

TAITherm / CoTherm 2022.1

Public Release May 17, 2022

Steven Patterson

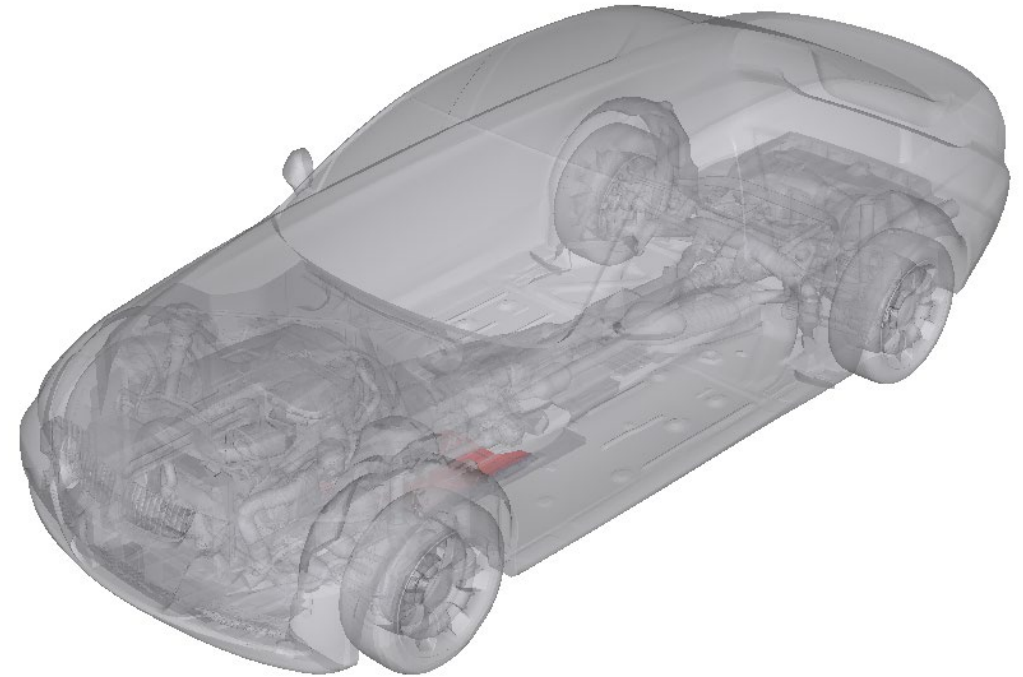
Josh Pryor

TAITherm 2022.1.0

- Advanced graphics engine performance with large models
- Realize full model setup automation through tdfiopy
- Advancements in human thermal modeling

Advanced Graphics Engine Migration

- New graphics engine with superior performance
 - Default graphics engine
 - Open models faster
 - Interact with models faster
 - Improved look and feel
- Previous graphics engine available as a fallback for legacy functionality



Benefits

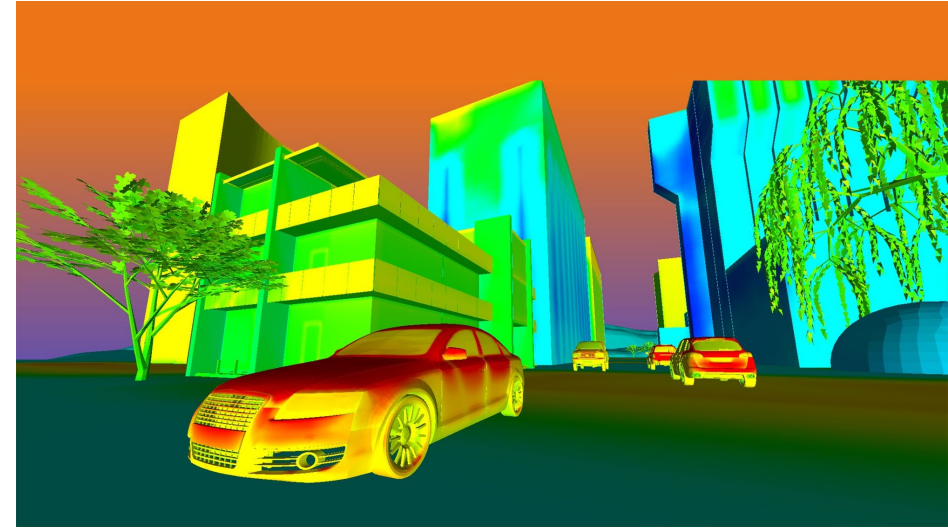
- Increase productivity
- Create better looking images for reports and presentations

