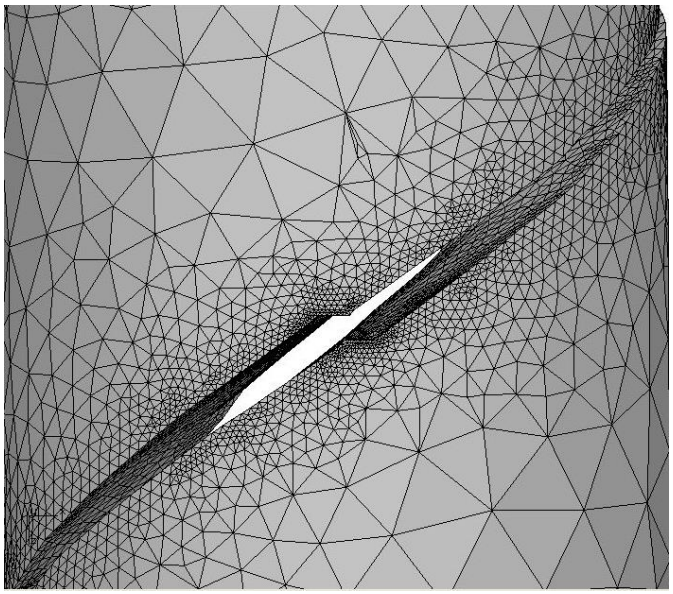
FRANC3D

Command Language & Python Extensions



Version 7

Fracture Analysis Consultants, Inc
www.fracanalysis.com

Revised: August 2016

Table of Contents:

1. Introduction

This document describes the FRANC3D command language and the Python extensions. The PyF3D module works with Python versions 2.7.6 - 2.7.10 (and probably 2.7.11). If your system does not have one of these versions, you can download and build Python yourself (see www.python.org).

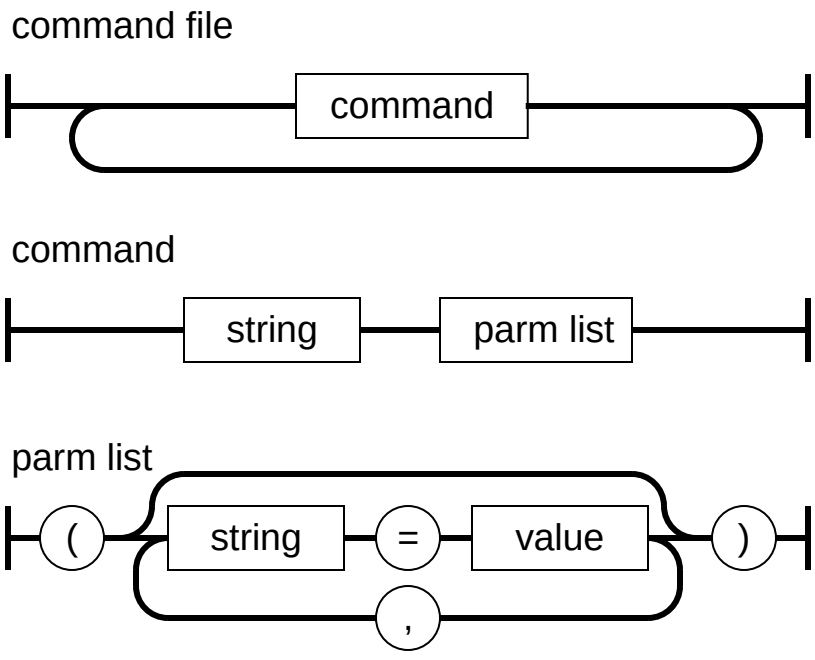
2. Command File Syntax

When running the FRANC3D using the standard GUI menu and dialogs, a session file is saved, which contains the commands that were executed through the GUI. A user can playback these commands to reproduce their actions or edit the file to execute different or modified commands without using the GUI.

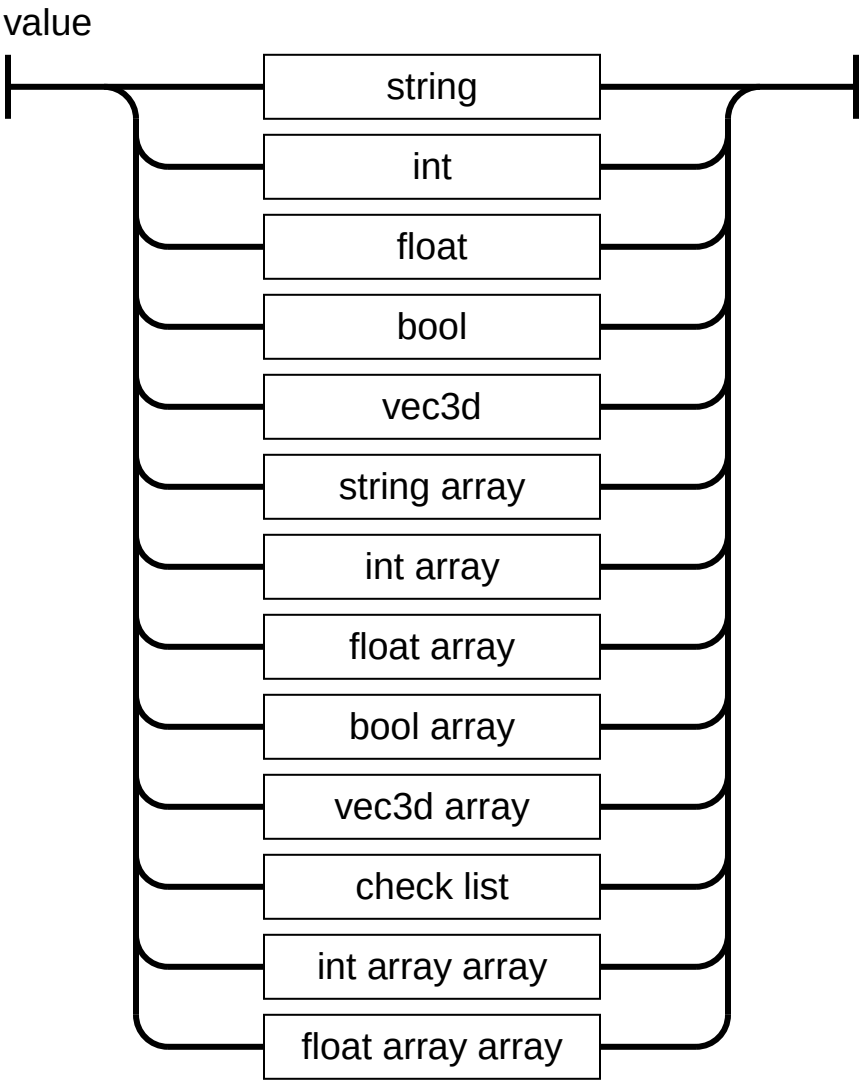
These session files contain a series of command statements. In command files, lines starting with '#' are comment lines. Informally, the command statements look something like:

```
command_a(param_1=value1,param_2="string param")
```

More formally, the syntax diagram for a command files is:

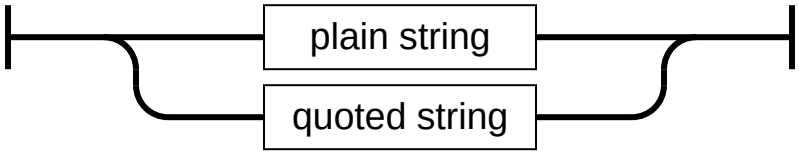


Each parameter value is one of the following types:

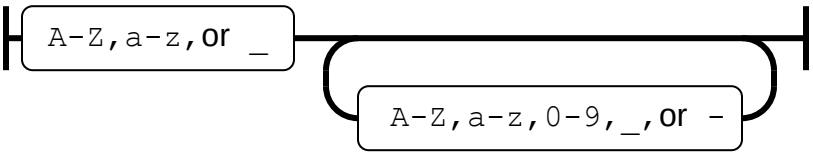


Where the types are defined as:

