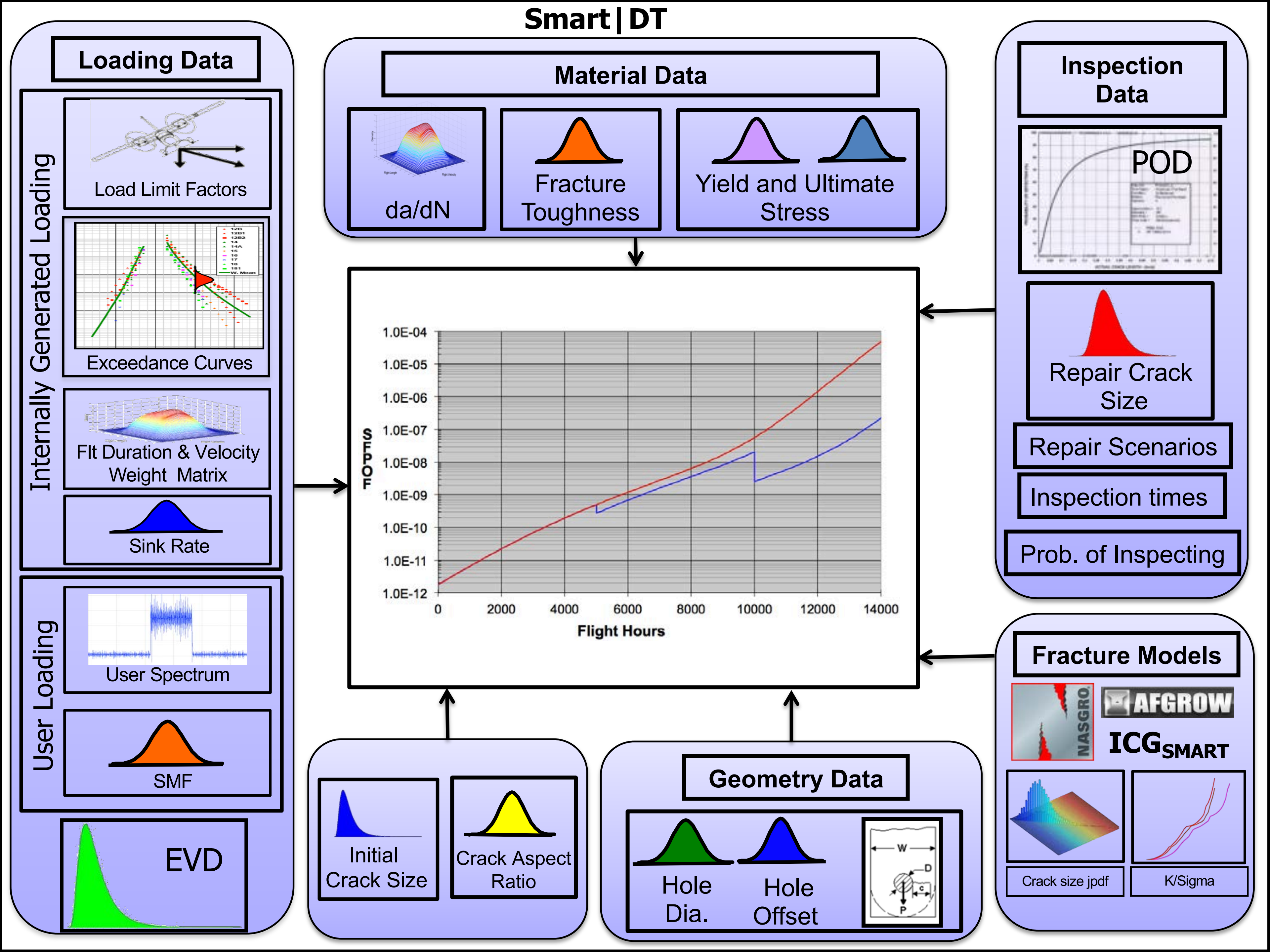
An Ultrafast Crack Growth Lifing Algorithm for Probabilistic Damage Tolerance Analysis

Harry Millwater, Nathan Crosby
University of Texas at San Antonio

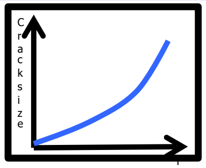
Juan D. Ocampo
St. Mary's University, San Antonio



Smart | DT
Loading Data
Internally Generated Loading
Load Limit Factors
Exceedance Curves
Flt Duration & Velocity Weight Matrix
Sink Rate
User Loading
User Spectrum
SMF
EVD
Material Data
da/dN
Fracture Toughness
Yield and Ultimate Stress
SFPOF
1.0E-04
1.0E-05
1.0E-06
1.0E-07
1.0E-08
1.0E-09
1.0E-10
1.0E-11
1.0E-12
0
2000
4000
6000
8000
10000
12000
14000
Flight Hours
Inspection Data
POD
Repair Crack Size
Repair Scenarios
Inspection times
Prob. of Inspecting
Fracture Models
NASGRO
AFGROW
ICG SMART
Crack size jpdf
K/Sigma
Initial Crack Size
Crack Aspect Ratio
Geometry Data
Hole Dia.
Hole Offset
W
D
c
P

Motivation

- Typical run times w Monte Carlo (1B samples):
- 1) Master Curve:
  - 1 CG (30 sec), 1B interpolations->3 hrs on 8 processors



(only 3 random variables)

(N random variables)

- 2) Kriging :
  - 400 CG (1/2 hr), 1B interpolations-> 20 hrs on 8 processors
- 3) Standard Monte Carlo, 1B samples
  - General CG: 30s/run on 8 processors = 43K days = 118 yrs!
  - If internal CG code 1000x faster -> 43 days
  - If internal CG code 10,000x faster -> 4.3 days
  - If internal CG code 100,000x faster -> 0.43 days = 10 hrs

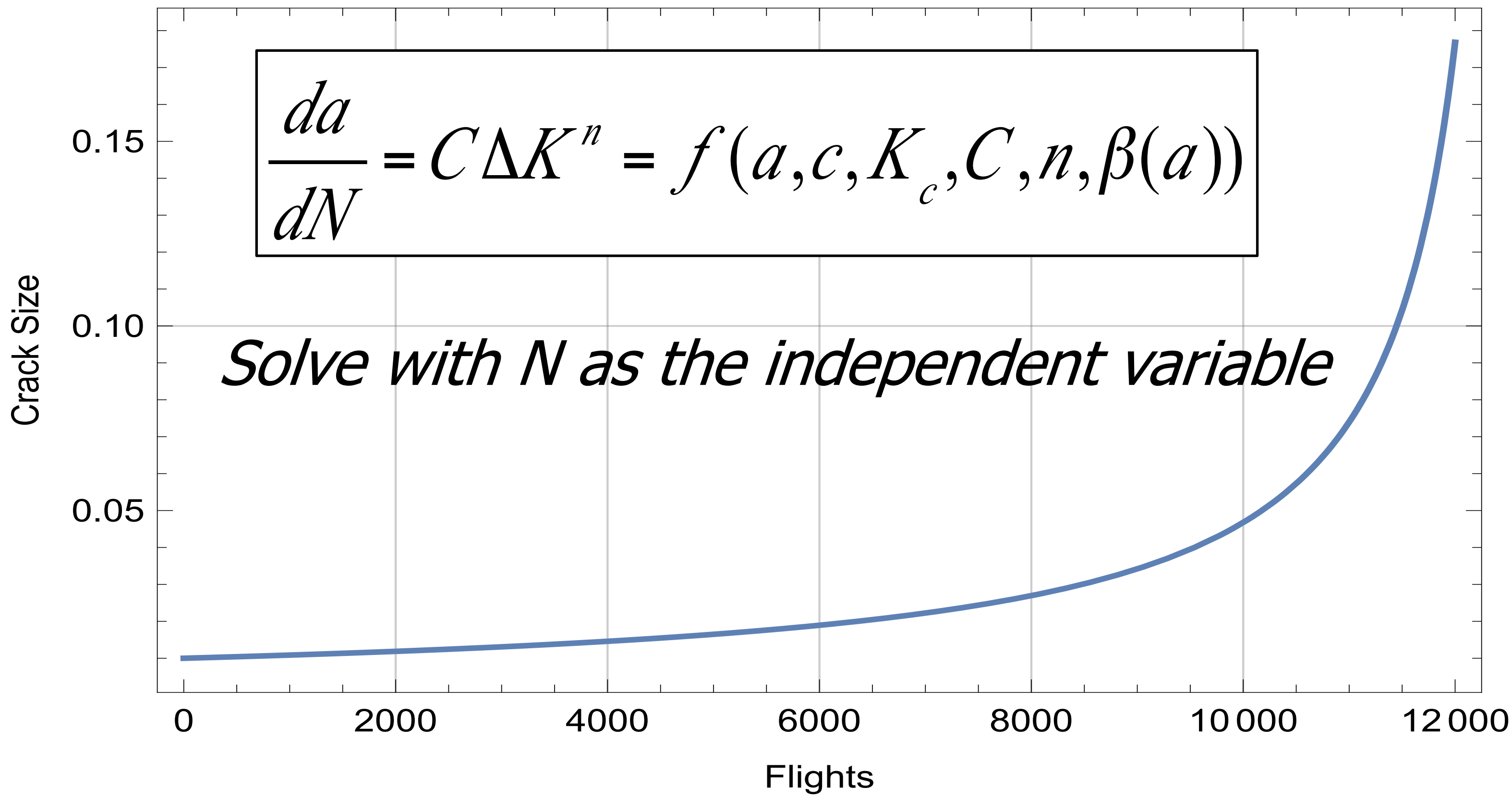
Minimum improvement

- 4) Numerical Integration
  - 100K CG -> 800 hrs on 1 processor
  - If internal CG code 1000x faster -> 0.8 hrs
- If internal CG code 1000x faster -> 0.8 hrs
- 5) Numerical Integration w Kriging
  - 400 ICG (2s), 100K interpolations-> 100s on 1 processor
- 6) Importance Sampling
  - Internal CG for optimization then 1K ICG -> 1 hr

w/o inspection

Components of CG Module

- ✓ Compute crack size vs. flights/cycle

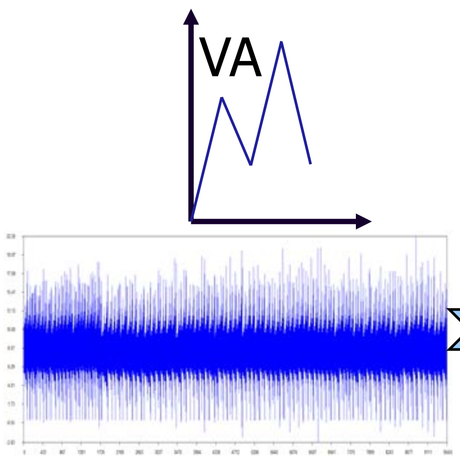


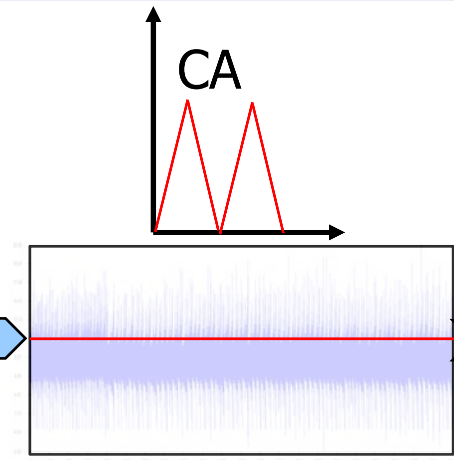
- ✓ Requires a crack growth integration routine (ODE solver)
- ✓ Requires K solutions or weight functions
- ✓ Net section yield calculations

Ultrafast Approach

1) Create an *equivalent constant amplitude* from an arbitrary spectrum
2) Use an internal *adaptive time stepping* Runge-Kutta algorithm to grow the crack (Cycles become the independent variable)
3) Collect the top 100 (or so) damaging realizations for further examination and potential reanalysis

