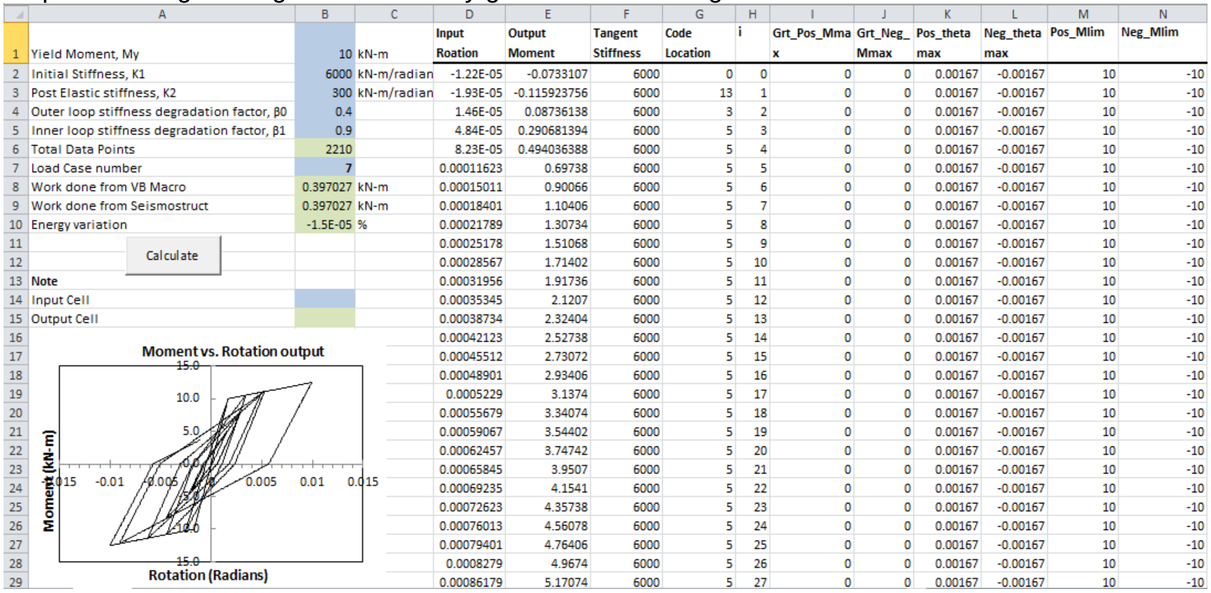

Begin
Selection of Hysteresis model (e.g. Bilinear Kinematic, Simplified Bilinear Takeda etc.)
Literature Survey for Hysteresis rules. Documentation of all the rules for algorithm development
Code development/Bug fixing in Visual Basic Macro
Collection of load-deformation hysteresis data from literature and other well established software for verfication purpose
Running the visual basic macro with deformation history to find out the load response.
Output hysteres == Literature/Other FEM Software's
No
Yes
Convert Visual Basic Code to Intel Visual Fortran code for implementation in the solver. Debug code for errors.
Further conversion is done to the code to match S-Frame's orginal coding style and variable naming convention and then integrated in the S-Frame Solver
Is S-Frame module providing the same output as the VB Macro ?
No
Yes
Modification of S-Frame Solver core for the implementation of the link element. Debug if necessary.
Build model with single element and single hysteresis model and apply cyclic loading to compare response with literature and other established softwares
Debug the hysteresis module's code. If necessary debug using VB Macro
Structural response == literature/FEM
No
Yes
Build multi degree of freedom models with multiple link elements and run analysis with cyclic loading. Compare result with other FEM softwares/Literature
Modify S-Frame Solver core and Debug hysteresis module code if necessary
Is there a good agreement between the results ?
No
Yes
End

Figure 1: Hysteresis rule implementation process in S-Frame (2012)

Column “G” in figure 2 shows the reporting of the code location along output moment values. Figure 3 shows a segment of the code where “location()” array have been inserted. This array keeps the addresses of the segment of the code being executed just above it. Whenever the program calculates a moment value and moves to the next step of the iteration the location array saves the address associated with that portion of the code. If a moment output does not make sense or seems incorrect, the programmer can directly go to the code location and fix that portion of the code just by looking at the associated value from column “G: code location”. This approach can drastically reduce the debugging time .