Performance Benchmarks

Comparing 2022.1 with advanced graphics to legacy graphics in the previous version:

- 30% faster Thermal Reports
- 40% reduction in RAM for graphically post processing large models
- 30% faster in head to head comparison of a 12 action sequence on a 7 million element model
- Able to open and work with 20M+ element models (previously impossible)

* Actual speedups are model and case dependent



Graphics Demo

Model Setup and Processing Automation

- The power of TDF I/O is now accessible to Python script writers (no compiler required)
- Reduce CAE turn-around time through model setup and results processing automation
- Access settings not available in the summary table

```
In [1]: import tdfiopy

In [2]: tdfio = tdfiopy.TdfIO()

In [3]: tdfio.readFile("C:/Program Files/TAITherm/2021.2.4/
examples/demoship.tdf")

In [4]: tdfio.numberOfParts()
Out[4]: 24

In [5]: tdfio.numberOfElements()
Out[5]: 9322

In [6]:
```

Benefits

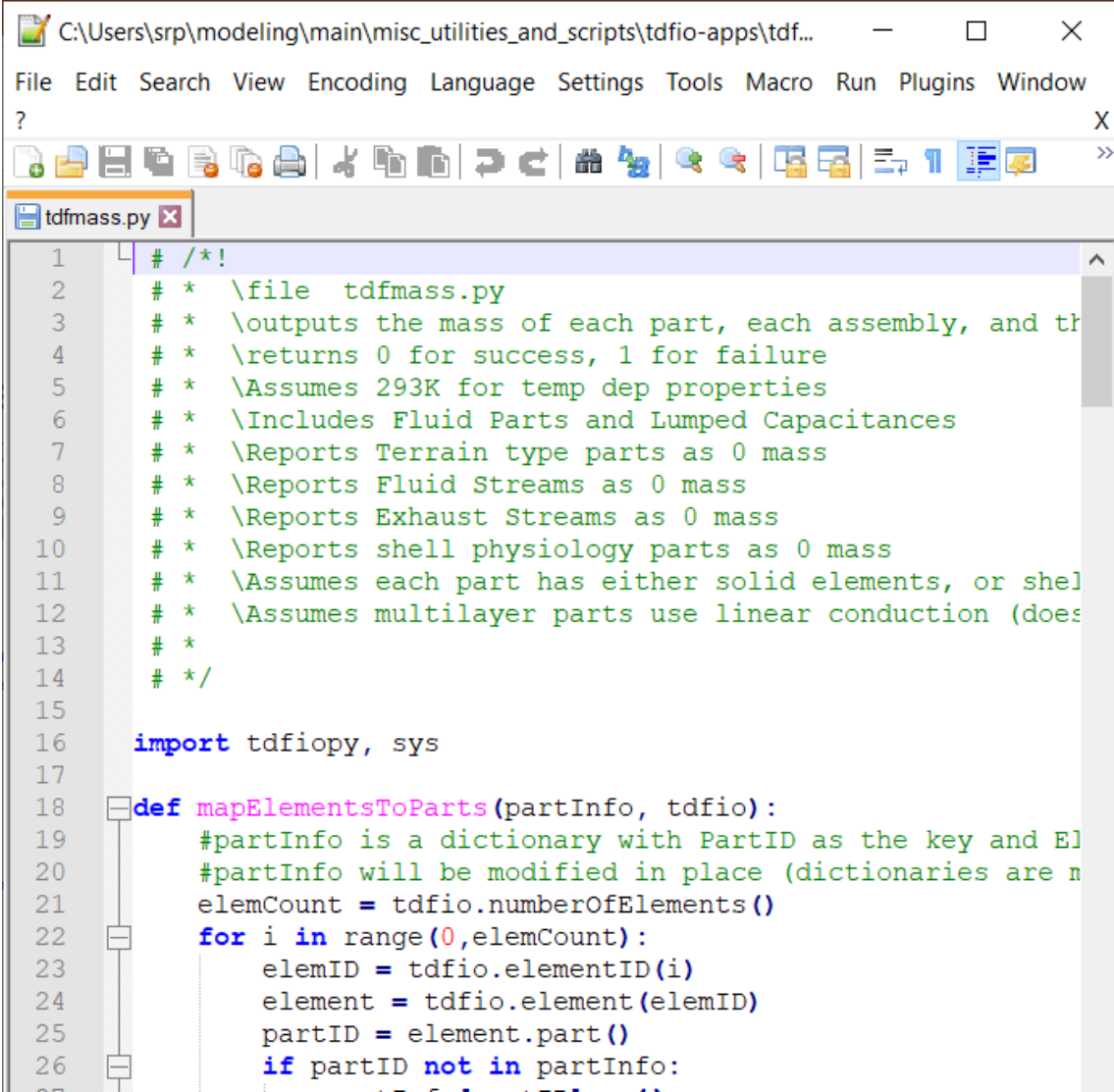
- Fully automate model setup
- Create simulations faster with fewer mistakes