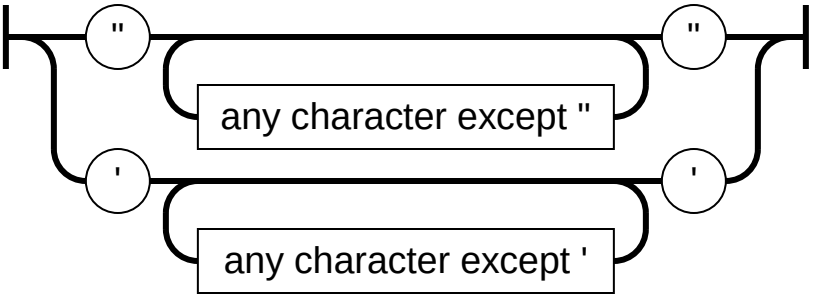
string



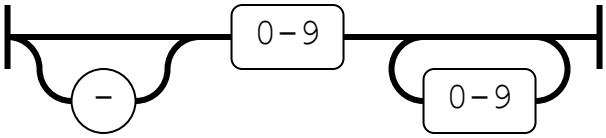
plain string



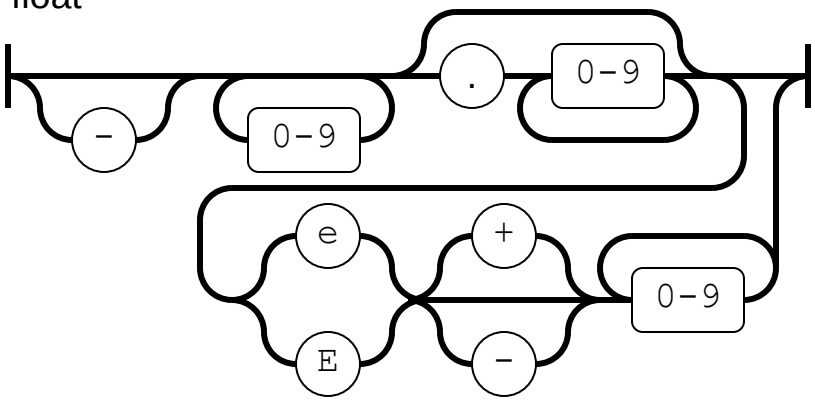
quoted string



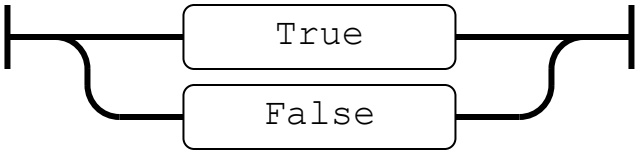
int



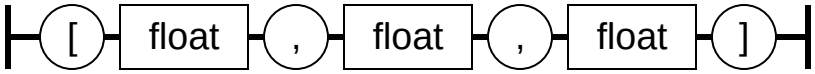
float



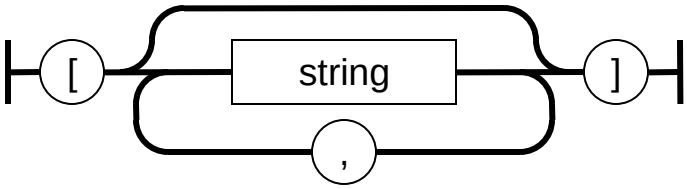
bool



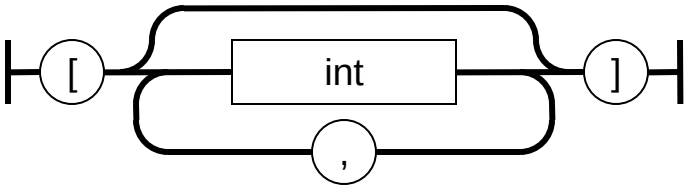
vec3d



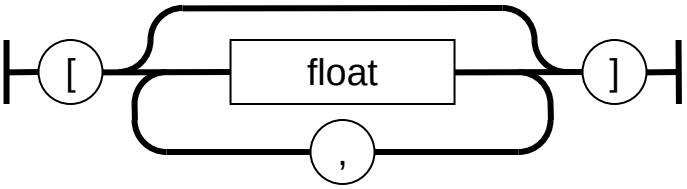
string array



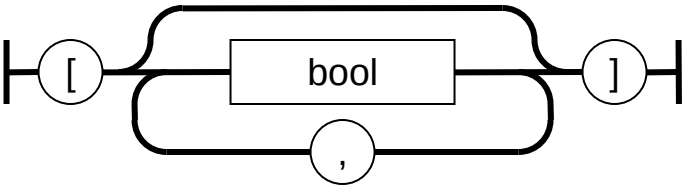
int array



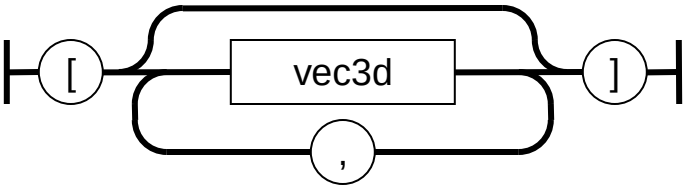
float array



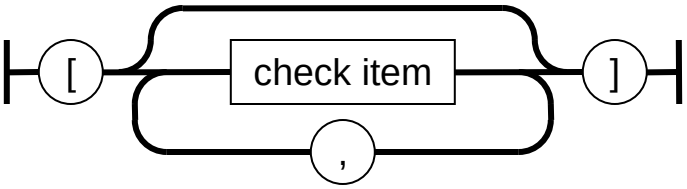
bool array

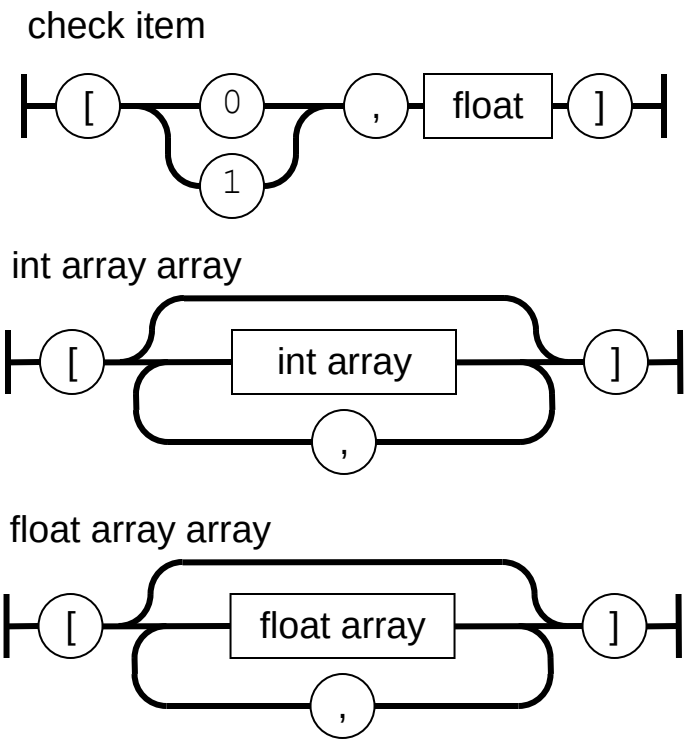


vec3d array



check list





2.1 Commands

The FRANC3D commands are listed next with their parameter list followed by an example. The parameter type is given after the parameter name in *italics*. Note that (req) means that a parameter is required and (opt) means that a parameter is optional.

2.1.1 AutoGrowth( )

Automatically grow crack(s) using the solver specified by the model type parameter.

parameters:

model_type = *string* (req) – model type – ABAQUS, ANSYS or NASTRAN
cur_step = *int* (req) – current crack growth step number
file_name = *single quoted string* (req) - base file name

sif_params (opt) – see 2.1.1.1
growth_params (opt) – see 2.1.1.2
growth_plan (opt) – see 2.1.1.3
template_params (opt) – see 2.1.1.4
front_fitting_params (opt) – see 2.1.1.5
general_analysis_options (req) – see 2.1.1.6

example:

```
AutoGrowth (
   model_type="ABAQUS",
   cur_step=1,
```

```
file_name='/home/bruce/models/abaqus_cube/junk',
paris_C=1e-10,
paris_N=2,
num_steps=3,
const_step_size=0.1,
temp_radius_type=ABSOLUTE,
temp_radius=0.05,
discard=[[0,0]],
extrapolate=[[3,3]],
flags=[NO_WRITE_TEMP,NO_RENUMBER,TRANSFER_BC,
          NO_CFACE_TRACT,NO_CFACE_CNTCT,QUADRATIC],
merge_tol=0.0001,
executable='/usr/local/abaqus/Commands/abaqus',
command='/usr/local/abaqus/Commands/abaqus
   job=junk_STEP_001_full -interactive -analysis ',
   global_model='abaqus_cube/small_cube_outer.inp',
merge_surf_labels=[CUT_SURF],
global_surf_labels=[C_SURF])
```

2.1.1.1 sif_params

parameters:

sif_method = *string* (opt) - SIF computation method
- M_INTEGRAL - use the *M*-integral (interaction integral) technique (default)
- DISP_CORR - use the displacement correlation technique

sif_flags = *string array* (opt) - array of SIF computation options:
- DO_THERM - include thermal terms in the *M*-integral
- REF_TEMP - *float* (opt) - reference temperature, default = 0.0
- DO_PRESS - include crack-face pressure terms in the *M*-integral
- DO_CONTACT - include contact terms in the *M*-integral

2.1.1.2 growth_params

growth_flags = *string* (opt) – list of crack growth flags
- FATIGUE - fatigue crack growth rule (default)
- QUASI_STATIC - quasi-static crack growth rule
- CONSTANT_AMP - constant amplitude fatigue cycles (default)
- VARIABLE_AMP - variable amplitude fatigue cycles
- CLOSURE - include crack closure effects (default)
- NO_CLOSURE - do not include crack closure effects
- SERR_EFF - (default)
- MODAL_EFF -
- DIFF_FRONT_N- (default)
- SAME_FRONT_N -
- PETTIT_THEORY - (default=true)
- KINK_TIP_FIELDS - (default=true)

