LexTech

AFGROW.NET

AFGROW Workshop 2026

Developing AFGROW Plugins Using StressCheck Automation

Jimmy Lambert

LexTech

AFGROW.NET

Introduction

This Presentation will contain two major sections:

- Narrative of what exactly I did to implement two StressCheck proof-of-concept plugins
- Results and recommendations after implementing these plugins

LexTech

AFGROW.NET

Agentic Coding

- Throughout this project, I used agentic coding agents with the hope of finding out if it would be a useful asset for creating plugins from scratch
- The project included a combination of implementation done by myself, as well as many components coded by coding agents with only prompting from me
- If something particularly notable happened, I will mention it on the slide, otherwise, I will give my opinions about the usefulness of agentic coding after the narrative portion of the presentation

LexTech

AFGROW.NET

Through Crack Tabular Model – Plugin Template

- To start, I decided to implement a through-crack at hole with a tabular solution
- I created a scaffold for the project, filling in all of the required functions for the AfgrowBetaPlugin interface with stubs and todos
- I also included a way of managing AFGROW parameter information in a custom class called “Parameter”
- These features are the components of a baseline template

LexTech

AFGROW.NET

Plugin Template – Beta Table

The template I created contained additional functions specific to reading a beta table:

- I created a sample Beta table for the plugin to read with dummy values
- The plugin template implemented the ability to read this beta table
- The plugin could interpolate over this table to get any arbitrary Beta for any crack length

LexTech

AFGROW.NET

Plugin Template – Build and COM Registration

- Next, I enabled building for COM interop, necessary for connecting to AFGROW
- I created a new GUID and ProgID for the plugin
- I added the ProgID to the AFGROW configuration file
- Finally, I built the project, and attempted to load it through AFGROW
  - Trying to load the project resulted in a “Class not registered error”
  - I had to debug this issue, and .NET 10 was not playing nice with AFGROW
  - I reverted to .NET 8.0
  - I may have been able to get .NET 10 to work, but I moved on at this point
  - (The coding agent did not handle this well. I debugged this by hand)

Through Crack Tabular Model – Specific Implementation

- Next, I created a copy of the project to house the actual implementation of the through crack at hole plugin
- This implementation would only be able to handle a static thickness, radius, and width. Crack length would be the only variable.
- Implemented the drawing of the plate, hole, and crack
- Successfully ran a prediction in AFGROW with dummy data

LexTech

AFGROW.NET

StressCheck Scripting – Parametric Approach

I opted for a semi-automated approach for running StressCheck batches

- I created a model inside of StressCheck by hand
- All potential input parameters were defined in the StressCheck parameters window (plate width, plate height, plate thickness, hole radius, and crack length)
- The mesh, loads, and constraints were configured within StressCheck
- Sets were defined for relevant geometric features

StressCheck 12.1

File Edit Class View Display Tools Help

3D | Elasticity | in/lbf/sec/F | All Objects

Display Controls

Transparency
Mesh 50 %
Geometry 50 %
Depth Peeling (fastest)