What is TDF I/O?

A programming library to read/edit data in a TDF

What can I do with it?

- Write custom utilities to:
 - Read / Create / Edit mesh
 - Read / Create / Edit thermal links
 - Assign Part properties
 - Assign Material properties
 - Assign Conduction Rules
 - Access model results
 - Etc.
- Create custom file converters
- Automate model setup
- Aggregate model settings
- Extract model results



```
C:\Users\srp\modeling\main\misc_utilities_and_scripts\tdfio-apps\tdf...
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window
?
tdfmass.py
1  # /*!
2  # * \file tdfmass.py
3  # * \outputs the mass of each part, each assembly, and th
4  # * \returns 0 for success, 1 for failure
5  # * \Assumes 293K for temp dep properties
6  # * \Includes Fluid Parts and Lumped Capacitances
7  # * \Reports Terrain type parts as 0 mass
8  # * \Reports Fluid Streams as 0 mass
9  # * \Reports Exhaust Streams as 0 mass
10 # * \Reports shell physiology parts as 0 mass
11 # * \Assumes each part has either solid elements, or shel
12 # * \Assumes multilayer parts use linear conduction (does
13 # *
14 # */
15
16 import tdfiopy, sys
17
18 def mapElementsToParts(partInfo, tdfio):
19     #partInfo is a dictionary with PartID as the key and El
20     #partInfo will be modified in place (dictionaries are n
21     elemCount = tdfio.numberOfElements()
22     for i in range(0,elemCount):
23         elemID = tdfio.elementID(i)
24         element = tdfio.element(elemID)
25         partID = element.part()
26         if partID not in partInfo:
```

TDF I/O vs User Routines

TDF I/O	User Routines
Executed from a shell or other program before or after running a solution	Executed within TAITherm while running a solution
Written in C++ or Python	Written in C++, Python, or JavaScript
Read and edit mesh and part definitions	Read only mesh and part definitions
Assign values in the model setup	Assign values within the solver
Access results after a run	Access results during a run
Accessible within CoTherm Journals	Accessible within TAITherm

NOTE: Make sure you are referencing the correct documentation with using either of these frameworks (many functions have similar names). Functions and classes for the Solver API cannot be used in TDFIO utilities and TDFIO functions and classes do not exist in the solver API.

Multiple ways to access tdfiopy

Execute through the included interpreter

Explore in the interactive shell

Install in your preferred Python environment

Expand automation within CoTherm journals

C:\tmp>"C
D-6\tdfiop>>>
>>>
C:\tmp>
>>> import
>>> tdfio
>>> tdfio
This vers
>>> tdfio
24
>>> tdfio
9322
>>>

C:\Program Files\TAITherm\2022.1.0-RC-BUILD

Spyder (Python 3.9)

File Edit Search Source R

C:\Program Files\TAITherm\2022.1.0-RC-BUILD

py x tdfGeom2Huma

1 # \file te
2 # \brief Us
3 #
4 #-----
5 # Copyright
6 #
7
8 from tdfiopy
9
10 def tetrahe
11 # verte
12 nVertic
13 vertexI
14 vertex
15
16
17
18
19 # part

