Automating STAR-CCM+

Speakers:

1. Adam Green, ATA Engineering Inc.
2. Tim Palmer, ATA Engineering Inc.

Date:

6/19/2019

13290 Evening Creek Drive S, Suite 250, San Diego CA 92128

(858) 480-2000

www.ata-e.com

ata-engineering

@ATAEngineering

Agenda

Automating STAR-CCM+

1. Automation with the GUI – Design Manager
2. Basic JavaScript
   - Using Java to use adaptive meshing to track a shock in a transonic simulation
3. Advanced automation with JavaScript:
   - Modeling 100s of geometric shapes using Java to assess boundary size and meshing, then run.

13290 Evening Creek Drive S, Suite 250, San Diego CA 92128

(858) 480-2000

www.ata-e.com

ata-engineering

@ATAEngineering

Who We Are

We are an employee-owned small business with a full-time staff of around 150, more than 125 of whom are degreed engineers

14
Registered Professional Engineers

15
Average years of experience

ENGINEERING, INC.

What We Do

ATA Engineering's high-value engineering services help solve the most challenging product design challenges

Aerospace

Robotics & Controls

Themed Entertainment

Defense

Industrial & Mining Equipment

Consumer Products

ATA Engineering - Timeline

➢ A Legacy of Engineering Excellence:

1975

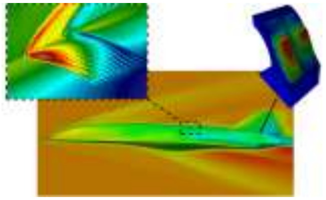
SDRC was an early pioneer of CAE tools starting in 1967.

After a series of acquisitions, SDRC was purchased by Siemens and their I-DEAS software was integrated with Unigraphics into the well known NX product line.

2000

ATA Engineering was formed in April 2000 after a management buyout from SRDC of the Advanced Test and Analysis Division.

Given this shared corporate heritage, ATA maintains its strong relationship with Siemens today



2002

ATA opens Eastern regional Office (ERO) in Herndon VA

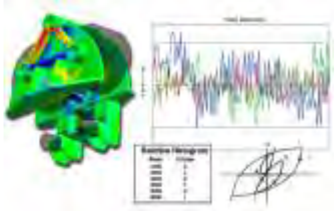
2005

ATA opens LA office in the heart of the Southern California Aerospace Industry

2007

2007

ATA opens Denver office and labels it RMO: Rocky Mountain Office

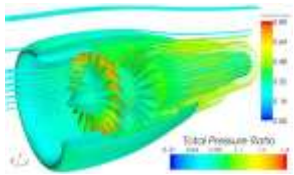


2009

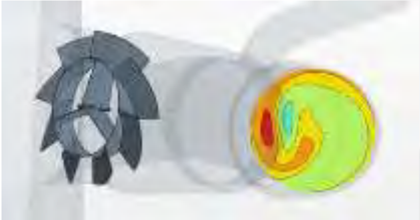
2009

ATA opens Huntsville Office to service South Eastern Aerospace clients

ATA becomes a full VAR for Siemens selling NX, Femap and Nastran



2010



2018

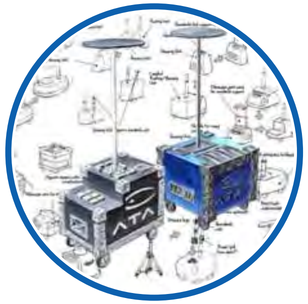
2018

ATA extends Siemens VAR relationship to include Sales and Support for STAR-CCM+ & HEEDS

ATA opens Berkeley, California Office

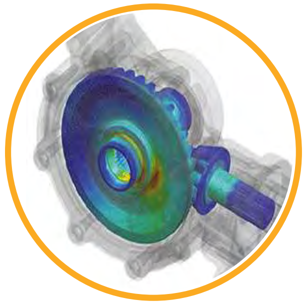
Our Services

We provide our customers with complete, integrated solutions



Design

From initial concept development to detailed structural design



Analysis

Comprehensive structural, fluid, acoustic, and thermal analysis services

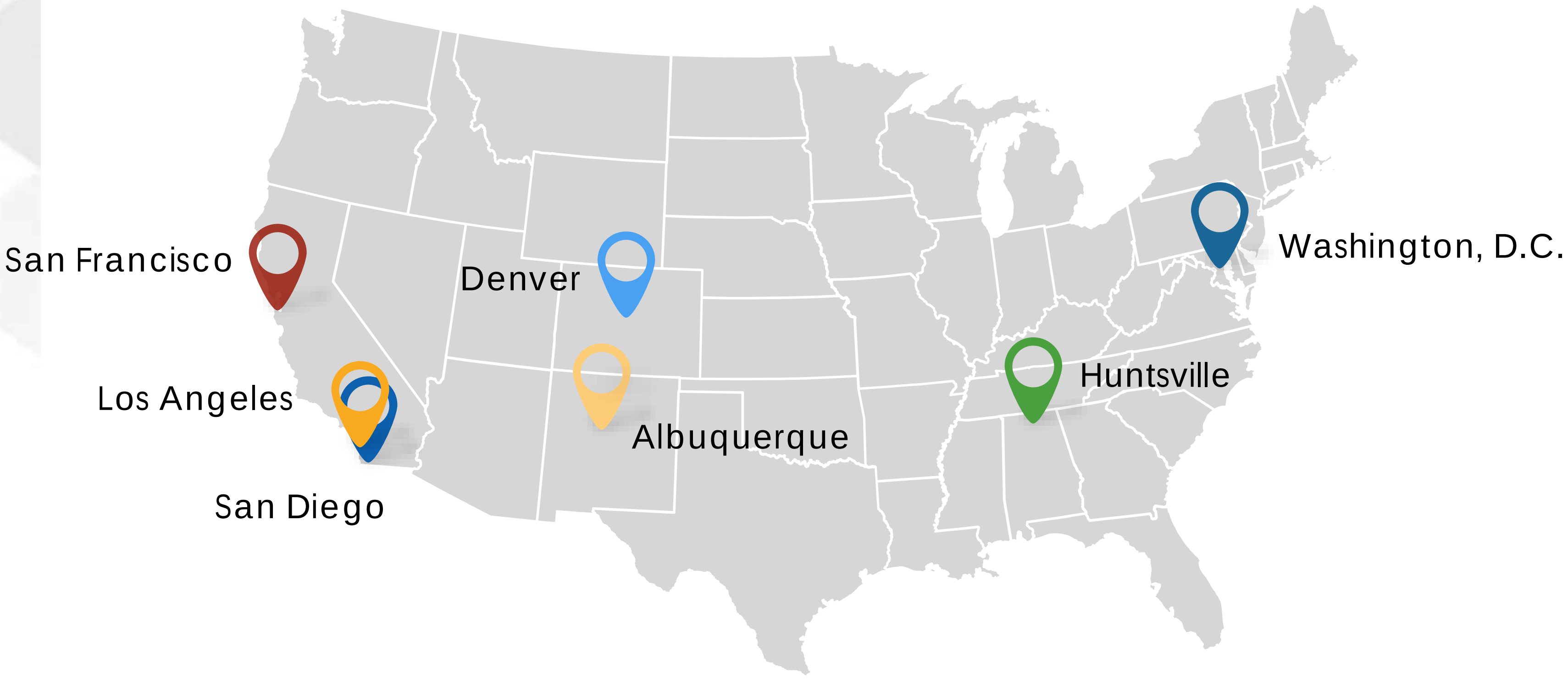
Test

Industry-leading structural test services for extreme loading environments

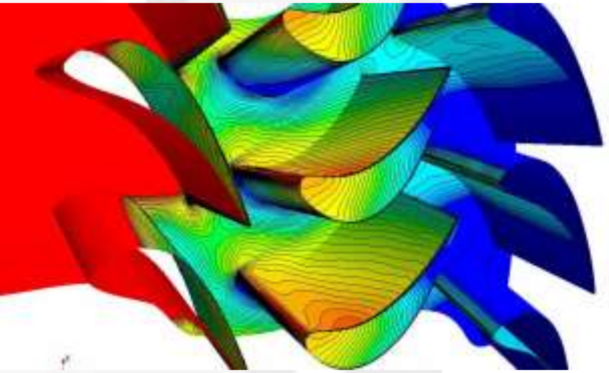
ATA ENGINEERING, INC.

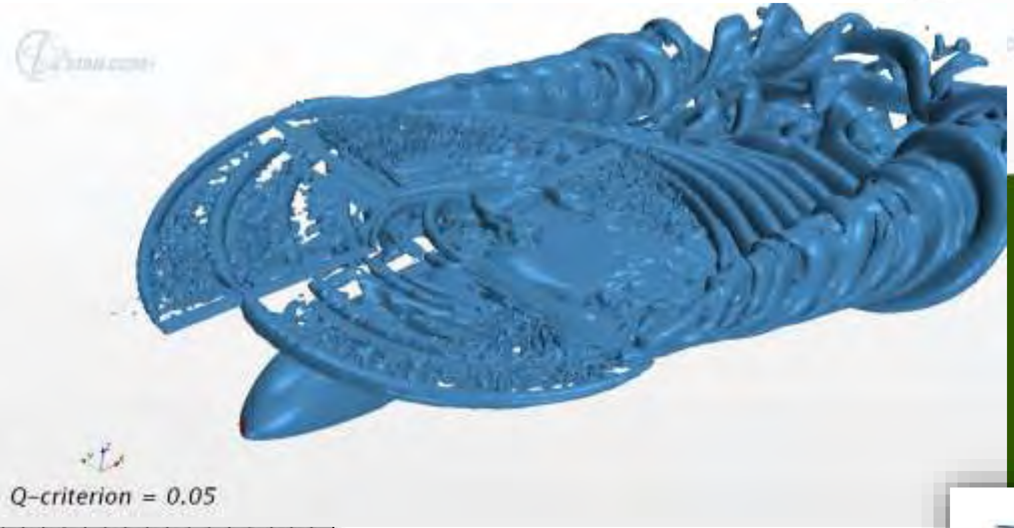
Our Offices

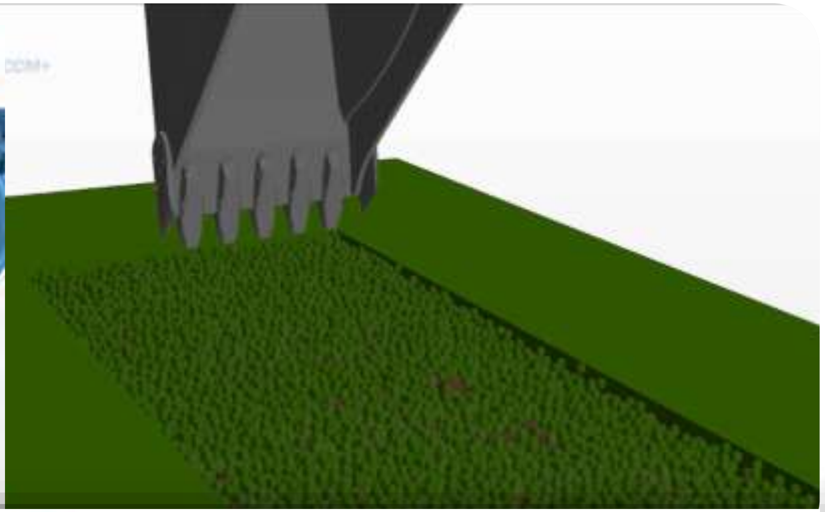
Our 7 nationwide locations provide local full-service capabilities and personal support to our customers

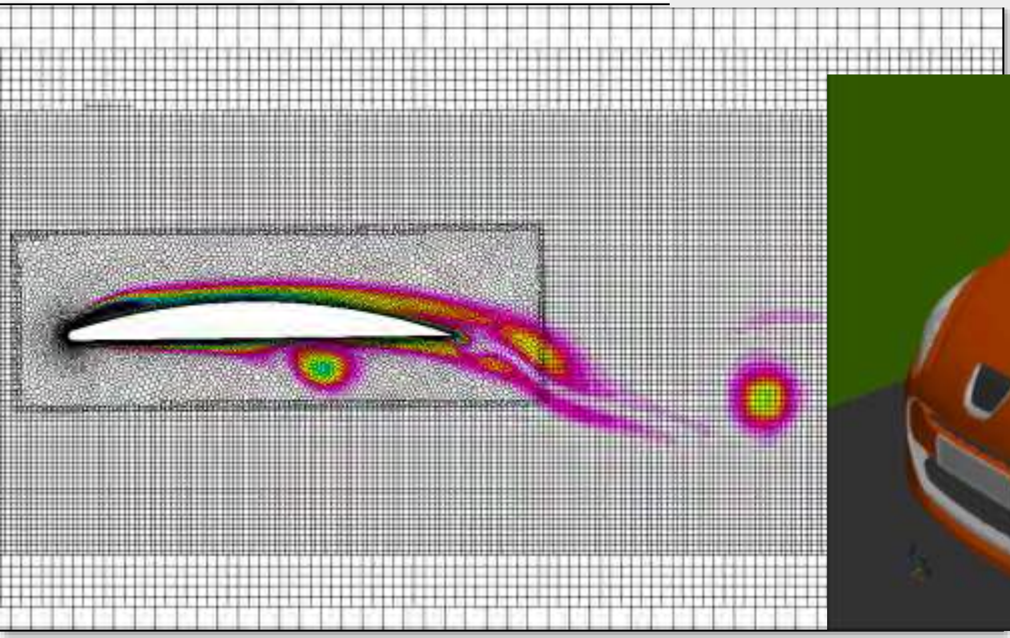


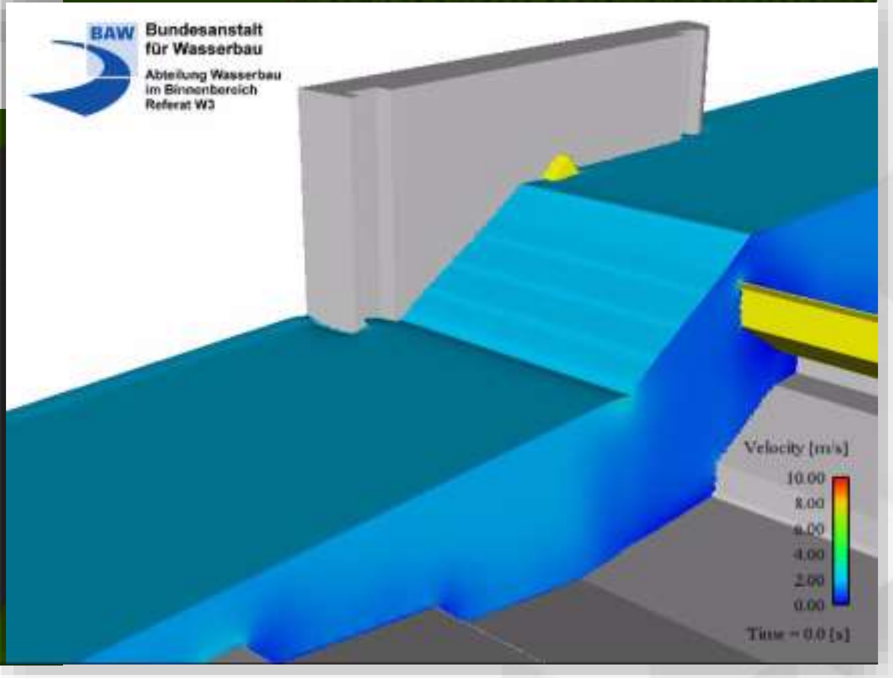
STAR-CCM+®




Q-criterion = 0.05






BAW
Bundesanstalt für Wasserbau
Abteilung Wasserbau im Binnenbereich Referat W3
Velocity [m/s]
10.00
8.00
6.00
4.00
2.00
0.00
Time = 0.0 [s]

Agenda

Automating STAR-CCM+

1. Automation with the GUI – Design Manager
2. Basic JavaScript
   - Using Java to use adaptive meshing to track a shock in a transonic simulation
3. Advanced automation with JavaScript:
   - Modeling 100s of geometric shapes using Java to assess boundary size and meshing, then run.