In Figure 2 column H to column N shows some other important variable values. For space limitation only few of these variables are shown here. During debugging it is very important to track the changes of all associated variables which is very difficult to do with traditional debugging tools. Sometimes the addition or subtraction of two different variables are needed for verification purpose. By tracking all the variables at the same time this can be done very effectively. On the other hand, after “code location” pinpoints the faulty code segment it sometimes becomes difficult to pinpoint the variable that is causing the error, by tracking all variable values it becomes very easy to pinpoint that variable.

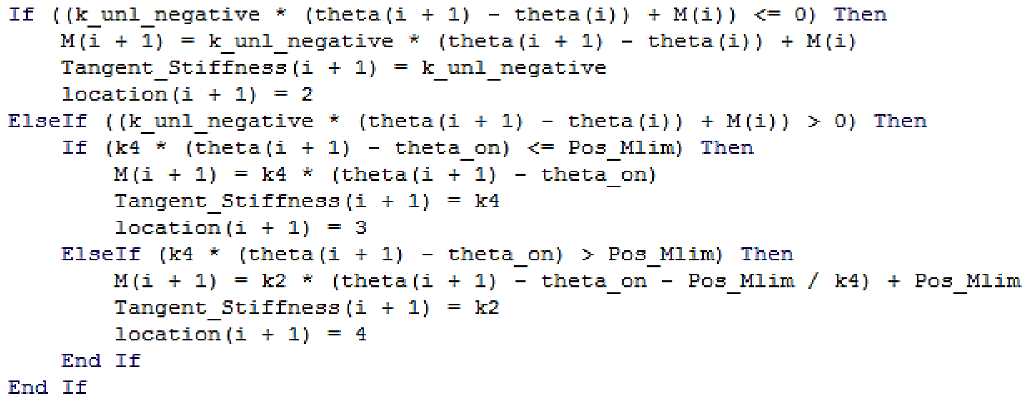
Step 5: Algorithm is converted to excel visual basic macro code. Code gets reviewed thoroughly for possible bugs. If bugs are found they get fixed at this stage.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Yield Moment, My	10	kN-m	Input Roation	Output Moment	Tangent Stiffness	Code Location	i	Grt_Pos_Mmax	Grt_Neg_Mmax	Pos_theta max	Neg_theta max	Pos_Mlim	Neg_Mlim
2	Initial Stiffness, K1	6000	kN-m/radian	-1.22E-05	-0.0733107	6000	0	0	0	0	0.00167	-0.00167	10	-10
3	Post Elastic stiffness, K2	300	kN-m/radian	-1.93E-05	-0.115923756	6000	13	1	0	0	0.00167	-0.00167	10	-10
4	Outer loop stiffness degradation factor, β0	0.4		1.46E-05	0.08736138	6000	3	2	0	0	0.00167	-0.00167	10	-10
5	Inner loop stiffness degradation factor, β1	0.9		4.84E-05	0.290681394	6000	5	3	0	0	0.00167	-0.00167	10	-10
6	Total Data Points	2210		8.23E-05	0.494036388	6000	5	4	0	0	0.00167	-0.00167	10	-10
7	Load Case number	7		0.00011623	0.69738	6000	5	5	0	0	0.00167	-0.00167	10	-10
8	Work done from VB Macro	0.397027	kN-m	0.00015011	0.90066	6000	5	6	0	0	0.00167	-0.00167	10	-10
9	Work done from Seismostruct	0.397027	kN-m	0.00018401	1.10406	6000	5	7	0	0	0.00167	-0.00167	10	-10
10	Energy variation	-1.5E-05	%	0.00021789	1.30734	6000	5	8	0	0	0.00167	-0.00167	10	-10
11				0.00025178	1.51068	6000	5	9	0	0	0.00167	-0.00167	10	-10
12				0.00028567	1.71402	6000	5	10	0	0	0.00167	-0.00167	10	-10
13	**Note**			0.00031956	1.91736	6000	5	11	0	0	0.00167	-0.00167	10	-10
14	Input Cell			0.00035345	2.1207	6000	5	12	0	0	0.00167	-0.00167	10	-10
15	Output Cell			0.00038734	2.32404	6000	5	13	0	0	0.00167	-0.00167	10	-10
16				0.00042123	2.52738	6000	5	14	0	0	0.00167	-0.00167	10	-10
17				0.00045512	2.73072	6000	5	15	0	0	0.00167	-0.00167	10	-10
18				0.00048901	2.93406	6000	5	16	0	0	0.00167	-0.00167	10	-10
19				0.0005229	3.1374	6000	5	17	0	0	0.00167	-0.00167	10	-10
20				0.00055679	3.34074	6000	5	18	0	0	0.00167	-0.00167	10	-10
21				0.00059067	3.54402	6000	5	19	0	0	0.00167	-0.00167	10	-10
22				0.00062457	3.74742	6000	5	20	0	0	0.00167	-0.00167	10	-10
23				0.00065845	3.9507	6000	5	21	0	0	0.00167	-0.00167	10	-10
24				0.00069235	4.1541	6000	5	22	0	0	0.00167	-0.00167	10	-10
25				0.00072623	4.35738	6000	5	23	0	0	0.00167	-0.00167	10	-10
26				0.00076013	4.56078	6000	5	24	0	0	0.00167	-0.00167	10	-10
27				0.00079401	4.76406	6000	5	25	0	0	0.00167	-0.00167	10	-10
28				0.0008279	4.9674	6000	5	26	0	0	0.00167	-0.00167	10	-10
29				0.00086179	5.17074	6000	5	27	0	0	0.00167	-0.00167	10	-10