Shrink Factor 70 %

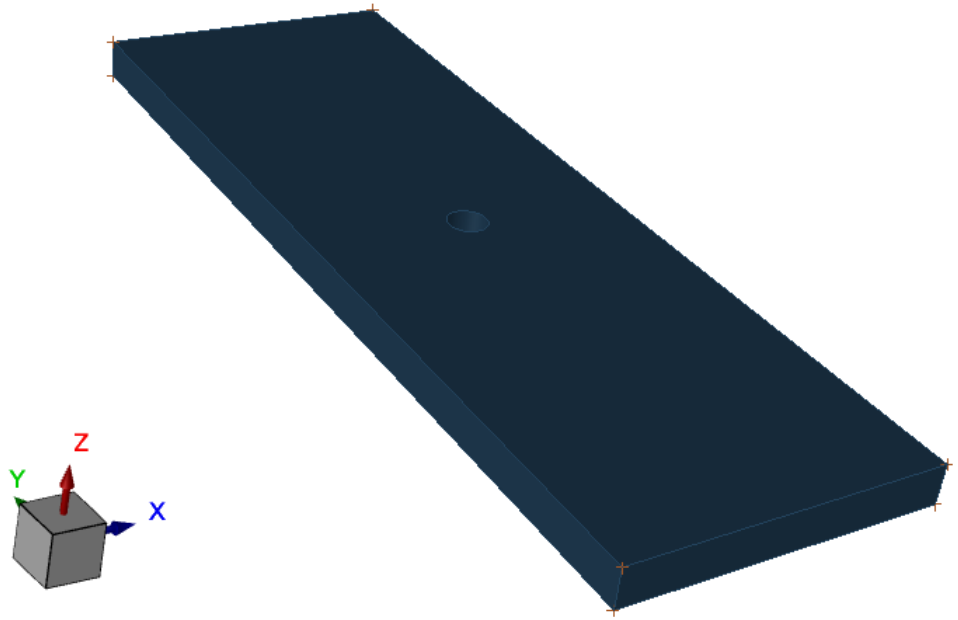
Element Edges: Dark

Settings
Geometry Resolution: 100
Element Resolution: 8
Silhouette Sensitivity: 1.0
Display Format: %10.4e

Suppress Small Features
Pixel Boundary Box: 5

Apply

Model View



StressCheck V12.1

Parameters

Name	Desc	Expr	Value	Limit	Class	Label
pWidth			4.0000e+00		General	
pThickness			5.0000e-01		General	
pHeight		pWidth * 3	1.2000e+01		General	
hRadius			2.5000e-01		General	
hDiameter		hRadius * 2.0	5.0000e-01		General	
cLength			2.0000e-01		General	
aLength			2.0000e-01		General	

Accept | Delete | Auto Step | 0.2 | Settings | Dependen

Simulation

Input
Solve
Results
>>

Geometry | Mesh | Section Prop. | Thickness | Material

Select | Any Object | Locate

Screen:
X: 0.0000e+00
Y: -6.0000e+00
Z: 0.0000e+00
Rot-X:
Rot-Y:
Rot-Z:

Data | Index

Move | Replace | Delete | DeLast
Copy | Assoc. | Disassoc. | Deselect

Message Log

Info | Warn | Error | File | Input | Undo | Solve | Clear

```
Sep 11 2026 03:17:55PM  Opened project
Sep 11 2026 03:17:55PM  Parasolid session precision =  1.0e-08
```

Message Log | Project Log | Error Log

Action: Select ANY GEOM. OBJECT at screen LOCATION.

