Title:	CrossLink - Geometry API
Author(s):	Drayna, Travis William Lang, Laura M. Long, Louie Joseph Stam, Henry Russell
Intended for:	Public release of recent advancements in CrossLink software.
Issued:	2023-01-20

CrossLink

Geometry API

Travis Drayna

Laura Lang

Louie Long

Henry Stam

January 19, 2023

Problem & Solution

Problem

The mesh generation process is very challenging and time consuming when working with complex CAD models. The process of creating and sorting geometric entities into groups appropriate for meshing is labor intensive and prone to error. In addition, the common data exchange formats such as STEP and IGES do not propagate information such as entity names that may be defined in the original model. Finally, entity counts change frequently with parameter variation as a result of tolerance-based geometry operations. Thus, sorting by index does not provide a robust and repeatable means for grouping.

Solution

xGeom is a geometry library that enables the creation of NURBS curves and surfaces via a python scripting interface. xGeom is ideal for studying relatively simple models and is fully integrated with CrossLink's mesh generation capabilities. For more complex models, xCAD is a python-based Creo Parametric CAD model driver that enables the model to be generated, queried, parametrically modified, regenerated, and exported without data loss and in a fully repeatable manner.

xGeom: Overview

xGeom – Overview

Geometry Library

- Implemented in C++
- Shared by GUI, mesh engine, geometry library
- NURBS-based implementation (curves and surfaces)
- Includes STEP file data translator (OCCT)

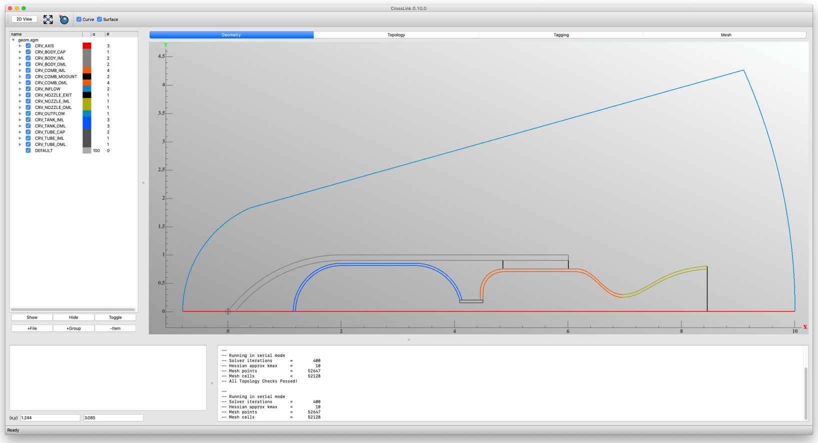
Python API

- Simple python interface
- Enables workflow scripting
- Works with xMesh scripting interface

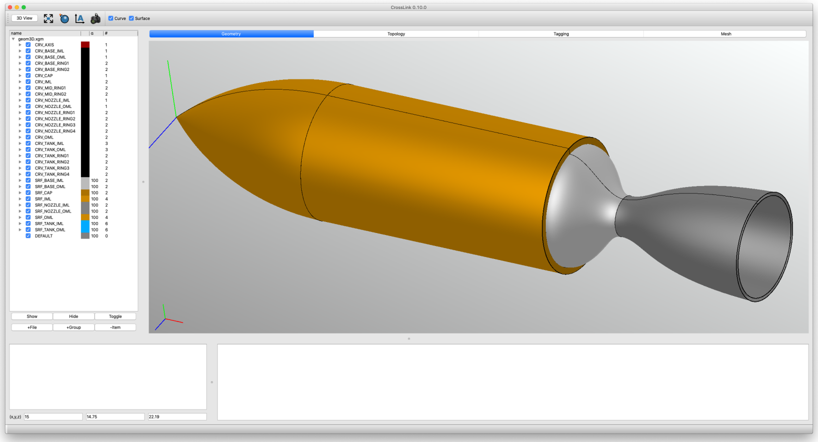
xGeom - Features

- Python API: scriptable and intuitive
- Natively creates 2D/3D NURBS geometry
- Geometry grouping integrated into API
- Currently have basic functionality shown here
- GUI, mesh engine, and geometry API use the same library
- API under active development
- Includes comprehensive unit testing

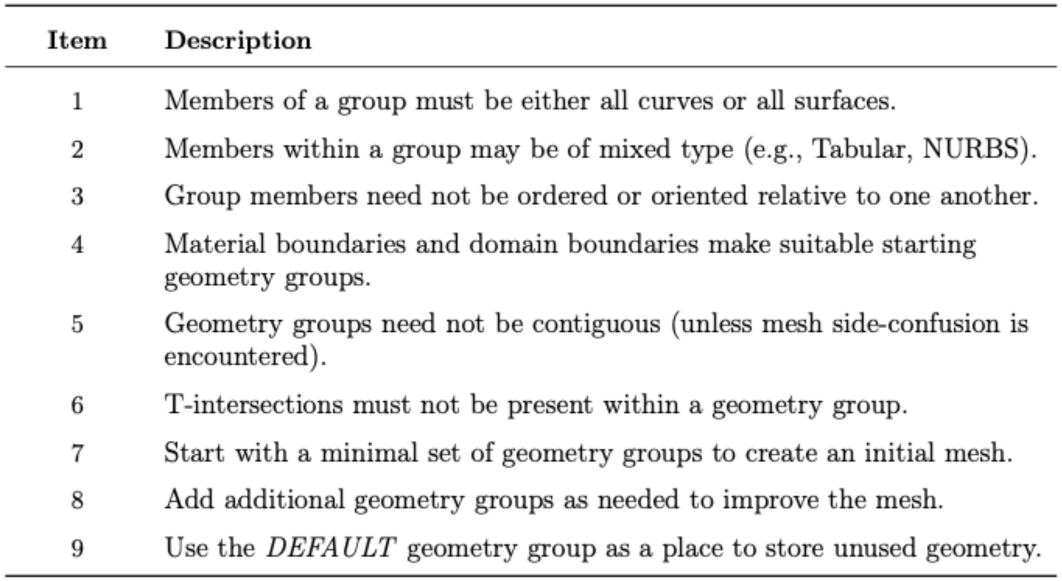
```
xgeom.Group();            // create a geometry group
xgeom.NURBScurve();       // create a NURBS curve explicitly
xgeom.NURBSsurface();     // create a NURBS surface explicitly
xgeom.line();             // create a line between two points
xgeom.arc();              // create a circular arc
xgeom.splitCurve();       // split a curve
xgeom.splitSurface();     // split a surface
xgeom.intersectCurves();  // find the intersection of two curves
xgeom.spin();             // spin curves into surfaces
xgeom.transform();        // transformation operators
```



