What's Ahead in OpenTD: An API for Thermal Desktop

Version 6.2

Presented at TFAWS 2020

by Matt Garrett/CRTech

www.crtech.com

C&R TECHNOLOGIES, INC.
Phone 303.971.0292
Fax 303.971.0035

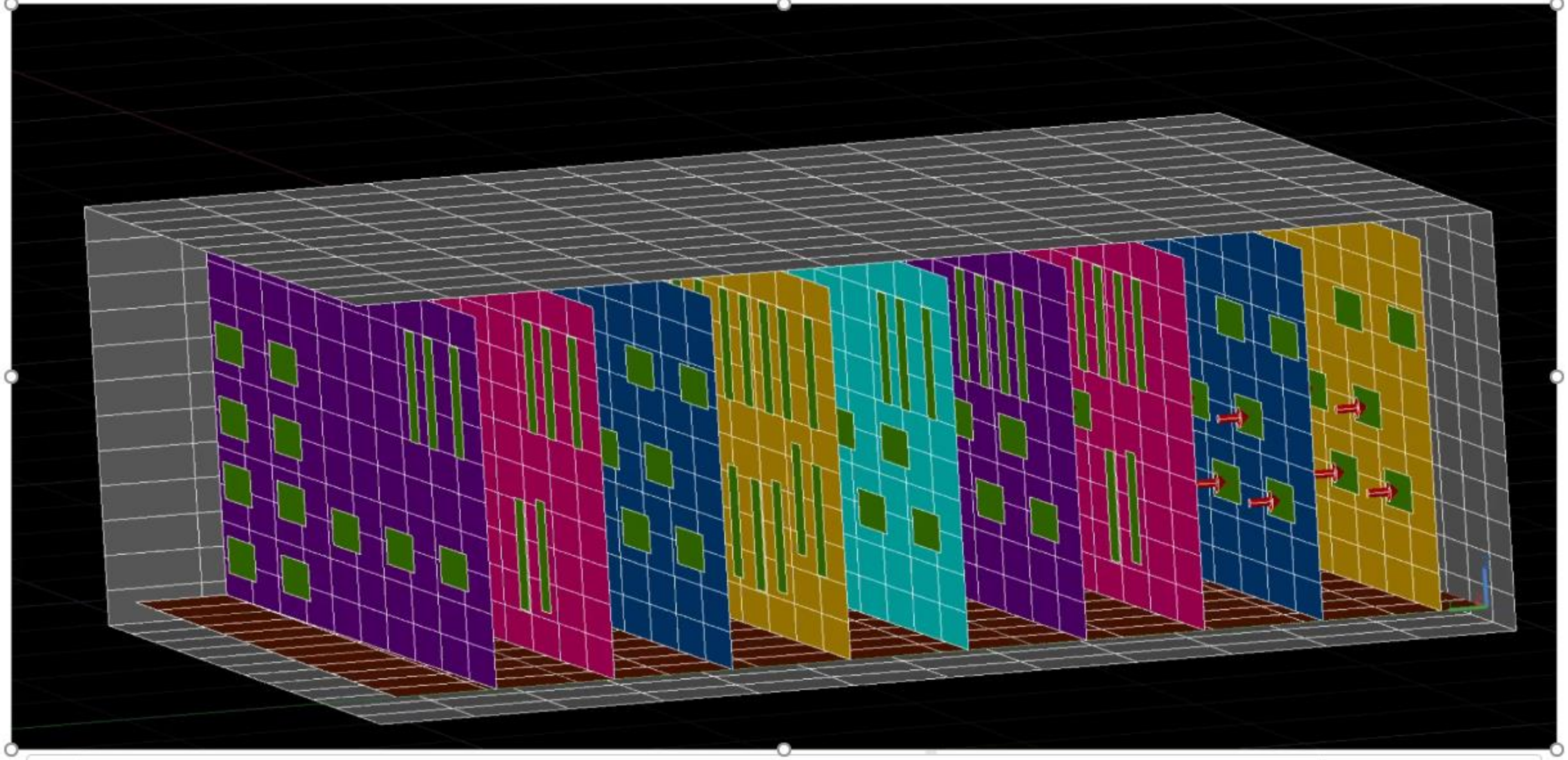
What is OpenTD?

- Application Programming Interface (API)
  - ✓ Allows you to write programs to control TD without human interaction.
- Installed with TD 6.0 and above
- Accessible with any language that can talk to .NET
  - ✓ C#, MATLAB, Python, Powershell, VB.NET, F#, etc.
  - ✓ We support C#.
- Version-controlled
  - ✓ Your 6.1 programs won't break when we release 6.2.
- Documented
  - ✓ Getting Started guide and Class Reference installed with TD
  - ✓ Additional examples on CRTech forums

What can you do with OpenTD?

- Connect your program to TD
  - ✓ Attach to already-running instance or start new TD
  - ✓ Connect to one or many instances of TD simultaneously
- Create, query, edit, delete most TD entities
  - ✓ Nodes, conductors, heat loads, etc.
  - ✓ FD surfaces and solids
  - ✓ FE meshes
  - ✓ Lumps, paths, ties, etc.
  - ✓ Case sets, orbits
  - ✓ Thermophysical and optical properties
  - ✓ Much more!
- Work with units, symbols, and expressions
- Run cases
- Explore, compare and plot results