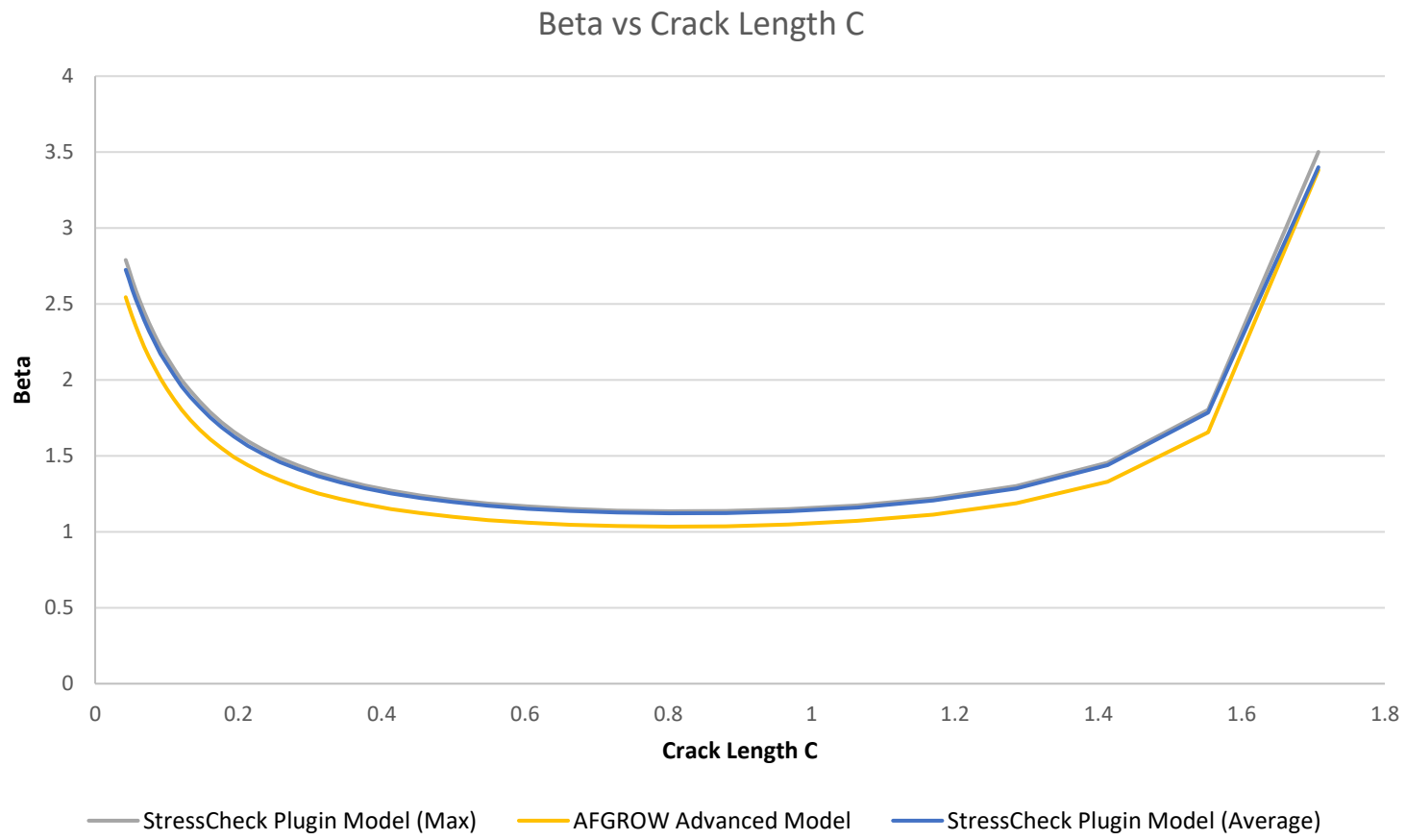
LexTech

AFGROW.NET

Parametric Approach – Batch Script

- The actual batch script for running StressCheck mainly interacts with the properties already defined within the base .scp file
- The script edits the named parameters from the base file
- After editing the parameters (plate width, height, etc.), Automesh() is called to ensure the mesh is re-built with the correct parameters
- It creates a solution ID named “AUTOCONFIG” and solves the model
- Extracts K1 values along the crack front
- Averages, or takes the max value of the extracted K1 values to obtain a single K value for the whole crack front
- Converts Ks to Betas, and writes those values to a table

Through Crack Tabular Model – Beta vs Crack Length C



LexTech

AFGROW.NET

Corner Crack Live Script

- After getting the through crack at hole script working, I copied the through crack plugin as a starting point to create the corner crack at hole live script
- I updated the project to have a new GUID and ProgID
- I removed all of the existing table reading and interpolating functions

LexTech

AFGROW.NET

Corner Crack Live Script – Enabling Additional Parameters

This live script would be able to support more user-defined parameters, because it is a live script.

- Added the second crack direction parameter (A)
- Enabled the plate width for editing
- Enabled the plate thickness for editing
- Enabled the hole diameter for editing

Corner Crack Live Script – Modify the Baseline StressCheck Model

- I copied the baseline StressCheck model used for the through crack at hole script
- I deleted the mesh and crack front
- I created a new variable for crack length a
- I defined a new crack front based on an ellipse
- I re-defined the automesh
- I re-defined the crack front set with the new curvatures

LexTech

AFGROW.NET

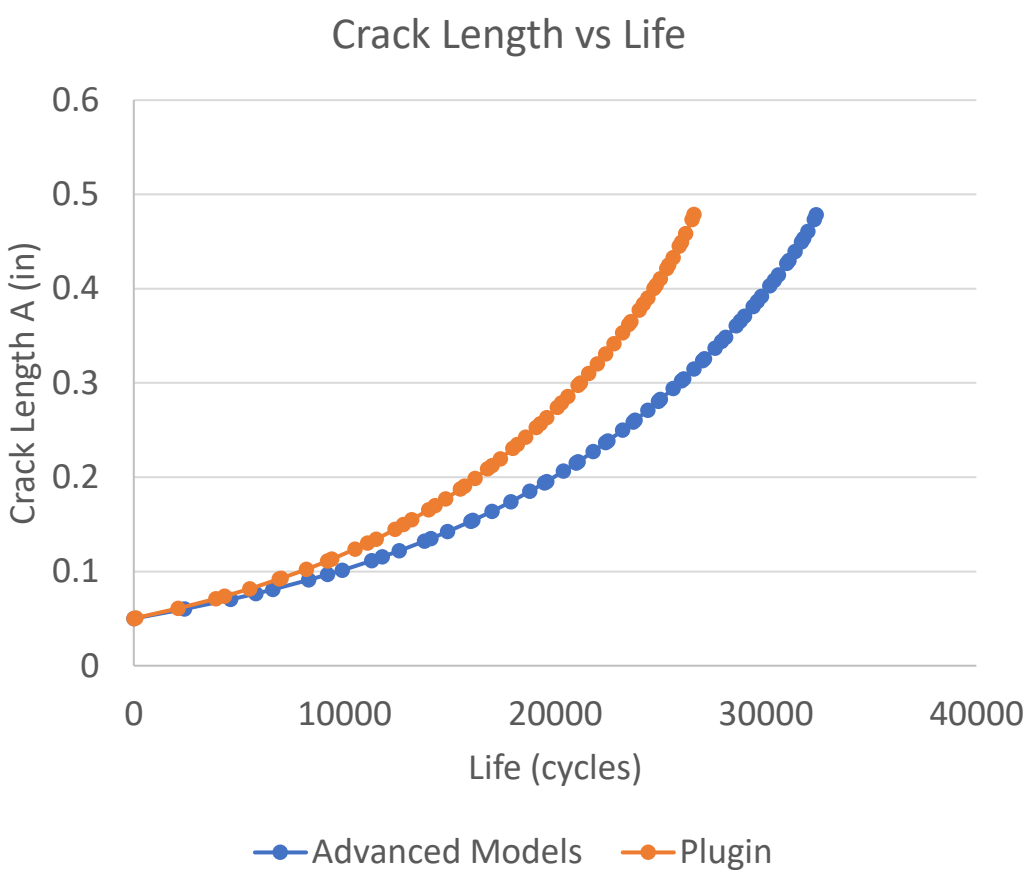
Corner Crack Live Script – Adapt the Through Crack at Hole Script