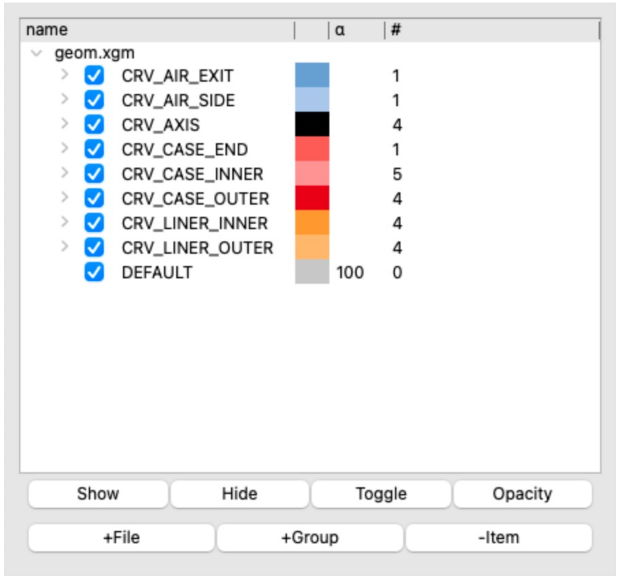
```
--
-- Running in serial mode
-- Solver iterations          =      400
-- Hessian approx kmax        =       10
-- Mesh points                =    52647
-- Mesh cells                 =    52128
-- All Topology Checks Passed!

--
-- Running in serial mode
-- Solver iterations          =      400
-- Hessian approx kmax        =       10
-- Mesh points                =    52647
-- Mesh cells                 =    52128
```



Geometry Groups



Item	Description
1	Members of a group must be either all curves or all surfaces.
2	Members within a group may be of mixed type (e.g., Tabular, NURBS).
3	Group members need not be ordered or oriented relative to one another.
4	Material boundaries and domain boundaries make suitable starting geometry groups.
5	Geometry groups need not be contiguous (unless mesh side-confusion is encountered).
6	T-intersections must not be present within a geometry group.
7	Start with a minimal set of geometry groups to create an initial mesh.
8	Add additional geometry groups as needed to improve the mesh.
9	Use the *DEFAULT* geometry group as a place to store unused geometry.



name		α	#
geom.xgm			
✓ CRV_AIR_EXIT	🟦		1
✓ CRV_AIR_SIDE	🟦		1
✓ CRV_AXIS	⬛		4
✓ CRV_CASE_END	🟥		1
✓ CRV_CASE_INNER	🟥		5
✓ CRV_CASE_OUTER	🟥		4
✓ CRV_LINER_INNER	🟧		4
✓ CRV_LINER_OUTER	🟧		4
✓ DEFAULT	⬜	100	0

- A *geometry group* is a collection of geometric entities that define an internal or external boundary.
- Rules given in table are general guidelines for grouping of curves and surfaces.

xGeom – Example Script

```
################################################################################
#
# Example - Noh2D geometry (xgeom)
#
################################################################################

#### Initialization ####

from xlink import xgeom

#### Define geometric parameters ####

hbox    = 1.00                 # box height
wbox    = 1.00                 # box width
arad    = 0.70                 # arc radius
awall   = 0.10                 # arc wall thickness
sfaci   = (1.0,1.0,0.0)        # inner arc scale factors
sfaco   = (1.0,1.0,0.0)        # outer arc scale factors

#### Create curve groups ####

allGroups = []

DEFAULT       = xgeom.curveGroup("DEFAULT",       "gray1", allGroups)
CRV_BOX_B     = xgeom.curveGroup("CRV_BOX_B",     "black", allGroups)
CRV_BOX_T     = xgeom.curveGroup("CRV_BOX_T",     "blue2", allGroups)
CRV_BOX_R     = xgeom.curveGroup("CRV_BOX_R",     "blue",  allGroups)
CRV_BOX_L     = xgeom.curveGroup("CRV_BOX_L",     "green", allGroups)
CRV_ARC_INNER = xgeom.curveGroup("CRV_ARC_INNER", "red2",  allGroups)
CRV_ARC_OUTER = xgeom.curveGroup("CRV_ARC_OUTER", "white", allGroups)
```