OpenTD Demo: Organize Model with LINQ



00:03.27

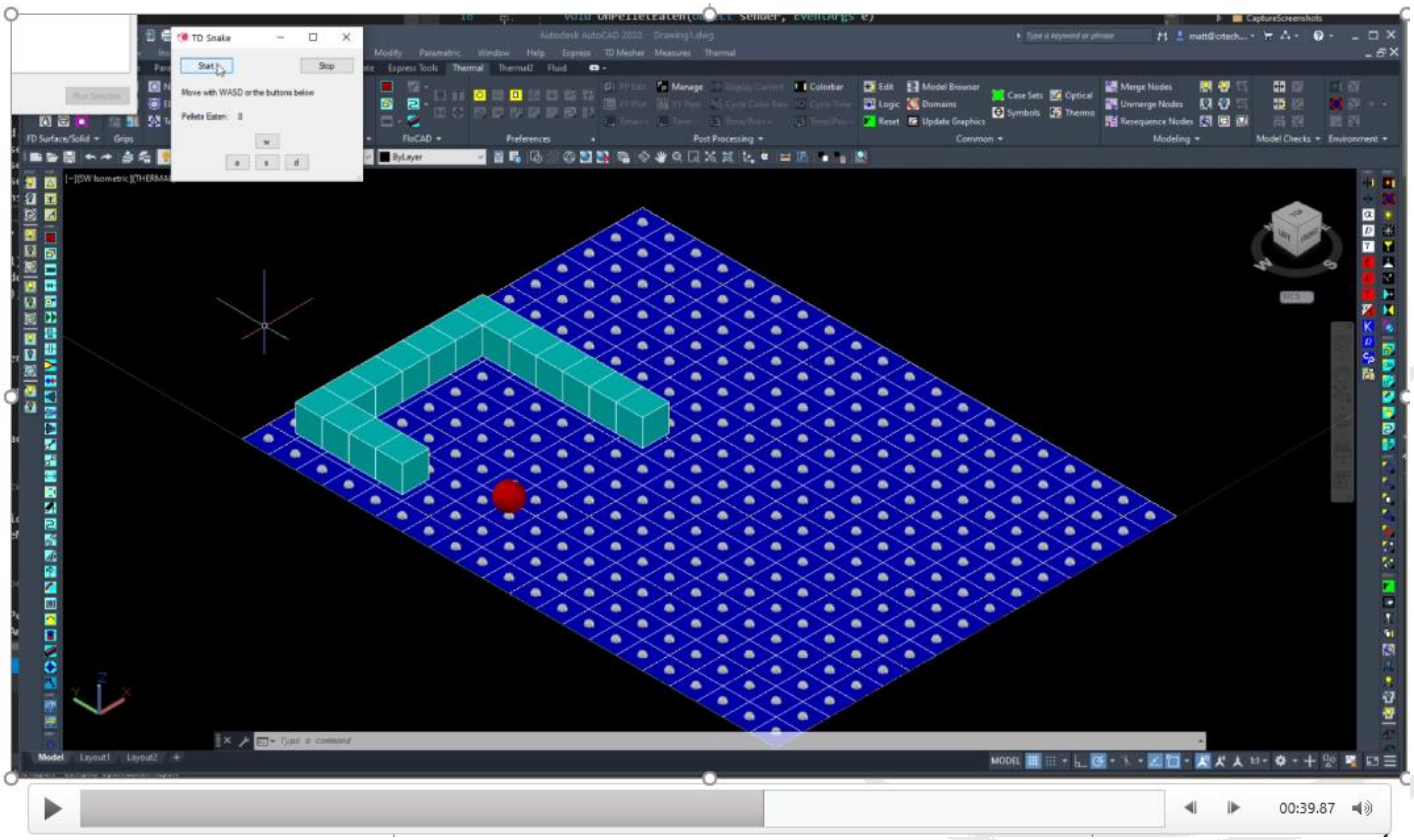
What's Ahead in OpenTD:
An API for Thermal Desktop

NEW FEATURES

Support for More TD Entities

- FD Solids
- Full support for FEM's
- Compartments
- Contour Plots
- Case Set Manager for batch runs
- Pressure Loads
- AutoCAD arcs, circles, ellipses, splines, helices, and polylines
- PID Controller Logic Objects
- Array Interpolation Logic Objects
- Qflow Manager
- UDFA's

OpenTD FD Solids Demo



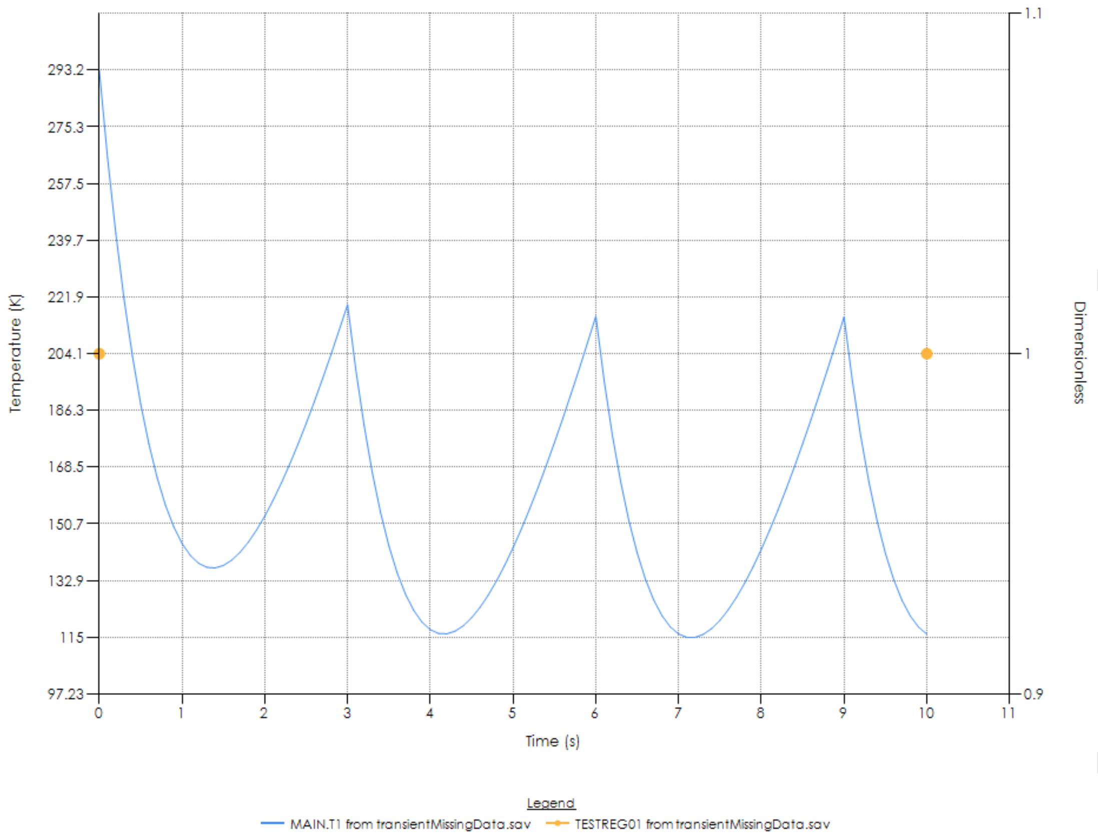
Determine Model Topology from Results

- Read PCS files to determine model topology from solution results
- Use the *Results.Browser* class to create heat maps between submodels or arbitrary groups of nodes/lumps
  - ✓ Combines PCS topology with sav/CSR/etc. results

Handle Missing Data Gracefully

- *GetData* methods now return NaN for records with missing data, and plots/*DerivedDataArrays* handle NaN