Figure 2: A partial screenshot of the excel spreadsheet designed for simplified bilinear takeda hysteresis model (Otani 1974) development.

Step 6:. Input data array is provided directly in a column (e.g. Column “D” in Figure 2) which gets read out by the macro. The macro then processes the input data and writes the output directly on the same worksheet (e.g. Column “E” in Figure 2). The worksheet contains a linked plot (Figure 2) which generates the hysteresis plot each time the macro is run.



```
If ((k_unl_negative * (theta(i + 1) - theta(i)) + M(i)) <= 0) Then
    M(i + 1) = k_unl_negative * (theta(i + 1) - theta(i)) + M(i)
    Tangent_Stiffness(i + 1) = k_unl_negative
    location(i + 1) = 2
ElseIf ((k_unl_negative * (theta(i + 1) - theta(i)) + M(i)) > 0) Then
    If (k4 * (theta(i + 1) - theta_on) <= Pos_Mlim) Then
        M(i + 1) = k4 * (theta(i + 1) - theta_on)
        Tangent_Stiffness(i + 1) = k4
        location(i + 1) = 3
    ElseIf (k4 * (theta(i + 1) - theta_on) > Pos_Mlim) Then
        M(i + 1) = k2 * (theta(i + 1) - theta_on - Pos_Mlim / k4) + Pos_Mlim
        Tangent_Stiffness(i + 1) = k2
        location(i + 1) = 4
    End If
End If
```

Figure 3: Utilization of the code location identifier array

A sample of four input rotation history used for the VB Macro result verification is presented in Figure 4. Series 1 is a saw tooth shaped curve, series 2 is another jagged curve, series 3 is a complex shape with both sine wave and some sharply changing data points and series 4 is an inversion of series 3 with some additional data and two sine waves instead of one.