LEAD_CRACK_FIELDS -
units = *string* (opt) – US or SI units
constant_amp_R = *float* (opt) – R value
spectrum_file = *string* (opt) – load spectrum file name
spectrum_file_type = *string* (opt) – load spectrum file type
spectrum_file_delim = *string* (opt) – load spectrum file data delimiter
spectrum_file_cols = *int list* (opt) – load spectrum file number of columns
spectrum_label = *string list* (opt) – load spectrum column labels
spectrum_mult = *float* (opt) – load spectrum factor
spectrum_offset = *float* (opt) – load spectrum offset
spectrum_groups = *float* (opt) – load spectrum groups
spectrum_group_units = *string* (opt) – load spectrum group units
load_case_strategy = *string* (opt) – strategy for using load cases
kink_angle_strategy = *string* (opt) – strategy for determining kink angle
beta_II = *float* (opt) –
beta_III = *float* (opt) –
aniso_tough_params = *float list* (opt) – anisotropic toughness
extension_type = *string* (opt) – extension method
extension_model = *string* (opt) – extension model
front_n = *float* (opt) – extension model
front_ext_mult = *float list* (opt) – extension multiplier
const_median_step = *float* (opt) – constant median extension
const_cycles_step = *float* (opt) – constant number of cycles for extension
paris_C = *float* (opt) – Paris model C parameter
paris_n = *float* (opt) – Paris model n parameter
walker_C = *float* (opt) – Walker model C parameter
walker_N = *float* (opt) – Walker model n parameter
walker_M = *float* (opt) – Walker model m parameter
nasgro3_props = *float list* (opt) – NASGRO V3 parameter list
nasgro3_thick = *float* (opt) – NASGRO V3 thickness
nasgro5_props = *float list* (opt) – NASGRO V5 parameter list
nasgro5_props = *float* (opt) – NASGRO V5 thickness
user_model_label = *string* (opt) – user-defined crack growth rate data label
user_model_file = *string* (opt) – user-defined crack growth rate data file
tip_closure = *string* (opt) –
closure_COF = *string* (opt) –
newman_alpha = *float* (opt) – Newman closure model alpha parameter
newman_C = *float* (opt) – Newman closure model C parameter
newman_stress_ratio = *float* (opt) – Newman closure model stress ratio parameter

2.1.1.3 growth_plan

num_steps = *int* (opt) – number of crack growth steps
step_type = *string* (opt) – type of crack growth increment model
const_step_size = *float* (opt) – constant crack growth median increment

lin_step_start = *float* (opt) – linear model crack growth start value
lin_step_inc = *float* (opt) – linear model crack growth increment
user_step = *float list* (opt) – user-defined crack growth increment list
user_program = *string* (opt) –
user_front_file = *string* (opt) –
user_points_file = *string* (opt) –

2.1.1.4 template_params

use_templates = *bool* (opt) – use template flag
temp_radius_type = *string* (opt) – template radius type
temp_radius = *float* (opt) – template radius type
temp_prog_ratio = *float* (opt) – template progression ratio
temp_num_rings = *int* (opt) – template number of rings of elements
temp_num_circ = *int* (opt) – template number of elements around front
temp_max_aspect = *float* (opt) – template element aspect ratio
temp_simple_interserct = *bool* (opt) – use simple intersections

2.1.1.5 front_fitting_params

smoothing_method = *string list* (opt) – smoothing type for each crack front
- KINK_EXTEN_POLY - polynomial fit to kink angles and extension
- FIXED_ORDER_POLY – polynomial fit through front points
- CUBIC_SPLINE – cubic-spline fit through front points
- MOVING_POLY – moving polynomial fit through front points
- NOFIT_EXTRAP – partial fitting and extrapolation
- NOFIT_NOEXTRAP – no fitting or extrapolation
- HERMITIAN – Hermitian polynomial fit through closed-front points

polynomial_order = *int list* (opt) – polynomial order for each front
discard = *int list list* (opt) – discard end points for each front
extrapolate = *float list list* (opt) – extrapolate curve ends for each front
retain_all_nodes = *bool list list* (opt) – retains nodes as geometric points on the new crack front; only applies to GrowCrackFromFile

2.1.1.6 general_analysis_options

flags = *string* (req) - list of analysis options:

LINEAR -	use first order elements
QUADRATIC -	use second order elements (default)
RENUMBER -	renumber nodes and elements (default)
NO_RENUMBER -	do not renumber nodes and elements
TRANSFER_BC -	transfer boundary conditions from uncracked mesh (default)

NO_TRANSFER_BC - renumber nodes and elements (dense numbering from 1)
OUTPUT_MAT - output material definitions (default = false)
OUTPUT_CS - output coordinate systems (default = false)
WRITE_TEMP - write nodal temperatures
NO_WRITE_TEMP - do not write nodal temperatures (default)
CFACE_CNTCT - enforce crack face contact
NO_CFACE_CNTCT - do not enforce crack face contact (default)
CFACE_TRACT - write crack face tractions (default)
NO_CFACE_TRACT - do not write crack face tractions
FILE_ONLY - write analysis files only
CHECK_ONLY - perform a data check analysis only
CONTOUR_INTEGRAL - perform contour integral calculation (ABAQUS only)

start_node = *int* (opt) – start node id

start_elem = *int* (opt) – start element id

merge_tol = *float* (opt) - tolerance used to join nodes in the submodel to nodes in the global model

step_info = *string* (opt) - output step information:
DEFAULT - define default load step information, uses load step information from uncracked model file if it exists
SUPPRESS - do not define any load step information
EXTERNAL - read load step information from an external file

front_elem_type = *string* (opt) - type of elements to place at the crack front:
WEDGE - natural wedge elements (default)
COLLAPSED - collapsed brick, front nodes constrained
BLUNTED - collapsed brick, front nodes unconstrained

connection_type = *string* (opt) - method to join the submodel and global model:
MERGE - combine coincident nodes (default)
CONSTRAIN - join using constraint equations
CONTACT - insert contact conditions between models
SHELL_SOLID - insert shell to solid coupling conditions

file_name = *single quoted string* (req) - mesh file name

global_model = *single quoted string* (opt) - name of the global model

load_input = *single quoted string* (opt) - external load step

executable = *single quoted string* (opt) - path to the analysis executable

command = *single quoted string* (opt) - analysis command passed to the OS

merge_surf_labels = *string array* (opt) - labels for the merge surfaces on the submodel

global_surf_labels = *string array* (opt) - labels for the merge surfaces on the global model

2.1.2 CloseModel( )

Close the current model.

example:

```
CloseModel()
```

2.1.3 ComputeCOD( )

Compute crack-front stress intensity factors.

parameters:

distance = *float* (req) - distance from the crack front
at_nodes = *bool* (opt) - compute at node points (default = true)

example:

```
ComputeCOD(distance=0.1,
   at_nodes=false)
```

2.1.4 ComputeGrowthParams( )

Compute crack growth parameters; calls ComputeSifs() and ComputeGrowth().

parameters:

sif_params => see Section 1.1.10: ComputeSif()
growth_params => see Section 2.1.3.1: growth_params

example:

```
ComputeGrowthParams(sif_method=M_INTEGRAL,
   growth_flags=[QUASI_STATIC],
   paris_C=0,
   paris_N=2)
```

2.1.5 ComputeSif( )

Compute crack-front stress intensity factors.

example:

```
ComputeSif(sif_method=M_INTEGRAL,
   sif_flags=[DO_TEMP,SUM]
   load_case_factors=[[1,3.4],[0,1],[1,0.5]])
```

2.1.6 CrackTractConst( )

Define or edit constant crack-face tractions.

parameters:

index = *int* (req) - crack traction index
press = *float* (req) - constant pressure magnitude
load_case = *float* (req) - traction load case

example:

```
CrackTractConst(index=1,
  pressure=10.0,
  load_case=1)
```

2.1.7 CrackTractDelete( )

Delete a crack-face traction.

parameters:

index = *int* (req) - crack traction index

example:

```
CrackTractDelete(index=1)
```

2.1.8 CrackTractExternalDist( )

Define or edit external distribution crack-face tractions.

parameters:

index = *int* (req) - crack traction index
mesh_file = *single quoted string* (req) - external mesh file name
stress_file = *single quoted string* (req) - external stress file name
external_case = *int* (req) - external load case ID
load_case = *float* (req) - traction load case

example:

```
CrackTractExternalDist(index=1,
  mesh_file='my_dist.fem',
  stress_file='my_dist.str',
  external_case=1,
  load_case=1)
```

2.1.9 CrackTractSurface( )

Define or edit surface treatment crack-face tractions.