Untitled* - CoTherm

File Edit View Process Help

Key Pr
Docum

Filter

Bas

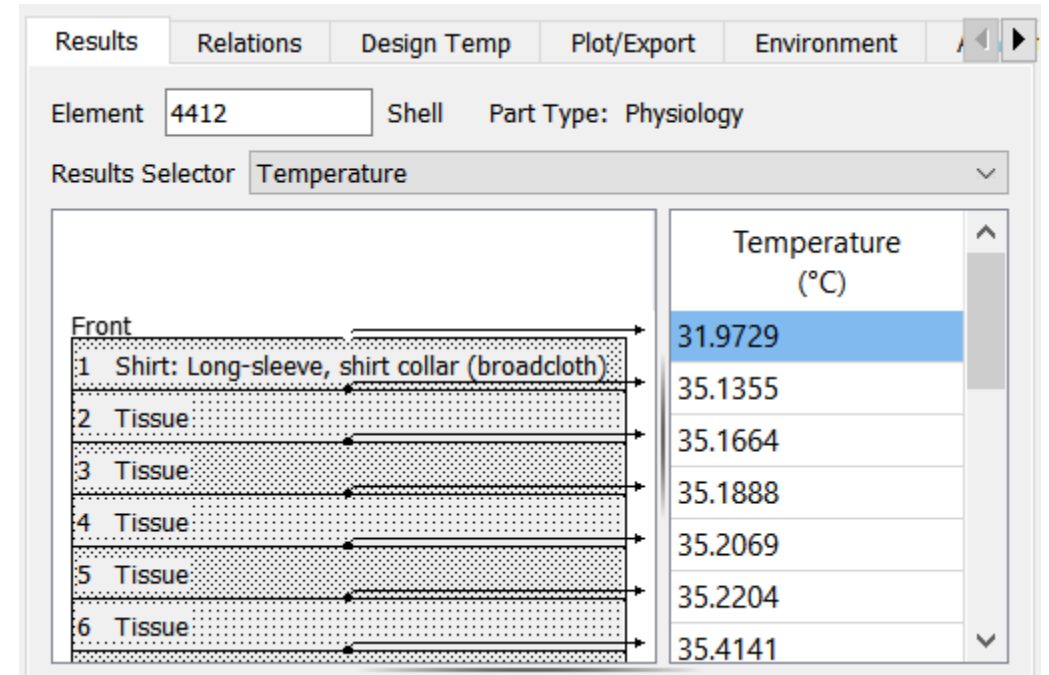
Edit File Contents

```
1 # \file modelinfo.py
2 # \brief Reads some model information from a TDF file and prints it out
3
4 #-----
5 # Copyright (C) 2022 ThermoAnalytics, Inc.
6 #-----
7 from tdfiopy import *
8 import sys
9
10 #
11 # Reads some model information from a TDF file and prints it out
12 #
```