Import Python module

Define parameters

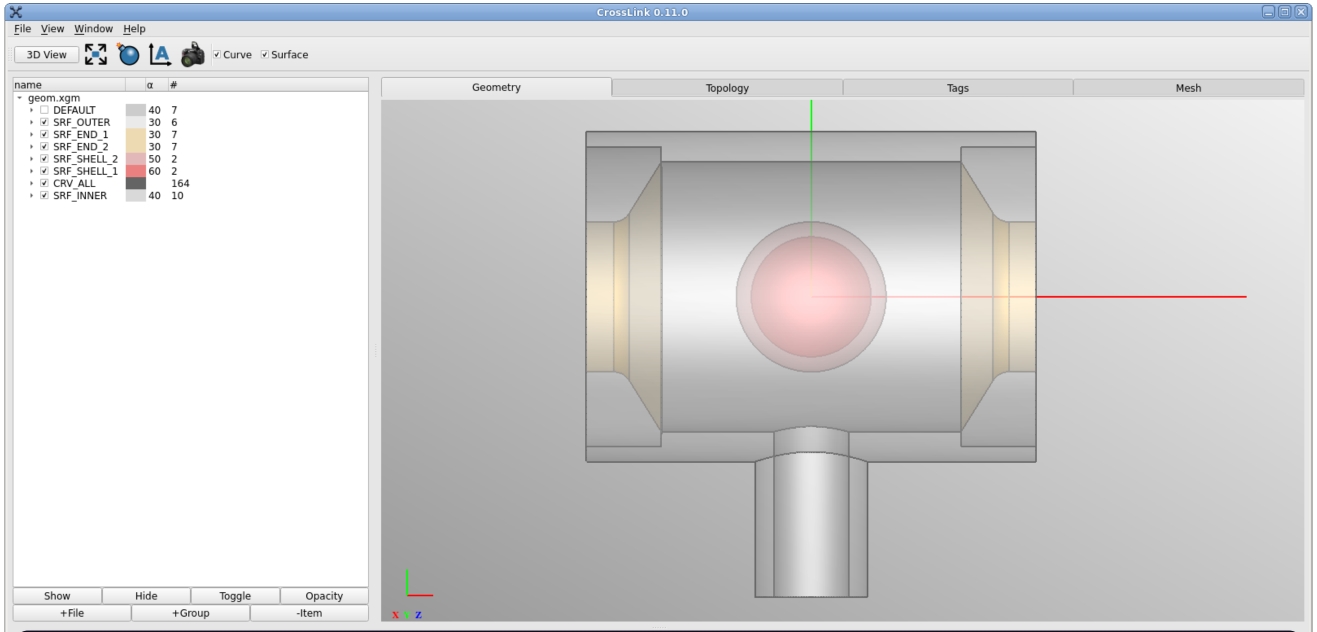
Create geometry groups

```
#### Create geometry ####

CRV_BOX_B.c1     = xgeom.line([ 0.0,  0.0,  0.0],[ wbox, 0.0,  0.0])
CRV_BOX_T.c2     = xgeom.line([ 0.0,  hbox, 0.0],[ wbox, hbox, 0.0])
CRV_BOX_R.c3     = xgeom.line([ wbox, 0.0,  0.0],[ wbox, hbox, 0.0])
CRV_BOX_L.c4     = xgeom.line([ 0.0,  0.0,  0.0],[ 0.0,  hbox, 0.0])
CRV_ARC_OUTER.c5 = xgeom.scale(xgeom.arc( arad, 0.0,  90.0),sfaco)
CRV_ARC_INNER.c6 = xgeom.scale(xgeom.arc( arad-awall, 0.0, 90.0),sfaci)

#### Write XGM file ####

xgeom.writeXGM(allGroups,"geom")

#### End of Script ####
```

Create geometry

Write *xgm* file

CAD Geometry



geom.xgm

name	α	#
☐ DEFAULT	40	7
✓ SRF_OUTER	30	6
✓ SRF_END_1	30	7
✓ SRF_END_2	30	7
✓ SRF_SHELL_2	50	2
✓ SRF_SHELL_1	60	2
✓ CRV_ALL		164
✓ SRF_INNER	40	10

- CrossLink handles geometry created with CAD software.
- These models are translated as BREP NURBS models.
- Supported data exchange format is STEP.