parameters:

index = *int* (req) - crack traction index
dist = *float list list* (req) - distribution distance/traction pairs
load_case = *float* (req) - traction load case

example:

```
CrackTractSurface(index=1,
   dist=[[0,10],[0.25,0]],
   load_case=2)
```

2.1.10 CrackTract1DRad( )

Define or edit a 1D radial crack-face traction distribution.

parameters:

index = *int* (req) - crack traction index
axis = *int* (req) - axis of rotation (x = 1, y = 2, z = 3)
offset = *vec3d* (req) - axis offset from origin
dist = *float list list* (req) - distribution distance/traction pairs
load_case = *float* (req) - traction load case

example:

```
CrackTract1DRad(index=1,
   axis=1,
   offset=[0,0,1],
   dist=[[0,10],[0.25,0]],
   load_case=2)
```

2.1.11 CrackTract2DRad( )

Define or edit a 2D radial crack-face traction distribution.

parameters:

index = *int* (req) - crack traction index
axis = *int* (req) - axis of rotation (x = 1, y = 2, z = 3)
axial = *float list* (req) - list of axial locations
radial = *float list* (req) - list of radial locations
dist = *float list list* (req) - table of traction values for all radial and axial locations
load_case = *float* (req) - traction load case

example:

```
CrackTract2DRad(index=0,
   axis=1,
```

```
axial=[-0.5,0.0,0.5],
radial=[0.0,1.0],
dist=[[0,1,0],[1,1.25,1]],
load_case=2)
```

2.1.12 FretModelImport( )

Import fretting data model files.

parameters:

model_type = *string* (req) – model file type
- ABAQUS - ABAQUS .inp file
- ANSYS - ANSYS .cdb file

file_name = *single quoted string* (req) – model file name
load_case_flags = *bool list* (req) – list of flags
results_files = *string list* (req) – list of results file names
results_load_cases = *int list* (opt) – list of load case ids in results
retained_mats = *string list* (opt) – list of retained materials
retained_master = *string list* (opt) – list of retained master surfaces
retained_slave = *string list* (opt) – list of retained slave surfaces
retained_eoc = *string list* (opt) – list of retained edge-of-contact nodes
color_regions = *bool* (opt) – color regions

example:

```
FretModelImport(model_type=ANSYS,
      file_name='C:\temp\uncracked.cdb',
      load_case_flags=[false,true,true],
      results_files=['NONE','C:\ \fretting test rig\fretting_rig_ls1.str','
            C:\ fretting test rig\fretting_rig_ls2.str'],
      results_load_cases=[0,0,0],
      retained_mats=[ALL],
      color_regions=true)
```

2.1.13 FretNucleationCycles( )

parameters:

fretting_model_type = *string* (req) – model file type
- FRET_SEQ - equivalent stress model
- FRET_GCRIT - critical shear stress model
- FRET_SWT - Smith-Watson-Topper model
- FRET_RAIQ - crack analog model

fretting_ params = *float list* (req) – fretting model parameters
do_averaging = *bool* (opt) – do volume averaging of stress / strain
save_file = *string* (opt) – file name to save fretting cycles

example:

```
FretNucleationCycles(fretting_model_type=FRET_SEQ,
        fretting_params=[0.43,52476,-0.6471,450.85,-0.03582],
        do_averaging=false)
```

2.1.14 FretNucleationDataImport( )

parameters:

fretting_raw_data_file = *single quoted string* (req) – file name for raw fretting nucleation data

fretting_function_type = *string* (req) – function type for fitting raw data

POLY_FIT - polynomial fit

LM_FIT - nonlinear exponential fit

fretting_function_id = *int* (req) – function id

example:

```
FretNucleationDataImport(fretting_raw_data_file= 'C:\temp\fret_data.txt')
```

2.1.15 GrowCrack( )

Grow crack front(s); calls ComputeSifs(), ComputeGrowth(), ComputeFitFront().

parameters:

sif_params => see Section 2.1.10: ComputeSifs()

growth_params => see Section 2.1.9: ComputeGrowthParams()

template_params => see Section 2.1.3.3: template_params

fit_params => see Section 2.1.3.4: front_fitting_params

file_name = *string (opt)* - file name

example:

```
GrowCrack(sif_method=M_INTEGRAL,
   const_median_step=0.1,
   growth_flags=[QUASI_STATIC],
   paris_C=0,
   paris_N=2,
   temp_radius_type=ABSOLUTE,
   temp_radius=0.05,
   discard=[[0,0]],
   extrapolate=[[3,3]])
```

2.1.16 GrowCrackFromFile( )

Grow crack front(s) from a file; calls ComputeFitFront().

parameters:

read_file = *single quoted string (req)* - file name of new front points
template_params => see Section 2.1.3.3: template_params
fit_params => see Section 2.1.3.4: front_fitting_params
file_name = *single quoted string (opt)* - file name

example:

```
GrowCrackFromFile(temp_radius_type=ABSOLUTE,
   temp_radius=0.05,
   read_file='abaqus_cube/small_step.frt'))
```

2.1.17 Include( )

2.1.18 InsertFileFlaw( )

Insert a flaw defined in a .crk file. Note that flaw orientation and template options are defined in .crk files. Parameters for this command, if specified, will override the values in the file.

parameters:

file_name = *single quoted string* (req) - name of a flaw (.crk) definition file
rotation_axes = *int array* (opt) - ordered list of up to three rotation axes (x = 1, y = 2, z = 3)
rotation_axes = *float array* (opt) - ordered list of up to three rotation magnitudes
translation = *vec3d* (opt) - translation vector
refinement_level = *int* (opt) - number of times the flaw patches will be recursively subdivided
radius = *float* (opt) - crack-front template radius
progression_ratio = *float* (opt) - crack-front template element size progression ratio (default = 1)
num_rings = *int* (opt) - number of element rings in the crack-front template (default = 3)
num_circ = *int* (opt) - number of element inserted circumferentially about a crack-front (default = 8)
max_aspect_ratio = *float* (opt) - maximum allowable aspect ratio for crack-front elements (default = 3.0)
simple_ints_only = *bool* (opt) - if true crack-front templates intersect free surfaces only if the crack-front meets the surface at nearly a right angle, otherwise the template terminates within the body (default = false)

example:

```
InsertFileFlaw(
   file_name='my_flaw.crk',
   rotation_axes=[3],
   rotation_mag=[-45],
   translation=[0,0.1,0],
   radius=0.025)
```

2.1.19 InsertMultFileFlaw( )

2.1.20 InsertMultParamFlaw( )

Insert multiple parameterized flaws.

parameters:

flaw_type = *string array* (req) - list of flaw types:
- CRACK - zero volume crack
- VOID - finite volume void

crack_type = *string array* (req) - list of crack types:
- ELLIPSE - elliptical crack
- THRU - through the thickness crack
- CENTER - center crack
- LONG - long shallow crack
- NONE - place holder for void types

void_type = *string array* (req) - list of void types:
- ELLIPSOID - ellipsoidal flaw
- NONE - place holder for crack types

flaw_params = *float array array* (req) - list of lists of flaw size parameters
rotation_axes = *int array array* (opt) - list of ordered lists of up to three rotation axes (x = 1, y = 2, z = 3)
rotation_axes = *float array array* (opt) - list of ordered lists of up to three rotation magnitudes
translation = *vec3d array* (opt) - list of translation vectors
refinement_level = *int array* (opt) - list of the number of times the flaw patches will be recursively subdivided
radius = *float array* (opt) - list of crack-front template radii
progression_ratio = *float array* (opt) - list of crack-front template element size progressions ratio (default = 1)

num_rings = *int array* (opt) - list of the number of element rings in the crack-front templates (default = 3)
num_circ = *int array* (opt) - list of the number of elements inserted circumferentially

about the crack-fronts (default = 8)

max_aspect_ratio = *float array* (opt) - list of the maximum allowable aspect ratios for crack-front elements (default = 3.0)

simple_ints_only = *bool array* (opt) - list of flags, true indicates that crack-front templates intersect free surfaces only if the crack-front meets the surface at nearly a right angle, otherwise the template terminates within the body (default = false)

example:

```
InsertMultParamFlaw(
   flaw_type=[CRACK,CRACK],
   crack_type=[ELLIPSE,ELLIPSE],
   flaw_params=[[0.1,0.1],[0.15,0.15]],
   rotation_axes=[[1],[1]],
   rotation_mag=[[90],[90]],
   translation=[[0,0.1,1],[0,-0.15,1]],
   radius=[0.02,0.03])
```

2.1.21 InsertParamFlaw( )

Insert a parameterized flaw.