Because of the parametric approach using a baseline StressCheck model, the majority of code logic was able to be recycled from the original StressCheck script.

- I copied the through crack at hole script
- I populated the parameters from the baseline .scp file with the parameters which are set by the user through the AFGROW plugin interface
- I removed the portions of code that wrote the evaluated max and average K values to a file
- I added a function that calculates the local maxima for K values near the free surface for the c and a direction

Corner Crack Live Script – Results

- Ran with Constant Amplitude Spectrum
  - SMF = 14
  - R = 0.0
- Default AFGROW Material (2024-T3)
- **StressCheck pLevel set to 2**
- Running the prediction until through crack transition took approximately 40 minutes.



LexTech

AFGROW.NET

Conclusions – What was agentic coding most useful for?

- Coding agents can access registered COM Libraries
  - It automatically recognizes COM signatures, including input parameters, return parameters, etc., as they exist installed on your system.
- Implementing routine functions
  - File IO, drawing functions, numerical functions, etc. are all trivial for it to execute as long as you actually know what you want
- Debugging
  - Coding agents can build your C# project, read the errors, and correct them.
- Something to be wary of: Coding agents love to build in “Fallback” code
  - Read the code, make sure it passes the sniff test

LexTech

AFGROW.NET

Conclusions – Tabular Solutions

Advantages

- Fast at runtime
  - With a tabular solution, you can front-load all of the FEM work so that actual crack growth predictions are faster
- Runtime Simplicity
  - Interpolating on a table is easy to verify and a simple process at runtime

Disadvantages

- Many runs must be performed in order to ensure good interpolation
- Does not scale easily to high dimensional problems
  - Large problems require enormous computational power
  - Your table will become large and hard to verify if you do find the resources
- Interpolation on tables can be a source of error
  - This can be mitigated, but it is not trivial

LexTech

AFGROW.NET

Conclusions – Live Scripts

Advantages

- Model complexity does not scale the problem computationally
  - Any number of parameters can be added to the model without affecting runtime
- Accuracy
  - No interpolation means the K values you get from your FEA are exactly the K values AFGROW will use to grow your crack

Disadvantages

- Your FE solver **must** be accessible
  - This adds another piece of software that must be deployed to anyone who wants to run the model
- More complexity at runtime can interrupt trying to run an analysis with debugging sessions
- **Slow** at runtime
  - Approximately 40 minutes for a pLevel = 2 run

LexTech

AFGROW.NET

Recommendations – How Do I Build a Plugin?

1. Tabular vs Live Script
   - Determine which framework fits your use case the most.
2. Develop your parametric model in StressCheck
3. Create a base framework for your application that interfaces with AFGROW.
   - This should include all of the parameters initialized in AFGROW, and a structure for keeping track of them inside of your template.
   - This should also include the functions for drawing the model to the screen
4. Implement the FEA portion of your plugin.
   - If you are using a table, it will be a separate application that generates the table or a table created from manual StressCheck runs
   - If you are implementing a live script, it will mean implementing the StressCheck COM calls necessary to run your prediction.
5. Make sure your results are reasonable
   - Verifying your implementation is always necessary, especially if you use coding agents

LexTech

AFGROW.NET

Questions?