xCAD: Overview

xCAD Overview

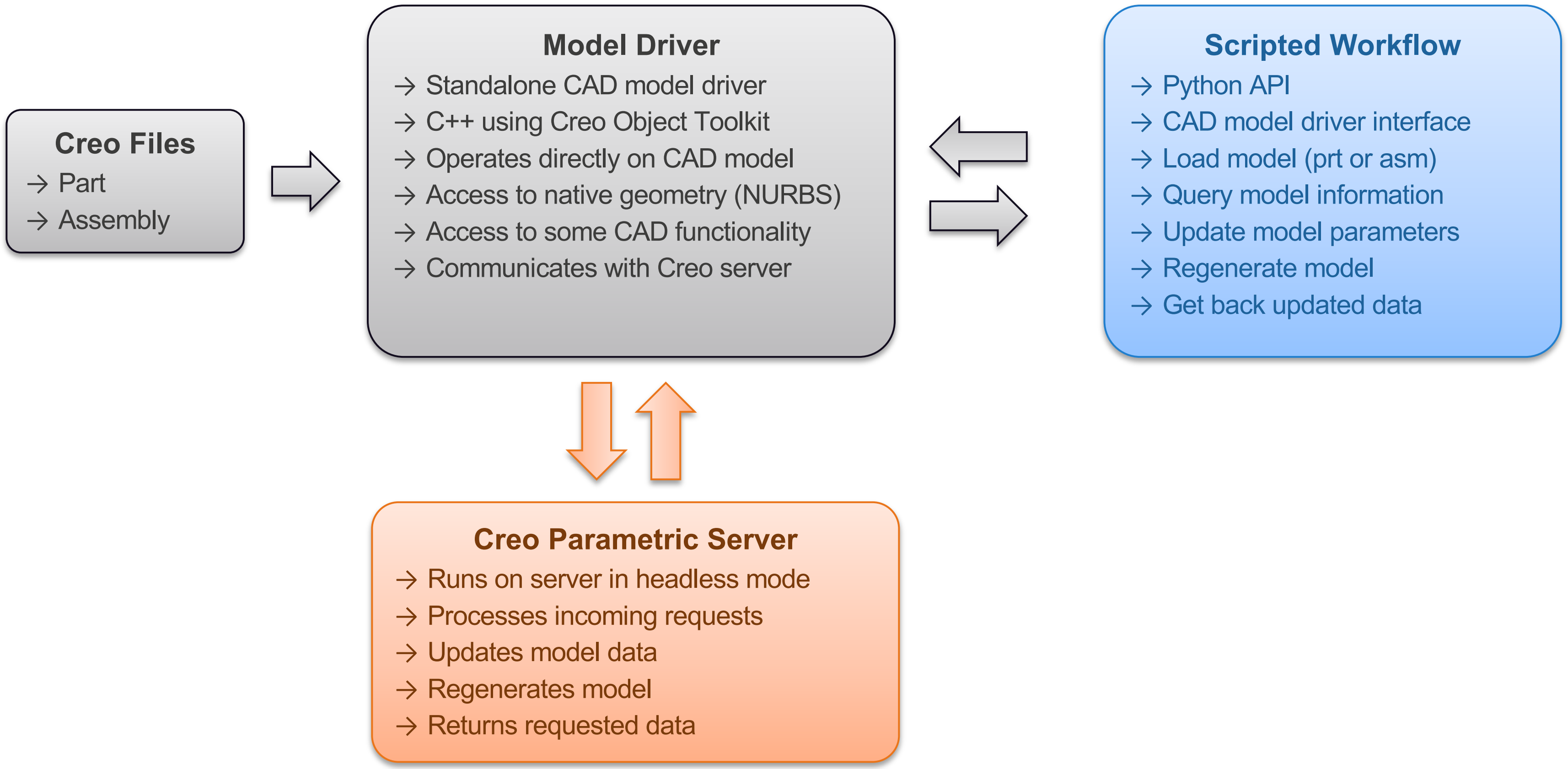
Core Model Driver

- Implemented in C++ using Creo Object Toolkit
- Model driver operates between CAD data and scripted workflow
- Model driver connects to Creo processing server on the network
- Driver loads native Creo part/assembly files
- Driver runs in headless mode (without opening the GUI)
- Driver capable of querying model, modifying model, regenerating model, returning geometric data, returning physical data

Python API

- Simple python interface for CAD model driver
- Integrates into scripted workflow
- Provides source for part/assembly files

xCAD – A Creo Parametric Model Driver



Creo Files
→ Part
→ Assembly

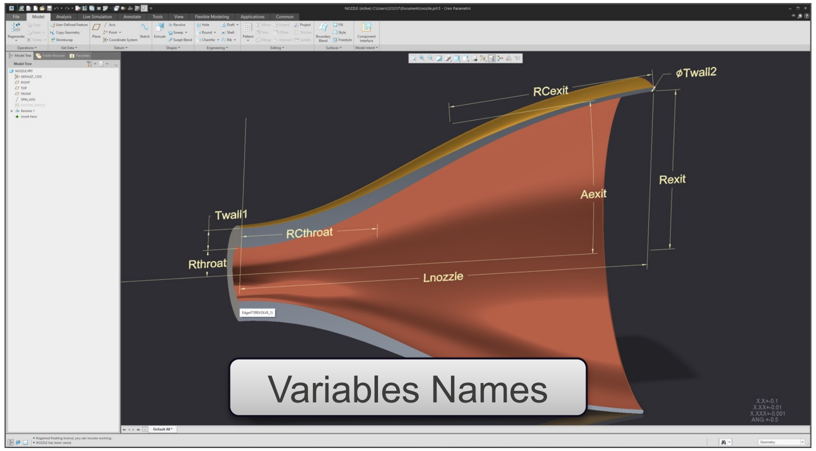
Model Driver
→ Standalone CAD model driver
→ C++ using Creo Object Toolkit
→ Operates directly on CAD model
→ Access to native geometry (NURBS)
→ Access to some CAD functionality
→ Communicates with Creo server

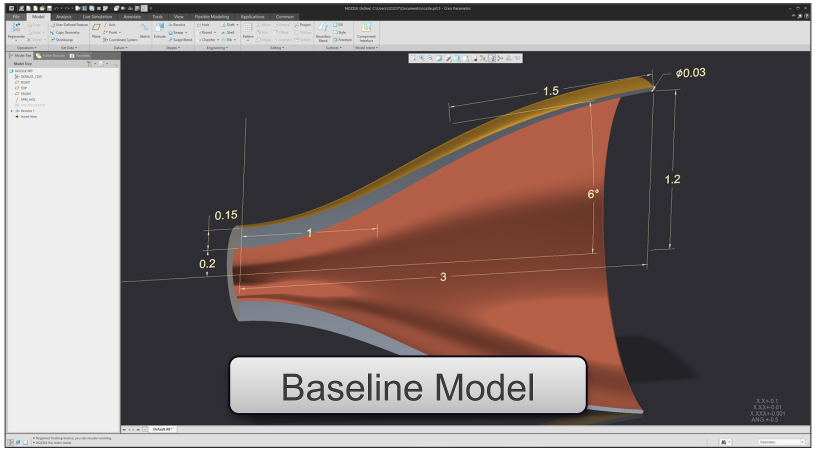
Scripted Workflow
→ Python API
→ CAD model driver interface
→ Load model (prt or asm)
→ Query model information
→ Update model parameters
→ Regenerate model
→ Get back updated data

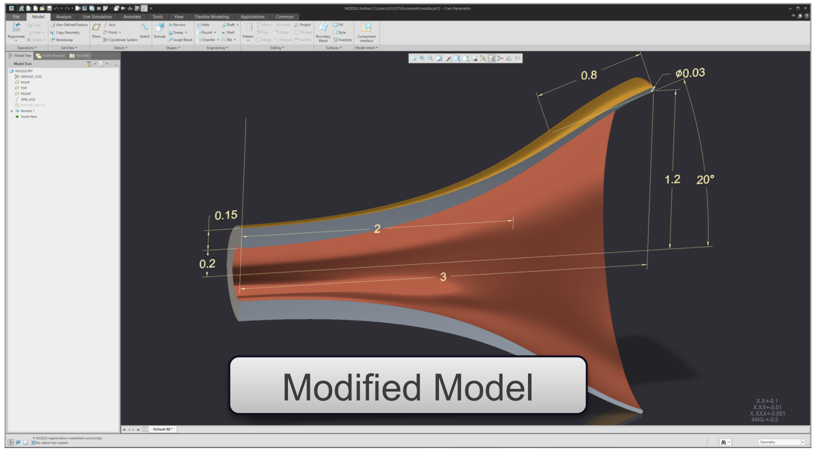
Creo Parametric Server
→ Runs on server in headless mode
→ Processes incoming requests
→ Updates model data
→ Regenerates model
→ Returns requested data