ATA ENGINEERING, INC.

Design Manager - built-in feature for Design Exploration

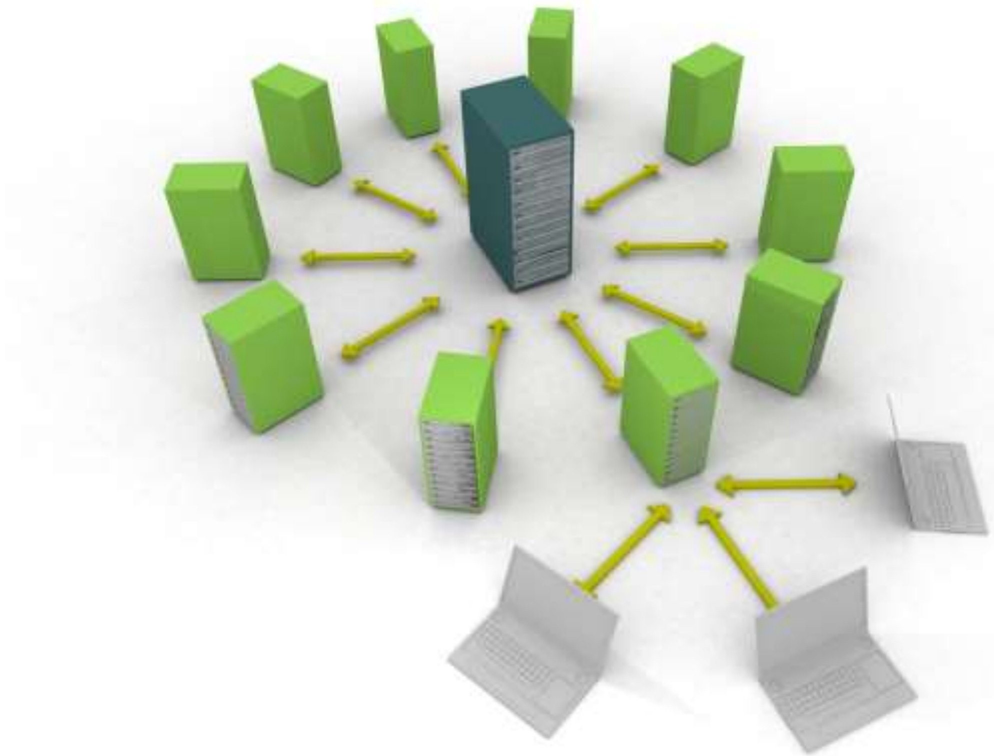
BUILD → TEST → EXPLORE → ASSESS

Design Manager

Simcenter STAR-CCM+ built-in feature for Design Exploration

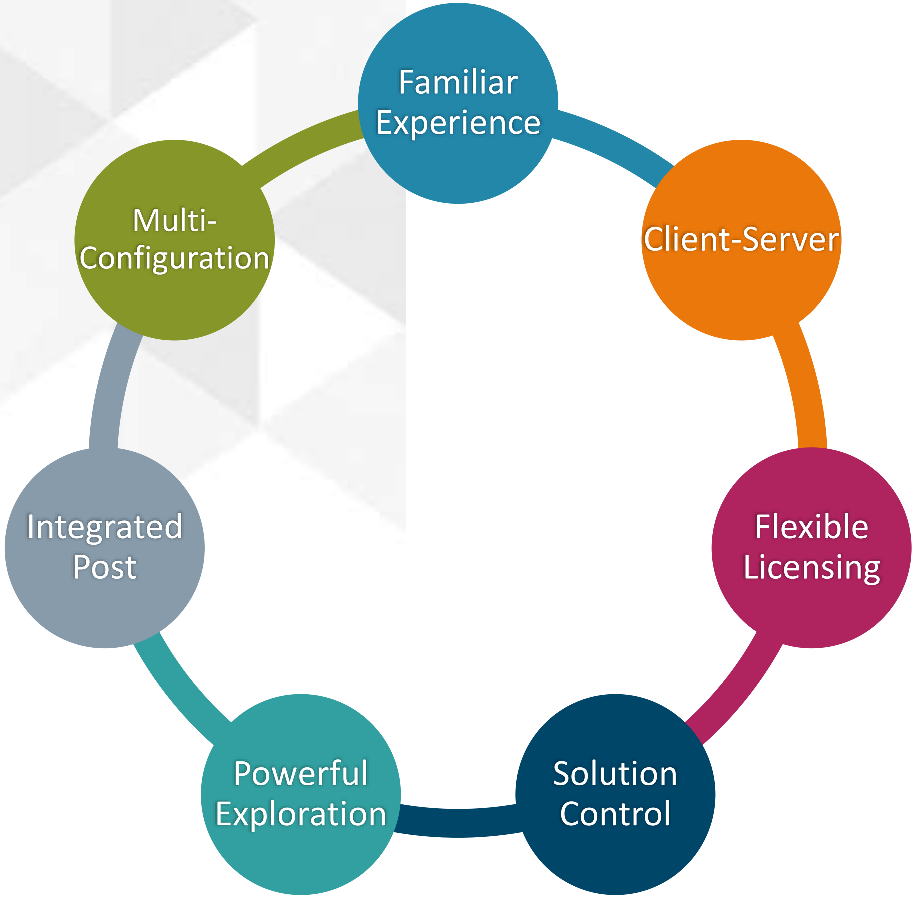
STAR-CCM+

- Integrated in Simcenter STAR-CCM+ for a familiar experience
- Automated design sweeps workflow accessible to every user
- Intelligent automated search with STAR-Innovate add-on license
- Powerful post-processing to quickly navigate and analyse results
- Multiple flexible comparison options to increase product knowledge
- Flexible licensing to deploy design exploration



ATA ENGINEERING, INC.

Design Manager



Entirely native Simcenter STAR-CCM+ design exploration feature

Integrated in the Simcenter STAR-CCM+ environment

- Easy to use, familiar experience & Rapid pace of development

Design Manager

- Facilitating parametric analysis
- No additional license required to enable sweeps

Design manager with STAR-Innovate add-on

- Explore the entire design space and find better designs
- Includes HEEDS technology for automatic intelligent search

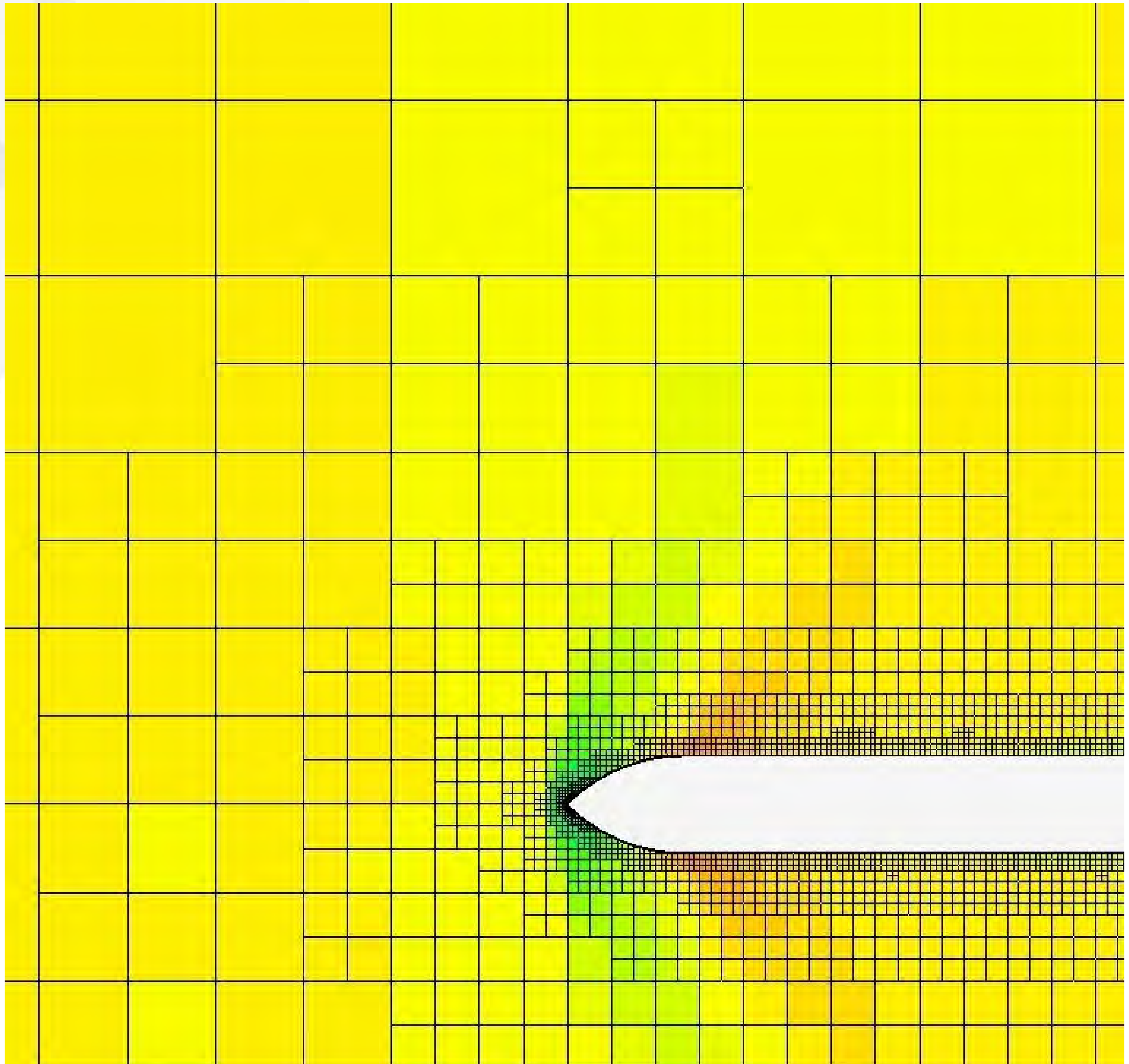
Agenda

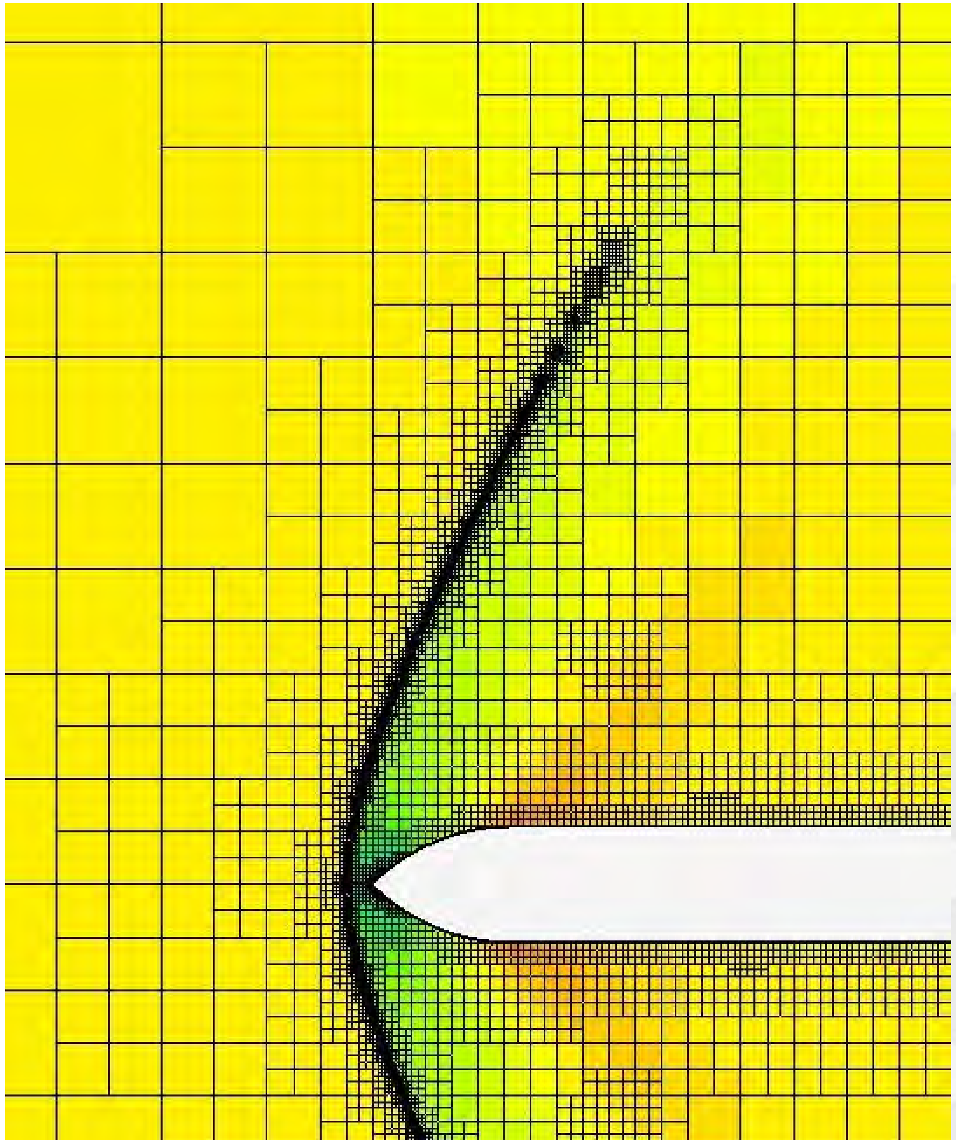
Automating STAR-CCM+

1. Automation with the GUI – Design Manager
2. Basic JavaScript
   - Using Java to use adaptive meshing to track a shock in a transonic simulation
3. Advanced automation with JavaScript:
   - Modeling 100s of geometric shapes using Java to assess boundary size and meshing, then run.