Internal CG Code





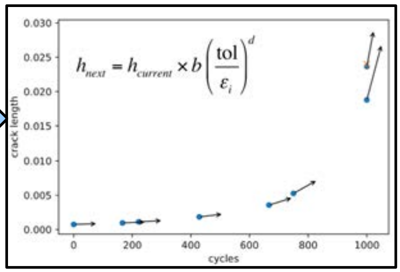
ODE Formulation

$$\frac{da}{dN} - C(\Delta K(a,c))^n = 0$$

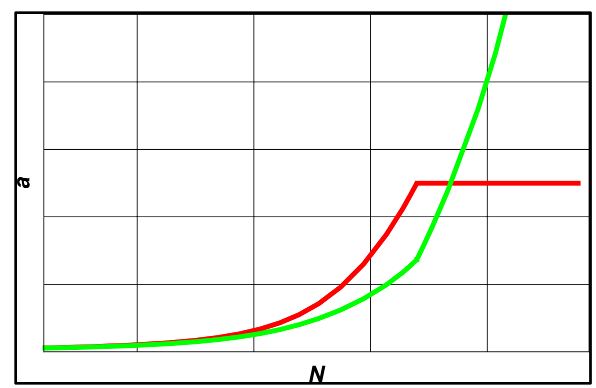
$$\frac{dc}{dN} - C(\Delta K(a,c))^n = 0$$

$$\textit{Initial Conditions}: a(0) = a_i, c(0) = c_i$$

RK ODE Solver

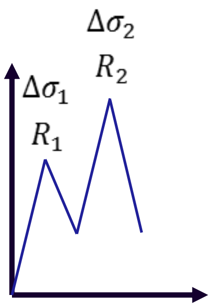


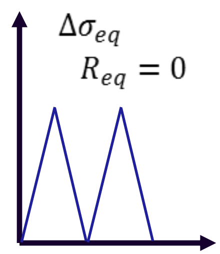
Crack Growth Result



ICG Capabilities	
Method	4-5th order Runge-Kutta
Accuracy	Error controlled by user tolerance
Speed	~7000/sec single proc.
Parallel	95% speedup on 8 proc.
K solutions	Newman-Raju, read beta tables

Equivalent Stress





$$N_{total} = N_1 + N_2 = \int_{a_0}^{a_1} \frac{1}{f(\Delta\sigma_1, R_1, a)} + \int_{a_1}^{a_2} \frac{1}{f(\Delta\sigma_2, R_2, a)}$$

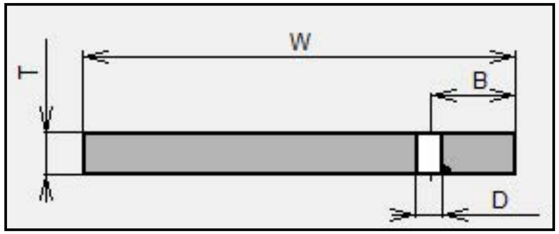
$$N_{total_{eq\_stress}} = \int_{a_0}^{a_2} \frac{1}{f(\Delta\sigma_{eq}, R_{eq}, a)}$$

$$\Delta\sigma_{eq} = \left[\sum_{i=1}^{K} \frac{n_i}{N_{Tot}} \left((1 - R_i)^{(m-1)n} \right) \Delta\sigma_i^n \right]^{1/n}$$

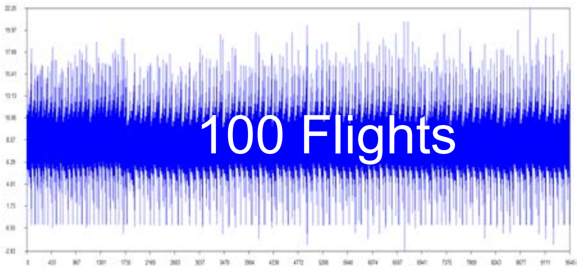
Equivalent Stress Transformation for Efficient Probabilistic Fatigue-Crack Growth Analysis under Variable Amplitude Loadings
Yibing Xiang and Yongming Liu

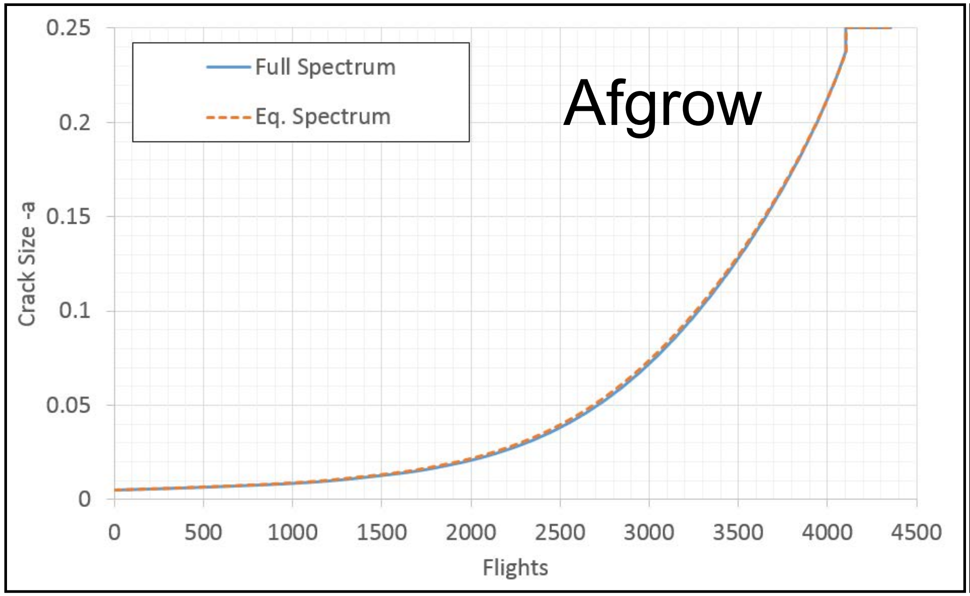
Eq. Stress Examples

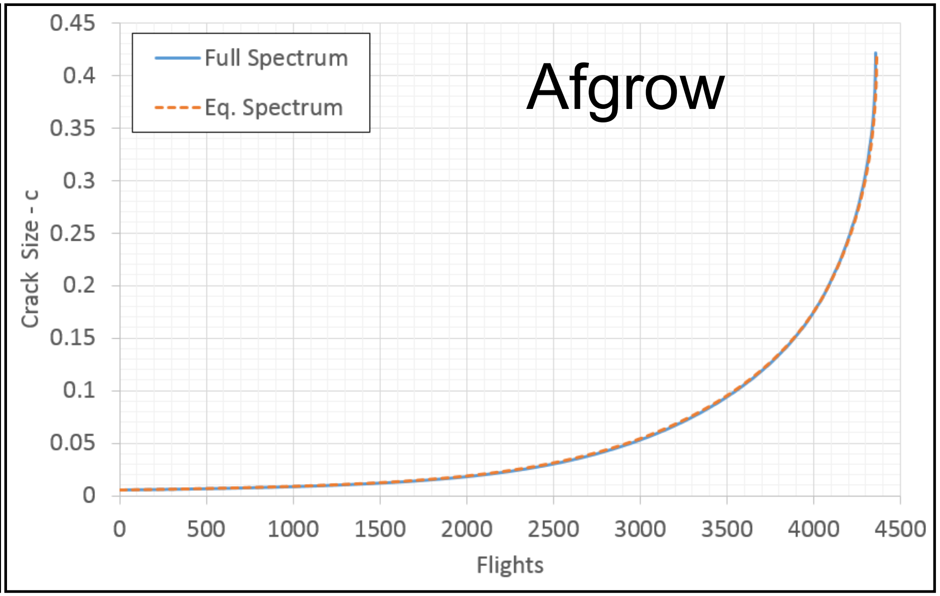
Corner Crack in a Hole



Variable	Value
Width	4 in.
Hole Offset	0.5
Thickness	0.25 in.
Hole Size	0.156 in.
Eq. spectrum	10.06 KSI
C	1.0E-09
Paris_m	3.8
Walker_m	0.5
$a_i = c_i$	0.005 in