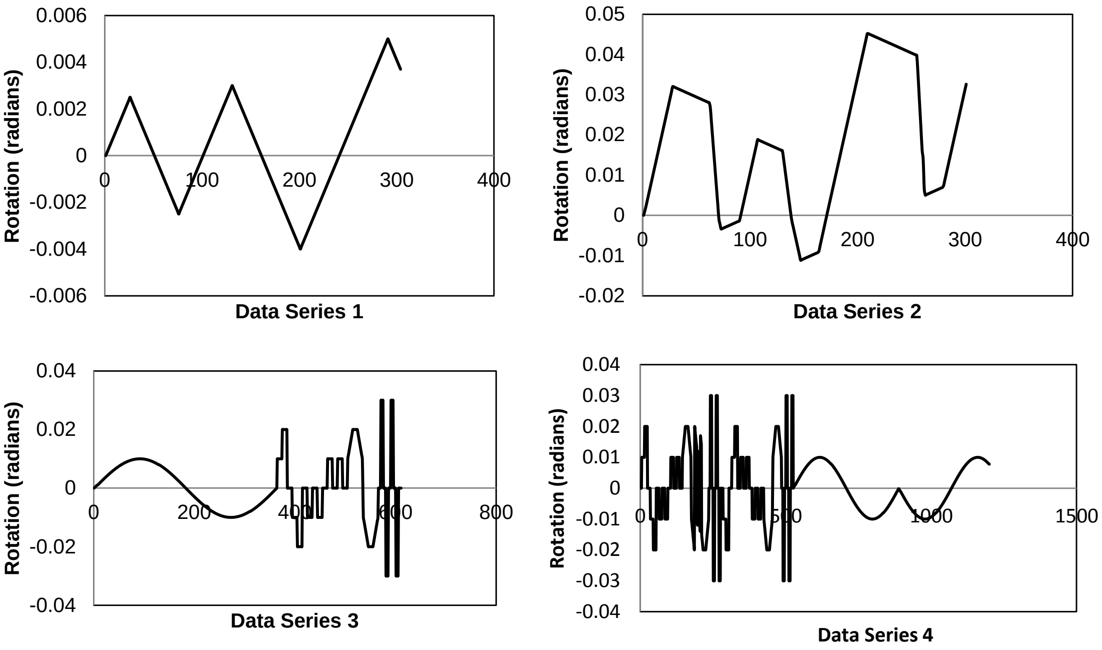


Figure 4: Input rotation history for VB Macro result verification

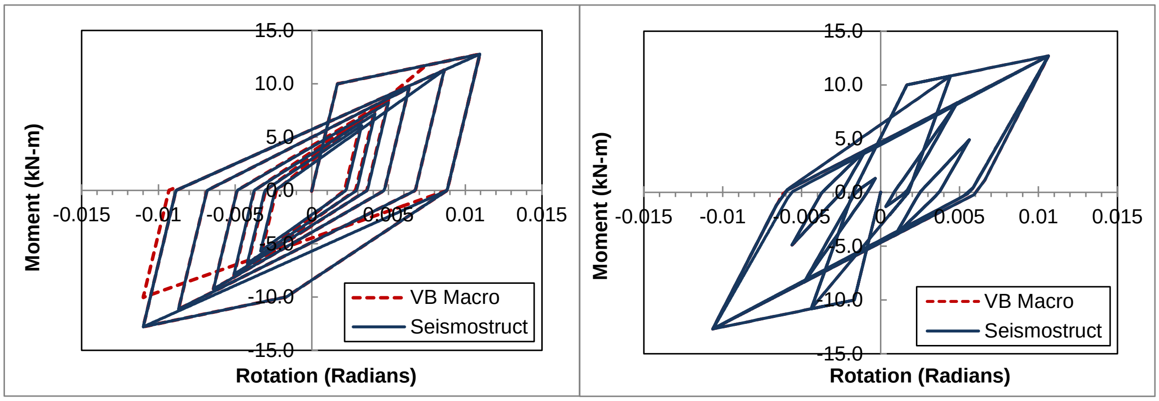
Step 7: The output hysteresis is then checked with expected hysteresis shapes collected from literature survey or other finite element modeling software's output. They are checked under same input deformation history. If a mismatch is found between the output and the expected hysteresis, the underlying principle, algorithm and the macro code is checked again for possible misunderstanding, code bugs and modified accordingly. If a good agreement is found between expected and output hysteresis with respect to both load-deformation data points and energy dissipation then project is moved to the next step. Figure 5(a) shows a condition of hysteresis mismatch between VB Macro and Seismostruct (2012) and Figure 5(b) shows a condition where there was an exact match between the two.

Step 8: The visual basic macro code is converted to Intel Visual Fortran (2011) and checked for conversion errors. Code debugging is done in this step.

Step 9: The converted Intel Visual Fortran (2011) code is then run with the same hysteresis models as previously used for the VB Macro code. The result is then checked against the expected hysteresis output. If a mismatch is found then “step 8” is executed otherwise “step 10” is executed.

Step 10: Intel Fortran (2011) Code gets converted into a module for the solver. Some variable naming convention and coding style is changed to match with the solvers existing style and structure. Code gets thoroughly debugged.

Step 11: S-Frame (2012) module is run again with the same input hysteresis data used for VB macro and Intel Visual Fortran (2011) code. The module’s output is checked against the expected output and if a disagreement is found then “Step 10” is executed until the output matches the expected.



(a) (b)

Figure 5: (a) Overlaid hysteresis from VB Macro and an FEM software (Seismostruct 2012) showing a mismatch. (b) Overlaid hysteresis from VB Macro and an FEM software (Seismostruct 2012) showing exact match.

Step 12: S-Frame link element’s implementation is reviewed, checked and corrected for implementation of the new hysteresis model.