Data from transientMissingData.sav



C&R TECHNOLOGIES
www.crtech.com

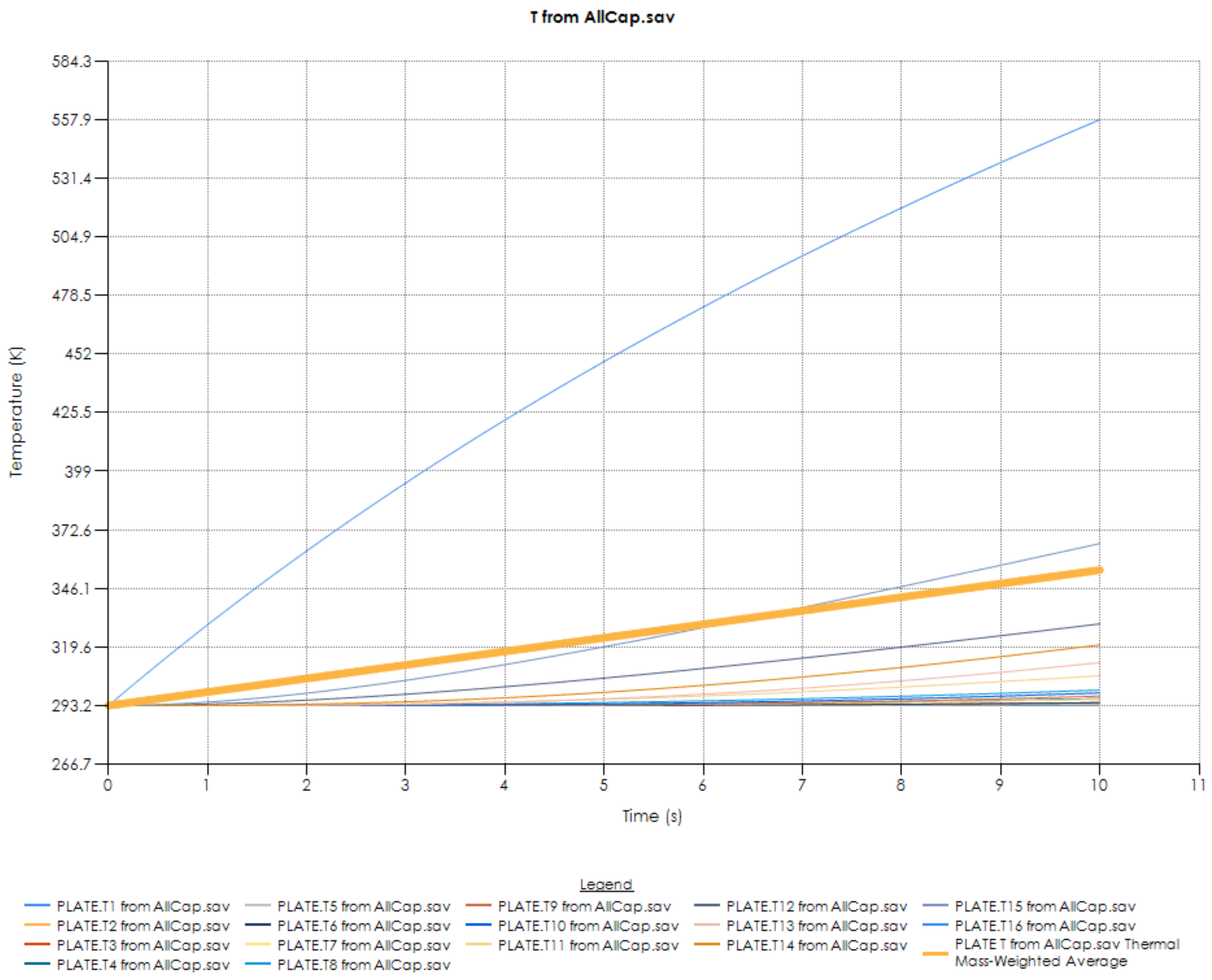
Read Text Transient and CSV Files

- New *IDataset* implementations:
  - ✓ *TextTransientFile*
  - ✓ *SpreadsheetDataFile* (for csv's and csv-like files)
- Create custom *DataSubtypes* to read data from these files and treat it like it came from a save/CSR

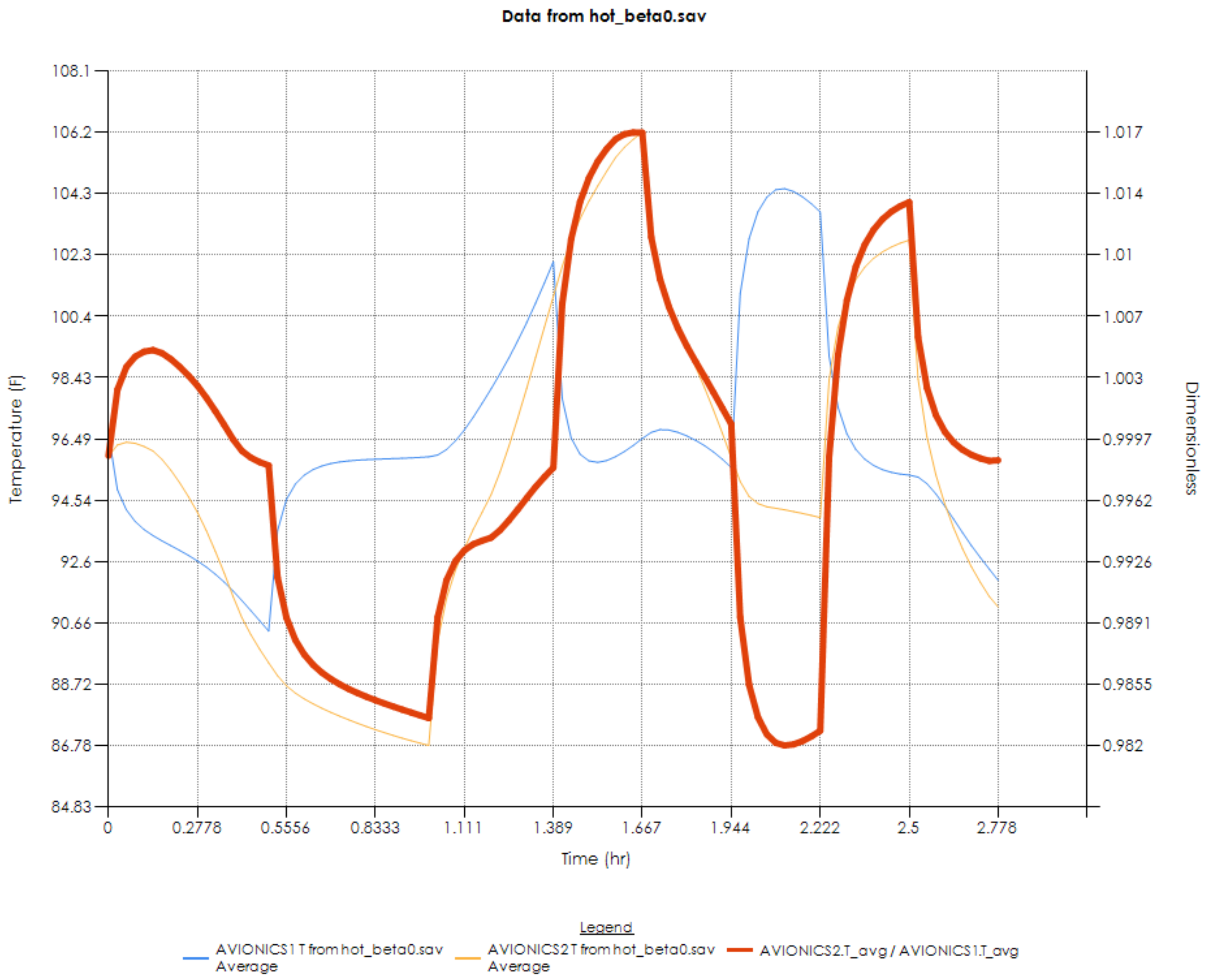
New *DerivedDataArray* Types

- *MaxDataArray* and *MinDataArray* return extreme values of their input *DataArrays* for each record
- *WeightedAverageDataArray*
- *FormulaDataArray* to combine *DataArrays* with an arbitrary, units-aware formula