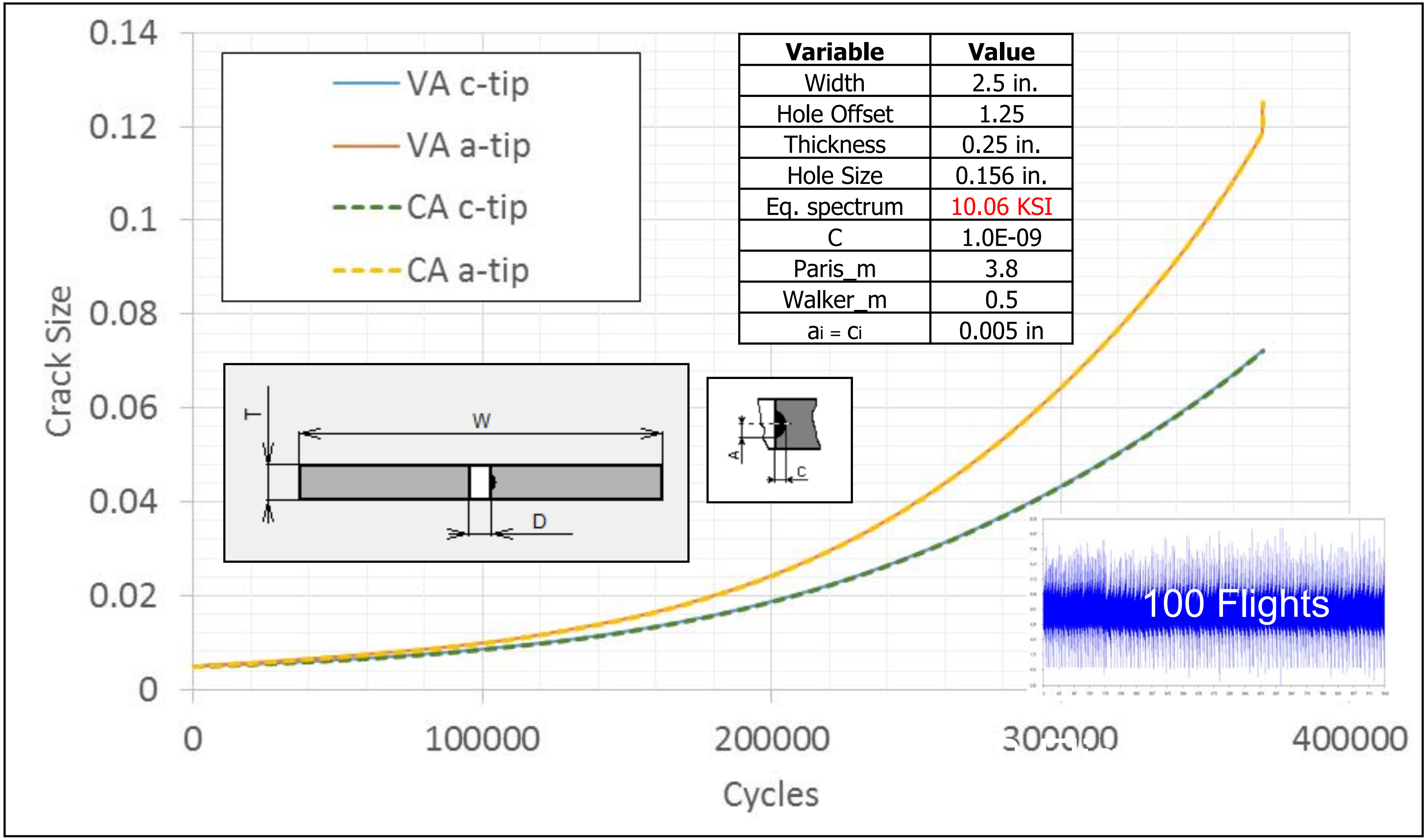






Eq. Stress Examples

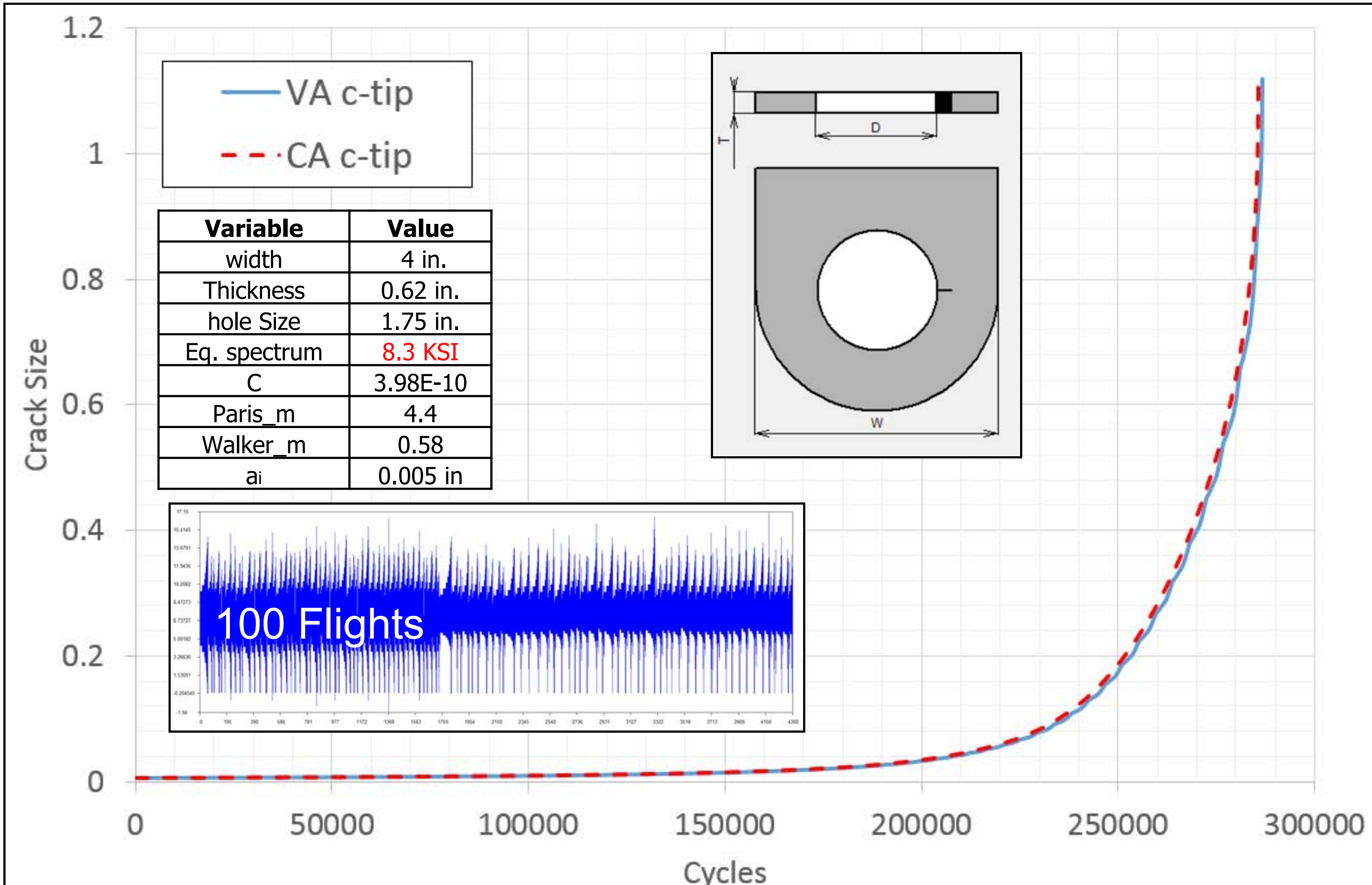
Surface Crack in a Hole



Variable	Value
Width	2.5 in.
Hole Offset	1.25
Thickness	0.25 in.
Hole Size	0.156 in.
Eq. spectrum	10.06 KSI
C	1.0E-09
Paris_m	3.8
Walker_m	0.5
$a_i = c_i$	0.005 in

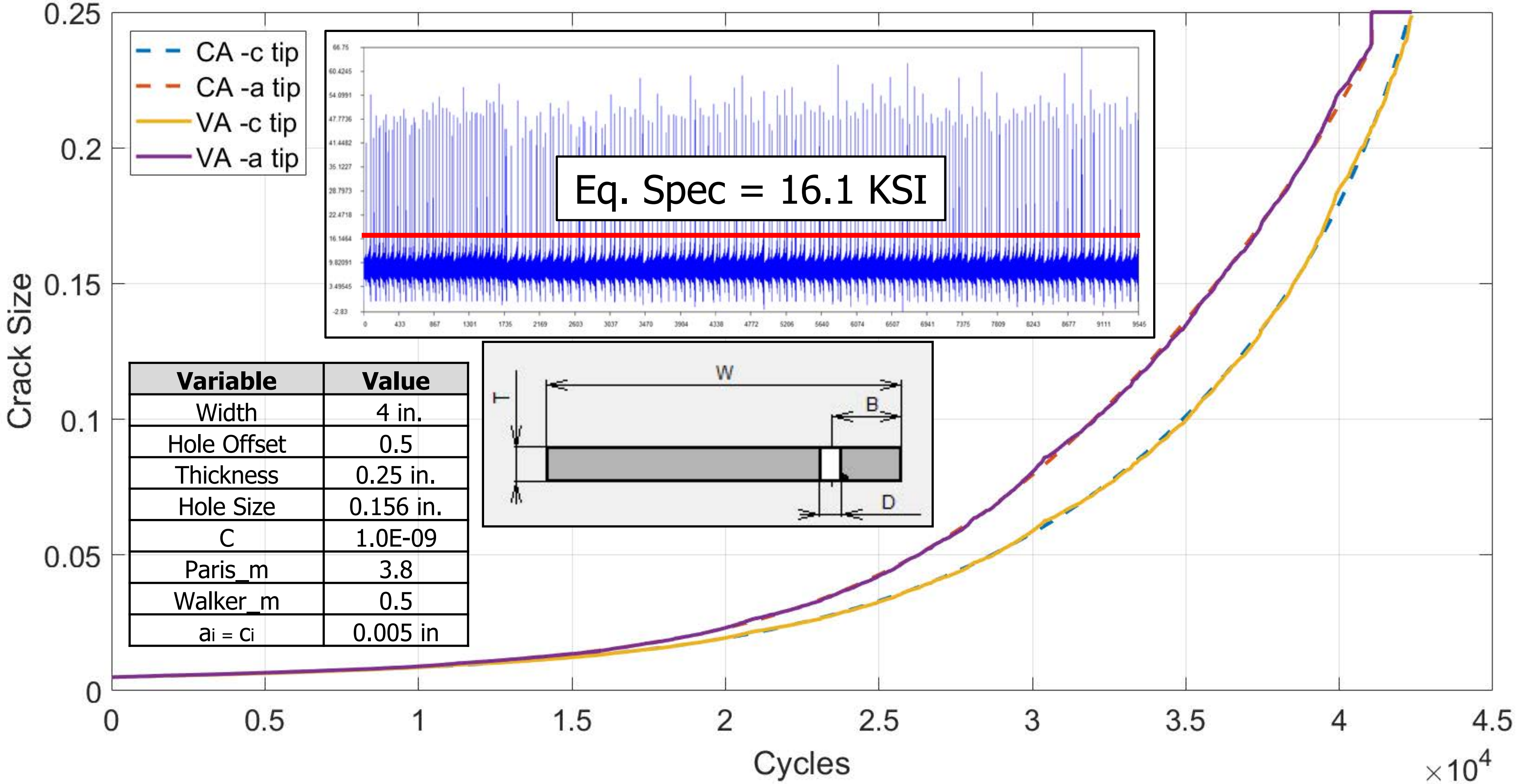
Eq. Stress Examples

Through Crack in a Lug



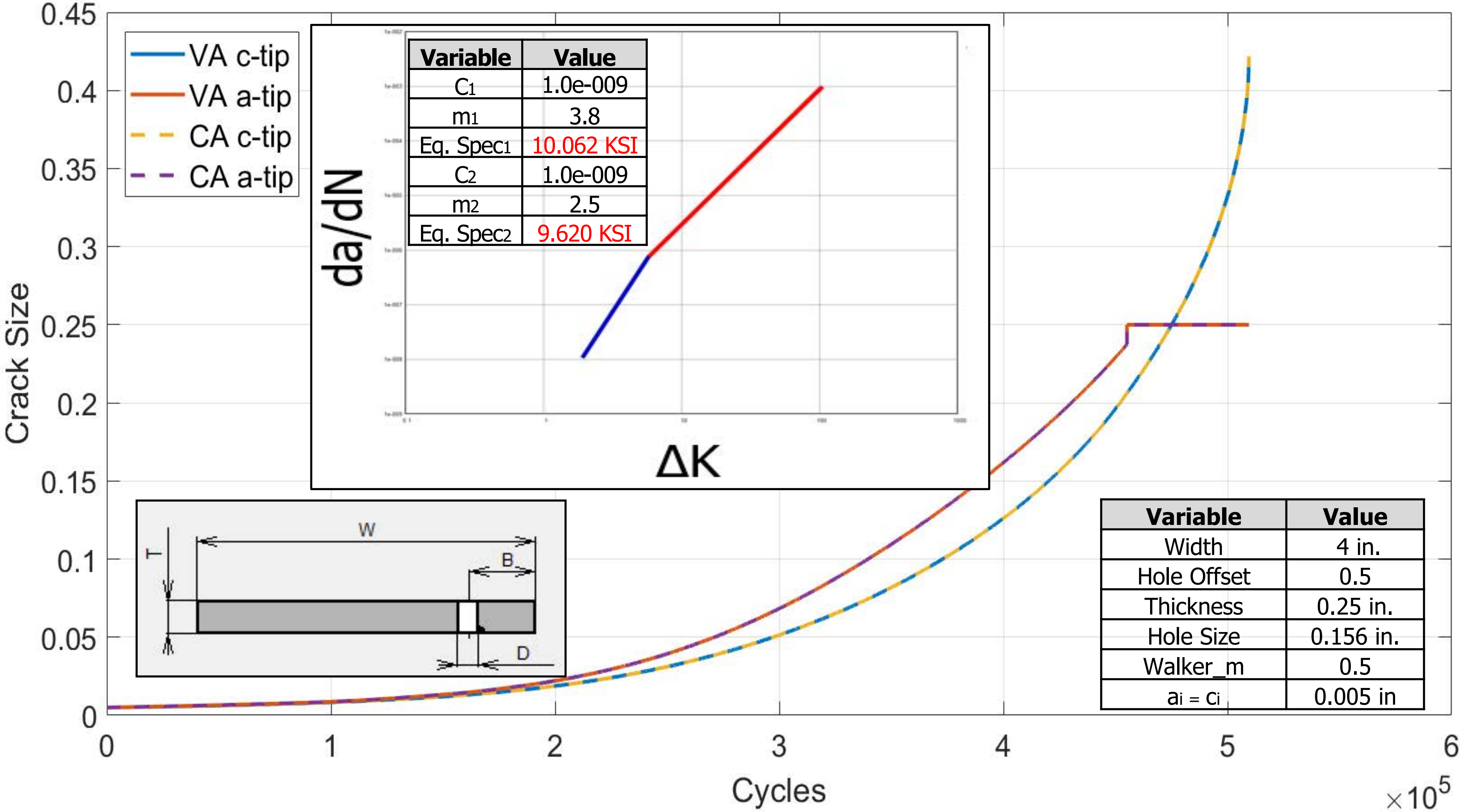
Variable	Value
width	4 in.
Thickness	0.62 in.
hole Size	1.75 in.
Eq. spectrum	8.3 KSI
C	3.98E-10
Paris_m	4.4
Walker_m	0.58
a_i	0.005 in

Over Load Example



Variable	Value
Width	4 in.
Hole Offset	0.5
Thickness	0.25 in.
Hole Size	0.156 in.
C	1.0E-09
Paris_m	3.8
Walker_m	0.5
$a_i = c_i$	0.005 in

Bilinear Paris Example



Variable	Value
C_1	1.0e-009
m_1	3.8
Eq. $Spec_1$	10.062 KSI
C_2	1.0e-009
m_2	2.5
Eq. $Spec_2$	9.620 KSI

Variable	Value
Width	4 in.
Hole Offset	0.5
Thickness	0.25 in.
Hole Size	0.156 in.
Walker_m	0.5
$a_i = c_i$	0.005 in