Physics & Engineering Shared CAD Model

- Engineering creates and maintains CAD model
- Engineering & Physics jointly define references, datums, variables, constraints, etc.
- Engineering & Physics jointly define common names for variables
- xCAD enables Physics to make model changes
- Physics explores design effects; recommends new values as needed
- Engineering maintains CAD model standard







Application: Examples

xGeom – Python geometry API (1)

```
#######################################################
# XGEOM - Simple rocket geometry
#######################################################

from xlink import xgeom
import math

#### Geometric Parameters ####

blen  = 4.00       # body length
bwall = 0.10       # body wall thickness
brad  = 1.00       # body radius
rnose = 2.50       # nose radius

tnoffset = 0.0     # nose-tank offset
tlen  = 3.00       # tank length
twall = 0.04       # tank wall thickness
trad  = 0.85       # tank radius

crad  = 0.75       # combustor radius
clen  = 1.30       # combustor length
csrad = 0.40       # combustor shoulder radius
cnfac = 0.40       # combustor-nozzle curvature
cnlen = 0.80       # combustor-nozzle length

tcoffset = 0.3     # tank-combustor offset
tcrad    = 0.2     # tank-combustor tube radius
tcwall   = 0.05    # tank-combustor tube wall thickness

nlen  = 1.50       # nozzle length
nwall = 0.05       # nozzle wall thickness
nrad1 = 0.30       # nozzle throat radius
nrad2 = 0.80       # nozzle exit radius
nphi2 = 4.0        # nozzle exit flare angle
nfac1 = 0.30       # nozzle curvature 1
nfac2 = 0.60       # nozzle curvature 2
```

```
irad  = 2.00       # inflow shock major radius
ilen  = 10.0       # inflow shock length
ibeta = 24.0       # inflow shock angle
ioff  = 1.20       # inflow shock offset

#### Create curve groups ####

DEFAULT          = xgeom.Group(name = "DEFAULT", color = "aaaaaa")
CRV_AXIS         = xgeom.Group(name = "CRV_AXIS", color = "ee0000")
CRV_INFLOW       = xgeom.Group(name = "CRV_INFLOW", color = "0088cc")
CRV_OUTFLOW      = xgeom.Group(name = "CRV_OUTFLOW", color = "0088cc")
CRV_BODY_OML     = xgeom.Group(name = "CRV_BODY_OML", color = "808080")
CRV_BODY_IML     = xgeom.Group(name = "CRV_BODY_IML", color = "808080")
CRV_BODY_CAP     = xgeom.Group(name = "CRV_BODY_CAP", color = "808080")
CRV_COMB_OML     = xgeom.Group(name = "CRV_COMB_OML", color = "ff5500")
CRV_COMB_IML     = xgeom.Group(name = "CRV_COMB_IML", color = "ff5500")
CRV_COMB_MOUNT   = xgeom.Group(name = "CRV_COMB_MOOUNT")
CRV_TUBE_OML     = xgeom.Group(name = "CRV_TUBE_OML", color = "505050")
CRV_TUBE_IML     = xgeom.Group(name = "CRV_TUBE_IML", color = "505050")
CRV_TUBE_CAP     = xgeom.Group(name = "CRV_TUBE_CAP", color = "505050")
CRV_TANK_OML     = xgeom.Group(name = "CRV_TANK_OML", color = "0055ff")
CRV_TANK_IML     = xgeom.Group(name = "CRV_TANK_IML", color = "0055ff")
CRV_NOZZLE_IML   = xgeom.Group(name = "CRV_NOZZLE_IML", color = "aaaa00")
CRV_NOZZLE_OML   = xgeom.Group(name = "CRV_NOZZLE_OML", color = "aaaa00")
CRV_NOZZLE_EXIT  = xgeom.Group(name = "CRV_NOZZLE_EXIT")
```