Java based mesh refinement

Shock Tracking





Java based mesh refinement

Shock Tracking

1. Generate an initial coarse mesh on the geometry.
2. Create a field function to threshold the cells that require refining by looking at the gradient of a scalar. For this example we will use Density.
3. Create a field function to specify the new cell size in the flagged cells and cap the minimum size for refinement.
4. Create a Refinement table with the refinement field function as the scalar and the region as your part, extract the values.
5. Add the Refinement table to your volume mesher expert options under 'Field Function Refinement table'
6. Re-volume mesh and you should see a solution based refinement.
7. This can be automated using a Java macro for several iterations of the refinement.

ATA ENGINEERING, INC.

Java based mesh refinement

Shock Tracking

```
public class Refinement2 extends StarMacro {

    public void execute() {
        execute0();
    }
private void execute0() {

        //INPUT DATA
        int totalRefineSteps = 20;
        int iterations = 150;
        int totalSteps = 5000;
        //DECLARE VARIABLES
```

Java based mesh refinement

Shock Tracking

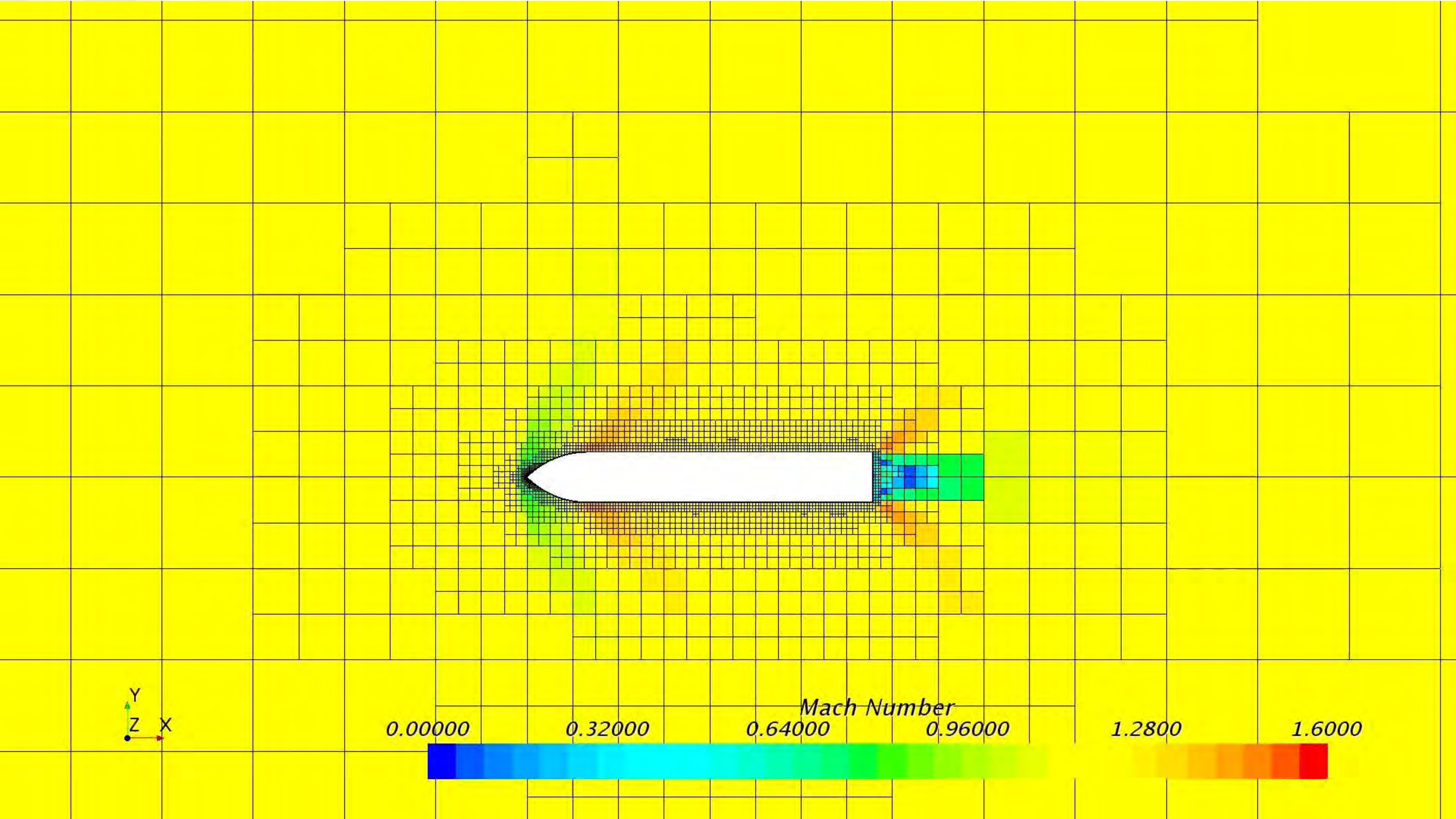
```
Simulation sim = getActiveSimulation(); //Get Current Simulation into Object 'sim'

//Set Maximum Steps Stopping Criteria
StepStoppingCriterion maxsteps = ((StepStoppingCriterion) sim.getSolverStoppingCriterionManager().getSolverStoppingCriterion("Maximum Steps"));
maxsteps.setMaximumNumberSteps(totalSteps);

//Get Automated Mesh into Object 'mesher'
AutoMeshOperation mesher = ((AutoMeshOperation) sim.get(MeshOperationManager.class).getObject("Automated Mesh"));

//Get Automated Mesh: Trimmed Cell Mesher Model into Object 'mesherTrim'
TrimmerAutoMesher mesherTrim =  ((TrimmerAutoMesher) mesher.getMeshers().getObject("Trimmed Cell Mesher"));

//Get Refine table into Object 'refineTable'
XyzInternalTable refineTable = ((XyzInternalTable) sim.getTableManager().getTable("Refine"));

//Get Simulation Iterator into Object 'simrunner'
SimulationIterator simrunner = sim.getSimulationIterator()
```

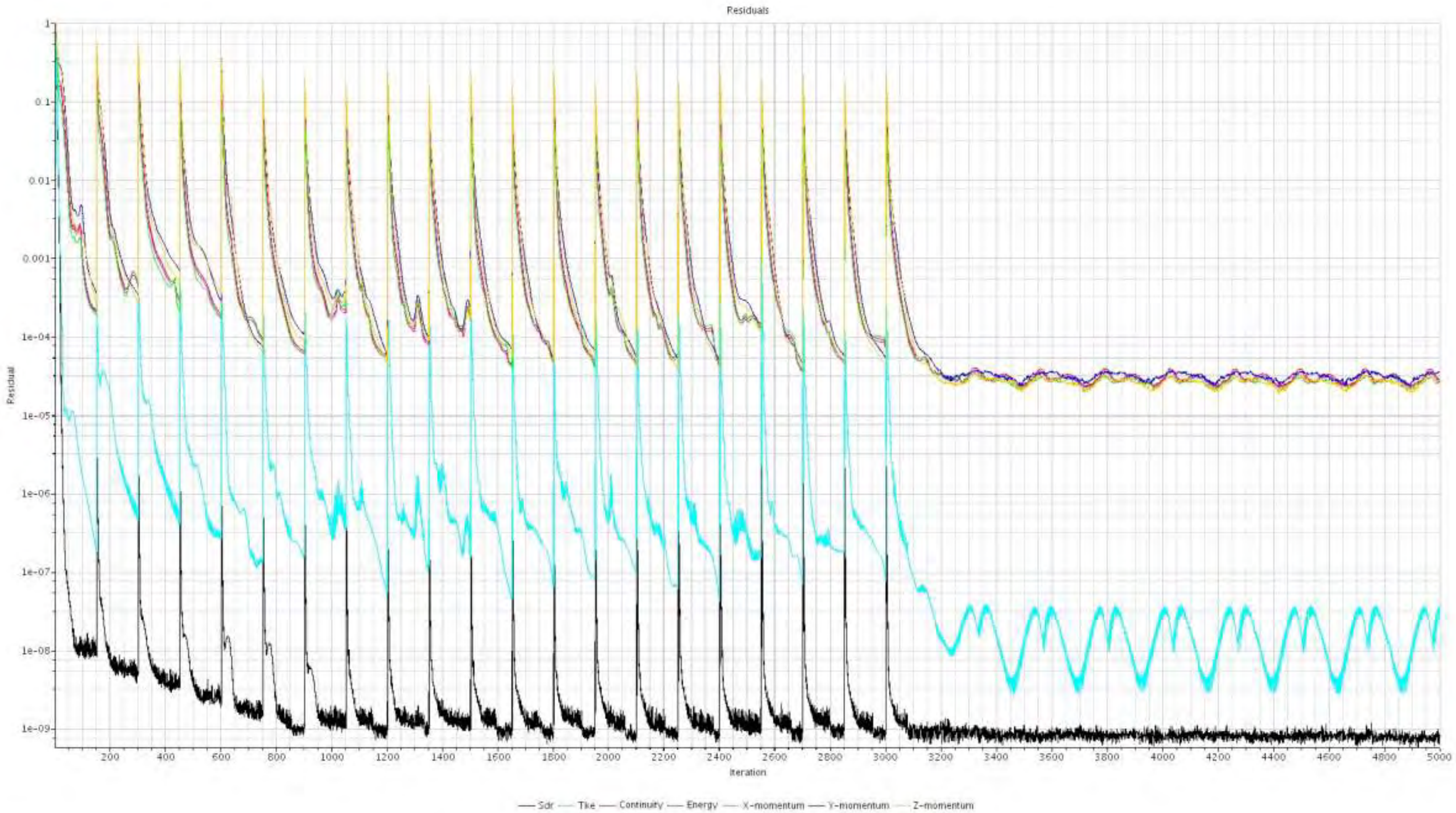
ATA ENGINEERING, INC.

Java based mesh refinement

Shock Tracking

```
//Loop number for number of refinement steps
    for (int i = 1; i < totalRefineSteps+1; i++) {
    //Run first run
    simrunner.run(iterations);
    //If first run, extract, table and set Trim mesh with field function refinement
      if (i==1){
        refineTable.extract();
        mesherTrim.setMeshSizeTable(refineTable);

        // else extract table
        } else {

        refineTable.extract();
    }
    //Re-mesh
    mesher.execute();
    }
    ///Run to stopping criteria
    simrunner.run();
```


Y
Z
X
Mach Number
0.00000
0.32000
0.64000
0.96000
1.2800
1.6000


Residuals
Residual
Iteration
1
0.1
0.01
0.001
1e-04
1e-05
1e-06
1e-07
1e-08
1e-09
200
400
600
800
1000
1200
1400
1600
1800
2000
2200
2400
2600
2800
3000
3200
3400
3600
3800
4000
4200
4400
4600
4800
5000
Sdr
Tke
Continuity
Energy
X-momentum
Y-momentum
Z-momentum

Agenda

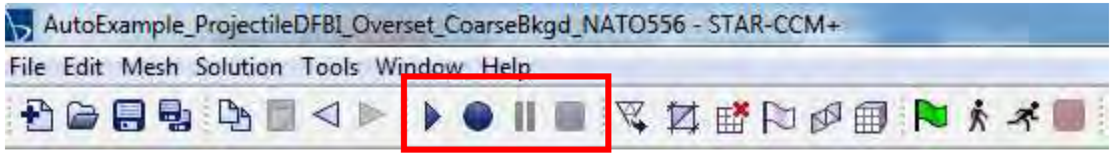
Automating STAR-CCM+

1. Automation with the GUI – Design Manager
2. Basic JavaScript
   - Using Java to use adaptive meshing to track a shock in a transonic simulation
3. Advanced automation with JavaScript:
   - Modeling 100s of geometric shapes using Java to assess boundary size and meshing, then run.

STAR-CCM+ includes a broad range of automation capability using Java macro scripting

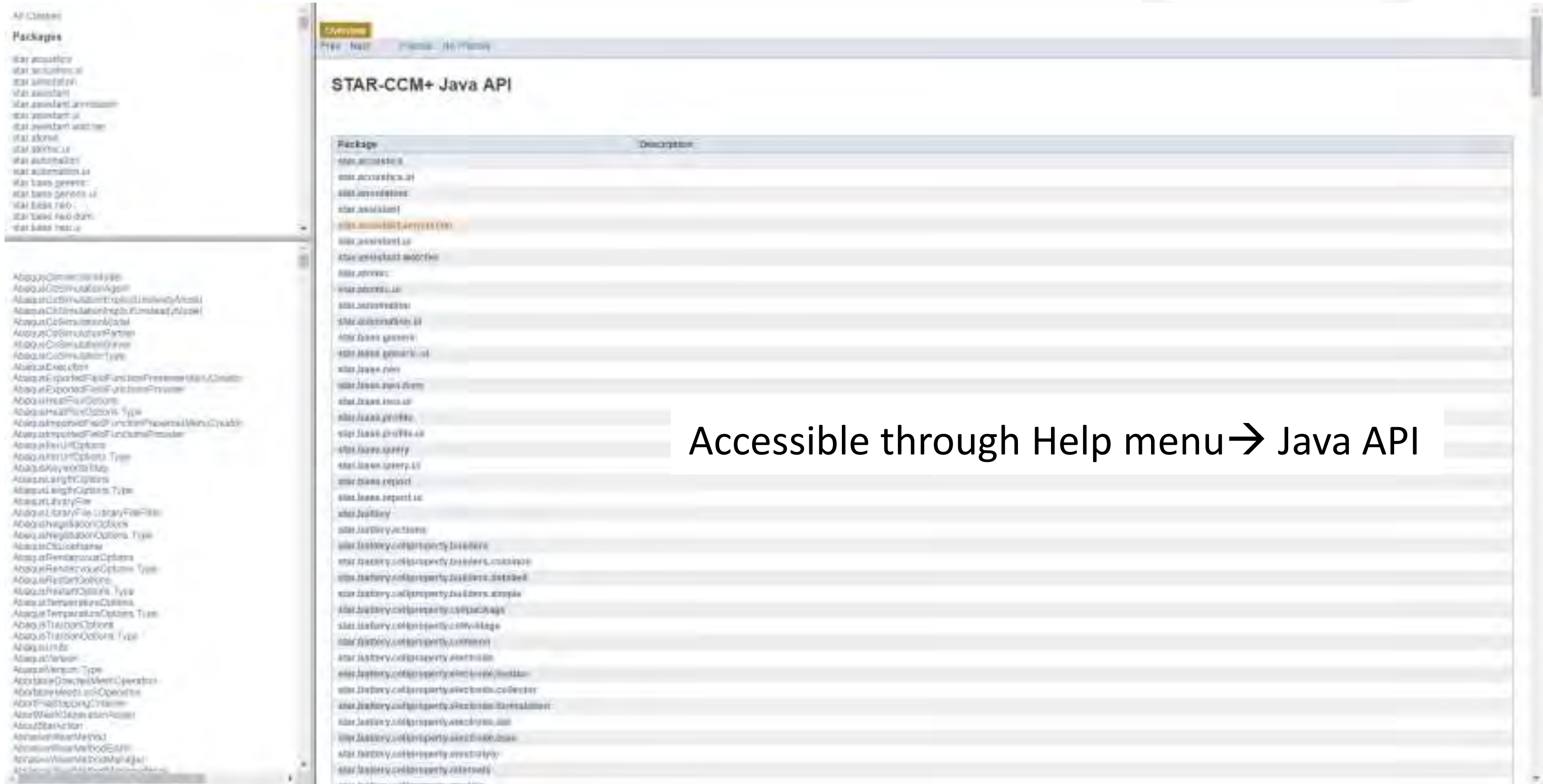
Java macros generated with built-in recording feature, manual programming

User actions in STAR GUI can be recorded as a Java macro using the highlighted controls in the upper toolbar (at right)



Alternatively, Java macros can be manually created by the user, using the STAR Java API library reference to identify the correct classes and methods.