Sigmoidal Crack Growth Law

- The equivalent stress is a function of the crack growth rate. Incorporate this relationship within the ODE solver.

$$\Delta\sigma_{eq}(n, a(N), c(N))$$

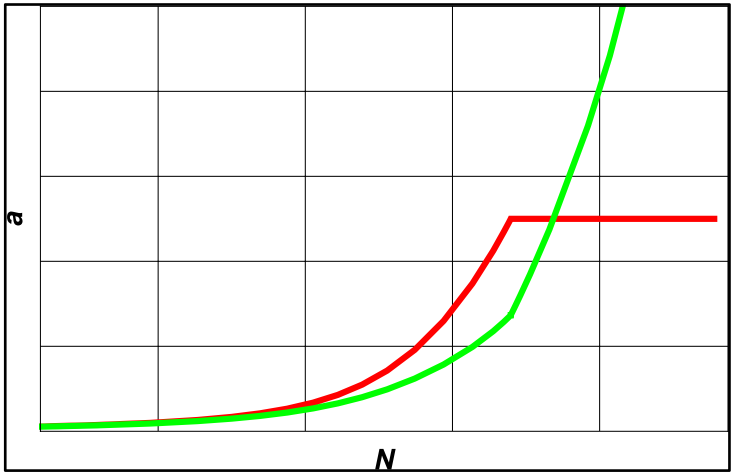
$$\frac{da}{dN} = C(\Delta K(\Delta\sigma_{eq}, a, c))^{n} = 0$$

$$\frac{dc}{dN} = C(\Delta K(\Delta\sigma_{eq}, a, c))^{n} = 0$$

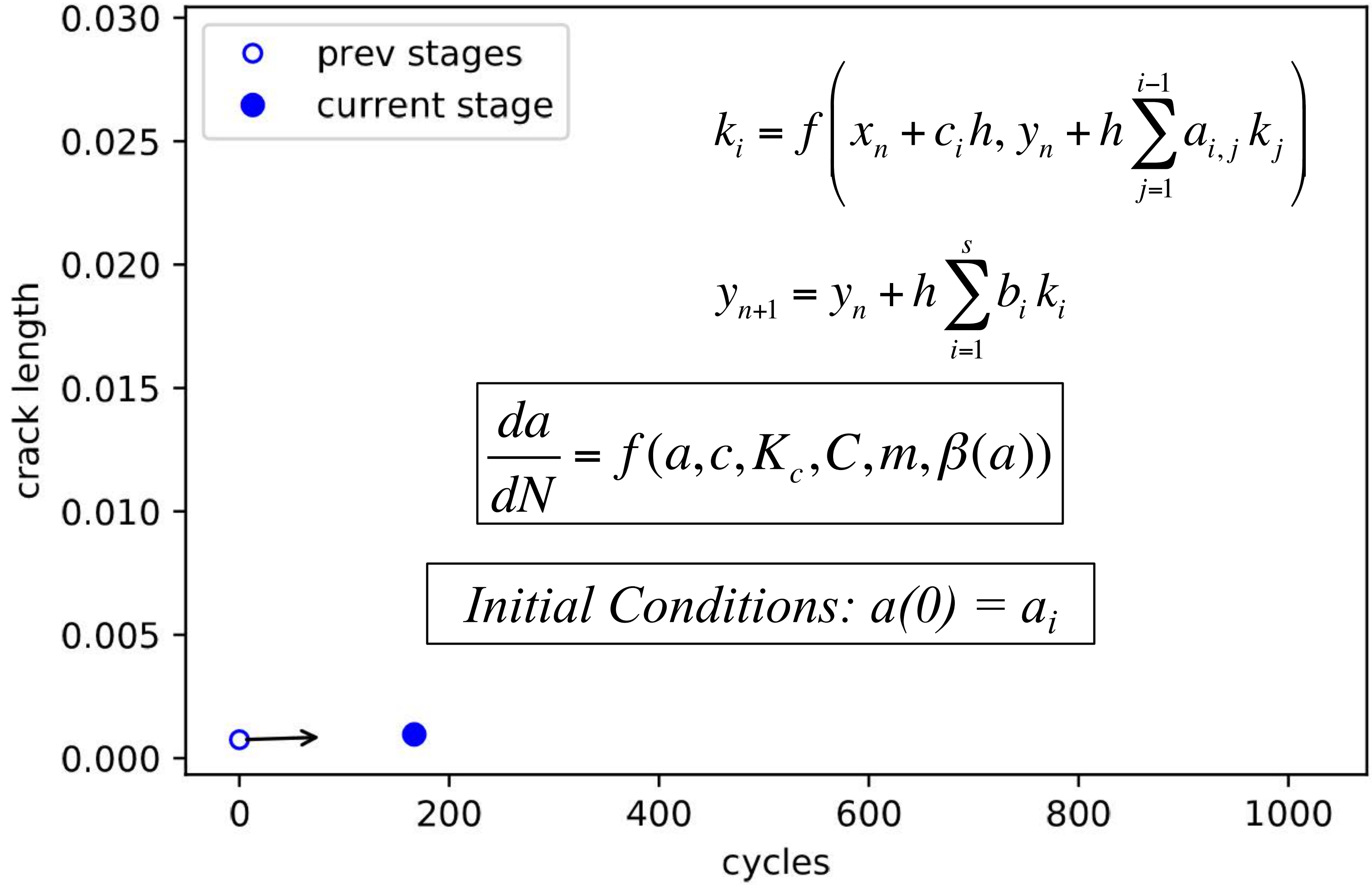
$$Initial\ \ Conditions : a(0) = a_i, c(0) = c_i$$

Fast ODE Solver

- Based on best practices from well known and available ODE solvers, e.g., Petsc, Sundials, RKSuite
- Paired Runge-Kutta implementations, 2(3), 4(5), 7(8), e.g., 4th and 5th order solutions computed simultaneously. Gives high quality error estimate.
- Automatically selects step size based on user input and error estimate. Produces large steps early in the life, smaller steps later.



Adaptive Step Size Control



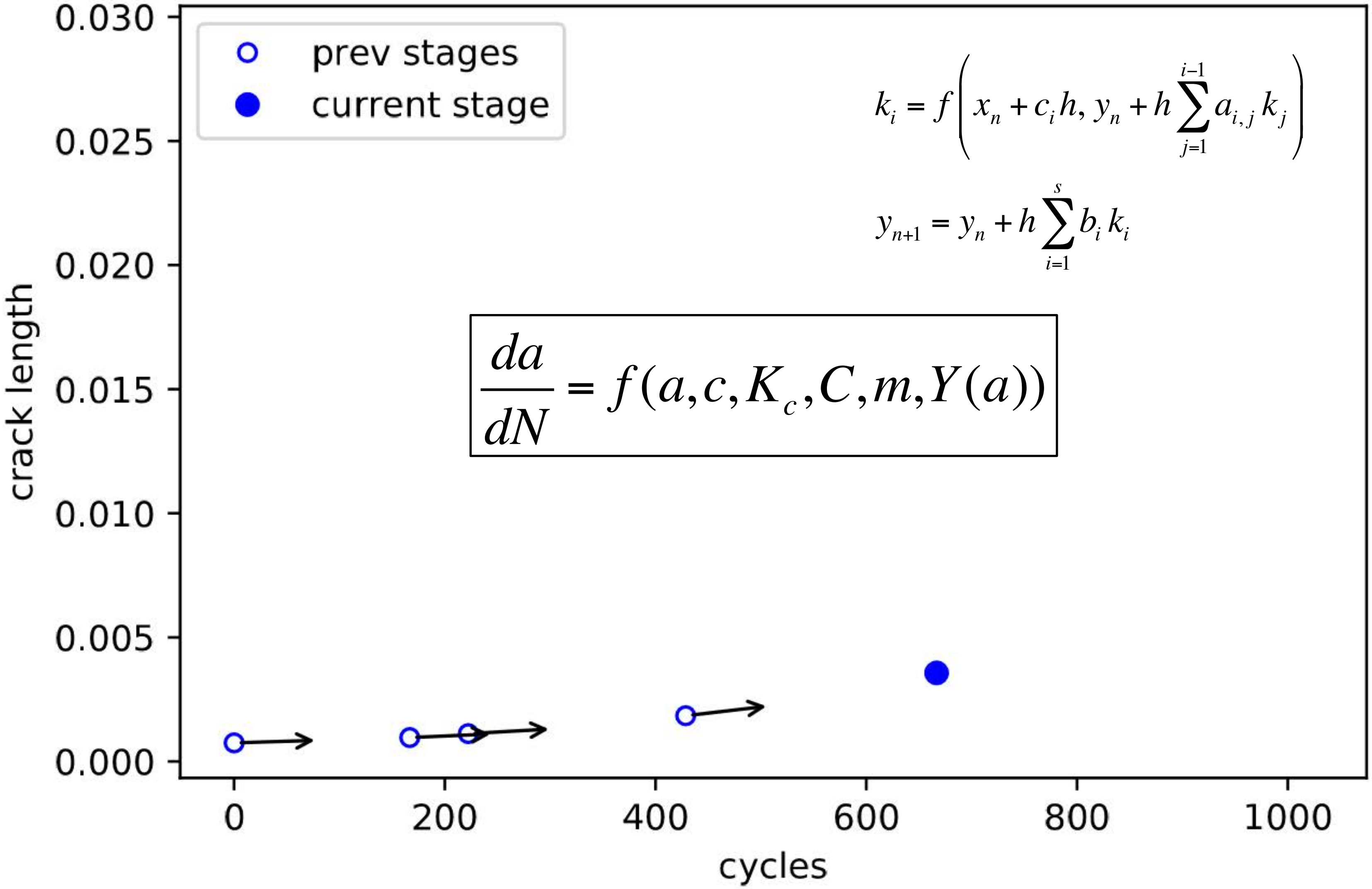
$$k_i = f\left(x_n + c_i h,\ y_n + h\sum_{j=1}^{i-1} a_{i,j} k_j\right)$$