parameters:

flaw_type = *string* (req) - flaw type:
- CRACK - zero volume crack
- VOID - finite volume void

crack_type = *string* (req if flaw_type = CRACK) - crack type:
- ELLIPSE - elliptical crack
- THRU - through the thickness crack
- CENTER - center crack
- LONG - long shallow crack

void_type = *string* (req if flaw_type = VOID) - list of void types:
- ELLIPSOID - ellipsoidal flaw

flaw_params = *float array* (req) - list of flaw size parameters

rotation_axes = *int array* (opt) - ordered list of up to three rotation axes (x = 1, y = 2, z = 3)

rotation_mag = *float array* (opt) - ordered list of up to three rotation magnitudes

x-axis* = *vec3d* (opt) – local x-axis of crack plane

y-axis* = *vec3d* (opt) – local y-axis of crack plane

translation = *vec3d* (opt) - translation vector

refinement_level = *int* (opt) - number of times the flaw patches will be recursively subdivided

radius = *float* (opt) - crack-front template radius

progression_ratio = *float* (opt) - crack-front template element size progression ratio (default = 1)
num_rings = *int* (opt) - number of element rings in the crack-front template (default = 3)
num_circ = *int* (opt) - number of element inserted circumferentially about a crack-front (default = 8)
max_aspect_ratio = *float* (opt) - maximum allowable aspect ratio for crack-front elements (default = 3.0)
simple_ints_only = *bool* (opt) - if true crack-front templates intersect free surfaces only if the crack-front meets the surface at nearly a right angle, otherwise the template terminates within the body (default = false)

**If local x and y axes are defined, the rotation angles are computed from the vectors.*

example:

```
InsertParamFlaw(
   flaw_type=CRACK,
   crack_type=ELLIPSE,
   flaw_params=[0.15,0.15],
   rotation_axes=[1],
   rotation_mag=[90],
   translation=[0,-0.15,1],
   radius=0.03)
```

2.1.22 MapState( )

Map material state variables between the previous and the current model – is not currently implemented.

parameters:

flags = *string array* (req) - list of state variables to map:
- TEMP - nodal temperatures
- DISP - nodal displacements
- STRESS - nodal stresses
- STRAIN - nodal strains

example:

```
MapState(
   flags=[DISP,STRESS,STRAIN])
```

2.1.23 OpenFdbModel( )

Open a FRANC3D database (.fdb) file.

parameters:

file_name = *single quoted string* (req) - .fdb file name
orig_mesh_file = *single quoted string* (opt) - file name for the original, uncracked, mesh model
extra_file = *single quoted string array* (opt) - list of names of extra (load case) analysis files
cur_mesh_file = *single quoted string* (req) - file name for the current crack step mesh model
cur_resp_file = *single quoted string* (req) - file name for the current crack step FEM results

example:

```
OpenFdbModel(
   file_name='my_cracked_model.fdb',
   orig_mesh_file='uncracked_mesh.cdb',
   cur_mesh_file='my_cracked_model.cdb',
   cur_resp_file='my_cracked_model.dtp')
```

2.1.24 OpenMeshModel( )

Open an FEM mesh file.

parameters:

model_type = *string* (req) - type of file to read:
  ABAQUS - ABAQUS .inp file
  ANSYS - ANSYS .cdb file
  NASTRAN - NASTRAN .bdf file
file_name = *single quoted string* (req) - mesh file name
extra_file = *single quoted string array* (opt) - list of names of extra load case files
retained_comps = *string array* (opt) - list of names of components to be retained
retained_mats = *string array* (opt) - list of names of materials to be retained
retained_css = *string array* (opt) - list of names of coordinate systems to be retained
retained_bcs = *string array* (opt) - list of names of boundary conditions to be retained
retained_const = *string array* (opt) - list of names of constraints to be retained
retained_resid = *string array* (opt) - list of names of residual stress components
retained_mesh_bcs = *string array* (opt) - list of mesh boundary condition flags:
  DISP - nodal displacements
  FORCE - nodal forces

DISP - nodal displacements
CPDOF - coupled degrees of freedom
CONST - constraints

ignore_mat_bdry = *bool* (opt) - if true, material boundaries are ignored (default = flase)
retain_inverse = *bool* (opt) - if true, mesh facets in the selected retained_comps are not retained while all other mesh facets are retained (default = false)

example:

```
OpenMeshModel(
  model_type=ANSYS,
  file_name='my_mesh.cdb',
  retained_mats=[ALL])
```

2.1.25 ReadFullGrowthHist ( )

Read a full crack growth history file.

parameters:

file_name = *single quoted string* (req) – crack growth history file name

example:

```
ReadFullGrowthHist(file_name='history.fcg')
```

2.1.26 ReadGrowthParams ( )

2.1.27 ReadResponse( )

Read a response file.

parameters:

file_name = *single quoted string* (req) - response file name
new_load_case = *bool* (opt) - if true a new load case is created

example:

```
ReadResponse(
  file_name='anal_rslts.dtp',
  new_load_case=True)
```

2.1.28 RunAnalysis( )

Run an analysis

parameters:

model_type (req) – see Section 2.1.1
general_analysis_options (req) – see Section 2.1.1.6

example:

```
RunAnalysis(model_type="ABAQUS",
   file_name='abaqus_cube/junk',
   flags=[NO_WRITE_TEMP,NO_RENUMBER,TRANSFER_BC,
             NO_CFACE_TRACT,NO_CFACE_CNTCT,QUADRATIC],
   merge_tol=0.0001,
   executable='/usr/local/abaqus/Commands/abaqus',
   command='/usr/local/abaqus/Commands/abaqus job=junk_full
              -interactive -analysis ',
   global_model='abaqus_cube/small_cube_outer.inp',
   merge_surf_labels=[CUT_SURF],
   global_surf_labels=[C_SURF])
```

2.1.29 SaveFdbModel( )

Save a FRANC3D database (.fdb) file.

parameters:

file_name = *single quoted string* (req) - .fdb file name
mesh_file_name = *single quoted string* (opt) - file name for the mesh model
rslt_file_name = *single quoted string* (req) - file name for the FEM results
anaysis_code = *string* (req) - analysis code:
    ABAQUS, ANSYS, or NASTRAN
flags = *string array* (opt) - list of analysis option flags

LINEAR -	use first order elements
QUADRATIC -	use second order elements (default)
RENUMBER -	renumber nodes and elements (default)
NO_RENUMBER -	do not renumber nodes and elements
TRANSFER_BC -	transfer boundary conditions from uncracked mesh (default)
NO_TRANSFER_BC -	renumber nodes and elements (dense numbering from 1)

start_node = *int* (opt) - starting node number for renumbering (default = 1)
start_elem = *int* (opt) - starting element number for renumbering (default = 1)

example:

```
SaveFdbModel(
   file_name='my_model.fdb',
   mesh_file_name='my_model.cdb',
   rslt_file_name='my_model.dsp',
   analysis_code=ANSYS)
```

2.1.30 SaveGrowthParams( )

2.1.31 SaveMeshModel( )

Save a FEM mesh file.

parameters:

file_name = *single quoted string* (req) - mesh file name
model_type = *string* (req) - analysis code:
ABAQUS, ANSYS, or NASTRAN
flags = *string array* (opt) - list of analysis option flags

LINEAR -	use first order elements
QUADRATIC -	use second order elements (default)
RENUMBER -	renumber nodes and elements (default)
NO_RENUMBER -	do not renumber nodes and elements
TRANSFER_BC -	transfer boundary conditions from uncracked mesh (default)
NO_TRANSFER_BC -	renumber nodes and elements (dense numbering from 1)

start_node = *int* (opt) - starting node number for renumbering (default = 1)
start_elem = *int* (opt) - starting element number for renumbering (default = 1)

example:

```
SaveMeshModel(
   file_name='my_model.cdb',
   model_type=ANSYS)
```

2.1.32 SetEdgeParameters( )

Set parameters for edge extraction.

parameters:

kink_angle = *float* (opt) - kink edge angle threshold

do_planar_seed = *bool* (opt) - use a seed growth algorithm to find planar regions
planar_angle = *float* (opt) - angle threshold for planar regions
planar_facets = *int* (opt) - minimum number of facets in planar regions

example:

```
SetEdgeParameters(
  do_planar_seed=True,
  planar_angle=178,
  planar_facets=5)
```

2.1.33 SetGrowthParams( )

2.1.34 SetLoadSchedule( )

2.1.35 SetMeshingParameters( )

Modify meshing parameters.

parameters:

max_gen_elems = *int* (opt) - maximum number of elements that will be generated
surf_refine_bdry_factor = *float* (opt) - boundary refinement factor for surface meshing
surf_near_bdry_factor = *float* (opt) - near boundary node factor for surface meshing
optimal_sphere_factor = *float* (opt) - optimal sphere size factor for volume meshing
optimal_size_factor = *float* (opt) - optimal octtree size factor for volume meshing
volume_refine_factor = *float* (opt) - optimal octtree refinement factor for volume meshing
all_planar_surf_meshing = *bool* (opt) - force planar surface meshing
do_surface_smoothing = *bool* (opt) - surface smoothing flag
do_coarsen_crack_mouth = *bool* (opt) - coarsen crack mouth flag
do_surface_refinement = *bool* (opt) - surface element refinement flag
max_vol_restarts = *int* (opt) - maximum number of volume meshing restarts
volume_meshing_method = *string* (opt) - meshing code: FRANC3D, ABAQUS, or ANSYS
ansys_executable = *string* (opt) - ANSYS executable for meshing
ansys_license = *string* (opt) - ANSYS license type
abaqus_executable = *string* (opt) - ABAQUS executable for meshing

example:

```
SetMeshingParameters(
  do_surface_refinement=True,
```

```
    max_vol_restarts=5)
```