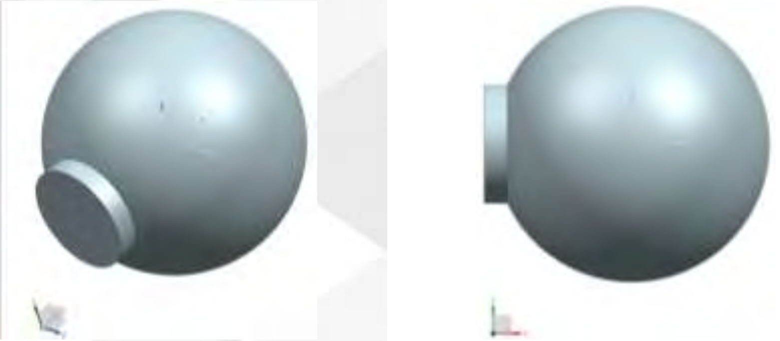
This is a particularly useful reference for classes and methods that are not captured by the macro recorder (e.g., methods that write to Output window).



Automation in STAR-CCM+ enables design space exploration of hundreds of designs

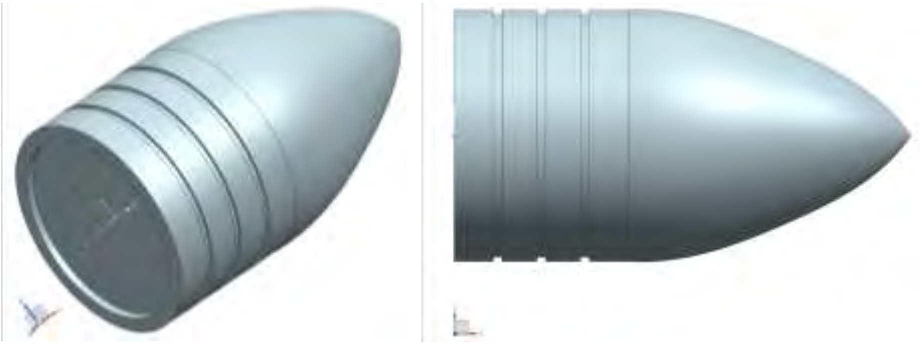
"Ballistics Through the Centuries"

18th century .58 caliber musket ball

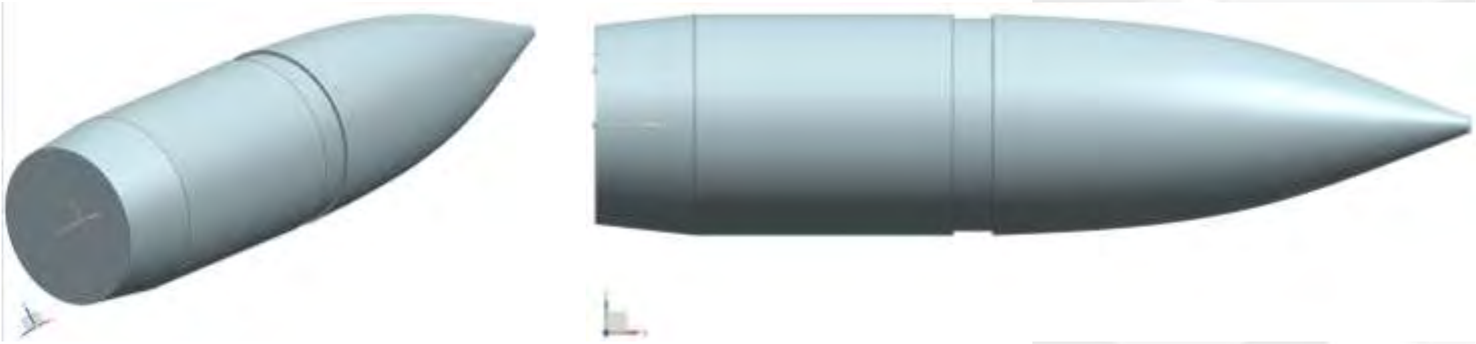


We will demonstrate the automated update and configuration of solver settings by look at different bullet designs from over the past few centuries.

19th century .58 caliber Minié ball



Modern NATO 5.56 mm (.224 caliber) round



Example simulation: initial flight and flow field of different bullets immediately after firing

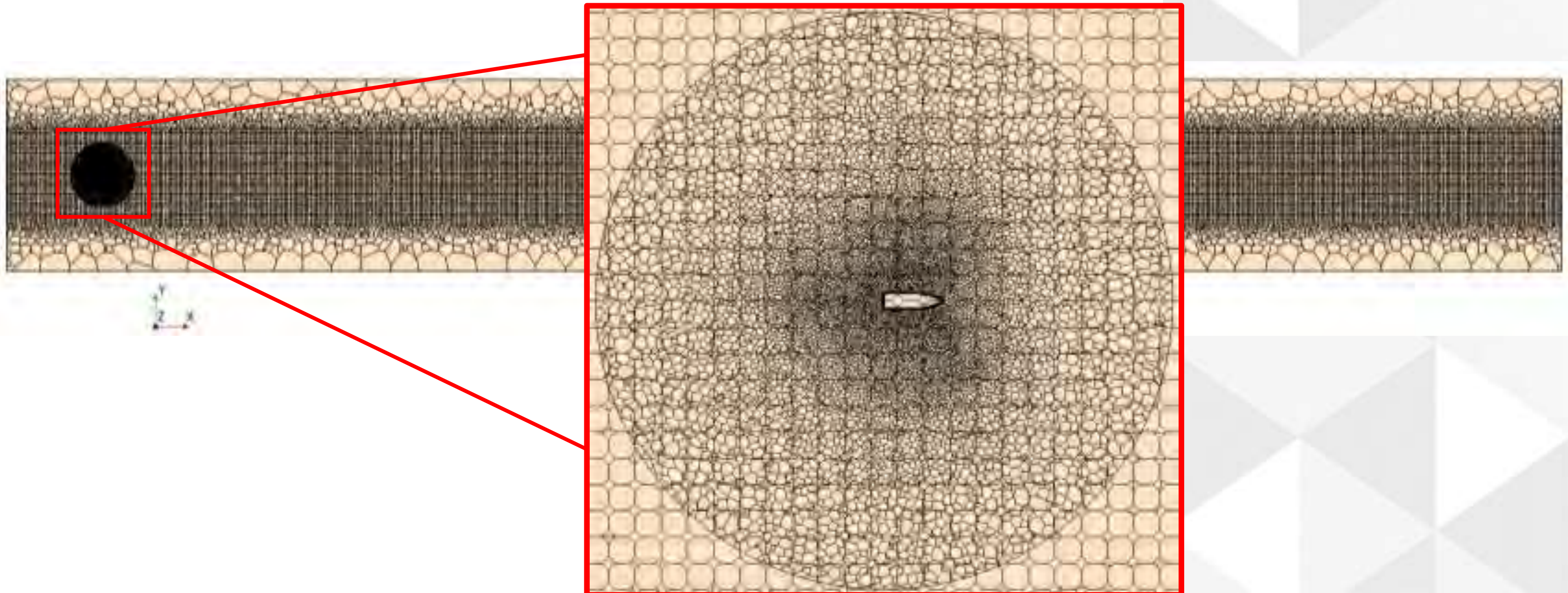
Time-accurate simulation with dynamic fluid-body interaction (DFBI)

Case #	Projectile	Size (caliber)	Mass (g)	Muzzle Velocity (m/s)	Angular Velocity (rps)
1	Musket ball	0.58	19.3	370	0
2	Minié ball	0.58	36.4	400	287†
3	NATO 5.56	0.224	4.1	850	4183‡

†Assumes 40” barrel with 1:72” twist rate
‡Assumes 24” barrel with 1:8” twist rate

Simulation uses time-accurate DFBI to compute the effect of drag and gravity on projectile velocity and trajectory.

Bullet motion captured by local, refined overset region around bullet, traveling through coarser background region.

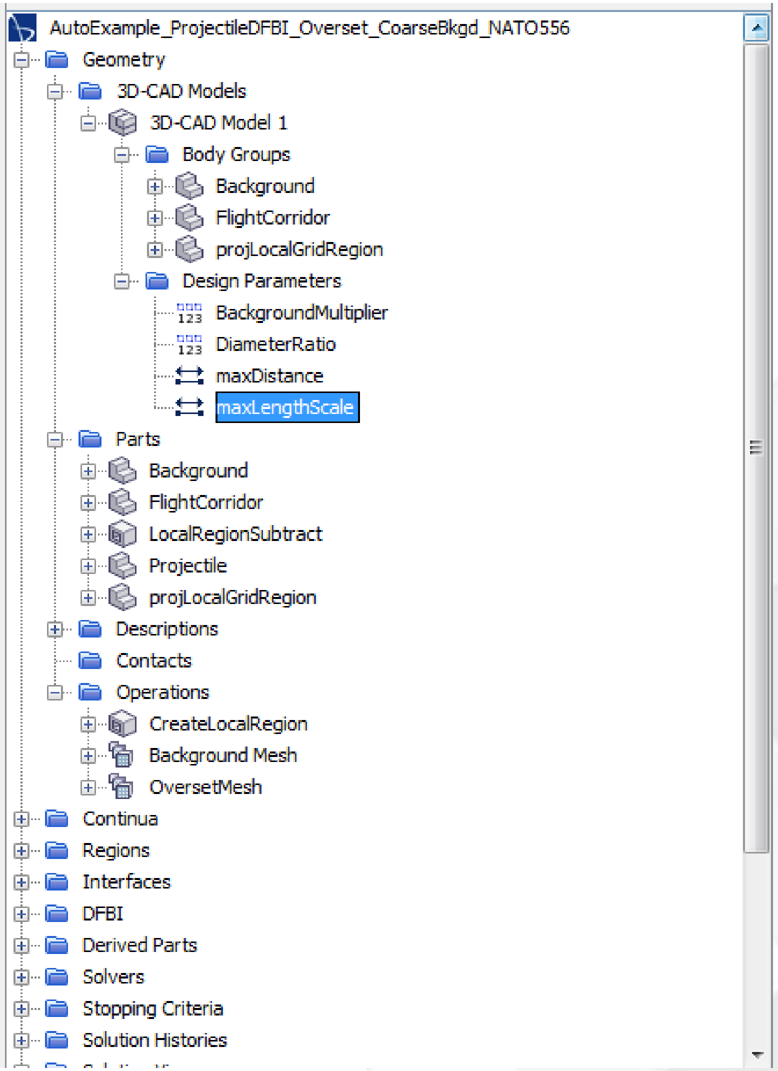


ATA ENGINEERING, INC.

Setting Up a Simulation for Automation

Automation takes advantage of STAR-CCM+'s pipeline-based organization

- First step is generally to set up one simulation case (including all mesh settings, boundary conditions, reports, monitors, plots, scenes, field functions, etc.)
  - These can be varied as needed for more complicated design/analysis campaigns, though best practice would be to limit any variations to pre-processing steps as much as possible and use Parameters for variation
- Geometry Operations manager contains many options for preparing surfaces, operating on parts, and generating meshes.
- Operations are executed in order, and can be executed with a single command.
- For the present study, automation script must accomplish several tasks before the Execute All command may be given (green are recorded, blue is manual):
  1. Load in new geometry part
  2. Import new projectile geometry into separate CAD model
  3. Extract solid body properties and store them for later use
  4. Update DFBI properties
  5. Re-center moving coordinate system on new projectile Center of Mass
  6. Combine projectile surfaces into single surface (optional)
  7. Update CAD parameters; Execute Update command for CAD model and associated parts
  8. Update projectile part in Subtract operation
  9. Delete old part and clear solution to reset overset region
  10. Execute all Operations (including mesh generation)



ATA ENGINEERING, INC.