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i$$

$$\frac{da}{dN} = f(a, c, K_c, C, m, \beta(a))$$

Initial Conditions: $a(0) = a_i$

Adaptive Step Size Control

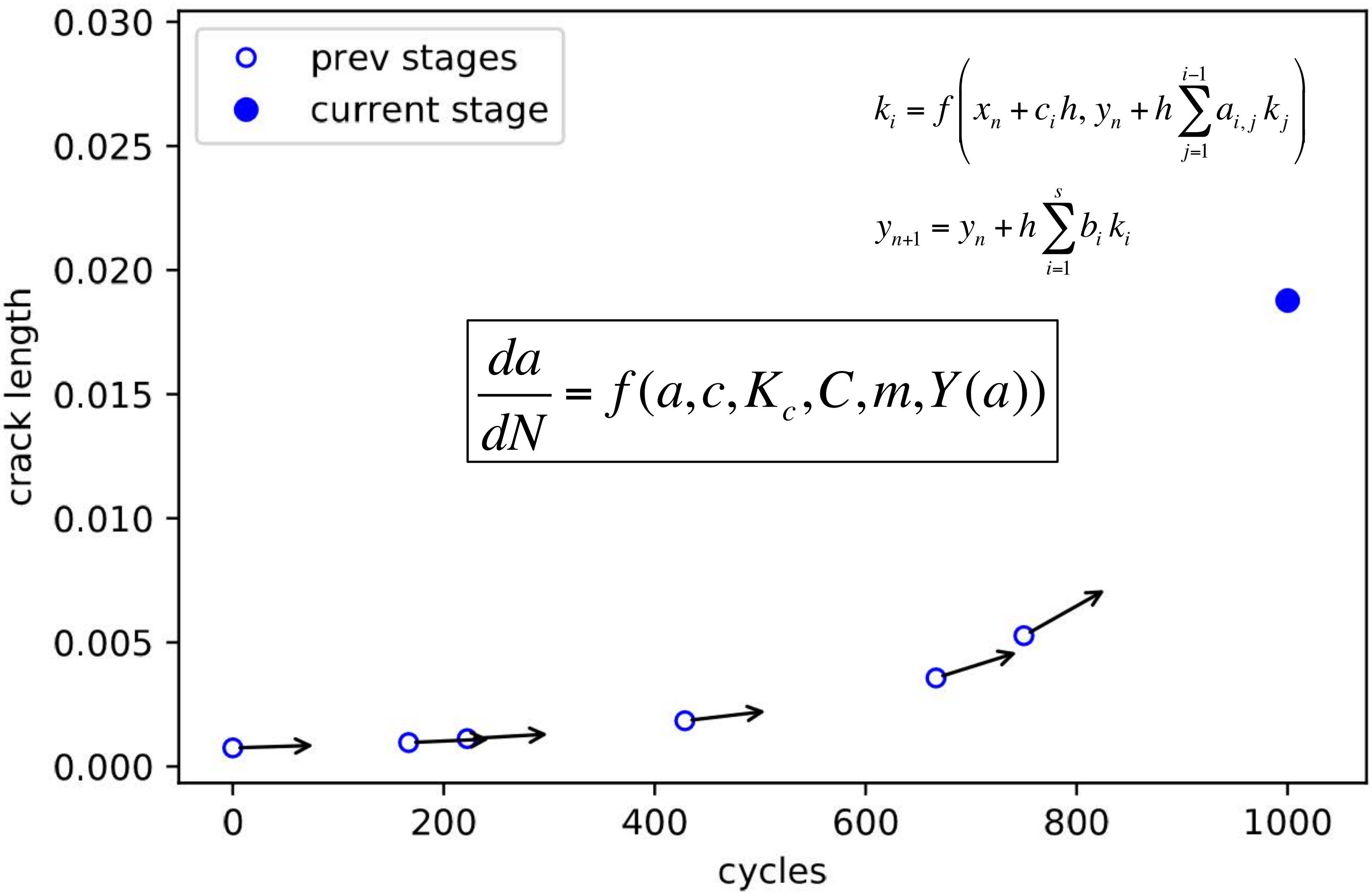


$$k_i = f\left(x_n + c_i h,\ y_n + h \sum_{j=1}^{i-1} a_{i,j} k_j \right)$$

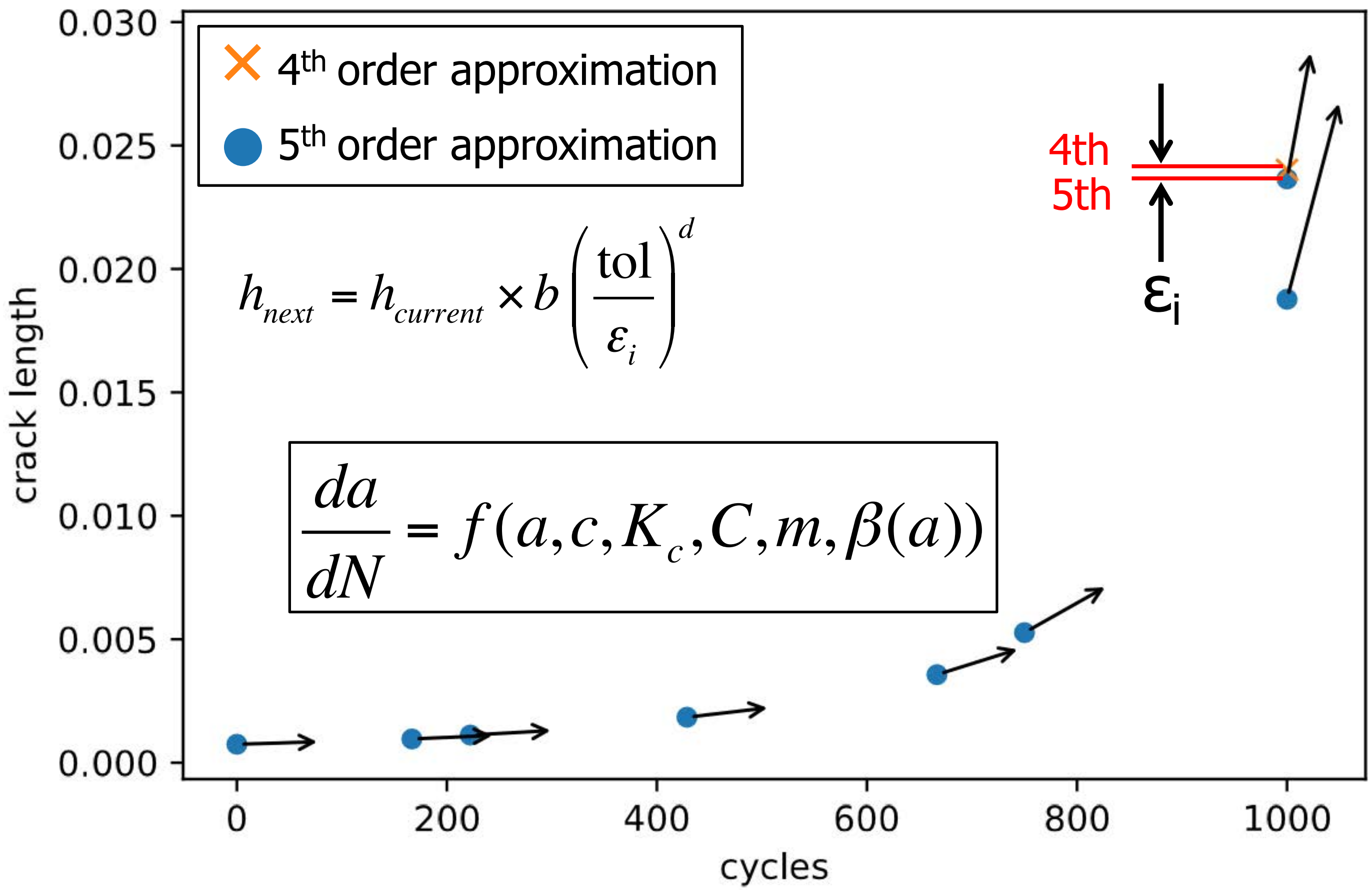
$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i$$

$$\frac{da}{dN} = f(a, c, K_c, C, m, Y(a))$$

Adaptive Step Size Control



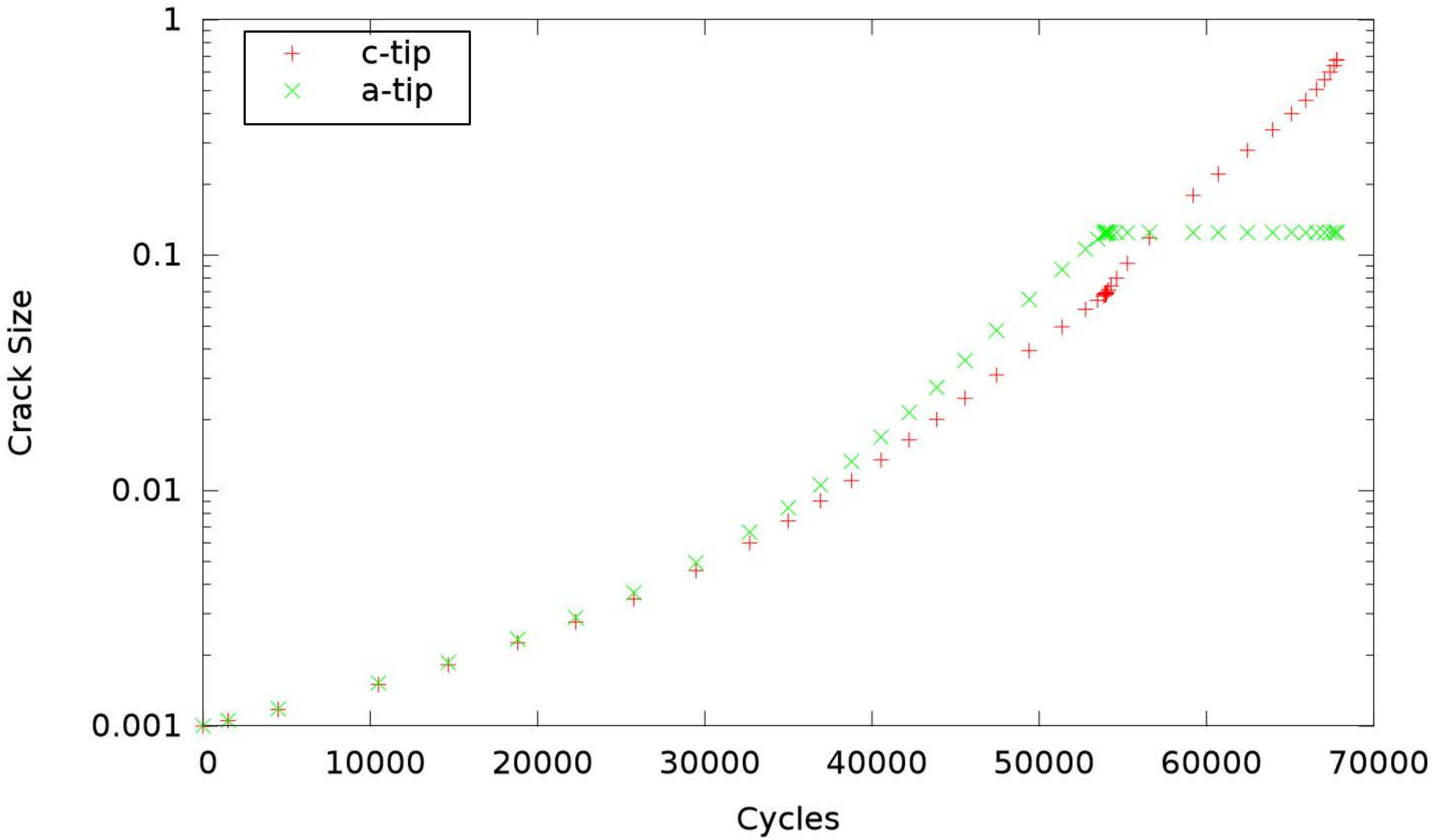
Adaptive Step Size Control



- ε_i is the absolute value of the difference between 5^{th} and 4^{th} order evaluations of the crack size
- Constants b and d determined empirically by the authors
- Step size is increased or decreased depending on the ratio of the user–defined tolerance to the error

Adaptive Step Size Control

Variable step sizes - corner crack integration

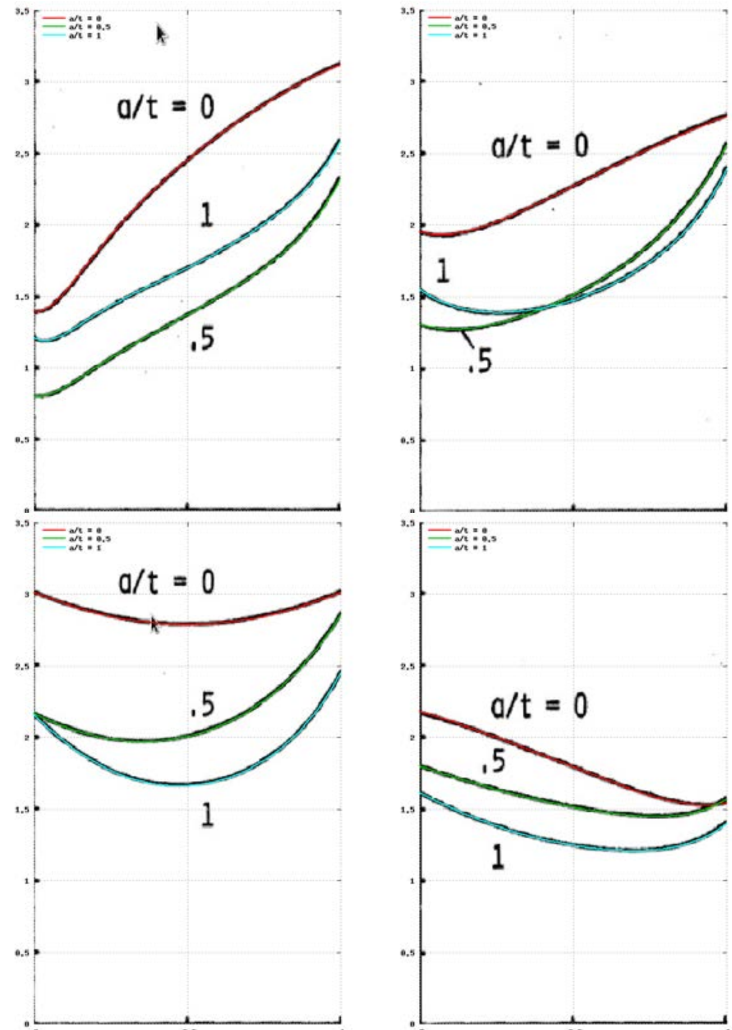


Internal K-Solutions

	Plate	Hole
Thru		
Corner (Newman-Raju)		
Surface (Newman-Raju)		