WeightedAverageDataArray Example

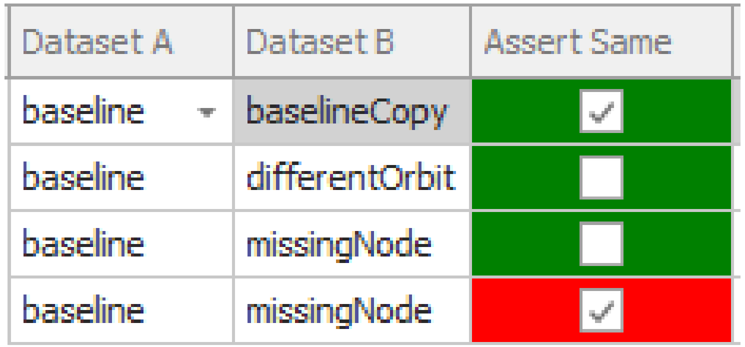


FormulaDataArray Example

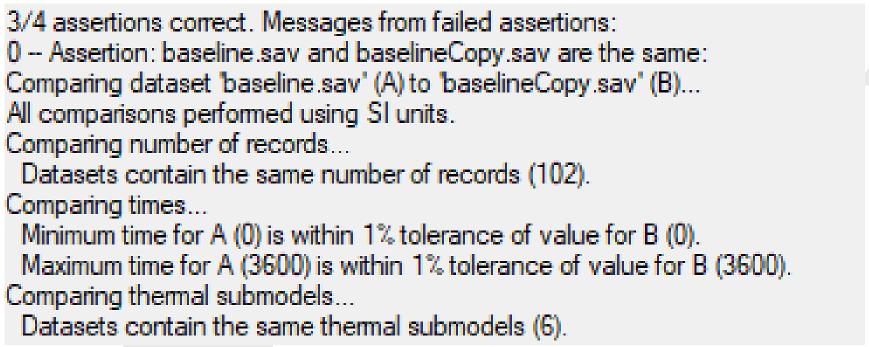


More Ways to Compare Datasets

- Added *CompareAssertion* and *CompareSuite* classes
  - ✓ Combine *Comparer* instances
  - ✓ Compare multiple pairs of Datasets automatically



Dataset A	Dataset B	Assert Same
baseline	baselineCopy	🟩 ✓
baseline	differentOrbit	🟩 ☐
baseline	missingNode	🟩 ☐
baseline	missingNode	🟥 ✓



```
3/4 assertions correct. Messages from failed assertions:
0 -- Assertion: baseline.sav and baselineCopy.sav are the same:
Comparing dataset 'baseline.sav' (A) to 'baselineCopy.sav' (B)...
All comparisons performed using SI units.
Comparing number of records...
  Datasets contain the same number of records (102).
Comparing times...
  Minimum time for A (0) is within 1% tolerance of value for B (0).
  Maximum time for A (3600) is within 1% tolerance of value for B (3600).
Comparing thermal submodels...
  Datasets contain the same thermal submodels (6).
```

Communicate with SINDA/FLUINT

- Use *OpenTDv62.CoSolver* namespace

Misc. New Features

- added *EllipticCylinder* class (accidentally left out of 6.1)
- added methods to get all rectangles, cones, etc. (all FD Surface types)
- added *TdConnectConfig.ShowAcadSplashScreen* and *.AdditionalAcadCommandline* members
- added *CreateIn(ThermalDesktop)* methods to all entities
- added *UpdateIn(ThermalDesktop)* methods to all entities
- added *Connection.IsEmpty()* method
- added *AttachedNodeHandles* member to all FD surfaces (already included for FD solids and finite elements)
- added *DataArrayCollection.Dimension* and *DataItemCount* members
- added *ThermalDesktop.GetOpticalPropDBPathname* and *.GetThermoPropDBPathname* methods
- added *RadiationAnalysisGroupManager.GetDefault* method
- added *DataArrayCollection.GetTranspose* method
- superceded *SubmodelDataArrayCollection* and *DomainNodeDataArrayCollection* with new methods for specifying groups
- added *CreatePipe(DbObject centerline)* convenience method
- added convenient *Write* and *WriteLine* methods to *StandardOutput*
- updated *Contactor/TEC* with new features for 6.2
- added *ThermalDesktop.GetViewNames* method
- now allow *CreatePort* to create ports with no connections by passing an empty Connection (Handle == "")
- update *Plot2d.AddSeries(DataArrayCollection)* to allow using first array as x data
- added *DataArray* copy constructor
- added *DataTypeFamilies* enum (thermal, fluid, other) and methods to determine from *DataTypes* or *DataSubtypes*
- added *ThermalDesktop.GetLayerByName* method
- added *ThermalDesktop.GetCurrentLayer()* and *.SetCurrentLayer(string name)*
- added added *SaveFile.Close()* method and make *SaveFile IDisposable*
- added *CaseSet.ReplaceFilenames* method
- added *UnitsData* ctor that accepts a units expression string
- added read-only members to *Node* class to help navigate FD/FEM networks (*AttachedObjectHandles*, etc.)
- added *Matrix3d.ToString* method
- *ExportNodeInfo* now defaults to returning a *List<string>* with the same strings that would otherwise be written to screen or file
- use *RcEntityData.AnalysisGroups* dictionary to simplify specifying FD Surface radiation analysis groups

Performance Improvements

- speed increases for
  - ✓ *SaveFile*
  - ✓ *CSR*
  - ✓ *Comparer*
  - ✓ *AverageDataArray*
  - ✓ *DataArrayCollection Dataset.GetData(...)*
- *SaveFile* now thread-safe

What's New in OpenTD:
An API for Thermal Desktop

QUESTIONS?