2.1.36 SetStatusFile( )

2.1.37 SetTrimRegions( )

Specify the ID's of regions into which crack will not grow. They will be trimmed at the bi-material interface.

parameters:

region = *int array* (req) - list of region ids

example:

```
SetTrimRegions(
  regions=[3])
```

2.1.38 SetUnits( )

2.1.39 SetWorkingDirectory( )

2.1.40 SifHistory( )

parameters:

path_type = *string* (opt) – SIF history path type
const_n_dist = *float* (opt) – normalized distance to compute SIFs
near_n_dist = *float* (opt) -
do_smooth = *bool* (opt) -
Ks_poly_order = *int* (opt) -
do_least_squares = *bool* (opt) -
poly_order = *int* (opt) -
min_n_dist = *float* (opt) -
max_n_dist = *float* (opt) -
min_smooth_n_dist = *float* (opt) -
max_smooth_n_dist = *float* (opt) -
plane_normal = *Vec3D* (opt) -
plane_point = *Vec3D* (opt) -
start_type = *string* (opt) – set start type (point or length)
start_point = *Vec3D* (opt) – set starting crack point (origin)

start_length = *float* (opt) – set starting crack length
start_step = *int* (opt) – set starting crack step id
start_front = *int* (opt) – set starting crack front id

example:

```
SifHistory(
   Start_front=1)
```

2.1.41 Submodeler( )

2.1.42 WriteCOD( )

Generate a file containing crack-front crack opening displacement data.

parameters:

file_name = *single quoted string* (req) - output SIF file name
front_id = *int* (opt) - ID (index) of the crack front (0 .. n-1), default = 0
flags = *string array* (opt) - array of output options:

SPACE -	use spaces as delimiters
TAB -	use tabs as delimiters (default)
COMMA -	use commas as delimiters
COD -	include crack opening displacements
CSD -	include crack sliding displacements
CTD -	include crack tearing displacements
KI -	include mode one stress intensity factors
KII -	include mode two stress intensity factors
KIII -	include mode three stress intensity factors
CRD -	include crack-front Cartesian coordinates
PNT -	include COD evaluation point coordinates
AXES -	include crack-front local coordinate axes
MODULI -	include the Youngs' moduli at the front and evaluation points

example:

```
WriteCOD(
   file_name='sif_data.sif'
   flags=[TAB,COD,CSD,CTD,CRD,PNT])
```

2.1.43 WriteCrackData( )

Generate a file containing predicted crack growth data.

parameters:

file_name = *single quoted string* (req) - output file name

example:

```
WriteCrackData(
   file_name='crack_info.dat')
```

2.1.44 WriteFatigueData( )

2.1.45 WriteGrowthParams( )

Generate a file containing crack-growth data.

parameters:

file_name = *single quoted string* (req) - output file name
front_id = *int* (opt) – crack front id
flags = *string list* (opt) – array of output options:

SPACE -	use spaces as delimiters
TAB -	use tabs as delimiters (default)
COMMA -	use commas as delimiters
XYYY -	first column is position, remaining columns are values (default)
XYXY -	odd columns are position, even columns are values
KI -	include mode one stress intensity factors
KII -	include mode two stress intensity factors
KIII -	include mode three stress intensity factors
J -	include J-integral values
T -	include T-stress values
CRD -	include crack-front Cartesian coordinates
AXES -	include crack-front local coordinate axes
ANGLE -	include the predicted kink angle
DIR -	include the normalized predicted propagation direction
REVERSE -	reverse the ordering of the values
DK -	include delta K1
EXT -	include extension
R -	include R ratio
FULL_STEP -	
GI -	include mode I energy release rate
GII -	include mode II energy release rate
GIII -	include mode III energy release rate

example:

```
WriteGrowthParams(
   file_name='growth_params.dat')
```

2.1.46 WriteSif( )

Generate a file containing crack-front fracture parameters and data.

parameters:

file_name = *single quoted string* (req) - output SIF file name
front_id = *int* (opt) - ID (index) of the crack front (0 .. n-1), default = 0
flags = *string array* (opt) - array of output options:

SPACE -	use spaces as delimiters
TAB -	use tabs as delimiters (default)
COMMA -	use commas as delimiters
XYYY -	first column is position, remaining columns are values (default)
XYXY -	odd columns are position, even columns are values
KI -	include mode one stress intensity factors
KII -	include mode two stress intensity factors
KIII -	include mode three stress intensity factors
J -	include J-integral values
CRD -	include crack-front Cartesian coordinates
AXES -	include crack-front local coordinate axes
ANGLE -	include the predicted kink angle
DIR -	include the normalized predicted propagation direction
REVERSE -	reverse the ordering of the values

example:

```
WriteSif(
   file_name=sif_data.sif
   flags=[TAB,XYYY,KI,KII,KIII])
```

2.1.47 WriteSifPath( )

2.1.48 WriteSifHist( )

2.1.49 WriteStdTempData( )

Generate a file containing the ID's and coordinates of nodes in the crack-front template.

parameters:

file_name = *string* (req) - output file name

example:

```
WriteStdTempData(
   file_name=template_info.dat)
```

2.2 Example Command File

An example command file might look like this:

```
OpenMeshModel(
   model_type=ANSYS,
   file_name='C:\cube\cube.cdb',
   retained_mats=[ALL],
   retained_bcs=[ALL])

InsertFileFlaw(
   file_name='C:\cube\cube_05.crk')

RunAnalysis(
   model_type=”ANSYS”,
   file_name='C:\cube\cracked_cube',
   flags=[QUADRATIC],
   executable='C:\ANSYS Inc\v121\ansys\bin\WINX64\ANSYS121.exe',
   command='"C:\ANSYS Inc\v121\ansys\bin\WINX64\ANSYS121.exe" -b -p
     struct -i "C:\cube\cracked_cube.macro" -o "C:\cube\cracked_cube.out"',
   license=struct)

ComputeSif(
   load_case_factors=[[1,1,1]])

GrowCrack(
   growth_flags=[QUASI_STATIC],
   const_median_step=0.05,
   paris_C=0,
   paris_N=2,
   temp_radius_type=ABSOLUTE,
   temp_radius=0.03,
   discard=[[0,0]],
   extrapolate=[[3,3]])

SetMeshingParameters()
```

```
AutoAnsys(
  model_type=”ANSYS”,
   cur_step=1,
   file_name='C:\ cube\cracked_cube',
   growth_flags=[QUASI_STATIC],
   paris_C=0,
   paris_N=2,
   num_steps=2,
   const_step_size=0.05,
   temp_radius_type=ABSOLUTE,
   temp_radius=0.03,
   discard=[[0,0]],
   extrapolate=[[3,3]],
  flags=[OUTPUT_MAT,OUTPUT_CS,TRANSFER_BC,
          NO_CFACE_TRACT,NO_CFACE_CNTCT],
   merge_tol=0.0001,
   executable='C:\ANSYS Inc\v121\ansys\bin\WINX64\ANSYS121.exe',
   command='"C:\ANSYS Inc\v121\ansys\bin\WINX64\ANSYS121.exe" -b -p struct -i
  "cracked_cube'_STEP_001.macro" -o "cracked_cube_STEP_001.out"',
   license=struct)

ComputeSif(
   load_case_factors=[[1,1,1]])

WriteCrackData(
   file_name='C:\cube\cracked_cube.fcg')
```

3. Python Module

The Python module has extensions that mirror the commands described in Section 2. It allows the user to write more elaborate Python scripts that include these commands.

3.1 Command Converter

The commands described in Section 2 can be converted to Python commands using the Fcl2Py executable. This program requires one argument, which is the command file name. It reads the commands from the file, converts them to the equivalent Python commands, and then writes this information to stdout, which can be piped to a file.

For example: C:/examples/Fcl2Py.exe session01.log > ses01.py

3.2 Python Module

The PyF3D.dll (or PyF3D.pyd) must be imported into Python. The module requires the Vec3D.py module, which is distributed with the PyF3D module. The PyF3D module is linked against Python libraries. The current version 7.0 module is linked with Python 2.7. A typical Python script would start by importing the "sys" module and appending the path to the PyF3D.dll file before importing the PyF3D module:

```
import sys
sys.path.append("C:\FRANC3D")
import PyF3D
import Vec3D
```

Note that for MSWindows, PyF3D.pyd is the same as PyF3D.dll; just the extension is changed.

There are eight classes defined in this module: 1) F3DApp, 2) FemModel, 3) FemResults, 4) Flaw, 5) CrackGrowthData, 6) CrackStep, 7) CrackFront, and 8) FrontPoint.