- Tension Loading only, bending / pin loading not implemented yet
- Centered Hole only
- Weight functions not implemented

Newman-Raju



Beta Tables

! Thru crack betas

c_1	β_1
c_2	β_1
…	…
c_N	β_1

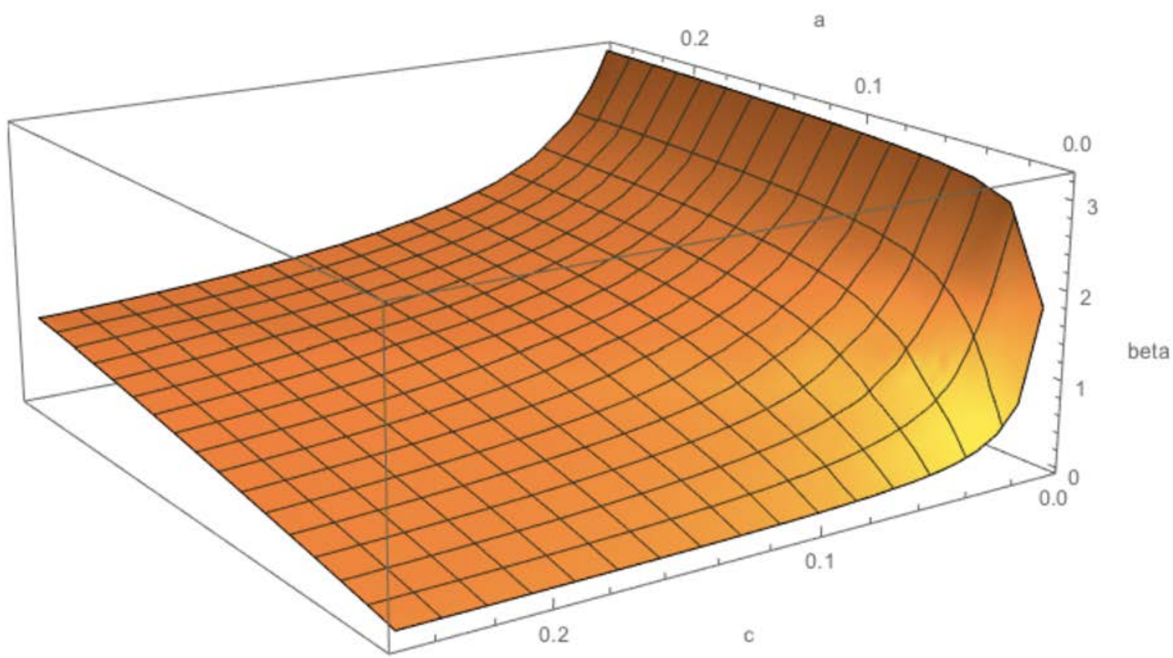
! C-tip direction

	a_1	a_2	…	a_N
c_1	β_{11}	β_{12}	…	β_{1N}
c_2	β_{21}	β_{22}	…	β_{2N}
…	…	…	…	…
c_N	β_{N1}	β_{N2}	…	β_{NN}

! A-tip direction

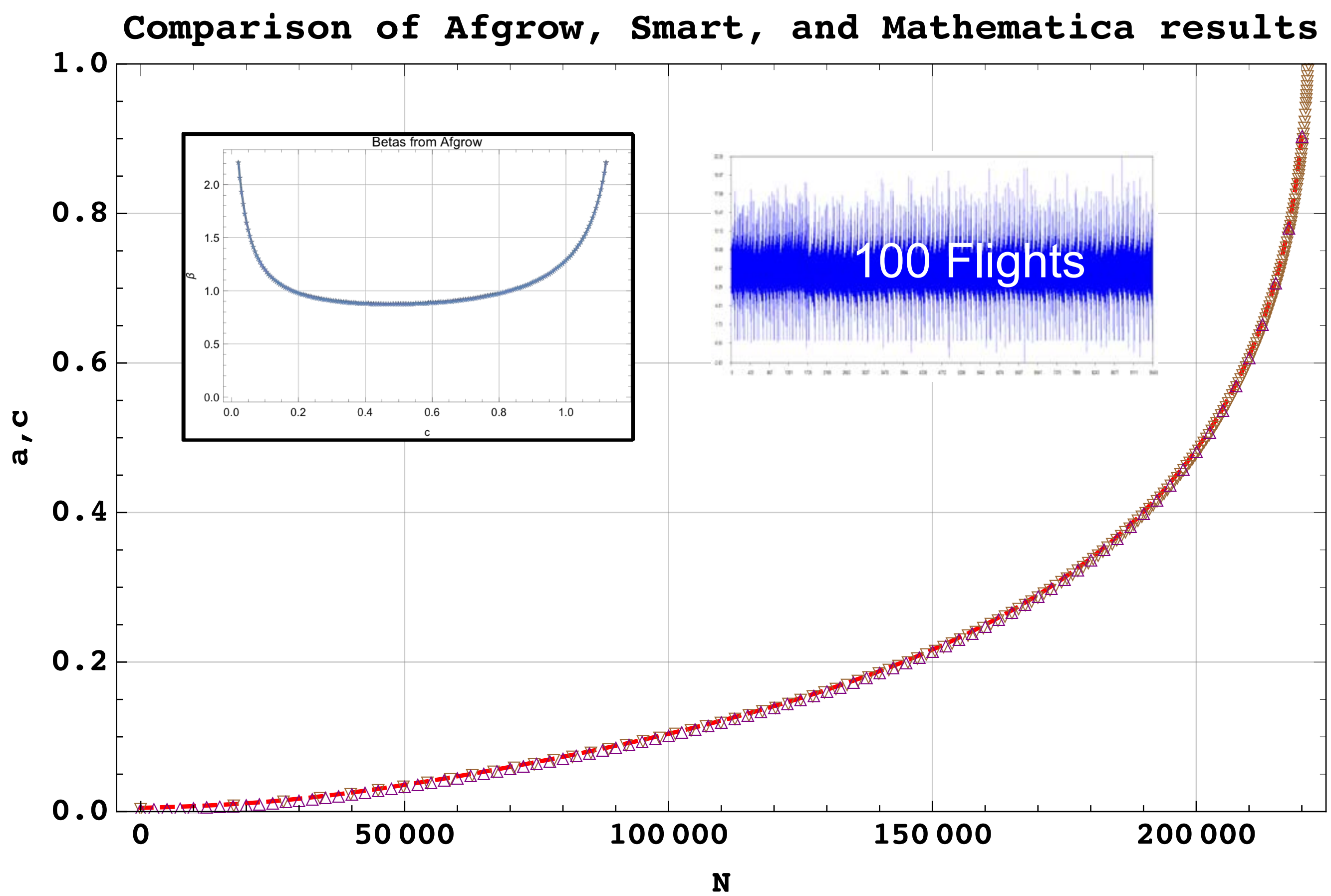
	a_1	a_2	…	a_N
c_1	β_{11}	β_{12}	…	β_{1N}
c_2	β_{21}	β_{22}	…	β_{2N}
…	…	…	…	…
c_N	β_{N1}	β_{N2}	…	β_{NN}

- Use Afgrow /Nasgro/other to generate beta tables for any K solution. ICG reads the table and interpolates to get betas.
- Allows ICG to solve any crack model



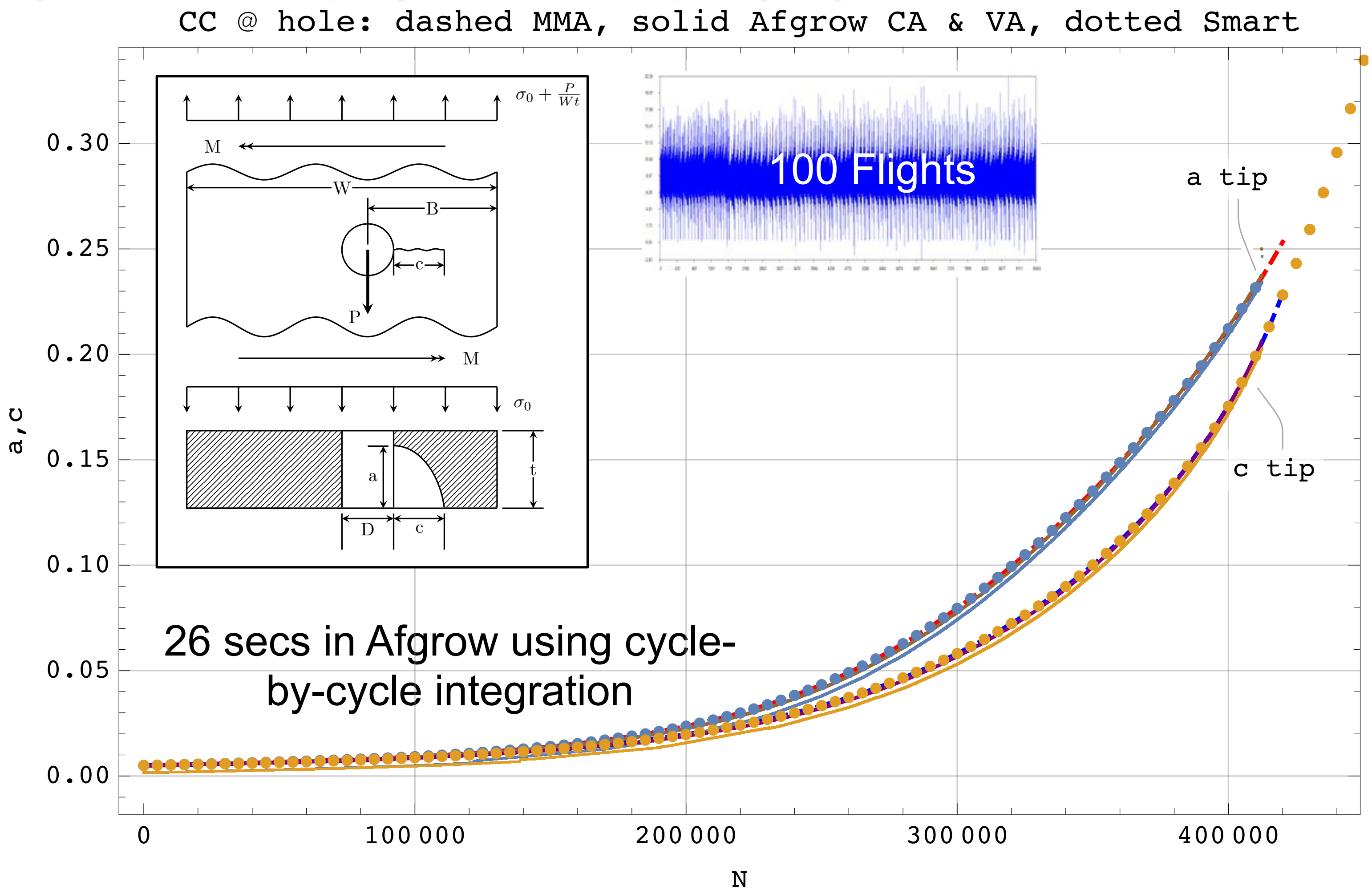
Through Crack at Hole
(Tension)