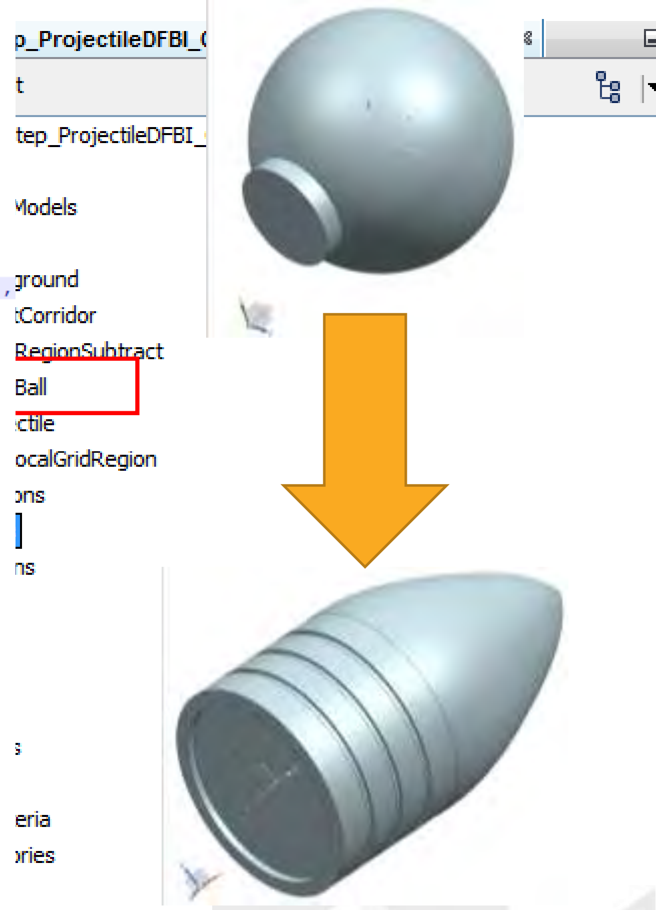
1. Load new geometry part

Loads replacement bullet alongside previous bullet geometry

Code below is generated by Import→Import Surface Mesh menu option when recording macro

```java
public class BulletUpdateMacro extends StarMacro {

// Main function
  public void execute() {
    cadUpdate();
  }

  private void cadUpdate() {

    // Set active simulation. Required as first line for every macro
    Simulation simulation_0 =
      getActiveSimulation();
    // Starts part importing procedure
    PartImportManager partImportManager_0 =
      simulation_0.get(PartImportManager.class);
    //The following line imports the replacement part. REPLACE resolvePath argument with appropriate file path
    partImportManager_0.importCadPart(resolvePath("H:\\6_BusinessDevelopment\\STAR_VAR\\AutomationWebinar\\Ballistics_thru_Years\\Projectiles\\Parasolids\\MinieBall.x_t"),
    // Create new scene to visualize new part
    simulation_0.getSceneManager().createGeometryScene("Geometry Scene", "Outline", "Geometry", 1);
    Scene scene_2 =
      simulation_0.getSceneManager().getScene("Geometry Scene 2");
    scene_2.initializeAndWait();
    PartDisplayer partDisplayer_2 =
      ((PartDisplayer) scene_2.getDisplayerManager().getDisplayer("Outline 1"));
    partDisplayer_2.initialize();
    PartDisplayer partDisplayer_3 =
      ((PartDisplayer) scene_2.getDisplayerManager().getDisplayer("Geometry 1"));
    partDisplayer_3.initialize();
    SceneUpdate sceneUpdate_1 =
      scene_2.getSceneUpdate();
    HardcopyProperties hardcopyProperties_1 =
      sceneUpdate_1.getHardcopyProperties();
    hardcopyProperties_1.setCurrentResolutionWidth(25);
    hardcopyProperties_1.setCurrentResolutionHeight(25);
    Scene scene_1 =
      simulation_0.getSceneManager().getScene("Lab-Frame Scene");
    SceneUpdate sceneUpdate_0 =
      scene_1.getSceneUpdate();
    HardcopyProperties hardcopyProperties_0 =
      sceneUpdate_0.getHardcopyProperties();
    hardcopyProperties_0.setCurrentResolutionWidth(1572);
    hardcopyProperties_0.setCurrentResolutionHeight(619);
    hardcopyProperties_1.setCurrentResolutionWidth(1570);
    hardcopyProperties_1.setCurrentResolutionHeight(618);
    scene_2.resetCamera();
```

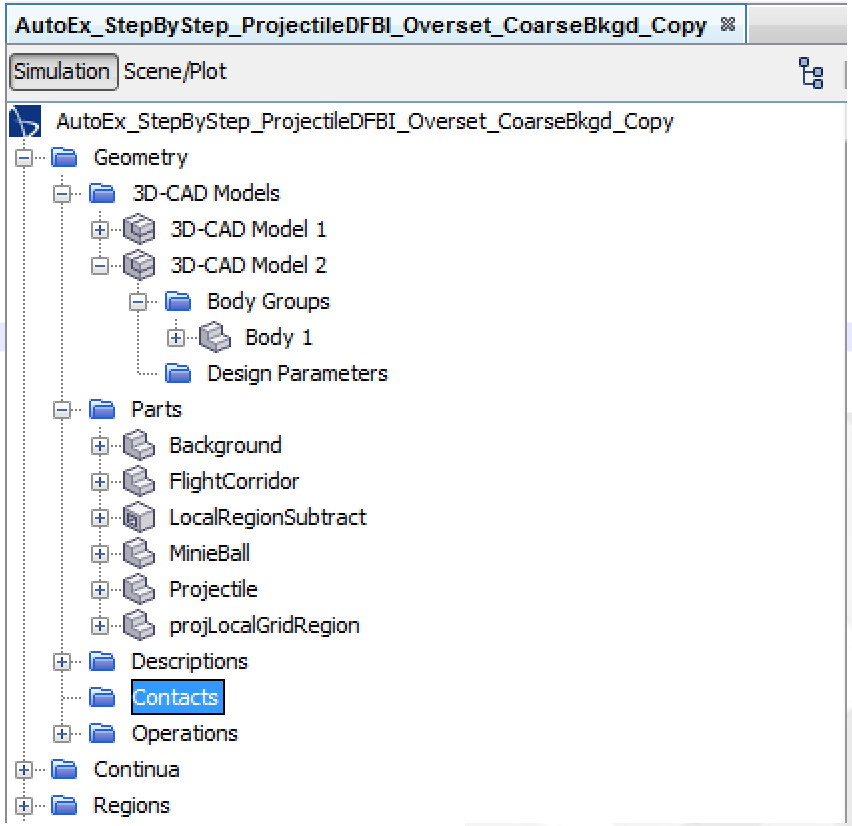
Optional (generated by way macro was recorded)



2. Import new projectile into separate CAD geometry

When executing macro, happens in background (user does not see CAD modeler open in scene)

```
// Import new part into CAD for evaluation
Scene scene_3 =
  simulation_0.getSceneManager().createScene("3D-CAD View");
scene_3.initializeAndWait();
// Choose part for import to CAD. REPLACE getPart argument with name of <part>.x_t file imported above (plus integer if same name as previously existing part. Keep quotes).
CadPart cadPart_0 =
  ((CadPart) simulation_0.get(SimulationPartManager.class).getPart("MinieBall"));
simulation_0.get(SolidModelManager.class).createSolidModelForCadParts(scene_3, new NeoObjectVector(new Object[] {cadPart_0}));
scene_3.open();
scene_3.setAdvancedRenderingEnabled(false);
SceneUpdate sceneUpdate_2 =
  scene_3.getSceneUpdate();
HardcopyProperties hardcopyProperties_2 =
  sceneUpdate_2.getHardcopyProperties();
hardcopyProperties_2.setCurrentResolutionWidth(25);
hardcopyProperties_2.setCurrentResolutionHeight(25);
hardcopyProperties_1.setCurrentResolutionWidth(1572);
hardcopyProperties_1.setCurrentResolutionHeight(619);
hardcopyProperties_2.setCurrentResolutionWidth(1570);
hardcopyProperties_2.setCurrentResolutionHeight(618);
scene_3.resetCamera();
CadModel cadModel_0 =
  ((CadModel) simulation_0.get(SolidModelManager.class).getObject("3D-CAD Model 2"));
```



Code above is equivalent to right-click→Edit in 3D-CAD on a given Part

3. Export solid body properties and store for use

First manual step as macro recorder does not record where solid body properties are stored, only the CAD modeler command to get them

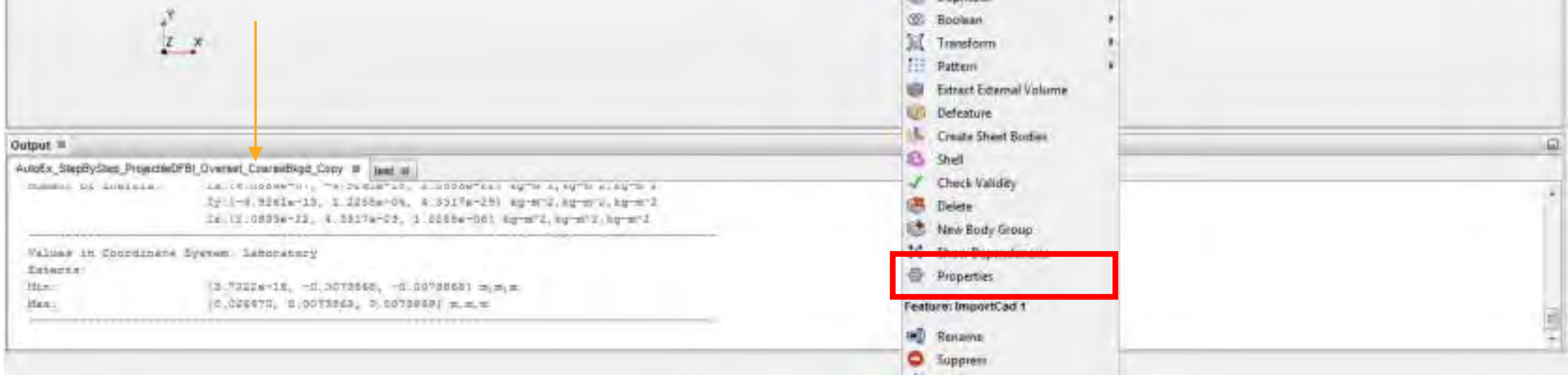
```
// Select CAD body to measure properties
star.cadmodeler.Body cadmodelerBody_0 =
  ((star.cadmodeler.Body) cadModel_0.getBody("Body 1"));
// Get CAD body properties and save for later referencing
NeoProperty bodyProp_0 =
  cadModel_0.getBodyManager().getBodyProperties(new NeoObjectVector(new Object[] {cadmodelerBody_0}));
Units units_0 =
  simulation_0.getUnitsManager().getPreferredUnits(new IntVector(new int[] {0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}));
Units units_1 =
  simulation_0.getUnitsManager().getPreferredUnits(new IntVector(new int[] {0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}));
Units units_2 =
  simulation_0.getUnitsManager().getPreferredUnits(new IntVector(new int[] {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0}));
Units units_3 =
  simulation_0.getUnitsManager().getPreferredUnits(new IntVector(new int[] {1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}));
// Extract body properties table
NeoProperty bodyPropTable = bodyProp_0.getNeoProperty("1");
simulation_0.get(SolidModelManager.class).endEditCadModel(cadModel_0);
simulation_0.getSceneManager().deleteScenes(new NeoObjectVector(new Object[] {scene_3}));
hardcopyProperties_1.setCurrentResolutionWidth(1570);
hardcopyProperties_1.setCurrentResolutionHeight(618);
SolidModelPart solidModelPart_0 =
  ((SolidModelPart) simulation_0.get(SimulationPartManager.class).getPart("MinieBall"));
solidModelPart_0.setPresentationName("NewProjectile");
```

Recorded

Manual storage of properties hastable

Exit CAD modeler and rename new part

window, not stored.

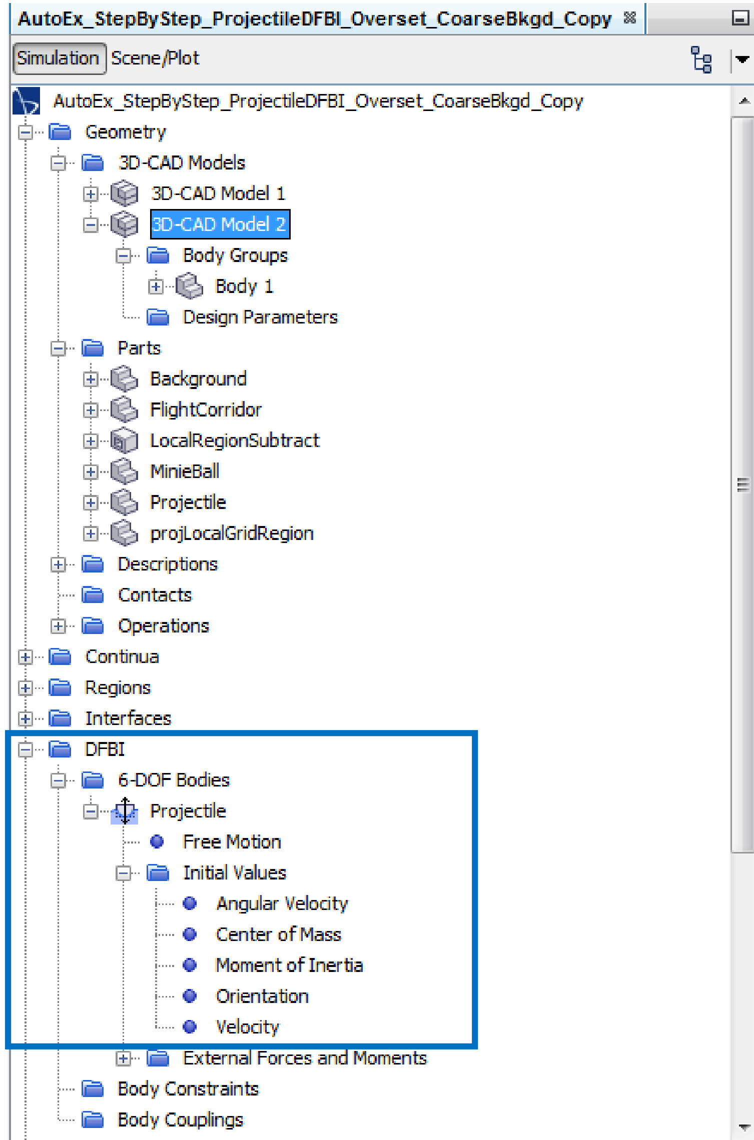


4. Update DFBI Properties

DFBI Initial Values updated based on stored body properties, case initial conditions

```java
// Select DFBI body
ContinuumBody continuumBody_0 =
  ((ContinuumBody) simulation_0.get(star.sixdof.BodyManager.class).getObject("Projectile"));
// Update DFBI Center of Mass
CenterOfMass centerOfMass_0 =
  ((CenterOfMass) continuumBody_0.getInitialValueManager().getObject("Center of Mass"));
Units units_4 =
  ((Units) simulation_0.getUnitsManager().getObject("m"));
(centerOfMass_0.getPosition()).setCoordinate(units_4,units_4,units_4,bodyPropTable.getDoubleVector("CenterOfGravity"));
// Update DFBI Moment of Inertia
MomentOfInertia momentOfInertia_0 =
  ((MomentOfInertia) continuumBody_0.getInitialValueManager().getObject("Moment of Inertia"));
DoubleVector newMoI = bodyPropTable.getDoubleVector("MomentOfInertia");
Double Mxx = newMoI.getComponent(0);
Double Myy = newMoI.getComponent(4);
Double Mzz = newMoI.getComponent(8);
Double Mxy = newMoI.getComponent(1);
Double Mxz = newMoI.getComponent(2);
Double Myz = newMoI.getComponent(5);
momentOfInertia_0.getDiagonalComponents().setComponents(Mxx, Myy, Mzz);
momentOfInertia_0.getOffDiagonalComponents().setComponents(Mxy, Mxz, Myz);
// Update DFBI Velocity
Velocity velocity_0 =
  ((Velocity) continuumBody_0.getInitialValueManager().getObject("Velocity"));
// Replace velocity components as necessary for correct initial velocity vector
velocity_0.getValue().setComponents(400, 0.0, 0.0);
//Update DFBI Mass and Angular Velocity
continuumBody_0.getBodyMass().setValue(0.0364);
AngularVelocity angularVelocity_0 =
  ((AngularVelocity) continuumBody_0.getInitialValueManager().getObject("Angular Velocity"));
Units units_5 =
  ((Units) simulation_0.getUnitsManager().getObject("rps"));
angularVelocity_0.getValue().setUnits(units_0);
angularVelocity_0.getValue().setComponents(1969, 0.0, 0.0);
```