Step 13: The hysteresis module is then integrated into the link beam element module in S-Frame solver and a single degree of freedom (SDOF) model with a single link element is modeled using the S-Frame graphical user interface (GUI).

Step 14: Various loading history is used on the SDOF model and nonlinear time history analysis is carried out.

Step 15: Output data is checked with other FEM software’s output and literature (if any). If a disagreement is found then “step 12” is executed. If good disagreement is found step 16 is executed.

Step 16: Multi degree of freedom (MDOF) portal frames are built at this stage with multiple link elements at the member ends and supports. Similar structural models are built in other FEM software with same hysteresis model.

Step 17: Several nonlinear time history analyses are carried out on these models and results are compared. If results disagree then step 12 is executed. If there is a good agreement between the results from two models then “Step 18” is executed.

Step 18: Documentation (Flow chart, project report) of above mentioned processes are done and kept for future reference.

Step 19: Completion.

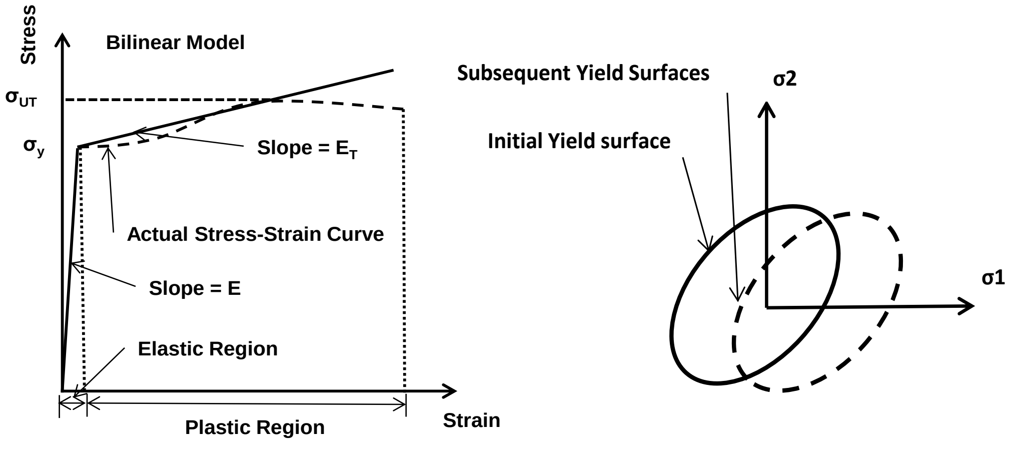
After the completion of the implementation of one hysteresis model the project continues with step1 and a new hysteresis model.

3. Example project

For space limitation, only one hysteresis model’s implementation is discussed here. The selected model is bilinear kinematic hysteresis. Presented below is the description of bilinear kinematic model, its rules and the implementation process.

3.1 Background

The bilinear elasto-plastic material model is one of the simplest material model available for modeling plastic deformation. As the name suggests this model uses two lines to represent the stress strain curve, one line for elastic portion and the other line for post elastic portion.



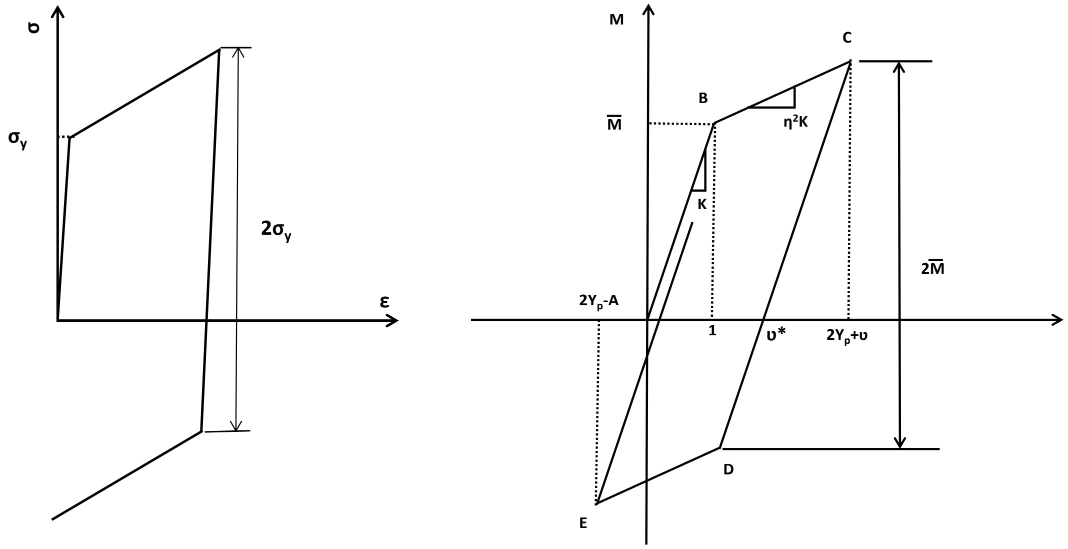
(a) (b)