xGeom – Python geometry API (2)

```python
#### Creat body curves ####

vecx = (-1.0, 0.0, 0.0)
vecy = ( 0.0, 1.0, 0.0)

theta1 = math.acos((rnose - brad)/rnose)
theta2 = math.acos((rnose - brad)/(rnose - bwall))
xref  = rnose*math.sin(theta1)
yref  = -(rnose - brad)

CRV_BODY_OML.cnose_o = xgeom.arc((xref,yref,0.0), rnose, math.degrees(0.5*math.pi-
theta1), 90.0,vecx,vecy)
CRV_BODY_IML.cnose_i = xgeom.arc((xref,yref,0.0), rnose-bwall, math.degrees(0.5*math.pi-
theta2), 90.0,vecx,vecy)

p1 = CRV_BODY_OML.cnose_o(u=1.0)
p2 = CRV_BODY_IML.cnose_i(u=1.0)
lref = p1[0]

CRV_BODY_OML.cside_o = xgeom.line( p1, [lref+blen,brad,0.0])
CRV_BODY_IML.cside_i = xgeom.line( p2, [lref+blen,brad-bwall,0.0])

CRV_BODY_CAP.cap1 = xgeom.line( CRV_BODY_OML.cside_o(u=1.0), CRV_BODY_IML.cside_i(u=1.0))

lref = CRV_BODY_OML.cside_o(u=1.0)[0]

#### Create oxidizer tank ####

lref = CRV_BODY_OML.cnose_o(u=1.0)[0]

CRV_TANK_OML.c1 = xgeom.arc((lref+tnoffset,0.0,0.0), trad, 90.0, 180.0)
CRV_TANK_IML.c1 = xgeom.arc((lref+tnoffset,0.0,0.0), trad-twall, 90.0, 180.0)

p1 = CRV_TANK_OML.c1(u=0.0)
p2 = CRV_TANK_OML.c1(u=0.0)
p2[0] = p2[0] + tlen - 2.0*trad

CRV_TANK_OML.c2 = xgeom.line(p1,p2)

p1 = CRV_TANK_IML.c1(u=0.0)
p2 = CRV_TANK_IML.c1(u=0.0)
p2[0] = p2[0] + tlen - 2.0*trad
```

```python
CRV_TANK_IML.c2 = xgeom.line(p1,p2)

lref =  CRV_TANK_OML.c2(u=1.0)[0]
theta1 = math.asin(tcrad/trad)
theta2 = math.asin(tcrad/(trad-twall))

CRV_TANK_OML.c3 = xgeom.arc((lref,0.0,0.0), trad, math.degrees(theta1),
90.0)
CRV_TANK_IML.c3 = xgeom.arc((lref,0.0,0.0), trad-twall,
math.degrees(theta2), 90.0)

#### Create combustor ####

xref =  CRV_TANK_OML.c2(u=1.0)[0] + trad + tcoffset

p1 = [xref, tcrad, 0.0]
p2 = [xref, crad-csrad, 0.0]
p3 = [xref+nwall, tcrad, 0.0]
p4 = [xref+nwall, crad-csrad, 0.0]

CRV_COMB_OML.c1 = xgeom.line(p1,p2)
CRV_COMB_IML.c1 = xgeom.line(p3,p4)

CRV_COMB_OML.c2 = xgeom.arc((xref+csrad,crad-csrad,0.0), csrad, 90.0, 180.0)
CRV_COMB_IML.c2 = xgeom.arc((xref+csrad,crad-csrad,0.0), csrad-nwall, 90.0,
180.0)

xref = CRV_COMB_OML.c2(u=0.0)[0]

p1 = [xref,      crad, 0.0]
p2 = [xref+clen, crad, 0.0]
p3 = [xref,      crad-nwall, 0.0]
p4 = [xref+clen, crad-nwall, 0.0]

CRV_COMB_OML.c3 = xgeom.line(p1,p2)
CRV_COMB_IML.c3 = xgeom.line(p3,p4)
```

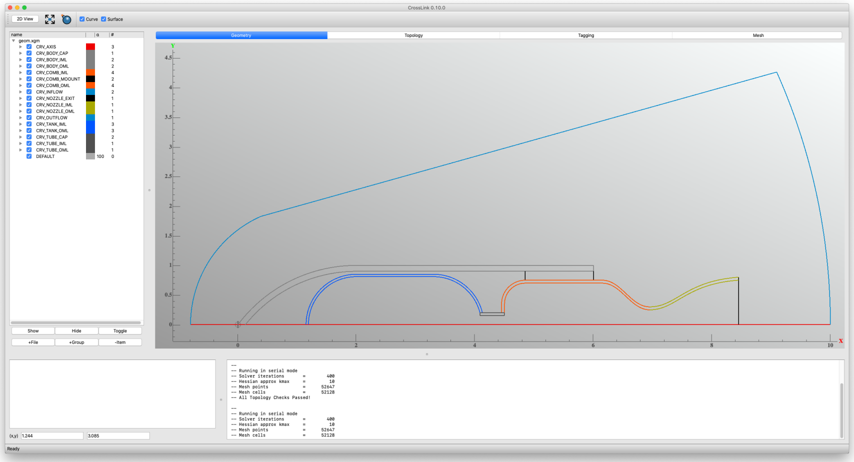
xGeom – Python geometry API (3)

```
xref = CRV_COMB_IML.c3(u=1.0)[0]

p1 = [xref, crad, 0.0, 1.0]
p2 = [xref+cnfac*cnlen, crad, 0.0, 1.0]
p3 = [xref+(1.0-cnfac)*cnlen, nrad1, 0.0, 1.0]
p4 = [xref+cnlen, nrad1, 0.0, 1.0]

CRV_COMB_OML.c4 = xgeom.NURBScurve(udeg=3, knots=[0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0,
1.0], xcp=[p1,p2,p3,p4])

p1 = [xref, crad-nwall, 0.0, 1.0]
p2 = [xref+cnfac*cnlen, crad-nwall, 0.0, 1.0]
p3 = [xref+(1.0-cnfac)*cnlen, nrad1-nwall, 0.0, 1.0]
p4 = [xref+cnlen, nrad1-nwall, 0.0, 1.0]

CRV_COMB_IML.c4 = xgeom.NURBScurve(udeg=3, knots=[0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0,
1.0], xcp=[p1,p2,p3,p4])

#### Create combustor mount ####

p1 = CRV_COMB_OML.c2(u=0.0)
p2 = [p1[0], brad-bwall, 0.0]

CRV_COMB_MOUNT.c1 = xgeom.line(p1,p2)

xref = CRV_BODY_IML.cnose_i(u=1.0)[0] + blen

p1 = [xref, crad , 0.0]
p2 = [xref, brad-bwall, 0.0]

CRV_COMB_MOUNT.c2 = xgeom.line(p1,p2)

#### Create tank-to-combustor tube ####

x1 = CRV_TANK_IML.c3(u=0.0)[0]
x2 = CRV_COMB_IML.c2(u=1.0)[0]

p1 = [x1, tcrad, 0.0]
p2 = [x2, tcrad, 0.0]
p3 = [x1, tcrad-tcwall, 0.0]
p4 = [x2, tcrad-tcwall, 0.0]
```

```
CRV_TUBE_OML.c1 = xgeom.line(p1,p2)
CRV_TUBE_IML.c1 = xgeom.line(p3,p4)
CRV_TUBE_CAP.c1 = xgeom.line(p1,p3)
CRV_TUBE_CAP.c2 = xgeom.line(p2,p4)

#### Create nozzle ####

xref = CRV_COMB_OML.c4(u=1.0)[0]

p1 = [xref, nrad1, 0.0, 1.0]
p2 = [xref + nlen*nfac1, nrad1, 0.0, 1.0]
p3 = [xref + (1.0-nfac2)*nlen, nrad2 -
math.tan(math.radians(nphi2))*nlen*nfac2, 0.0, 1.0]
p4 = [xref+nlen,nrad2,0.0, 1.0]

CRV_NOZZLE_OML.n1 = xgeom.NURBScurve(udeg=3, knots=[0.0, 0.0, 0.0, 0.0, 1.0,
1.0, 1.0, 1.0], xcp=[p1,p2,p3,p4])

p1 = [xref, nrad1-nwall, 0.0, 1.0]
p2 = [xref + nlen*nfac1, nrad1-nwall, 0.0, 1.0]
p3 = [xref + (1.0-nfac2)*nlen, nrad2 - nwall -
math.tan(math.radians(nphi2))*nlen*nfac2, 0.0, 1.0]
p4 = [xref+nlen,nrad2-nwall,0.0, 1.0]

CRV_NOZZLE_IML.n2 = xgeom.NURBScurve(udeg=3, knots=[0.0, 0.0, 0.0, 0.0, 1.0,
1.0, 1.0, 1.0], xcp=[p1,p2,p3,p4])

px = CRV_NOZZLE_OML.n1(u=1.0)
p1 = [px[0], 0.0, 0.0]
p2 = [px[0], px[1], 0.0]

CRV_NOZZLE_EXIT.n3 = xgeom.line(p1,p2)
```