What's New in OpenTD:
An API for Thermal Desktop

BACKUP

Overall Organization

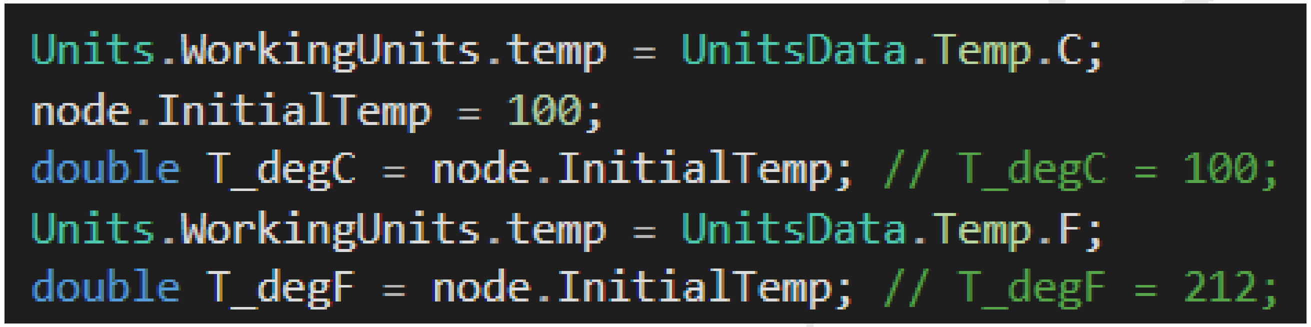
- OpenTD syntax is frozen with each TD release.
- Each version of OpenTD exists in unique dll's and uses unique namespaces.
  - ✓ Programs written for older releases will continue to work unchanged – or they can be updated to the latest OpenTD release by changing the dll's and namespaces they reference (and changing any necessary syntax).
- For 6.2, the following is installed to the GAC:
  - ✓ OpenTDv62.dll (the main OpenTD assembly)
  - ✓ OpenTDv62.Results.dll (for working with sav/CSR)
  - ✓ OpenTDv62.CoSolver.dll (for working with SINDA/FLUINT)
  - ✓ All 6.0 and 6.1 OpenTD dll's
- 6.2 uses the OpenTDv62 base namespace.

The *ThermalDesktop* Class

- Represents one instance of TD.
  - ✓ Can manage as many instances as licenses allow.
- Default behavior is to start a new instance using the latest AutoCAD with a blank drawing. Use *ThermalDesktop.ConnectConfig* to control behavior:
  - ✓ Connect to an already-running instance or start AutoCAD.
  - ✓ Open or connect to a specific dwg file.
  - ✓ Choose which version of AutoCAD to use.
  - ✓ Choose to make AutoCAD invisible.

Units

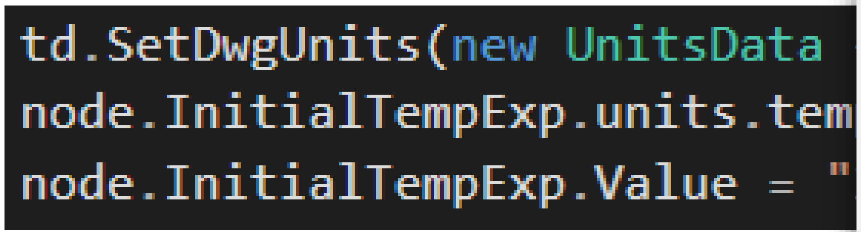
- Can get/set dwg units with *ThermalDesktop.GetDwgUnits()* and *.SetDwgUnits(…)*
- OpenTD also maintains its own unit system, independent of the dwg units.
  - ✓ All dimensional quantities are presented in the static *Units.WorkingUnits* system. Example:



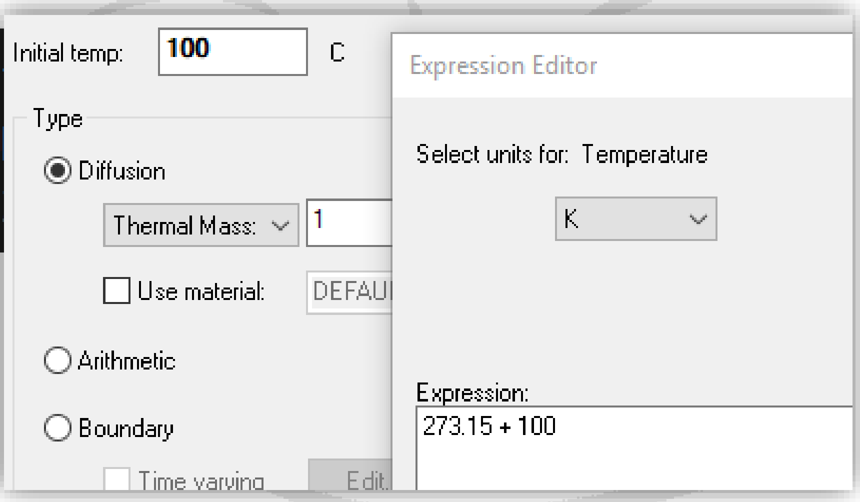
```
Units.WorkingUnits.temp = UnitsData.Temp.C;
node.InitialTemp = 100;
double T_degC = node.InitialTemp; // T_degC = 100;
Units.WorkingUnits.temp = UnitsData.Temp.F;
double T_degF = node.InitialTemp; // T_degF = 212;
```

Expressions

- OpenTD can use expressions anywhere that they can be used in the GUI.
- Like in the GUI, expressions have their own unit system, independent of the dwg units.
  - ✓ When you create a new expression, units default to the *Units.WorkingUnits* system.



```
td.SetDwgUnits(new UnitsData
node.InitialTempExp.units.tem
node.InitialTempExp.Value = "
```



Initial temp: 100 C

Type

[✓ Diffusion]