File names must be provided using single quoted strings. Note that for MSWindows, the file path separator is the '\' character. This must be defined using two of these characters, as '\\' so that Python will process the path correctly.

3.2.1 class F3DApp

The F3DApp is a FRANC3D application object. This is the Python analog to the main window in the GUI version. Most interesting things involve an instance of this object.

constructor:

F3DApp - No arguments.

example:

```
f3d = PyF3D.F3DApp()
```

methods:

All of the method examples below begin with "f3d", which is the F3DApp object defined above.

AutoGrowth – see Section 2.1.1

CloseModel – see Section 2.1.2

example:

```
f3d.CloseModel()
```

ComputeCOD – see Section 2.1.4

example:

```
f3d.ComputeCOD(distance=0.1)
```

ComputeGrowthParams – see Section 2.1.4.

example:

```
f3d.ComputeGrowthParams(sif_method="M_INTEGRAL")
```

ComputeSif – see Section 2.1.5

example:

```
f3d.ComputeSif(sif_method=M_INTEGRAL,
      sif_flags=["DO_TEMP","SUM"])
```

CrackTractConst – see Section 2.1.6

example:

```
f3d.CrackTractConst(index=1,pressure=10.0,
      load_case=1)
```

CrackTractDelete – see Section 2.1.7

example:

```
f3d.CrackTractDelete(index=1)
```

CrackTractExternalDist – see Section 2.1.8

example:

```
f3d.CrackTractExternalDist(index=1,
      mesh_file='C:\\temp\\my_dist.fem',
         stress_file='C:\\temp\\my_dist.str',
           load_case=1)
```

CrackTractSurface – see Section 2.1.9

example:

```
f3d.CrackTractSurface()
```

CrackTract1DRad – see Section 2.1.10

example:

```
f3d.CrackTract1DRad()
```

CrackTract2DRad – see Section 2.1.11

example:

```
f3d.CrackTract2DRad()
```

FretModelImport – see Section 2.1.12

example:

```
f3d.FretModelImport()
```

FretNucleationCycles – see Section 2.1.13

example:

```
f3d.FretNucleationCycles()
```

FretNucleationDataImport – see Section 2.1.14

example:

```
f3d.FretNucleationDataImport()
```

GrowCrack – see Section 2.1.15

example:

```
f3d.GrowCrack(growth_flags=["QUASI_STATIC"],
        const_median_step=0.05,
           paris_C=0,paris_N=2,
           temp_radius_type="ABSOLUTE",
           temp_radius=0.03,
           discard=[[0,0]],extrapolate=[[3,3]])
```

GrowCrackFromFile – see Section 2.1.16

example:

```
f3d.GrowCrackFromFile(file='C:\\temp\\new_front.txt')
```

Include – see Section 2.1.17

example:

```
f3d.Include(file='C:\\temp\\include_file.ext')
```

InsertFileFlaw – see Section 2.1.18

example:

```
f3d.InsertFileFlaw(file='C:\\temp\\init_crack.crk')
```

InsertMultFileFlaw – see Section 2.1.19

example:

```
f3d.InsertFileFlaw()
```

InsertMultParamFlaw – see Section 2.1.20

example:

```
f3d.InsertParamFlaw()
```

InsertParamFlaw – see Section 2.1.21

example:

```
f3d.InsertParamFlaw()
```

MapState – see Section 2.1.22

example:

```
f3d.MapState()
```

OpenFdbModel – see Section 2.1.23

example:

```
f3d.MapState(file_name='C:\\cube\\cracked_cube.fdb')
```

OpenMeshModel – see Section 2.1.24

example:

```
f3d.OpenMeshModel(model_type="ANSYS",
      file_name='C:\\cube\\small_cube_cutout.cdb',
         retained_comps=["CUT_SURF"],
         retained_mats=["ALL"])
```

ReadFullGrowthHist – see Section 2.1.25

example:

```
f3d.ReadFullGrowthHist(
      file_name='C:\\cube\\cracked_cube_history.fcg')
```

ReadGrowthParams – see Section 2.1.26

example:

```
f3d.ReadGrowthParams(
      file_name='C:\\cube\\cracked_cube_history.fcg')
```

ReadResponse – see Section 2.1.27

example:

```
f3d.ReadResponse(
      file_name='C:\\cube\\cracked_cube.dtp')
```

RunAnalysis – see Section 2.1.28

example:

```
f3d.RunAnalysis()
```

SaveFdbModel – see Section 2.1.29

example:

```
f3d.SaveFdbModel(file_name='C:\\temp\\my_model.fdb',
      mesh_file_name='C:\\temp\\my_model.cdb',
```

```
rslt_file_name='C:\\temp\\my_model.dsp',
analysis_code="ANSYS")
```

SaveGrowthParams – see Section 2.1.30

example:

```
f3d.SaveGrowthParams(file_name='C:\\temp\\my)
```

SaveMeshModel – see Section 2.1.31

example:

```
f3d.SaveMeshModel(file_name='C:\\temp\\my_model.cdb',
        model_type="ANSYS")
```

SetEdgeParameters – see Section 2.1.32

example:

```
f3d.SetEdgeParameters(do_planar_seed="True",
      planar_angle=178,
        planar_facets=5)
```

SetGrowthParams – see Section 2.1.33

SetLoadSchedule – see Section 2.1.34

SetMeshingParameters – see Sections 2.1.35

example:

```
f3d.SetMeshingParameters(do_surface_refinement="True",
        max_vol_restarts=5)
```

SetStatusFile – see Sections 2.1.36

SetTrimRegions – see Sections 2.1.37

example:

```
f3d.SetTrimRegions(regions=[3])
```

SifUnits – see Sections 2.1.38

SifWorkingDirectory – see Sections 2.1.39

SifHistory – see Sections 2.1.40

example:

```
f3d.SifHistory ()
```

StartRecording – writes the subsequent Python commands to a py_session.log file

example:

```
f3d.StartRecording ()
```

- this could be used if a user writes a complicated Python script and wants to record the commands in a session log to play back in the GUI later

Submodeler – see Sections 2.1.41

WriteCOD – see Sections 2.1.42

example:

```
f3d.WriteCOD(file_name='sif_data.sif'
         flags=["TAB","COD","CSD","CRD","PNT"])
```

WriteCrackData – see Sections 2.1.43

example:

```
f3d. WriteCrackData(file_name=crack_info.dat')
```

WriteFatigueData – see Sections 2.1.44

WriteGrowthParams – see Sections 2.1.45

example:

```
f3d.WriteGrowthParams()
```

WriteSif – see Sections 2.1.46

example:

```
f3d.WriteSifs(file_name='sif_data.sif'
         flags=["TAB","XYYY","KI","KII","KIII"])
```

WriteSifPath – see Sections 2.1.47

WriteSifHist – see Sections 2.1.48

WriteStdTempData – see Sections 2.1.49

example:

```
f3d.WriteStdTempData(file_name='template_info.dat')
```

The following methods do not have command line equivalents. They are implemented to support the ABAQUS interface where FRANC3D commands are called directly from the ABAQUS CAE.

GetCrackData – returns CrackGrowthData object

example:

```
cgd = f3d.GetCrackData()
```

GetFemModel – returns FemModel object

example:

```
fem = f3d.GetFemModel()
```

GetFemResults – returns FemResults object

example:

fem = f3d.GetFemResults()

InsertMemFlaw – inserts flaw object

example:

f3d.InsertMemFlaw(flaw)

OpenMeshMemModel – processes mesh information into a FemModel object

example:

fem = f3d.OpenMeshMemModel(fem)

ReadMemResponse – processes FE results into a FemResults object

example:

f3d.ReadMemResponse(res)

3.2.2 class FemModel

The FemModel class stores the FE model data.

constructor:

FemModel() - arguments.

example:

```
fem = PyF3D.FemModel()
```

methods:

Clear – clears the database

example:

```
fem.Clear()
```

AddNode – add a node to the database

parameters:

```
id – integer
coordinates – Vec3D
```

example:

```
fem.AddNode(1,Vec3D.Vec3D(0.0,1.0,2.0))
```

AddElem – add an element to the database

parameters:

```
id – integer
material id – integer
coordinate system id – integer
```

element type – string
TET_4
TET_10
PYRAMID_5
PYRAMID_13
WEDGE_6
WEDGE_15
BRICK_8
BRICK_20
BRICK_8

node list – [integer,integer,…]

example:

```
fem.AddElem(1,1,1,"BRICK_8",[1,2,3,4,5,6,7,8])
```

AddMaterial – add a material to database

parameters:

id – integer
type – integer

example:

```
fem.AddMaterial(1,0)
```

AddBc –

parameters:

id – integer
entity – string
NODE
ELEMENT
NODE_GROUP
ELEMENT_GROUP
face – int (if entity is ELEMENT)
type – string
UX
UY
UZ
FX
FY
FZ
MX
MY
MZ
CPRESS
VPRESS
TRACT
BF