- Cparis = 10^{-9}, nparis = 3.8, Eq. Spectrum =10.062 ksi



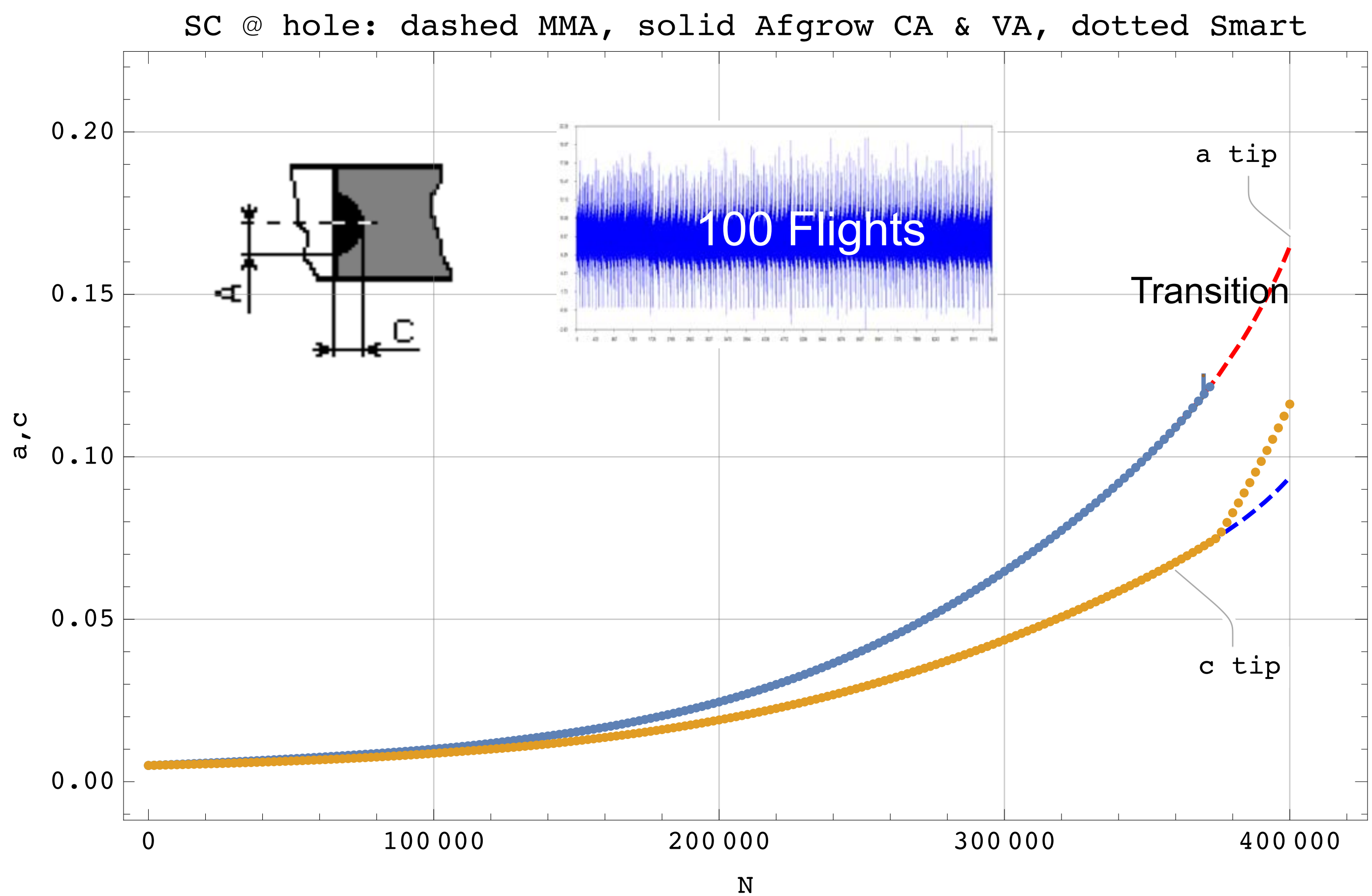
Corner Crack at Hole
(Tension)

- Cparis = 10^{-9}, nparis = 3.8, Eq. spectrum =10.062 ksi



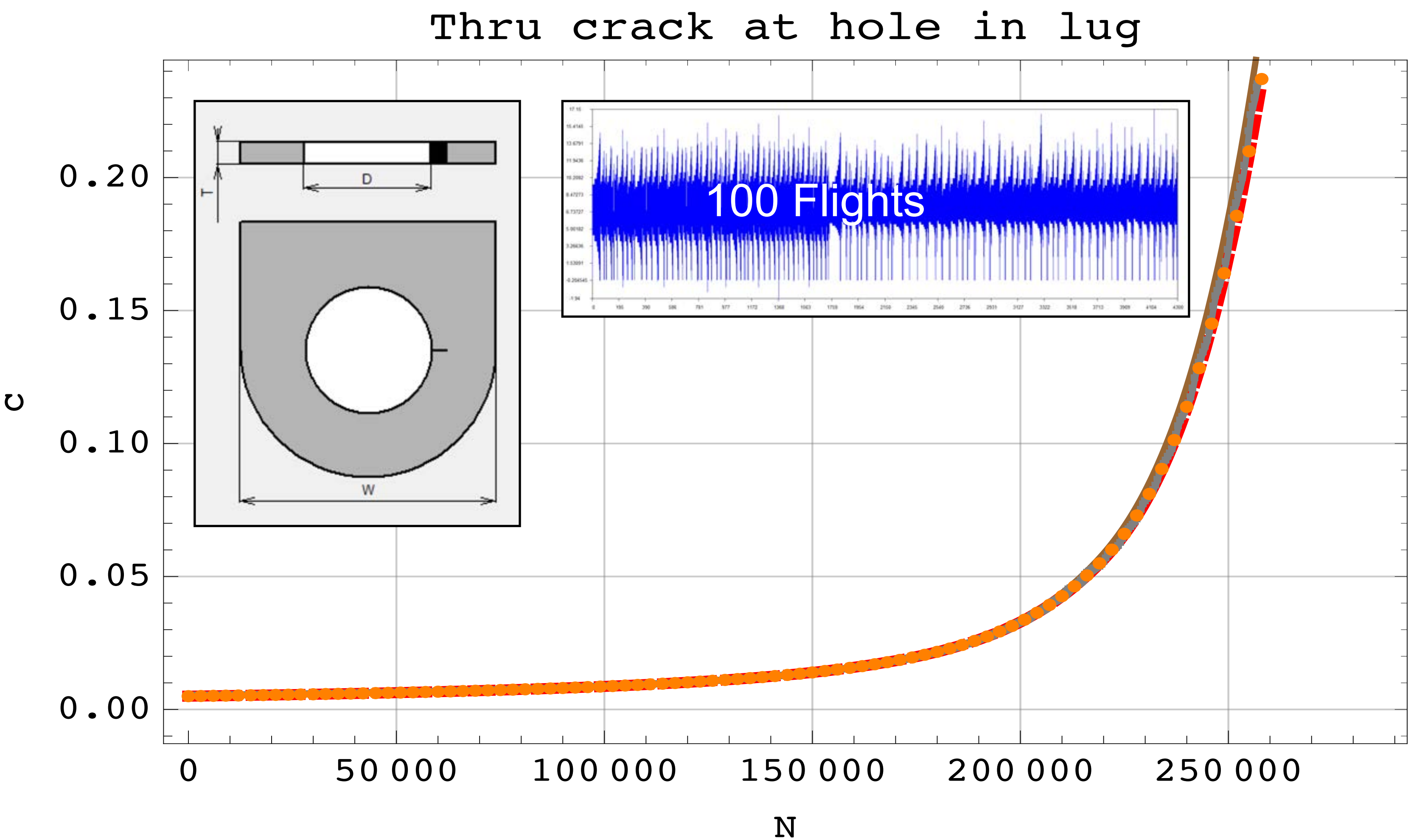
Surface Crack at Hole
(Tension)

- Cparis = 10^{-9}, nparis = 3.8, Eq. Spectrum =10.062 ksi

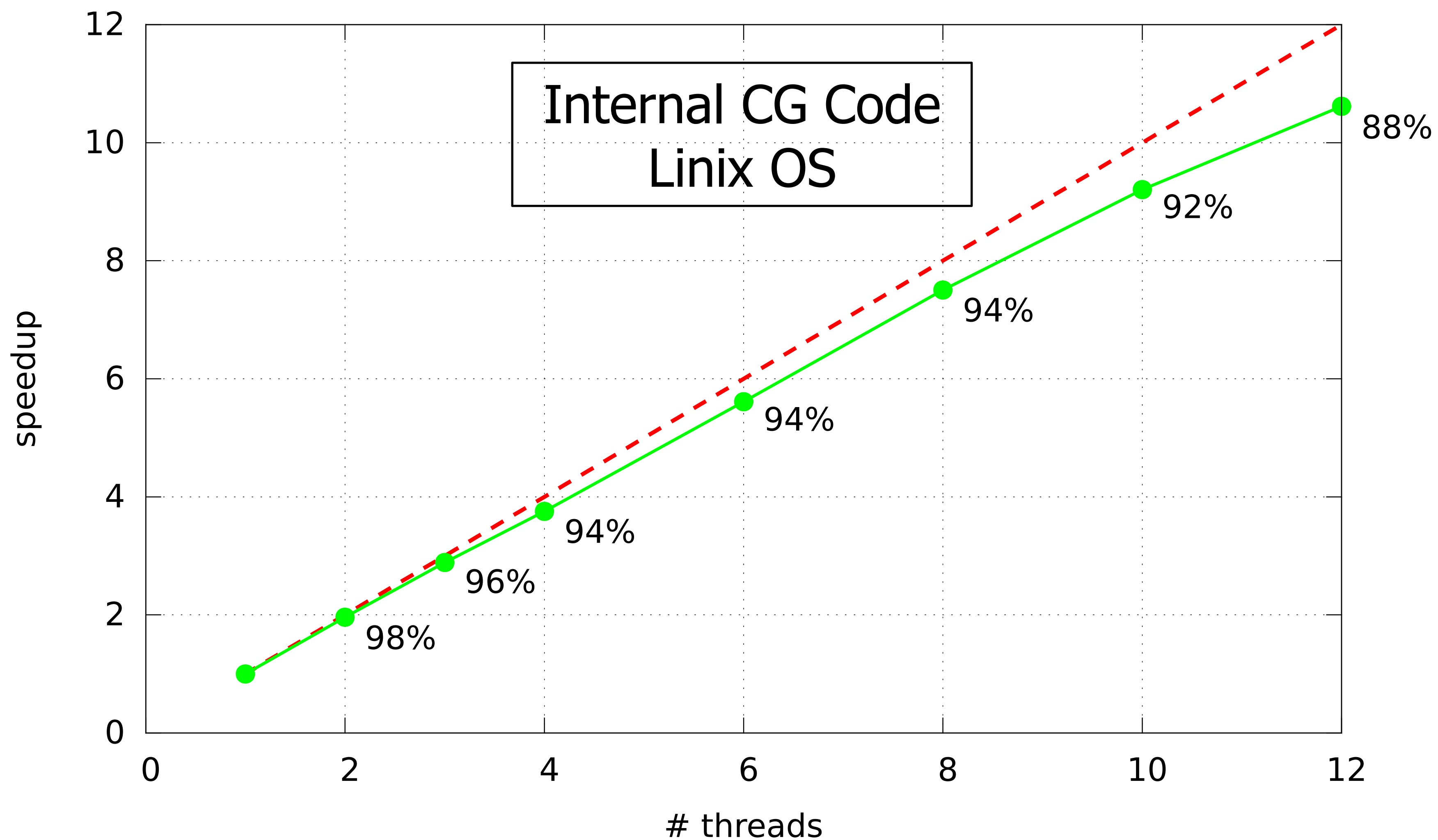


Thru Crack at Lug
(Tension)