Thermal Mass: 1

Use material: DEFAU

Arithmetic

Boundary

Time varying Edit

Expression Editor

Select units for: Temperature

K

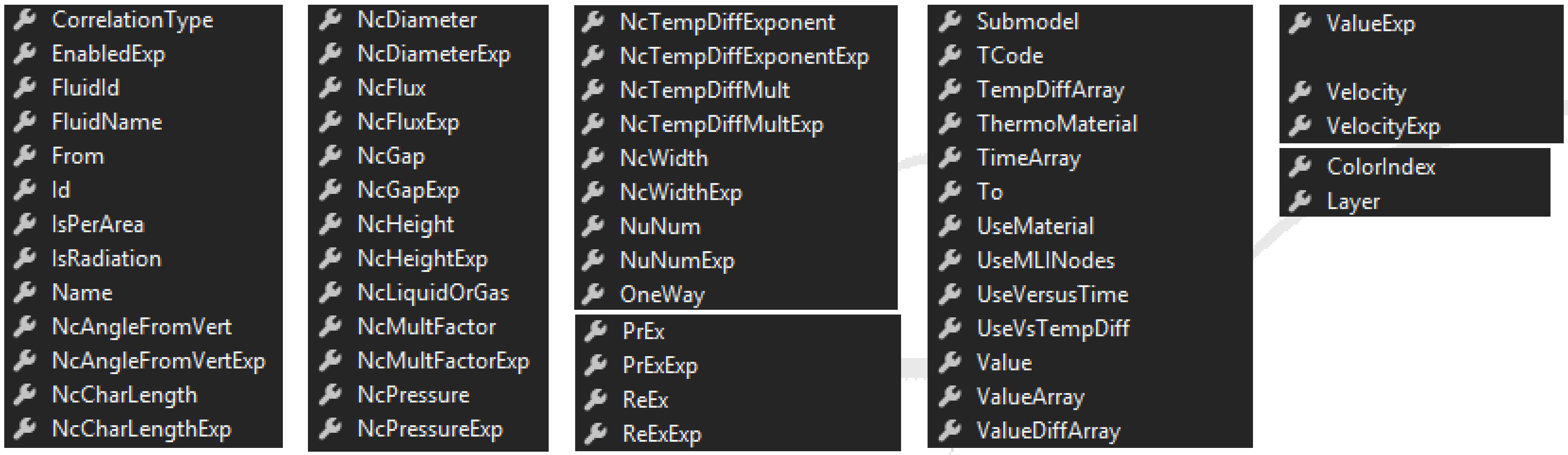
Expression:

273.15 + 100

C&R TECHNOLOGIES
www.crtech.com

Typical OpenTD Class Example: *Conductor*

- OpenTD *Conductor* class can do everything a conductor can do in the GUI
- *Conductor* properties:



CorrelationType
EnabledExp
FluidId
FluidName
From
Id
IsPerArea
IsRadiation
Name
NcAngleFromVert
NcAngleFromVertExp
NcCharLength
NcCharLengthExp

NcDiameter
NcDiameterExp
NcFlux
NcFluxExp
NcGap
NcGapExp
NcHeight
NcHeightExp
NcLiquidOrGas
NcMultFactor
NcMultFactorExp
NcPressure
NcPressureExp

NcTempDiffExponent
NcTempDiffExponentExp
NcTempDiffMult
NcTempDiffMultExp
NcWidth
NcWidthExp
NuNum
NuNumExp
OneWay

PrEx
PrExExp
ReEx
ReExExp

Submodel
TCode
TempDiffArray
ThermoMaterial
TimeArray
To
UseMaterial
UseMLINodes
UseVersusTime
UseVsTempDiff
Value
ValueArray
ValueDiffArray

ValueExp

Velocity
VelocityExp

ColorIndex
Layer

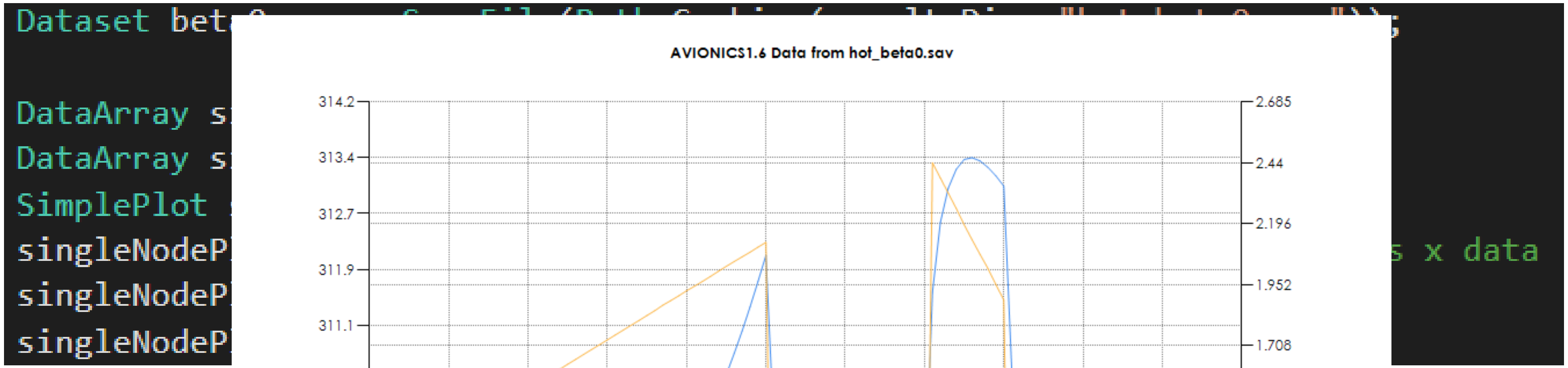
Working with Results

- Use the OpenTDv62.Results.dll to work with save files, CSR's, Text Transient Files, and csv files.
- Abstract *Dataset* class uses the same syntax for accessing all results files.
- Data is returned in *WorkingUnits*, regardless of what's on disk.
- Classes for working with domains, submodels or arbitrary groups of data.
- User-Extensible classes for finding averages, max/min, or performing other operations on data.
- Class for comparing datasets.