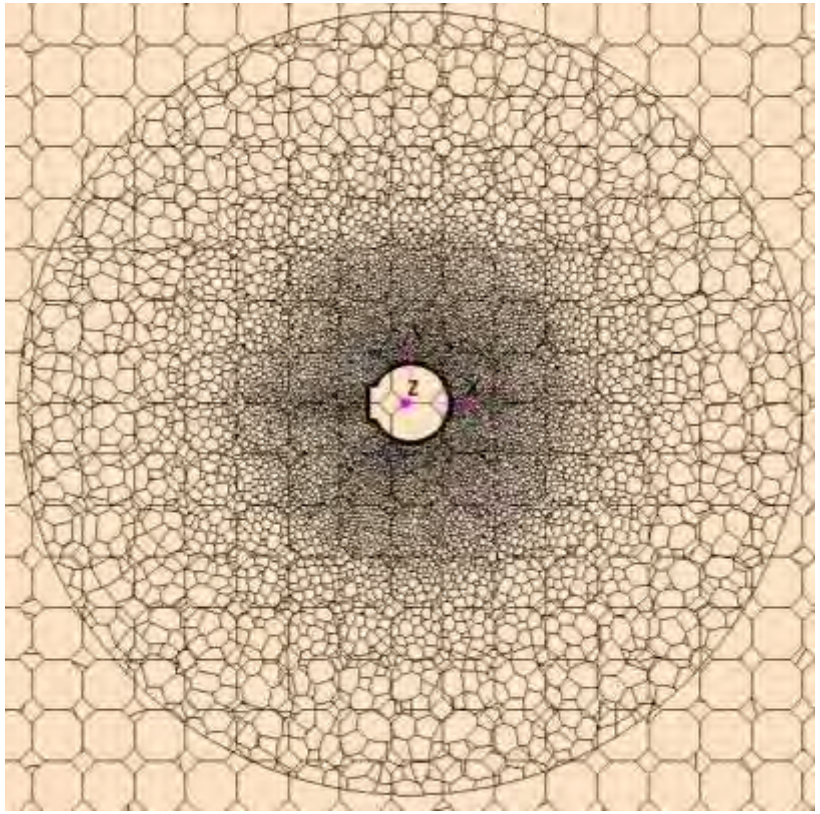
Case initial conditions

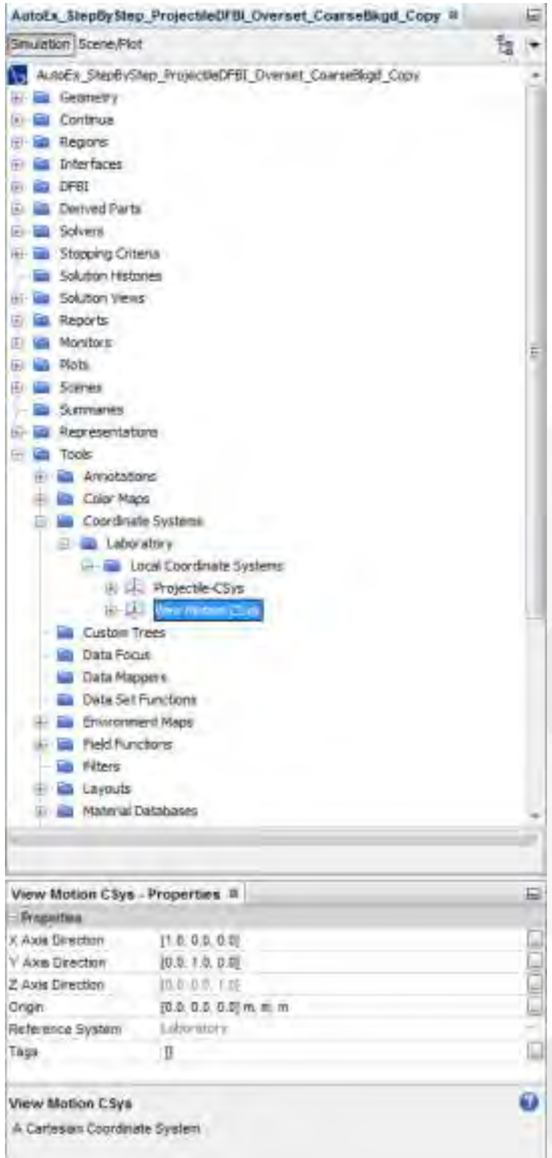


5. Re-center moving coordinate system

Moving coordinate system used to keep scenes centered on bullet CG

```
// Re-center moving reference frame on new projectile Center of Mass
LabCoordinateSystem labCoordinateSystem_0 =
  simulation_0.getCoordinateSystemManager().getLabCoordinateSystem();
CartesianCoordinateSystem cartesianCoordinateSystem_0 =
  ((CartesianCoordinateSystem) labCoordinateSystem_0.getLocalCoordinateSystemManager().getObject("View Motion CSys"));
Coordinate coordinate_0 =
  cartesianCoordinateSystem_0.getOrigin();
coordinate_0.setCoordinate(units_4, units_4, units_4, bodyPropTable.getDoubleVector("CenterOfGravity"));
```

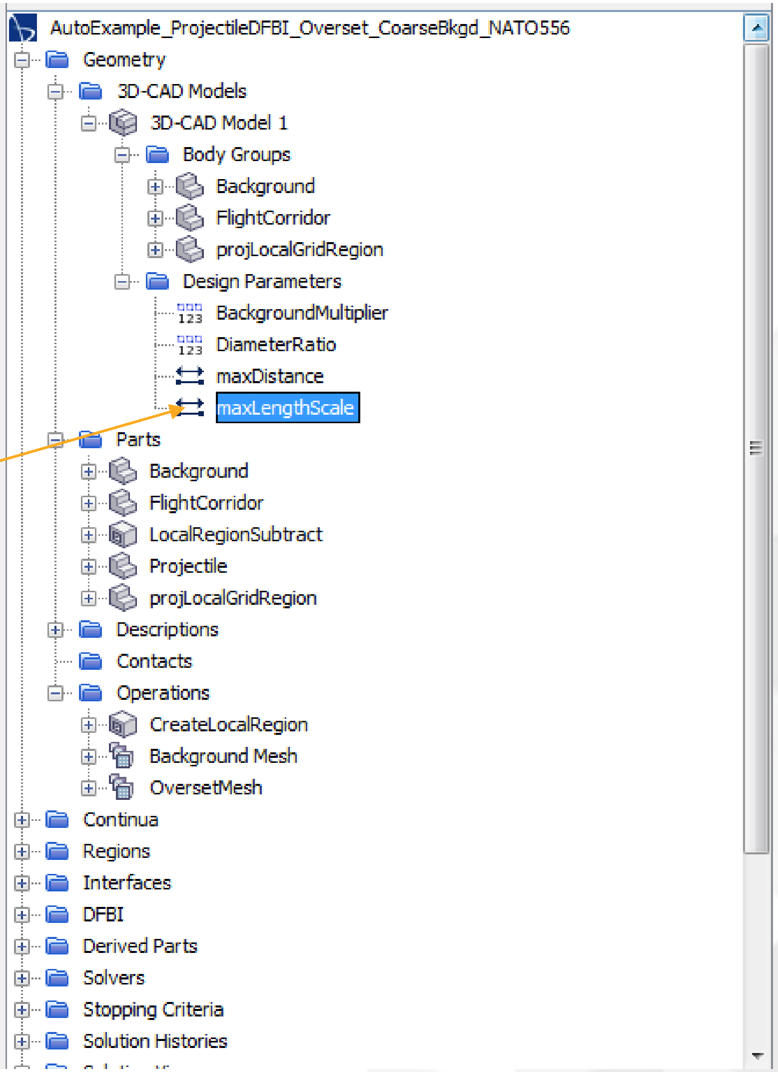




7&8. Update CAD and Subtract Operation parameters

Includes parameter values, arguments, and executing update to reflect CAD changes. Subtract Operation executed under Execute All command

```java
// Replace projectile in Subtract operation with new part
SubtractPartsOperation subtractPartsOperation_0 =
  ((SubtractPartsOperation) simulation_0.get(MeshOperationManager.class).getObject("CreateLocalRegion"));
subtractPartsOperation_0.getInputGeometryObjects().setQuery(null);
SolidModelPart solidModelPart_1 =
  ((SolidModelPart) simulation_0.get(SimulationPartManager.class).getPart("projLocalGridRegion"));
subtractPartsOperation_0.getInputGeometryObjects().setObjects(solidModelPart_1, solidModelPart_0);

// Get length properties to resize surrounding regions to fit new projectile
DoubleVector maxPos =
  bodyPropTable.getDoubleVector("UpperPos");
DoubleVector minPos =
  bodyPropTable.getDoubleVector("LowerPos");
Double maxLengthX = maxPos.getComponent(0)-minPos.getComponent(0);
Double maxLengthY = maxPos.getComponent(1)-minPos.getComponent(1);
Double maxLengthZ = maxPos.getComponent(2)-minPos.getComponent(2);
Double maxLength = Math.max(Math.max(maxLengthX,maxLengthY),maxLengthZ);
CadModel cadModel_1 =
  ((CadModel) simulation_0.get(SolidModelManager.class).getObject("3D-CAD Model 1"));
UserDesignParameter userDesignParameter_10 =
  ((UserDesignParameter) cadModel_1.getDesignParameterManager().getObject("maxLengthScale"));
userDesignParameter_10.getQuantity().setValue(maxLength);
cadModel_1.update();
SolidModelPart solidModelPart_2 =
  ((SolidModelPart) simulation_0.get(SimulationPartManager.class).getPart("Background"));
simulation_0.get(SimulationPartManager.class).updateParts(new NeoObjectVector(new Object[] {solidModelPart_2}));
```

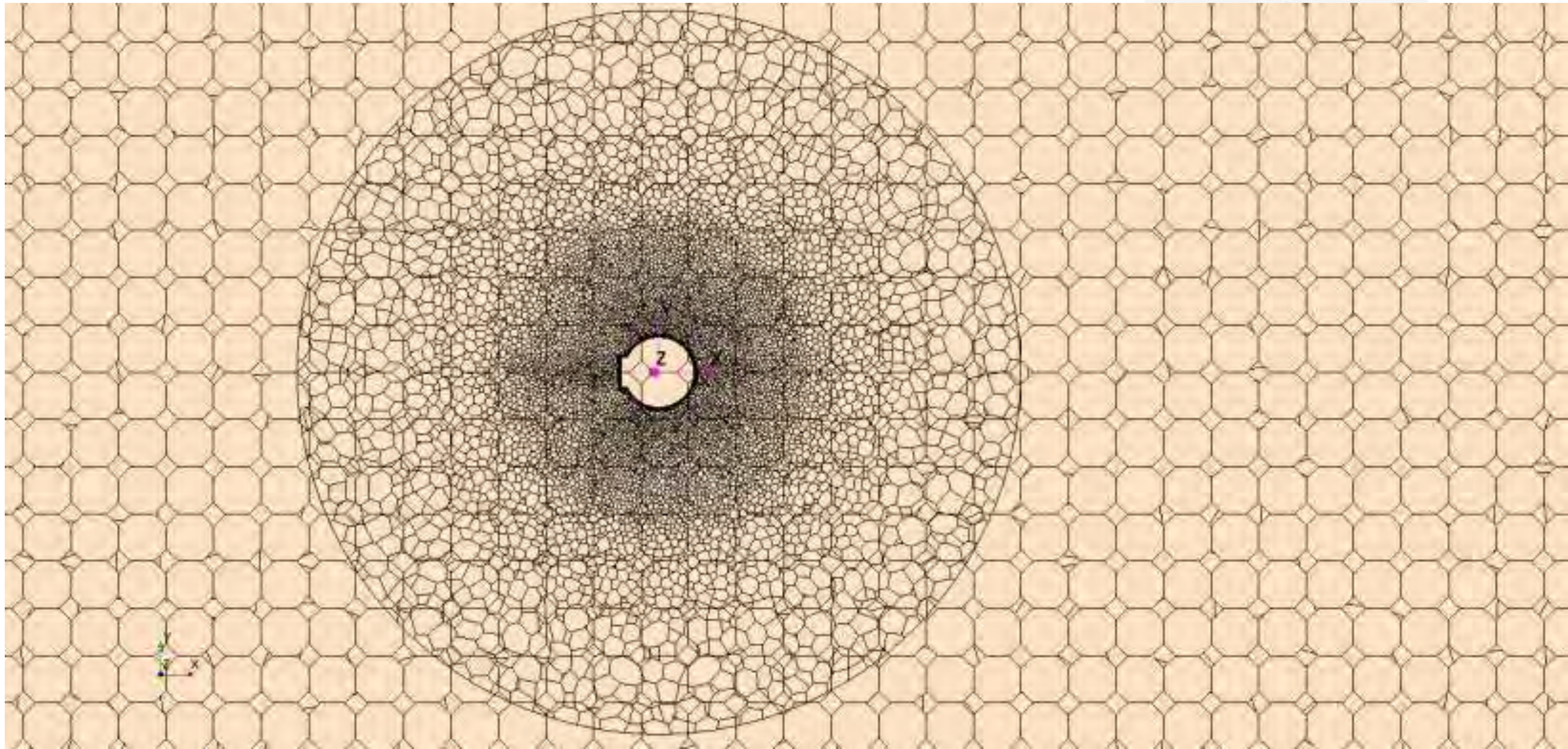


ATA ENGINEERING, INC.