TDFIOPY Demo

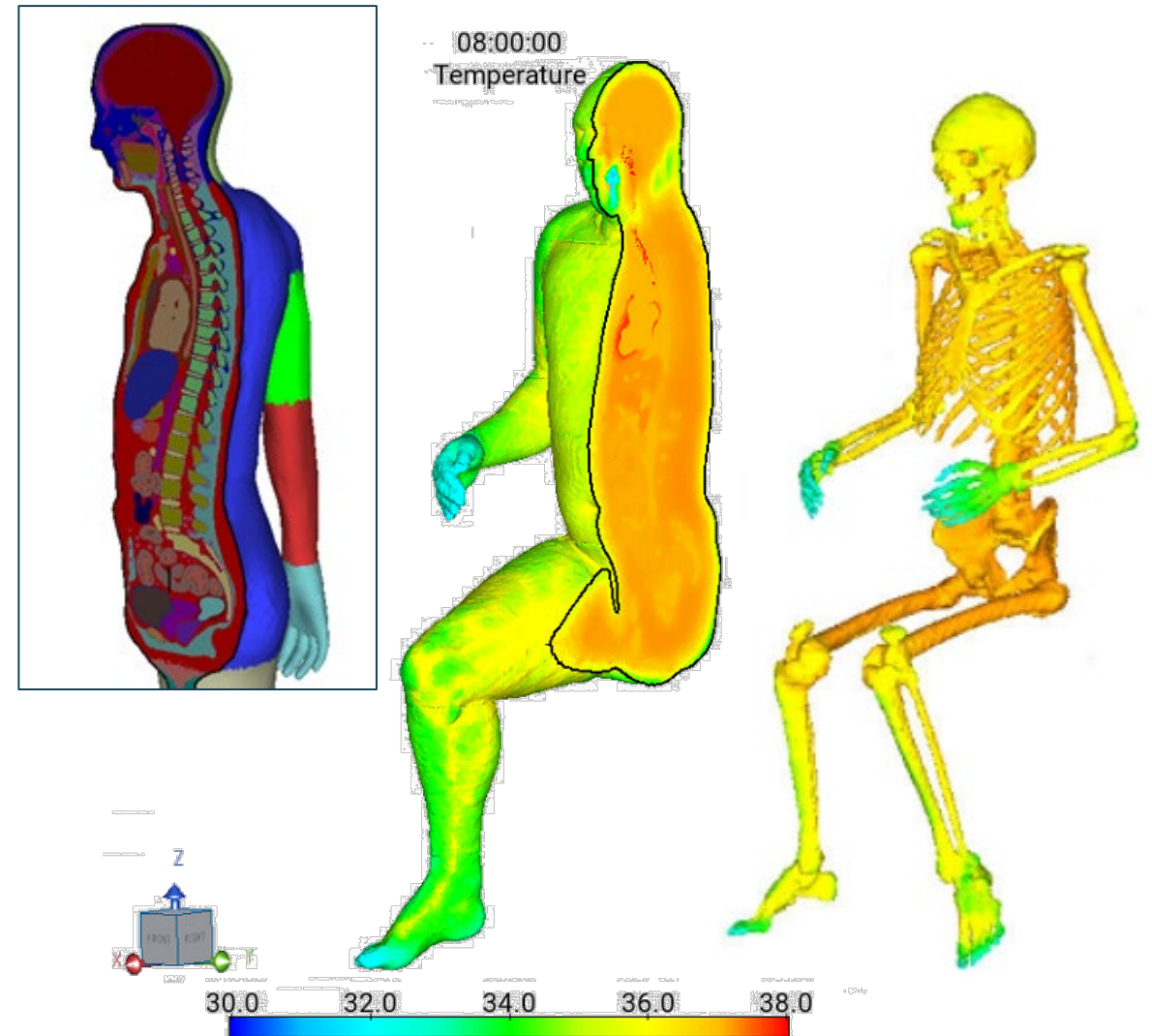
Human Thermal Enhancements

- Materials now shown in the post processor
- Berkeley setpoints utility log file
- The “AppendToExistingResults” keyword now copies results from the transient restart source files



Looking to the future of advanced human modeling: Detailed volume element humans

- Now available for research and evaluation purposes
- Predict tissue temperatures from highly localized heat sources
- Future applications:
 - RF tissue heating
 - Wearable electronics
 - Detailed hand temperatures from heated steering wheels
 - And more