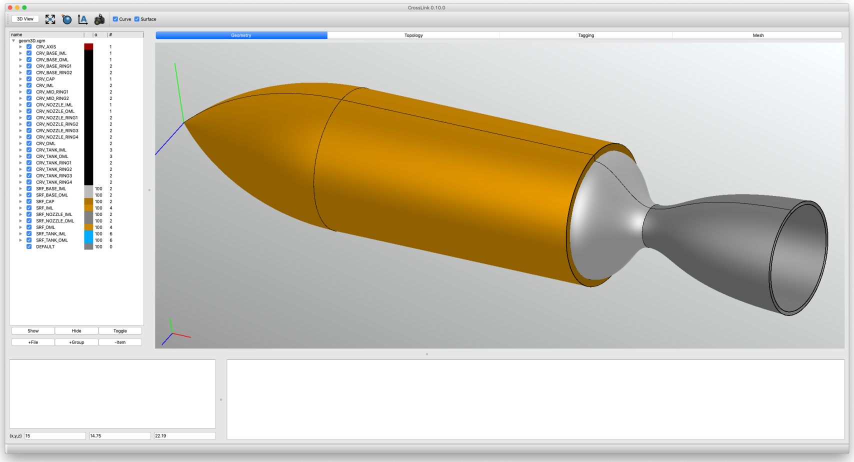
xGeom – Python geometry API (4)

```
#### Create inflow ####

CRV_INFLOW.c1 = xgeom.arc((0.0+ioff,0.0,0.0), irad, 90.0+ibeta, 180.0)

lref = CRV_INFLOW.c1(u=0.0)[0]/math.tan(math.radians(ibeta))
xref = CRV_INFLOW.c1(u=0.0)[0] - lref
rad = ilen - xref

CRV_OUTFLOW.c1 = xgeom.arc((xref,0.0,0.0), rad, 0.0, ibeta)

p1 = CRV_INFLOW.c1(u=0.0)
p2 = CRV_OUTFLOW.c1(1.0)

CRV_INFLOW.c2 = xgeom.line(p1,p2)

#### Create the axis ####

x1 = CRV_INFLOW.c1(u=1.0)[0]
x2 = CRV_BODY_OML.cnose_o(u=0.0)[0]
x3 = CRV_NOZZLE_OML.n1(u=1.0)[0]
x4 = CRV_OUTFLOW.c1(0.0)[0]

CRV_AXIS.a1 = xgeom.line([x1,0.0,0.0], [x2,0.0,0.0])
CRV_AXIS.a2 = xgeom.line([x2,0.0,0.0], [x3,0.0,0.0])
CRV_AXIS.a3 = xgeom.line([x3,0.0,0.0], [x4,0.0,0.0])

#### Write XGM file ####

xgeom.writeXGM(grouplist = [DEFAULT, CRV_AXIS, CRV_INFLOW, CRV_OUTFLOW,
CRV_BODY_IML,CRV_BODY_OML,CRV_BODY_CAP,
                            CRV_COMB_OML,CRV_COMB_IML,CRV_COMB_MOUNT,
CRV_TUBE_OML,CRV_TUBE_IML,CRV_TUBE_CAP,
                            CRV_TANK_IML,CRV_TANK_OML,
CRV_NOZZLE_IML,CRV_NOZZLE_OML,CRV_NOZZLE_EXIT
                           ], filename = "geom")
```