- Cparis = 10^{-9}, nparis = 3.8, Eq. Spectrum =8.3 ksi



Parallel & Vectorized

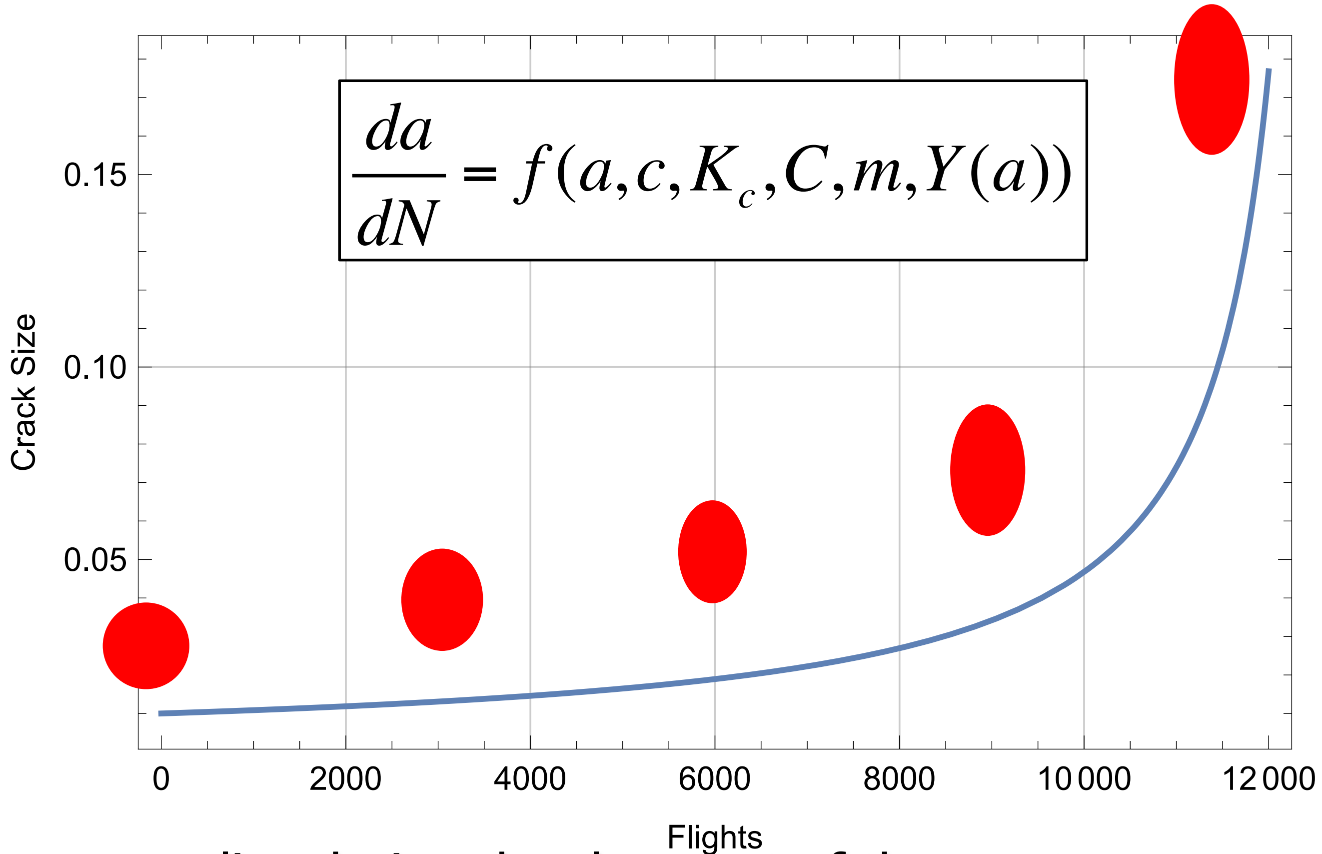


Compute Times

# samples/sec	# processors	Windows (s)[1]	Linux (s)[2]
Master Curve	1	55,000	51,000
Internal CG	1	3800/7500	3200/6500
Master Curve	8	412,000	380,000
Internal CG	8	28,500/57,000	24,000/50,000

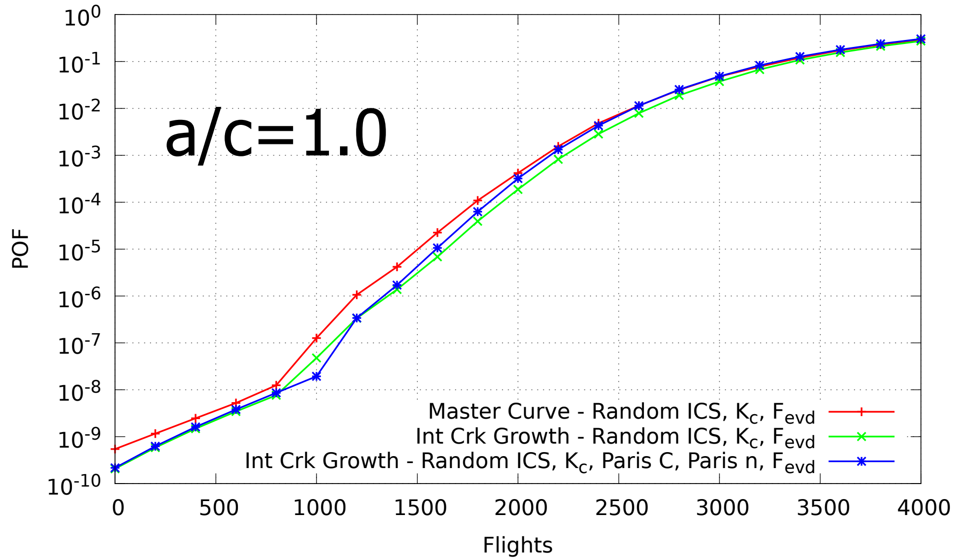
[1]2.8 GHz Intel core 7, 16 Gb Ram
[2]3.5 GHz Intel Xeon, 64 Gb ram
4-5 rule, 10^{-6} / 10^{-4} relative error

Master Curve Limitations

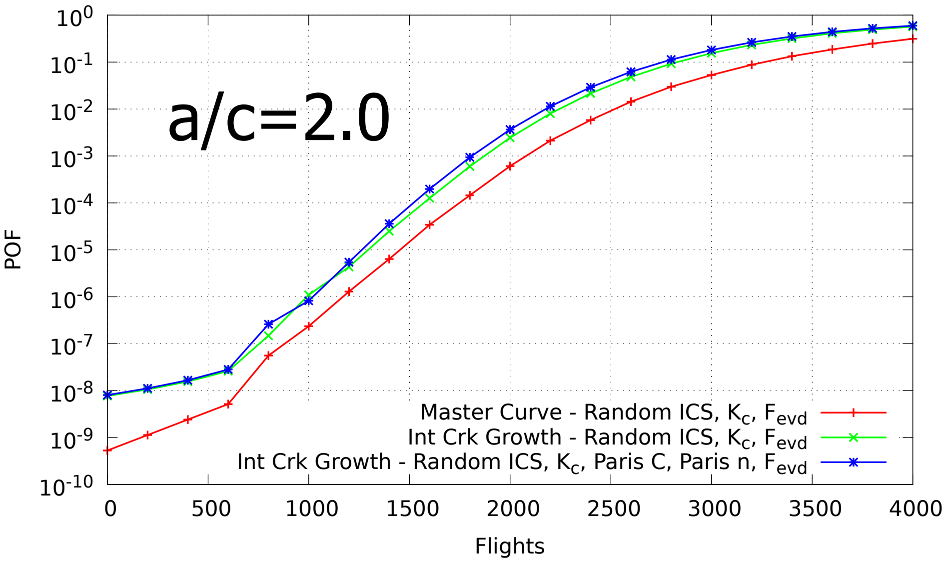


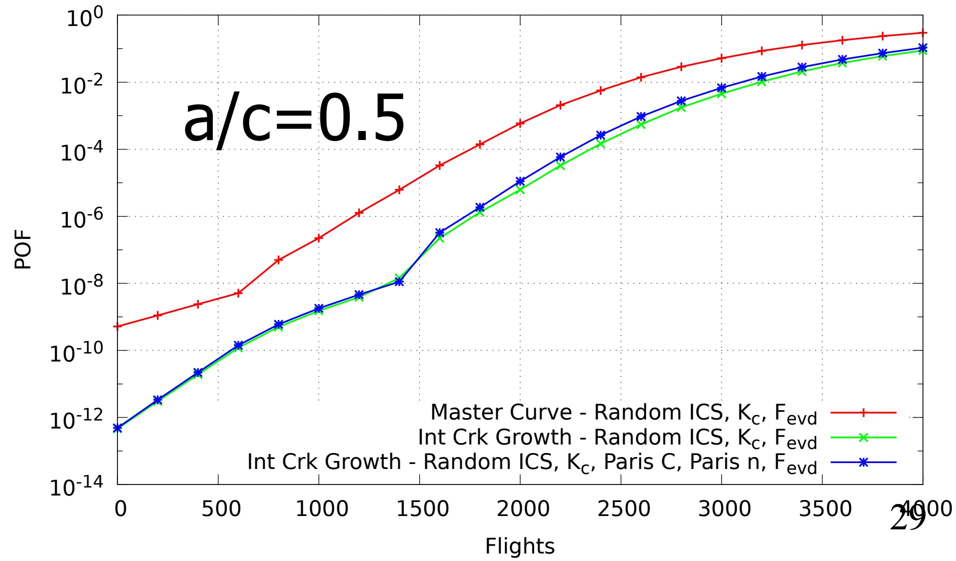
- ✓ Crack may ovalize during development of the master curve.
- ✓ This ovalization is ignored during the probabilistic analysis.
- ✓ This may or may not be conservative.

Risk Calculations
Master Curve Ovalization



Master Curve crack ovalizes during growth





Crack Growth Capabilities

	AFGROW	NASGRO	ICG
Create avsn	Y	Y	Y
MCS	Y	Y	Y
NI	Y	Y	Y
Kriging	Y	Y	coming
RUL	Y	Y	Y
K solutions	Comprehensive	Comprehensive	Newman-Raju Read Beta tables
Weight functions	Comprehensive	Comprehensive	N (maybe)
Net section yield	Y	Y	coming
Retardation	Y	Y	N
Adaptive error control	% Δa	% Δa	Adaptive based on RK
Parallel capable	N	Y	Y (multi-threaded)

Ultrafast Approach Conclusions

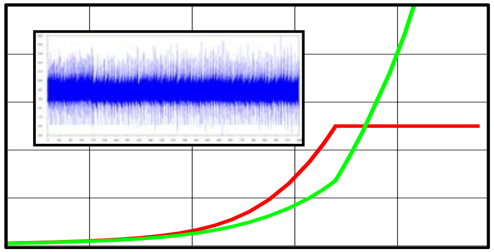
1) Equivalent constant amplitude is accurate at predicting variable amplitude crack growth – *for all problems to date.*
2) Adaptive RK algorithm to grow the crack is very effective (~7000 evaluations/sec/proc)
   - Capability to read beta tables provides an attractive method to incorporate a variety of crack models.
3) The top 100 (or so) damaging realizations can be further examined for potential reanalysis

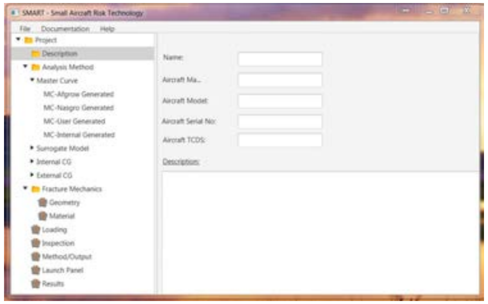
Future Work

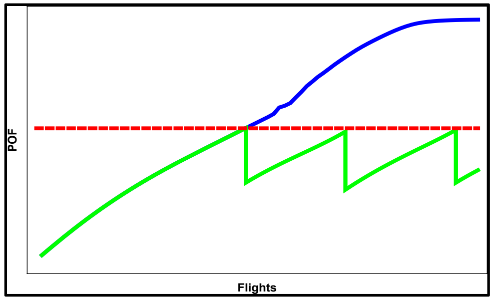
- Verify using more geometries and a larger variety of spectra. **Open to suggestions.**
- Compute beta tables on-the-fly with Afgrow & Nasgro.
- Build library of highly-used beta tables to include with the software.
- Expand the equivalent stress method to work with varying crack growth laws, e.g., bilinear Paris, Nasgro equation, and tabular da/dN input.

SMART|DT Current Development Activities

- Ultrafast crack growth code
- Probabilistic data base
  - (EIFS, POD, Kc, da/DN, etc.)
- MPI version for clusters
- New Java-based GUI
- Risk based inspections
- Importance Sampling
- Fleet management







Acknowledgements

- Probabilistic Fatigue Management Program for General Aviation, Federal Aviation Administration, Grant 12-G-012
  - Sohrob Mattaghi (FAA Tech Center) – Program Manager
  - Michael Reyer (Kansas City) - Sponsor