CoTherm v2022.1 product enhancements

Joshua Pryor, CoTherm Product Manager

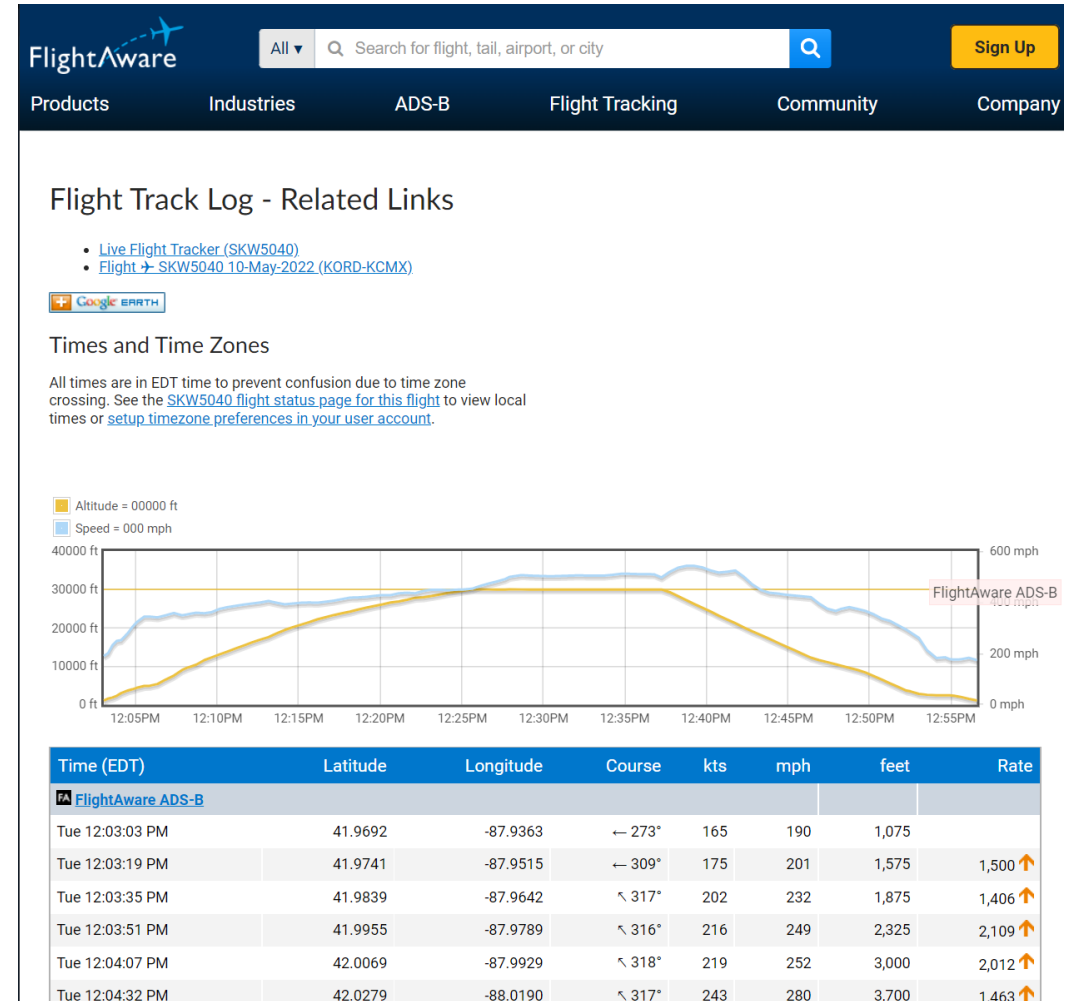


CoTherm 2022.1 Updates

- CoTherm's python includes *tdfiopy* library for easy custom journal operations on TAITherm TDF files
- **Higher performance** & reduced overhead for **fast-running processes**
 - Updates to monitors, plots, variables happen much more efficiently
 - Improves performance for system/1D models, AFSIM sensor modeling, and other fast models
- Improved stability for **parallel sub-processes**
 - Improves reliability when running multiple parallel threads with many tasks
 - Addresses GUI stability when navigating while running

CoTherm + TdfIO (Py) Example: Transient flight plan simulation

- Scenario: Aircraft thermal analysis using a transient flight plan dictating aircraft speed, altitude, GPS position and heading, etc
 - FlightAware or other data sources can be used to specify scenario conditions
- Solution: Use CoTherm to manage scenario conditions and TdfIO-Py to set boundary conditions on TAItherm thermal model for each timestep



CoTherm + TdfIO (Py) Example: Transient flight plan simulation

- FlightAware data copied and pasted into Excel spreadsheet
- Data arrays are linked into CoTherm using Excel Variables
 - Arrays are then assigned into Python for access by TdfIO-Py

	A	B	C	D	E	F	G	H
1	Time (EDT)	Latitude	Longitude	Course	kts	mph	feet	Rate
2	Tue 12:03:03 PM	41.9632	-87.9363	← 273°	165	190	1,075	
3	Tue 12:03:19 PM	41.9741	-87.9515	← 309°	175	201	1,575	1,500
4	Tue 12:03:35 PM	41.9839	-87.9642	↖ 317°	202	232	1,875	1,406
5	Tue 12:03:51 PM	41.9955	-87.9789	↖ 316°	216	249	2,325	2,109
6	Tue 12:04:07 PM	42.0069	-87.9929	↖ 318°	219	252	3,000	2,012
7	Tue 12:04:32 PM	42.0279	-88.019	↖ 317°	243	280	3,700	1,463
8	Tue 12:04:48 PM	42.04	-88.0344	↖ 317°	263	303	4,000	1,266
9	Tue 12:05:04 PM	42.057	-88.0492	↖ 338°	280	322	4,375	1,193
10	Tue 12:05:32 PM	42.0957	-88.0571	↗ 23°	299	344	4,875	670
11	Tue 12:05:51 PM	42.108	-88.0504	↗ 26°	296	341	5,400	1,650
12	Tue 12:06:21 PM	42.137	-88.0025	↗ 25°	303	349	6,550	2,175
13	Tue 12:07:21 PM	42.2351	-87.9793	↗ 24°	311	358	7,575	2,450
14	Tue 12:07:51 PM	42.2727	-87.956	↗ 24°	303	349	9,000	2,521
15	Tue 12:08:08 PM	42.296	-87.9431	↗ 18°	306	352	9,550	1,671
16	Tue 12:08:26 PM	42.3207	-87.9368	↗ 4°	309	356	9,975	1,457
17	Tue 12:08:43 PM	42.3458	-87.9386	↗ 355°	312	359	10,400	2,074
18	Tue 12:09:13 PM	42.3903	-87.944	↗ 356°	310	357	11,600	2,009
19	Tue 12:09:39 PM	42.4264	-87.9483	↗ 356°	314	361	12,275	1,580
20	Tue 12:10:09 PM	42.469	-87.9533	↗ 355°	325	374	13,075	1,575
21	Tue 12:10:39 PM	42.5171	-87.9595	↗ 355°	331	381	13,850	1,575
22	Tue 12:11:09 PM	42.5622	-87.964	↗ 356°	335	386	14,650	1,575
23	Tue 12:11:39 PM	42.6078	-87.9688	↗ 355°	340	391	15,425	1,575