TEMP
value – integer
load case – integer

example:
```
fem.AddBc(1,"NODE","UX",0.0,1)
```

AddGroup – add a group of nodes or elements to the database

parameters:
label – string
type – string
GNODE
GELEMENT
entity list – [integer,integer,…]

example:
```
fem.AddGroup("test_group","NODE",[1,2,4,8])
```

AddCs – add a coordinate system to the database

parameters:
id – integer
type – string
CARTESIAN
CYLINDRICAL
SHPERICAL
coordinate – Vec3D
angles – Vec3D

example:
```
fem.AddGroup("test_group","NODE",[1,2,4,8])
```

HasNode – tests whether a node exists in the database; returns True or False

parameters:
id – integer

example:
status = `fem.HasNode(1)`

HasElem – tests whether an element exists in the database; returns True or False

parameters:
id – integer

example:
status = `fem.HasElem(1)`

HasMat – tests whether a material exists in the database; returns True or False

parameters:

```
id – integer
```

example:

status = `fem.HasMat(1)`

HasBc – tests whether a boundary condition exists in the database; returns True or False

parameters:

```
id – integer
entity – string (see AddBc)
type – string (see AddBc)
```

example:

status = `fem.HasBc(1,"NODE","UX")`

HasGroup – tests whether a group exists in the database; returns True or False

parameters:

```
label – string
entity – string (see AddGroup)
```

example:

status = `fem.HasGroup("test","GNODE")`

HasCs – tests whether a coordinate system exists in the database; returns True or False

parameters:

```
id – integer
```

example:

status = `fem.HasCs(1)`

GetNode – get node from database

parameters:

```
id – integer
```

example:

nd = `fem.GetNode(1)`

GetElem – get element from database

parameters:

```
id – integer
```

example:

el = `fem.GetElem(1)`

GetMat – get node from database

parameters:

```
id – integer
```

example:

mat = `fem.GetMat(1)`

GetBc – get node from database

parameters:

```
id – integer
entity – string (see AddBc)
type – string (see AddBc)
```

example:

bc = `fem.GetBc(1,"NODE","UX")`

GetGroup – get node from database

parameters:

```
label – string
entity – string (see AddGroup)
```

example:

el = `fem.GetGroup("test","GNODE")`

GetCs – get node from database

parameters:

```
id – integer
```

example:

el = `fem.GetCs(1)`

GetNodeIterator – get node iterator

example:

ni = `fem.GetNodeIterator()`

GetElemIterator – get element iterator

example:

ei = `fem.GetElemIterator()`

GetMatIterator – get material iterator

example:

m = `fem.GetmatIterator()`

GetBcIterator – get boundary condition iterator

example:

bci = `fem.GetBcIterator()`

GetGroupIterator – get group iterator
example:
gi = `fem.GetGroupIterator()`

GetCsIterator – get node iterator
example:
csi = `fem.GetCsIterator()`

GetNumNode – returns the total number of nodes
example:
nn = `fem.GetNumNode()`

GetNumElem – returns the total number of elements
example:
ne = `fem.GetNumElem()`

GetNumMat – returns the total number of materials
example:
nm = `fem.GetNumMat()`

GetNumBc – returns the total number of boundary conditions
example:
nbc = `fem.GetNumBc()`

GetNumGroup – returns the total number of groups
example:
ng = `fem.GetNumGroup()`

GetNumCs – returns the total number of coordinate systems
example:
ncs = `fem.GetNumCs()`

3.2.3 class FemResults

The FemResults class stores the FE results data. By default, load case 0 is added when the object is created.

constructor:

FemResults()

example:

```
fres = PyF3D.FemResults()
```

methods:

Clear – clear the database

example:

```
fres.Clear()
```

AddScalarResults – add a scalar result to database

parameters:

```
id – integer
entity – string
        NODE
        ELEMENT
type – string
        TEMP
value – integer
load case – integer
```

example:

```
fres.AddScalarResults(1,”NODE”,”TEMP”,20.0,1)
```

AddVectorResults –

parameters:

```
id – integer
entity – string
        NODE
        ELEMENT
type – string
        DISP
        FORCE
value – Vec3D
load case – integer
```

example:

```
fres.AddVectorResults(1,”NODE”,”DISP”,Vec3D,1)
```

AddScalarList – add a list of scalar results

AddVectorList – add a list of vector results

HasResults – determine if database contains results; returns True or False

parameters:

```
id – integer
entity – string
        NODE
        ELEMENT
type – string
        TEMP
        …
load case – integer
```

example:

```
fres.AddHasResults(1,”NODE”,”TEMP”,1)
```

GetScalarResults – returns scalar result

parameters:

```
id – integer
entity – string
        NODE
        ELEMENT
type – string
        TEMP
load case – integer
```

example:

```
fres.GetScalarResults(1,”NODE”,”TEMP”,1)
```

GetVectorResults – returns vector results

parameters:

```
id – integer
entity – string
        NODE
        ELEMENT
type – string
        DISP
        FORCE
load case – integer
```

example:

```
fres.GetVectorResults(1,”NODE”,”DISP”,1)
```

GetScalarList – returns list of scalar results

GetVectorList – returns list of vector results

AddLoadCaseId – adds a load case id

parameters:

```
id – integer
```

example:

```
fres.AddLoadCaseId(1)
```

GetAvailableLoadCaseIds – returns the load case ids

example:

```
fres.GetAvailableLoadCaseIds()
```

GetNumLoadCases – returns the number of load cases

example:

```
fres.GetNumLoadCases()
```

3.2.4 class Flaw

The Flaw class stores parametric flaw data.

constructor:

Flaw()

example:

```
flaw = PyF3D.Flaw()
```

methods:

Save – saves the flaw data to a file

parameters:

```
file_name - string
```

example:

```
flaw.Save("crack_data.crk")
```

3.2.5 class CrackData

The CrackData class stores the crack growth data.

constructor:

CrackData()

example:

```
cgd = PyF3D.CrackData()
```

methods:

Step – returns the list of crack growth steps

example:

```
cgd.Step()
```

StartPoint – returns the crack start point or origin

example:

```
cgd.StartPoint()
```

StartLength – returns the initial crack length

example:

```
cgd.Startlength()
```

3.2.6 class CrackStep

The CrackStep class stores the crack growth data per step.

constructor:

CrackStep()

example:

```
cgs = PyF3D.CrackStep()
```

methods:

Front – returns the crack front

example:

```
cgs.Front()
```

Name –

UserIndx –

ExtensionAmount –

3.2.7 class CrackFront

The CrackFront class stores the crack growth data per front.

constructor:

CrackFront()

example:

```
cgf = PyF3D.CrackFront()
```

methods:

Point – returns list of front points

example:

```
cgf.Point()
```

Id – returns the crack front ID

example:

```
cgf.Id()
```

StartPoint – returns the start point coordinates of the crack front

example:

```
cgf.StartPoint()
```

StopPoint – returns the end point coordinates of the crack front

example:

```
cgf.StopPoint()
```

FitFront – returns the list of fitted points to the new front

example:

```
cgf.FitFront()
```

FitOldFront – returns the current front points corresponding to the new fitted points

example:

```
cgf.FitOldFront()
```

FitExtens – returns the list of extensions between the current and new front points

example:

```
cgf. FitExtens()
```

ExtensionMult – returns the extension multiplier

example:
```
cgf.ExtensionMult()
```

3.2.8 class FrontPoint

The FrontPoint class stores the crack growth data per crack front point.

constructor:

FrontPoint()

example:
```
fp = PyF3D.FrontPoint()
```

methods:

NPos – returns the normalized position at the point along the crack front

example:
```
fp.NPos()
```

K – returns the SIF values at the point along the crack front for the first load case

example:
```
fp.K()
```

Klc – returns the SIF values at the point along the crack front for all load cases

example:
```
fp.Klc()
```

J – returns the J-integral values at the point along the crack front for the first load case

example:
```
fp.J()
```

Jlc – returns the J-integral values at the point along the crack front for all load cases

example:
```
fp.Jlc()
```

T – returns the T-stress value at the point along the crack front for the first load case

example:

```
fp.T()
```

Tlc – returns the T-stress value at the point along the crack front for all load cases

example:

```
fp.Tlc()
```

Coord – returns the coordinate of the point along the crack front

example:

```
fp.Coord()
```

Axes – returns the local axes at the point along the crack front

example:

```
fp.Axes()
```

KinkAngle – returns the crack growth kink angle at the point along the crack front

example:

```
fp.KinkAngle()
```

Extension – returns the crack extension at the point along the crack front

example:

```
fp.Extension()
```

DKEffEquiv – returns the effective delta K_I value at the point along the crack front

example:

```
fp.DKEffEquiv()
```

REff – returns the effective R-ratio at the point along the crack front

example:

```
fp.REff()
```

NodeId – returns the node ID of the point along the crack front

example:

```
fp.NodeId()
```

ElemId – returns the element ID for the point along the crack front

example:

```
fp.ElemId()
```