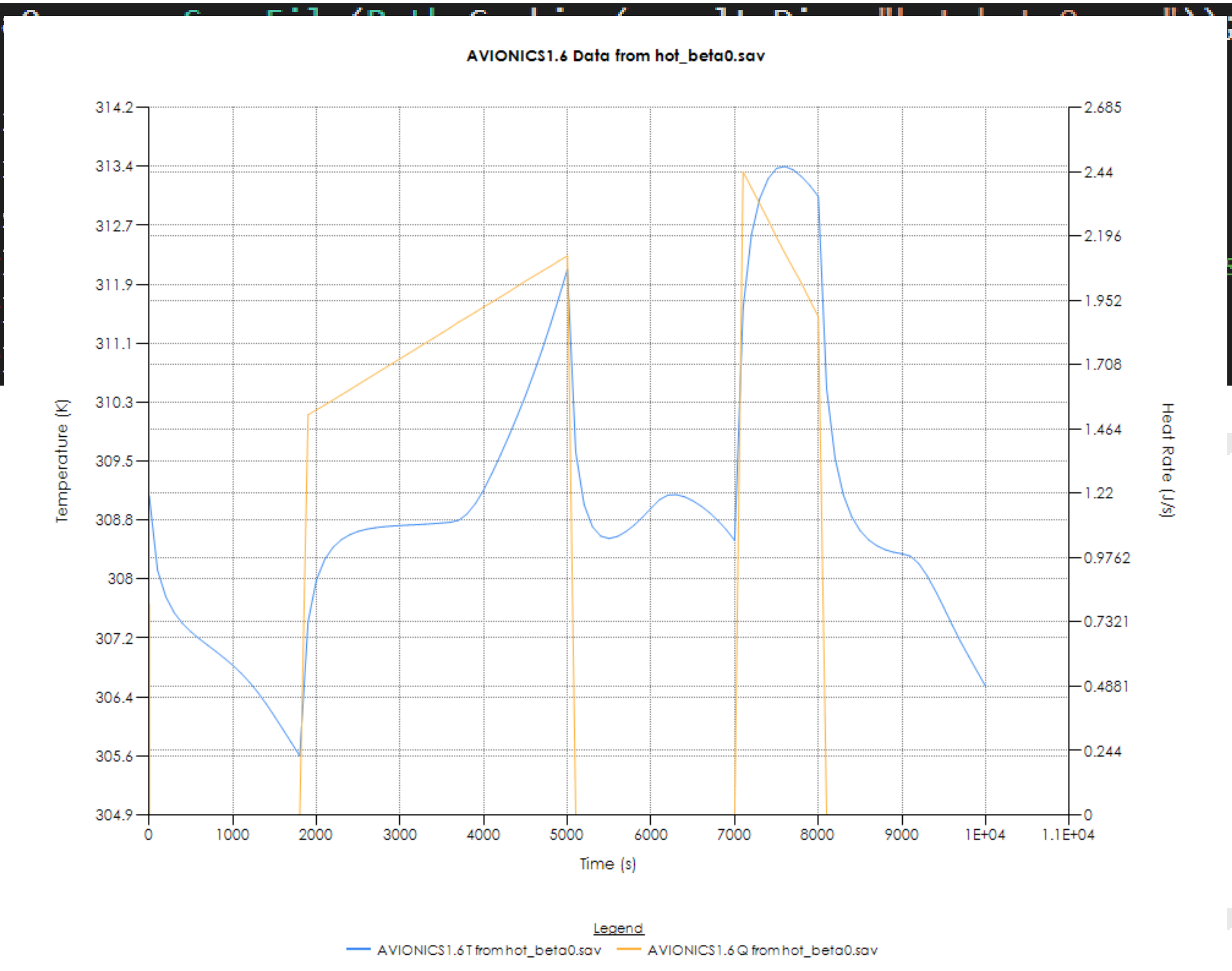
Example Use Cases

- Translate TD models to/from another thermal model format
  - ✓ ESA is working on STEP/TAS to TD and TD to STEP/TAS translators using OpenTD.
- Modify and run models as part of an optimization loop, to create reduced-order models, or within multi-discipline analysis frameworks
  - ✓ LoadPath uses OpenTD in its Veritrek product to create reduced-order thermal models
  - ✓ We are using OpenTD as we explore integrating TD into other analysis frameworks
- Create boundary surfaces based on bitmap data from IR cameras
- Produce statistics from results of 100's of cases
  - ✓ Even the cases could be automatically generated and run using OpenTD.
- Compare save files and CSR's
- Create a simplified front-end for a thermal model
- Capture screenshots of a model for a report
- Clean up a model by searching the AutoCAD database and automatically putting entities into layers
- Run cases in a batch and email the analyst after each run finishes
- Use OpenTD in an ad hoc way to automate small tasks

Exploring Results Demo #1: T and Q for 1 Node from 1 sav



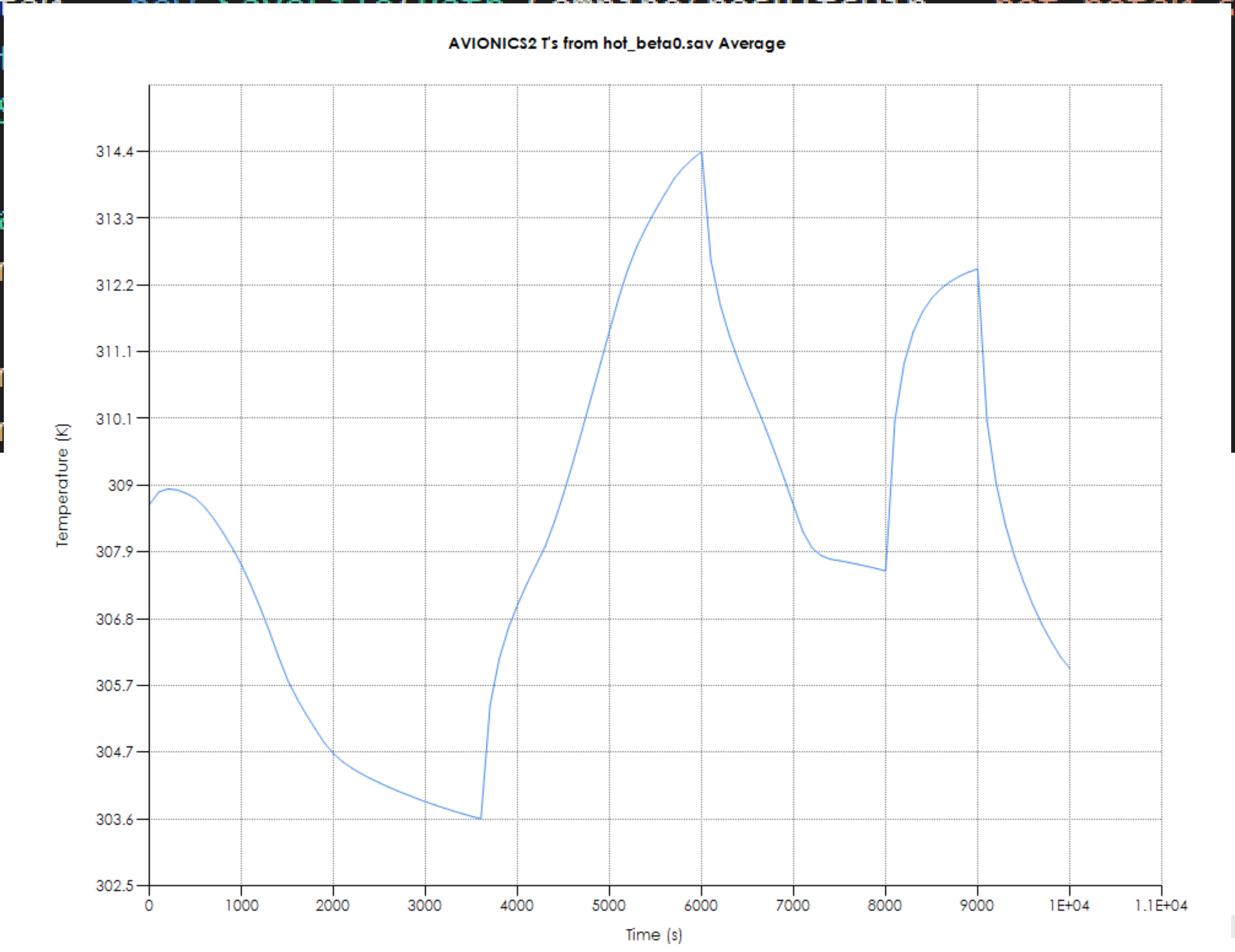
```
Dataset bet                                                    ;

DataArray s
DataArray s
SimplePlot s
singleNodeP                                                    s x data
singleNodeP
singleNodeP
```