Figure 6: (a) Bilinear idealization of stress strain behavior of metal (Bryan, 2007) (b) Kinematic hardening laws in two dimensions (Bryan, 2007)

Figure 6(a) shows a tress strain curve for a typical metal and illustrates how the tangent modulus can be used to represent the slope of the second line. In this model a straight line is used to approximate the highly curved plastic portion of the stress strain curve. The slope of this straight line is known as the tangent modulus. As its name suggests the tangent modulus may be obtained by drawing a tangent to the plastic portion of the stress strain curve and measuring its slope.

3.2 Bilinear Kinematic Hardening Rule

Bilinear kinematic hardening rule describe how the yield surface will change as the yielding continues, so that the stress states for subsequent yielding can be established. Any stress state that is the combination of principal stresses inside yield surface is elastic and outside is plastic. Kinematic hardening assumes that the yield surface will not change in size but will change location with progressive yielding as shown in figure 6(b). The fundamental assumption of this model states that the total stress (i.e. difference between the Ultimate Tensile Stress and the maximum compressive stress) is equal to twice the yield stress. This is also known as the ‘Bauschinger effect’ and is illustrated in figure 7(a) (Bryan, 2007).



(a) (b)

Figure 7: (a) Bauschinger effect (b) The characteristic diagram of elasto plastic spring (Pratap and Mukherjee, 1994)

A bilinear kinematic moment-rotation spring behavior is shown in Figure 7(b). In this figure bending moment and normalized rotation angle υ is plotted in two perpendicular axes. A rotational spring constant K is used in the elastic region and η^2K in the plastic region. Here η^2 is representing the kinematic hardening coefficient. The value, $\eta^2 = 0$ means perfectly plastic and $\eta^2 = 1$, relate to perfectly elastic behavior. In kinematic hardening, the total range of elastic deformation remains constant over time. Figure 7(b) shows that from point A to B the spring is elastic and during rotation between point B and C it shows plastic behavior. The spring starts elastically unloading from point C until the moment reduces by 2M to point D. After reaching point D the spring again yields in the opposite direction. The total moment between the points of elastic unloading to the next plastic yield is always same having a value of 2M. For this reason, the linear extensions of the lines BC and ED or the plastic yield lines, always remains parallel in the M- υ plane (Pratap and Mukherjee, 1994).

3.3 Algorithm, code and flowchart

Due to a non-disclosure agreement with the industrial partner (S-FRAME Software Inc.) the full algorithm, flowchart, and code is shown in this paper. Only a brief version of the algorithm is shown below for demonstration purpose.

1. Read input data: Yield moment, Initial, post elastic stiffness, initial rotation, and moment.
2. Read next rotation data

3. If current moment value is less than yield moment then check If rotation is increasing/decreasing. Check whether the calculated moment using initial stiffness is going to be higher/lower than yield moment. If not, then calculate the moment using initial stiffness otherwise predict the yield surface (using straight line's equation passing through yield point with post yield stiffness). Calculate the yield point and using the yield rotation value calculate moment using post-elastic stiffness and difference in rotation beyond the yield point. This point must lie on yield surface. Go to step 5.
4. If current moment value is higher than the positive yield moment or lower than the negative yield moment then calculate the moment value using post-elastic stiffness. Go to step 5.
5. Print Moment value and go to step 2.

3.4 Debugging of Bilinear Kinematic Model

The debugging approach shown in Figure 2 has been utilized for bilinear kinematic model which resulted in fast error detection. Also, data series 3 shown in figure 4 was used to check robustness of the code. This data series contains big differences in between each values within 358th to 610th data point of the series. These large differences causes the program skip one or more code branches. This happens because if there is a large difference between two rotation values then calculation with initial stiffness might generate a very large bending moment which might be out of the yield surface. For these cases, the program automatically calculates the yield point using initial stiffness and then calculates the actual bending moment using the remainder rotation and post yield stiffness. If the code is not written to take these jumps into consideration then the hysteresis will have large error which will result in wrong structural response.