9&10. Delete old part, reset solution, Execute All

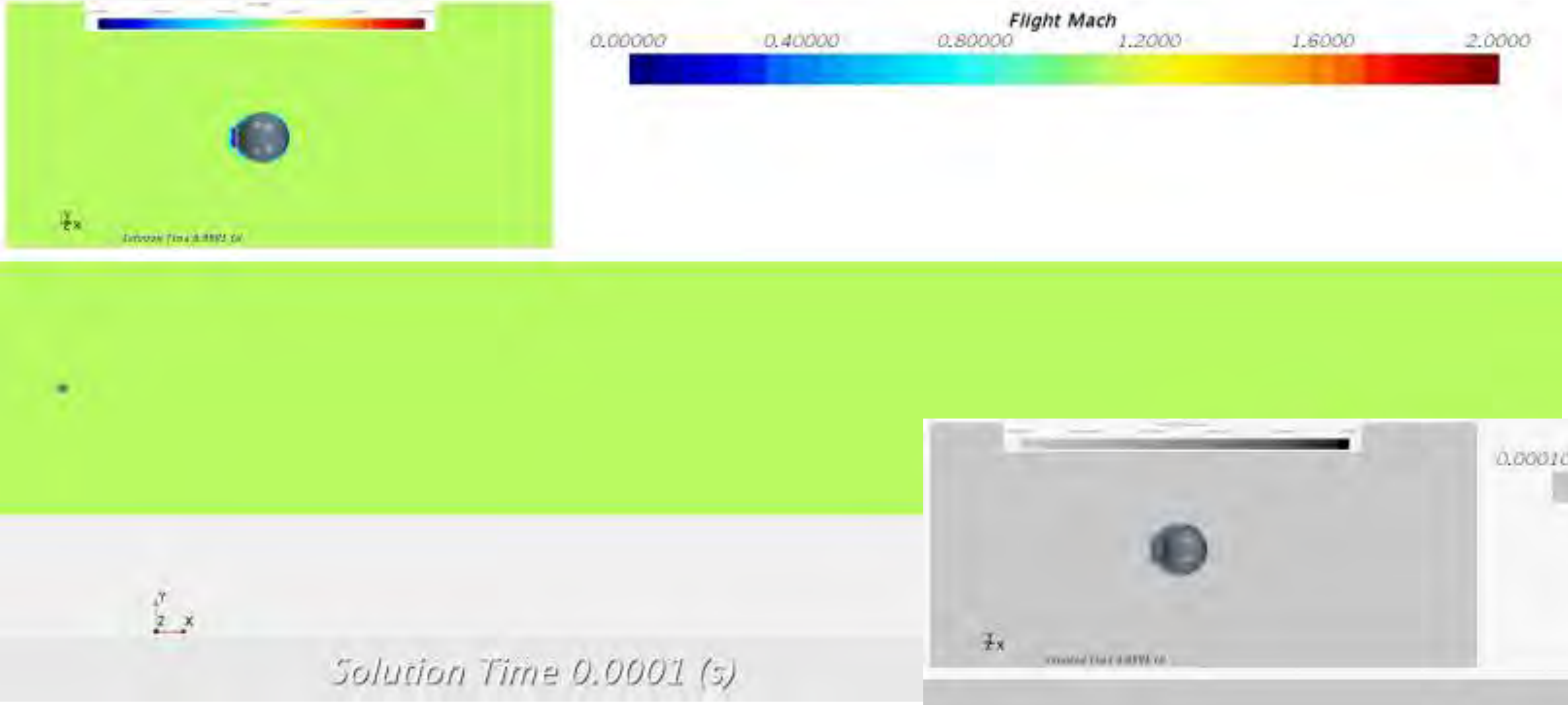
Reset solution is only required in this case because of time-accurate DFBI with motion; otherwise, clearing solution is optional and problem-dependent

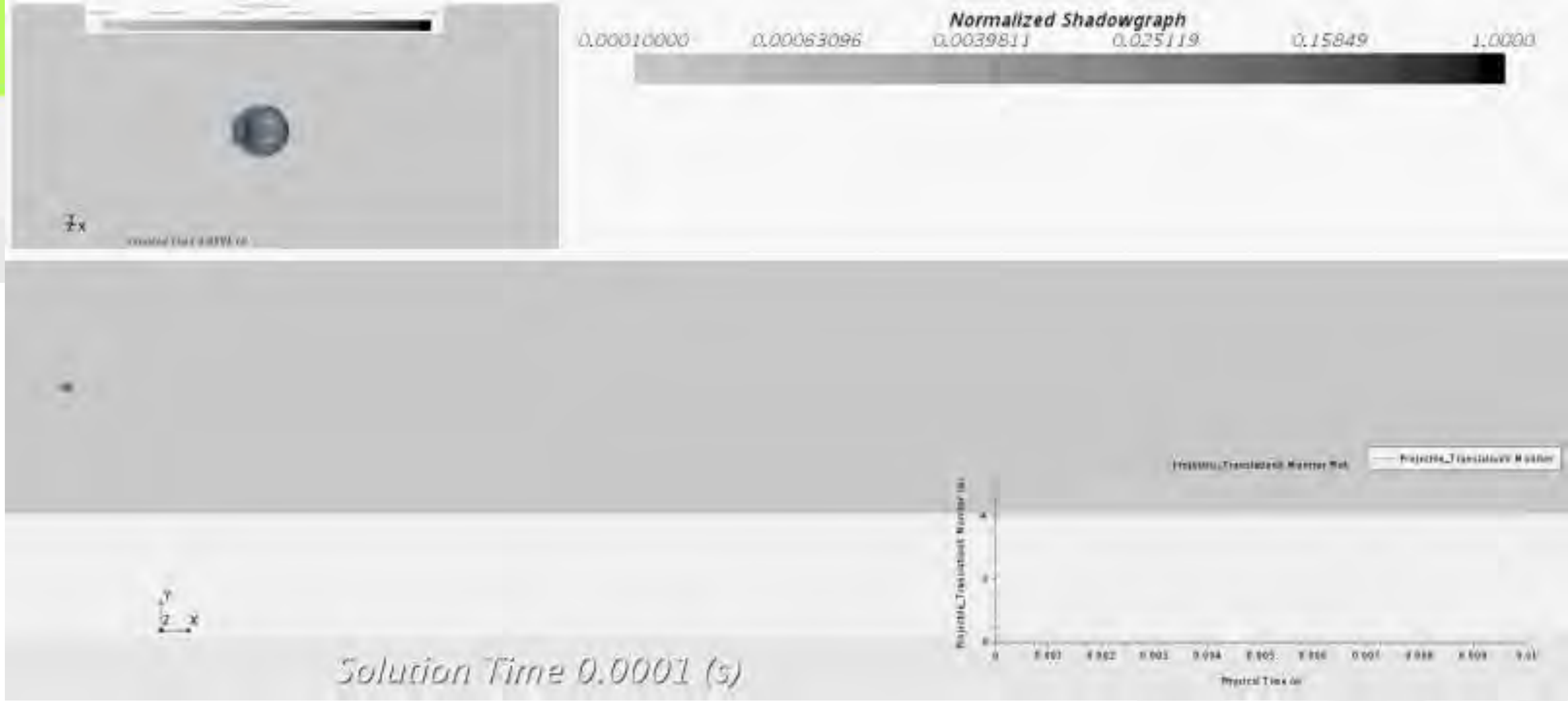
```
// Select old projectile part and remove it from the simulation (replace getPart argument with previous Part name, keeping quotes).
SolidModelPart solidModelPart_3 =
  ((SolidModelPart) simulation_0.get(SimulationPartManager.class).getPart("Projectile"));
simulation_0.get(SimulationPartManager.class).removeParts(new NeoObjectVector(new Object[] {solidModelPart_3}));
solidModelPart_0.setPresentationName("Projectile");

// Clear solution to reset coordinate systems for new parts
Solution solution_0 =
  simulation_0.getSolution();
solution_0.clearSolution();
// Execute Operations pipeline (Subtract projectile from Overset, and mesh Overset and Background)
simulation_0.get(MeshOperationManager.class).executeAll();
```



Simulations demonstrate advancement of bullet technology over the centuries

Musket ball with detached bow shock

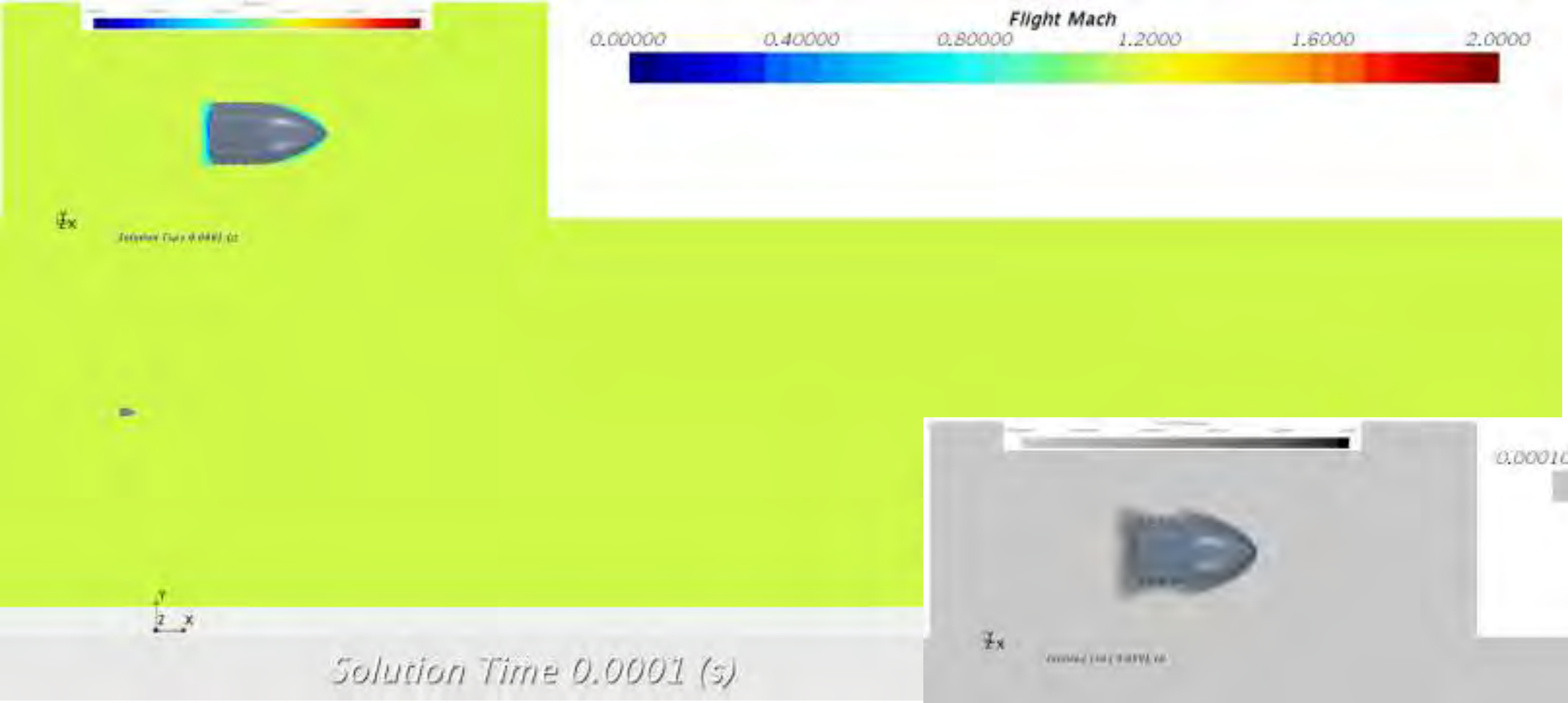


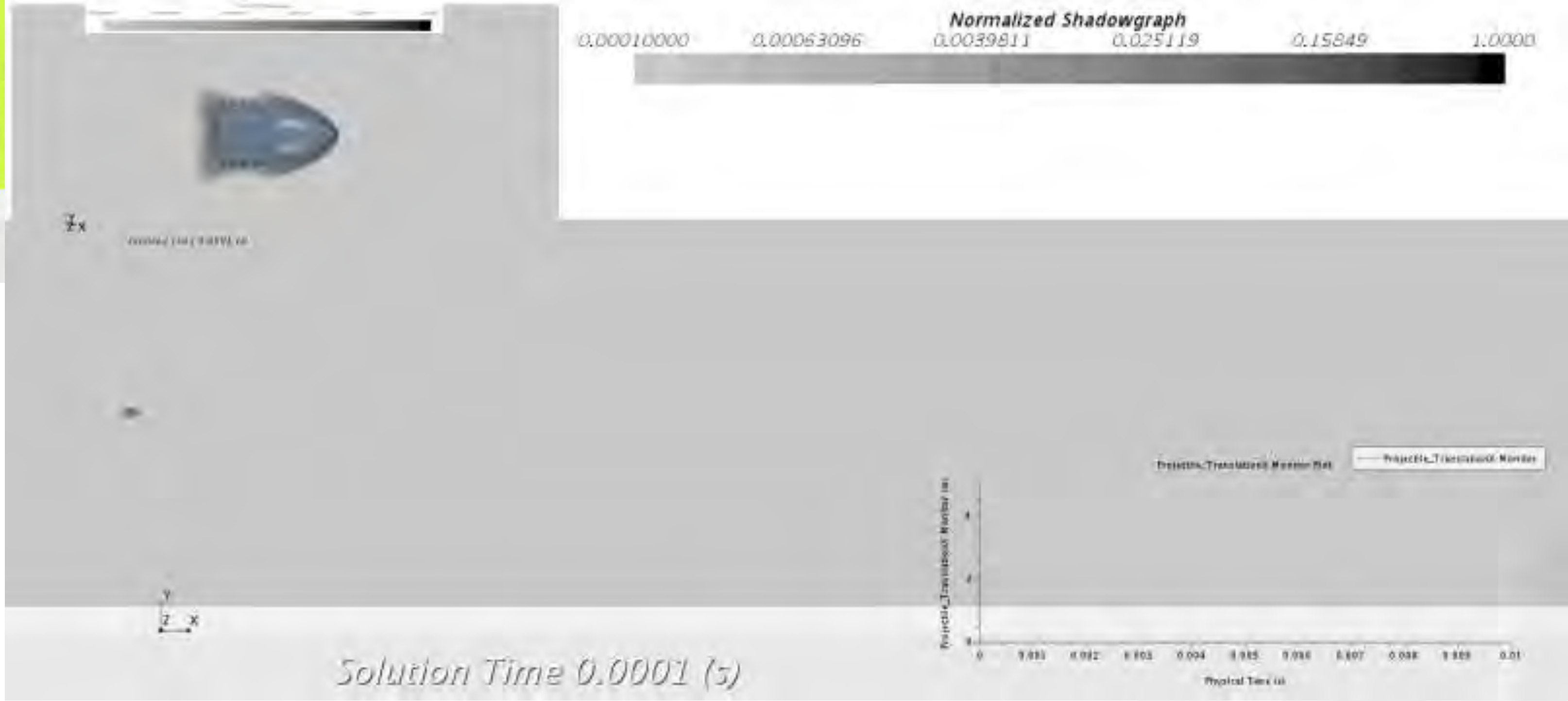


ATA ENGINEERING, INC.