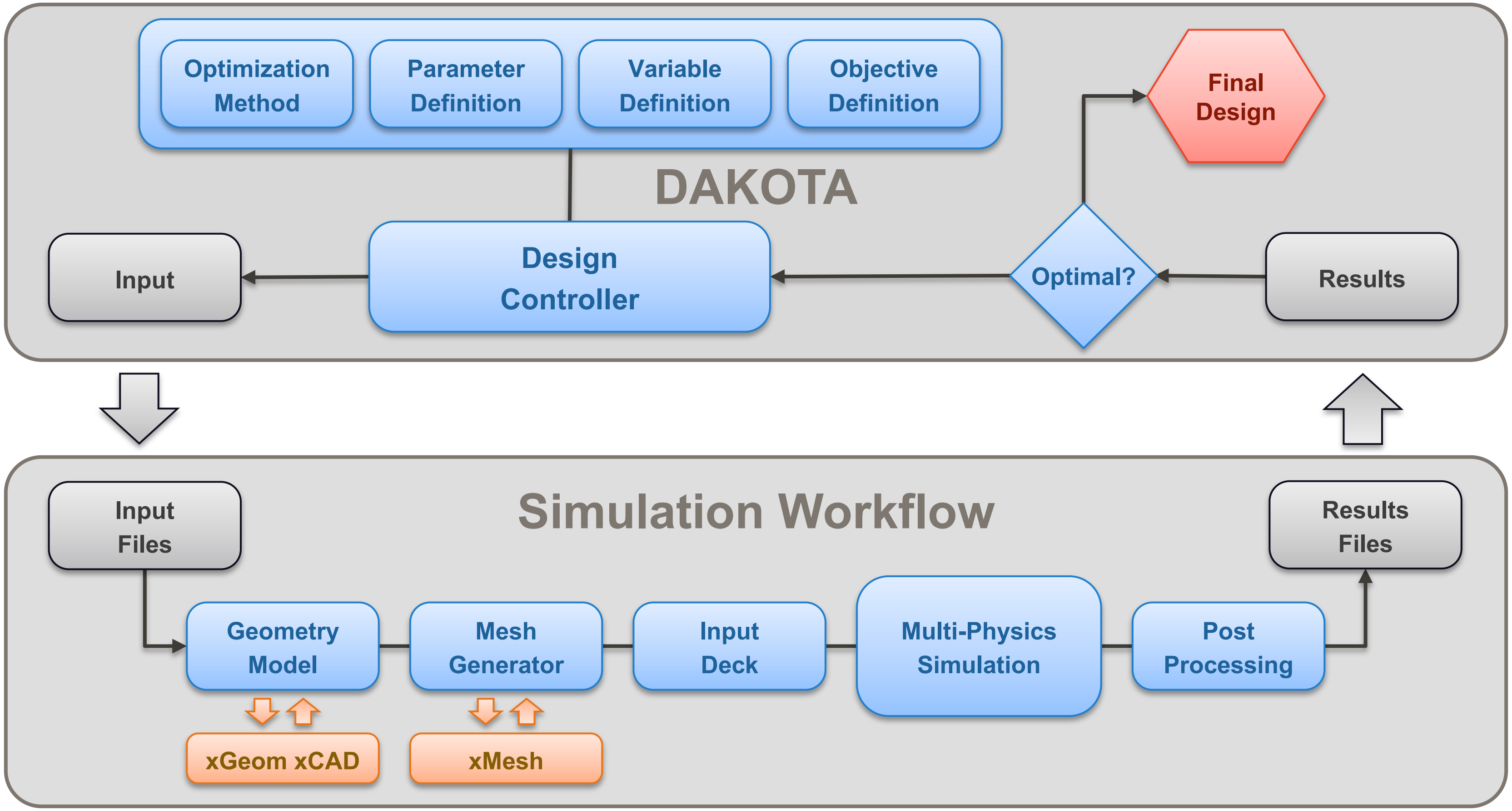
name		α	#
geom.xgm			
CRV_AXIS	🟥		3
CRV_BODY_CAP	▨		1
CRV_BODY_IML	▨		2
CRV_BODY_OML	▨		2
CRV_COMB_IML	🟧		4
CRV_COMB_MOOUNT	⬛		2
CRV_COMB_OML	🟧		4
CRV_INFLOW	🟦		2
CRV_NOZZLE_EXIT	⬛		1
CRV_NOZZLE_IML	🟨		1
CRV_NOZZLE_OML	🟨		1
CRV_OUTFLOW	🟦		1
CRV_TANK_IML	🟦		3
CRV_TANK_OML	🟦		3
CRV_TUBE_CAP	▨		2
CRV_TUBE_IML	▨		1
CRV_TUBE_OML	▨		1
DEFAULT	▨	100	0

```
--
-- Running in serial mode
-- Solver iterations        =      400
-- Hessian approx kmax      =       10
-- Mesh points              =    52647
-- Mesh cells               =    52128
-- All Topology Checks Passed!

--
-- Running in serial mode
-- Solver iterations        =      400
-- Hessian approx kmax      =       10
-- Mesh points              =    52647
-- Mesh cells               =    52128
```



name		α	#
geom3D.xgm			
CRV_AXIS	🟥		1
CRV_BASE_IML	⬛		1
CRV_BASE_OML	⬛		1
CRV_BASE_RING1	⬛		2
CRV_BASE_RING2	⬛		2
CRV_CAP	⬛		1
CRV_IML	⬛		2
CRV_MID_RING1	⬛		2
CRV_MID_RING2	⬛		2
CRV_NOZZLE_IML	⬛		1
CRV_NOZZLE_OML	⬛		1
CRV_NOZZLE_RING1	⬛		2
CRV_NOZZLE_RING2	⬛		2
CRV_NOZZLE_RING3	⬛		2
CRV_NOZZLE_RING4	⬛		2
CRV_OML	⬛		2
CRV_TANK_IML	⬛		3
CRV_TANK_OML	⬛		3
CRV_TANK_RING1	⬛		2
CRV_TANK_RING2	⬛		2
CRV_TANK_RING3	⬛		2
CRV_TANK_RING4	⬛		2
SRF_BASE_IML	▨	100	2
SRF_BASE_OML	▨	100	2
SRF_CAP	🟧	100	2
SRF_IML	🟧	100	4
SRF_NOZZLE_IML	▨	100	2
SRF_NOZZLE_OML	▨	100	2
SRF_OML	🟧	100	4
SRF_TANK_IML	🟦	100	6
SRF_TANK_OML	🟦	100	6
DEFAULT	▨	100	0

Design Optimization


Optimization Method
Parameter Definition
Variable Definition
Objective Definition
Final Design
DAKOTA
Input
Design Controller
Optimal?
Results
Simulation Workflow
Input Files
Results Files
Geometry Model
Mesh Generator
Input Deck
Multi-Physics Simulation
Post Processing
xGeom xCAD
xMesh