Data series 3 was created to check the code for these scenarios. For checking the robustness first yield surfaces are calculated and drawn manually on the plot (Figure 11). Next, the macro is executed and the resultant hysteresis is drawn over it. If all of the points fall on the yield surface then the code is assumed to have passed verification process for this step.

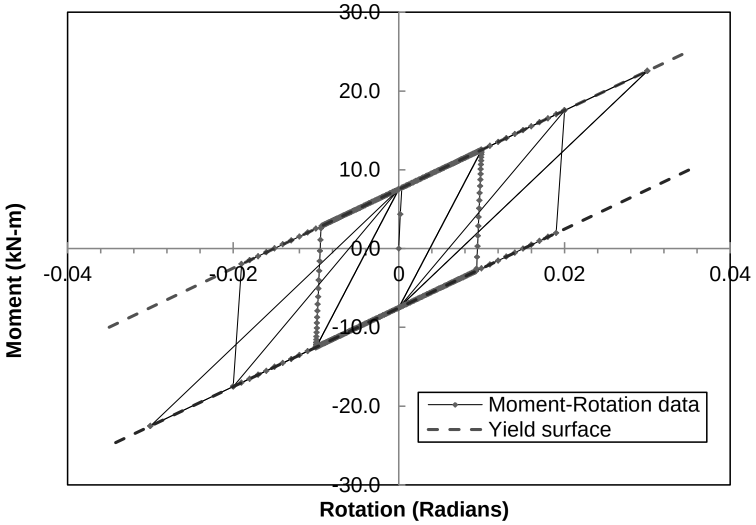


Figure 11: Debugging with yield surface plot

After the verification of VB Macro code, similar exercises were carried out on the Intel Fortran (2011) code. After Intel Fortran (2011) code passed the verification the code was implemented as a module inside the S-Frame (2012) solver and tested again. After the test and debugging, the module was integrated to the link element and the next phase of testing commenced.

4. SDOF and MDOF testing

Several SDOF model with link element were built in the S-Frame GUI and tested under cyclic loading condition. The load was carefully selected to create nonlinear response to the SDOF structure. An Identical model was built in Seismostruct (2012) environment and the result from similar analysis was found satisfactory. There were good agreements between these two structures' responses. After this step, a multi-degree of freedom portal frame was created in both S-frame (2012) and Seismostruct (2012) environment. A point load was applied at the corner of the frame and several loading time history was associated to it. This model was run under several time history records and the results were compared to the responses of identical Seismostruct (2012) model. All link elements responses were checked with their counterpart and was found to be in good agreement.

5. Conclusion

An efficient implementation method has been developed for the inclusion of hysteresis models in finite element software. This method utilizes the widely used and available Microsoft Excel software and its macro capability. The excel spreadsheet is used to track changes to all variables used by the program at each step of the program's iteration. Also, built in search, subtraction and other functions available in excel have drastically reduced debugging time. The automatic plotting of hysteresis shapes and overlapping with known shapes also dramatically reduced error detection time. This debugging process is simple and doesn't require much effort or technical knowledge. This method can be implemented in various software development projects which can result in time saving.

References

Bryan, J. M. 2007. Practical Stress Analysis with Finite Elements. Glasnevin Publishing, Dublin.

Dowell, R. K., Seible, F., & Wilson, E. L. 1999. Pivot Hysteresis Model for Reinforced Concrete Members. Aci Structural Journal, (95): 607–617.

Intel® FORTRAN. 2011. Version 2011.4.196. Intel® Visual FORTRAN Composer XE 2011 for Windows. Santa Clara, California, USA

Microsoft Excel. 2010. Microsoft Excel 2010 help. Microsoft corporation, Redmond, Washington, USA

Otani, S. 1974. SAKE a computer program for inelastic response of R/C frames to earthquakes. University of Illinois, Urbana, Illinois.

Pratap, R. and Mukherjee, S. 1994. Dynamic behavior of a bilinear hysteretic elasto-plastic oscillator, Part I: Free Oscillations. *Journal of sound and vibrations*, 172(3): 321-337

Seismostruct, 2012. version 6, Seismsoft Ltd. , Seismostruct Help, Perikleous Stavrou str, Chalkida, Greece

S-Frame. 2012. S-Frame reference manual. S-Frame Software Canada. Richmond, BC, Canada