Simulations demonstrate advancement of bullet technology over the centuries

Minié ball with high-drag detached bow shock

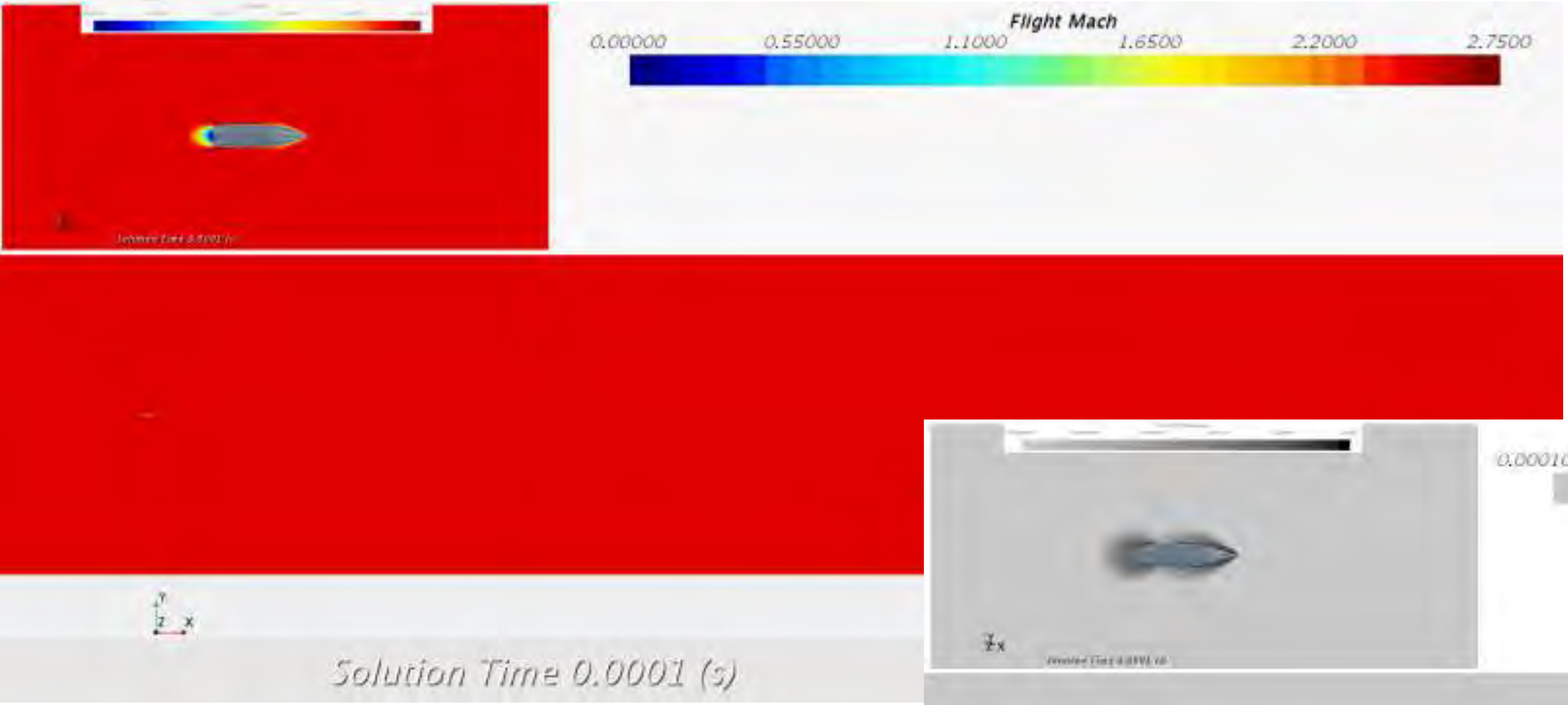


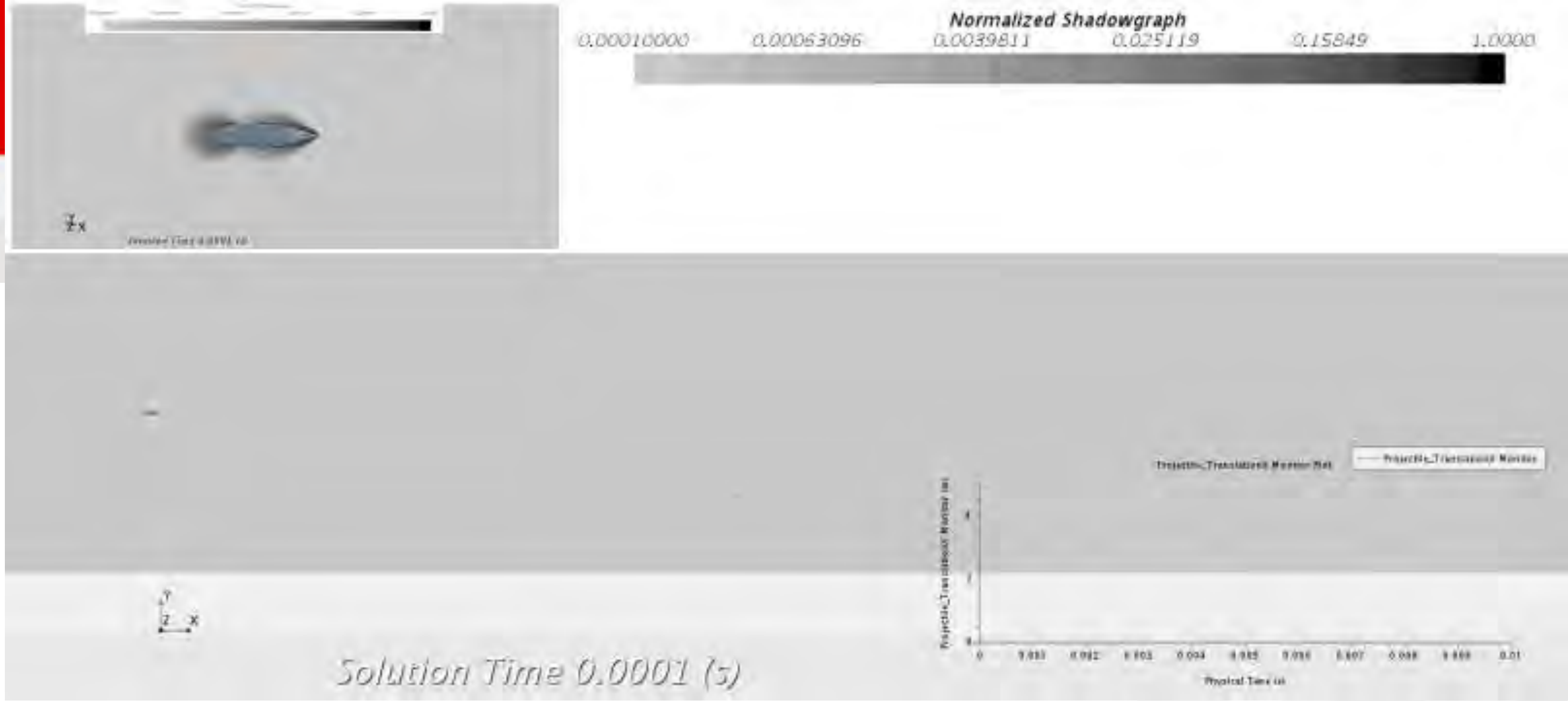


ATA ENGINEERING, INC.

Simulations demonstrate advancement of bullet technology over the centuries

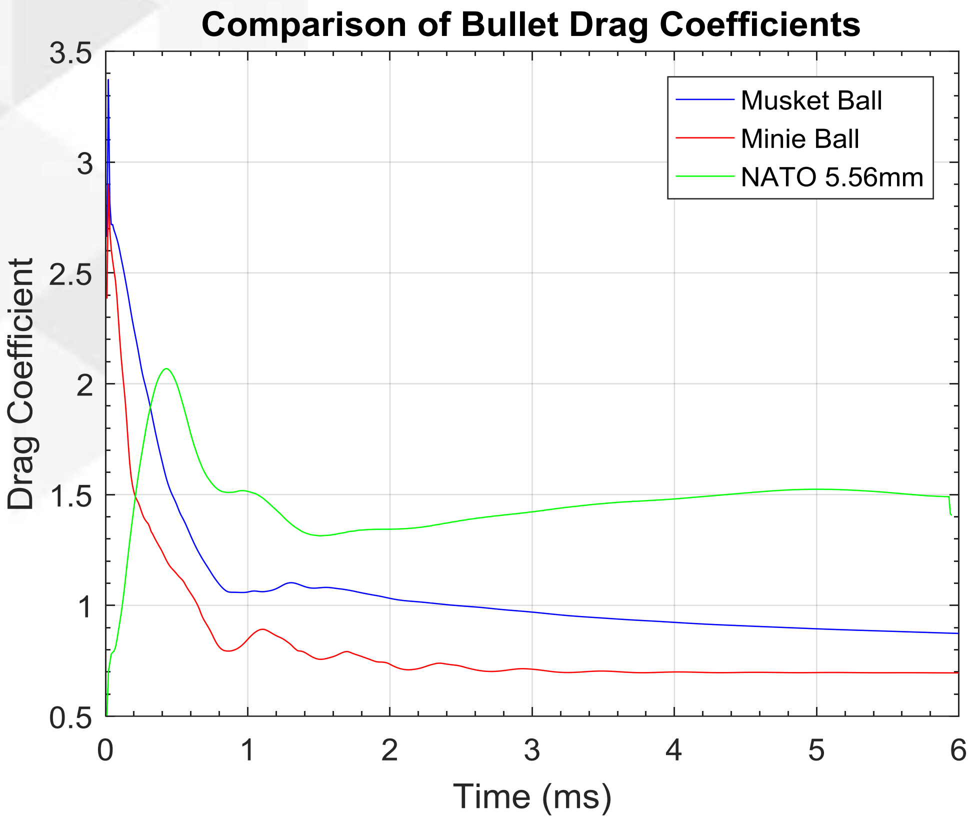
Supersonic NATO 5.56 round with oblique shock pattern

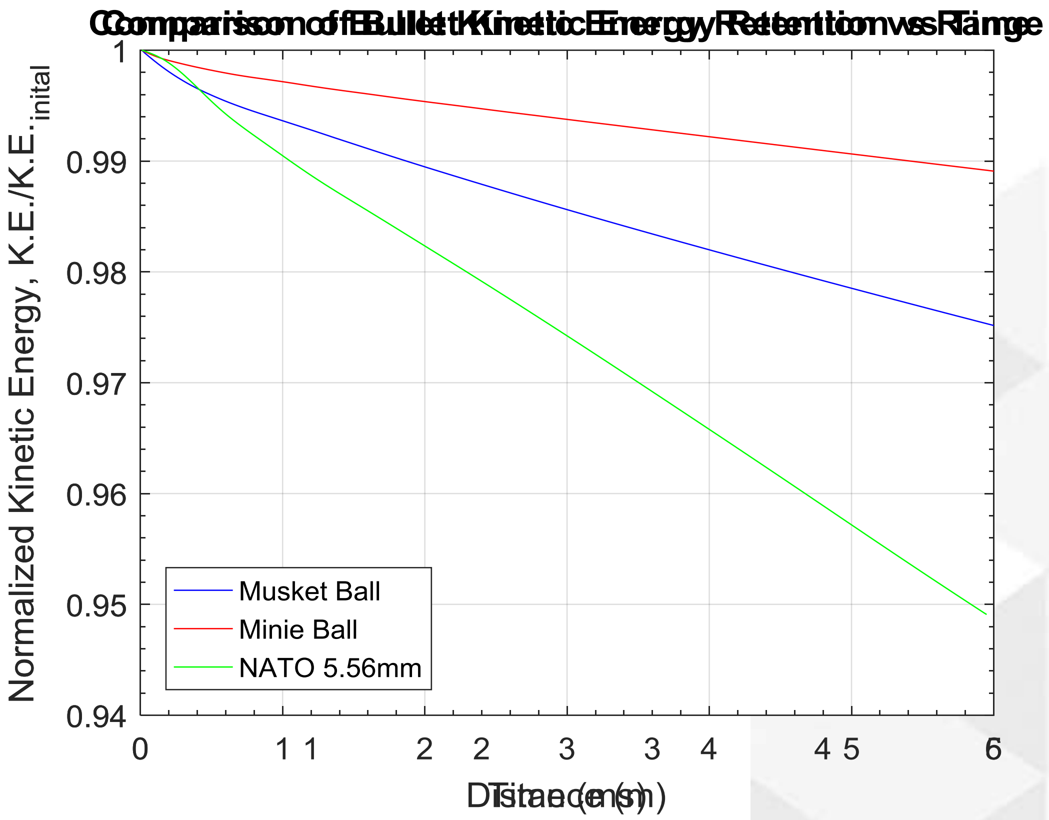




Coarse simulations show advantages of improved bullet shaping

NATO results inadvertently demonstrate importance of proper spin stabilization for maximum bullet performance

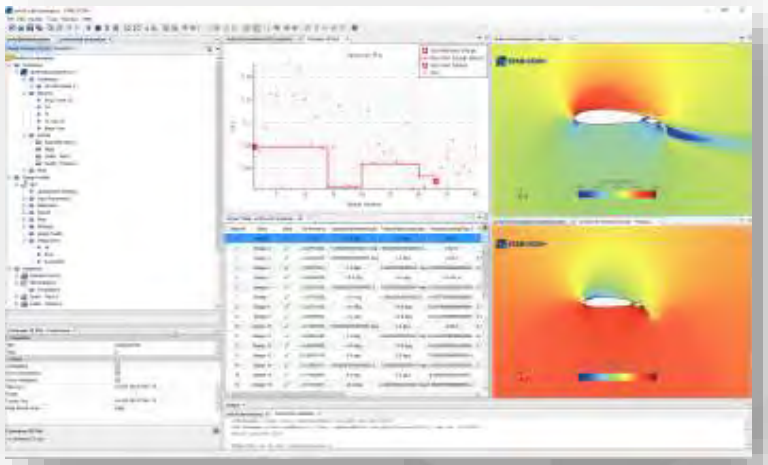




ATA ENGINEERING, INC.

Final Summary

- STAR-CCM+ offers several avenues to automate design and analysis of a myriad of products
- Design Manager provides built-in, efficient exploration of a parameterized design space with some optimization capability, scalable to advanced optimization methods using optional STAR-Innovate or HEEDS package
- Java macro functionality may also be used for highly-customizable, automated control over simulation settings to:
    - Automate analysis process for large set of designs
    - Dynamically update mesh or simulation parameters to improve accuracy/efficiency



ENGINEERING, INC.

Questions?