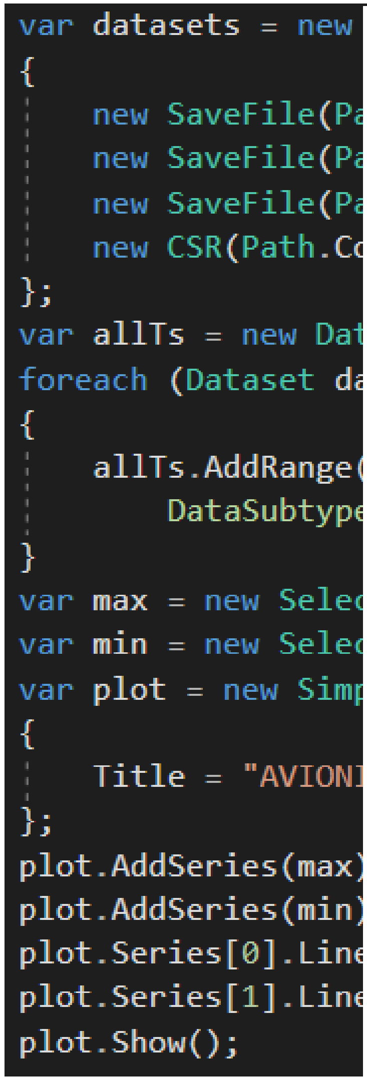
C&R TECHNOLOGIES
www.crtech.com

Exploring Results Demo #2: Average T for 1 Submodel from 1 sav

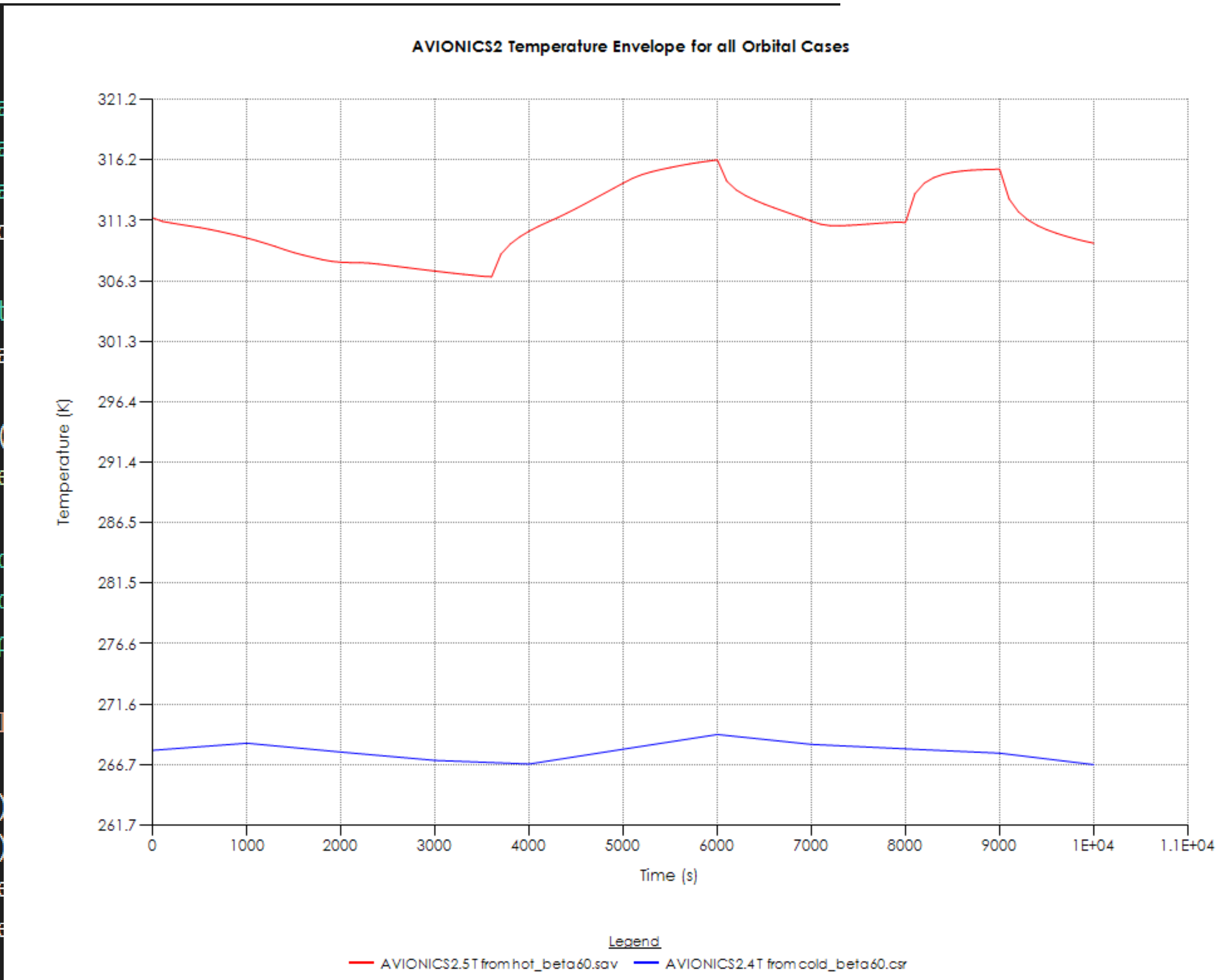
```
Dataset beta0 = new SaveFile(Path.Combine(resultsDir, "hot_beta0.sav"));
SubmodelDat
     = new S                                                  beta0);

AverageData
averageSubm
SimplePlot
averageSubm
averageSubm
```



Exploring Results Demo #3:
Max/Min T for 1 Submodel from 3 sav's and 1 CSR



```
var datasets = new
{
    new SaveFile(Pa
    new SaveFile(Pa
    new SaveFile(Pa
    new CSR(Path.Co
};
var allTs = new Dat
foreach (Dataset da
{
    allTs.AddRange(
        DataSubtype
}
var max = new Selec
var min = new Selec
var plot = new Simp
{
    Title = "AVIONI
};
plot.AddSeries(max)
plot.AddSeries(min)
plot.Series[0].Line
plot.Series[1].Line
plot.Show();
```


AVIONICS2 Temperature Envelope for all Orbital Cases
Temperature (K)
1E+04
1.1E+04
Time (s)
Legend
AVIONICS2.5T from hot_beta60.sav
AVIONICS2.4T from cold_beta60.csr