▼ Flight data spreadsheet	File Path	FlightData.xlsx
▼ Flight timestamp data	Sheet Name	Sheet1
	Cell Range	A3:A116
▼ Flight altitude data	Sheet Name	Sheet1
	Cell Range	G3:G116
▼ Flight speed data	Sheet Name	Sheet1
	Cell Range	E3:E116

	C	D	E	F	G	H
1	Longitude	Course	kts	mph	feet	Rate
2						
3	-87.9363	← 273°	165	190	1075	
4	-87.9515	← 309°	175	201	1575	1,500
5	-87.9642	↖ 317°	202	232	1875	1,406
6	-87.9789	↖ 316°	216	249	2325	2,109
7	-87.9929	↖ 318°	219	252	3000	2,012
8	-88.019	↖ 317°	243	280	3700	1,463
9	-88.0344	↖ 317°	263	303	4000	1,266
10	-88.0492	↖ 338°	280	322	4375	1,193
11	-88.0571	↗ 23°	299	344	4875	670
12	-88.0504	↗ 26°	296	341	5400	1,650
13	-88.0267	↗ 26°	296	341	5400	1,650

CoTherm + TdfIO (Py) Example: Transient flight plan simulation

```
18 import tdfiopy
19 import numpy as np
20
21 def setWeatherAltitude( tdfio, altitude ):
22     env = tdfio.environment()
23     if env.environmentType() == tdfiopy.TdfEnvironment.Natural:
24
25         wType = env.modelWeatherType()
26         if wType == tdfiopy.TdfEnvironment.AtAltitude:
27
28             envAlt = env.enviroAltitudeData()
29             envAlt.setAltitude(altitude)
30
31             env.setEnvironAltitudeData(envAlt)
32
33             tdfio.setEnvironment( env )
34
35         else:
36             print("Warning: attempted to set weather altitude, but model is using ground level")
37
38     else:
39         print("Warning: attempted to set weather altitude, but model is using bounding box")
40
41
42 def setAeroLibraryParams( tdfio, speed=None, altitude=None ):
43
44     speedProp = tdfiopy.TdfProperty()
45     if speed != None:
46         speedProp.setValue( speed )
47
48     for i in range( tdfio.numberOfParts() ):
49         partID = tdfio.partID( i )
50         part = tdfio.part( partID )
51
52         for f in [ tdfiopy.TDF.Front, tdfiopy.TDF.Back ]:
53
54             if part.convectionType( f ) == tdfiopy.TdfPart.Library:
55                 libConv = part.libraryConvection( f )
56
57                 if libConv.convectionType() == tdfiopy.TdfLibraryConvection.Aerodynamic:
58                     if speed != None:
59                         libConv.setFlowSpeed( tdfiopy.TdfLibraryConvection.Velocity, speedProp )
60                     if altitude != None:
61                         libConv.setAltitude( altitude )
62                     part.setLibraryConvection( libConv, f )
63                     tdfio.changePart( partID, part )
64
65
66 tdfio = tdfiopy.TdfIO()
67
```

```
15 tdfio.readFile( "$thermalModel" )
16
17 setWeatherAltitude( tdfio, flightAltitudeMeters )
18 setAeroLibraryParams( tdfio, flightSpeedMs, flightAltitudeMeters )
19
20 tdfio.writeFile( "$thermalModel" )
21
```

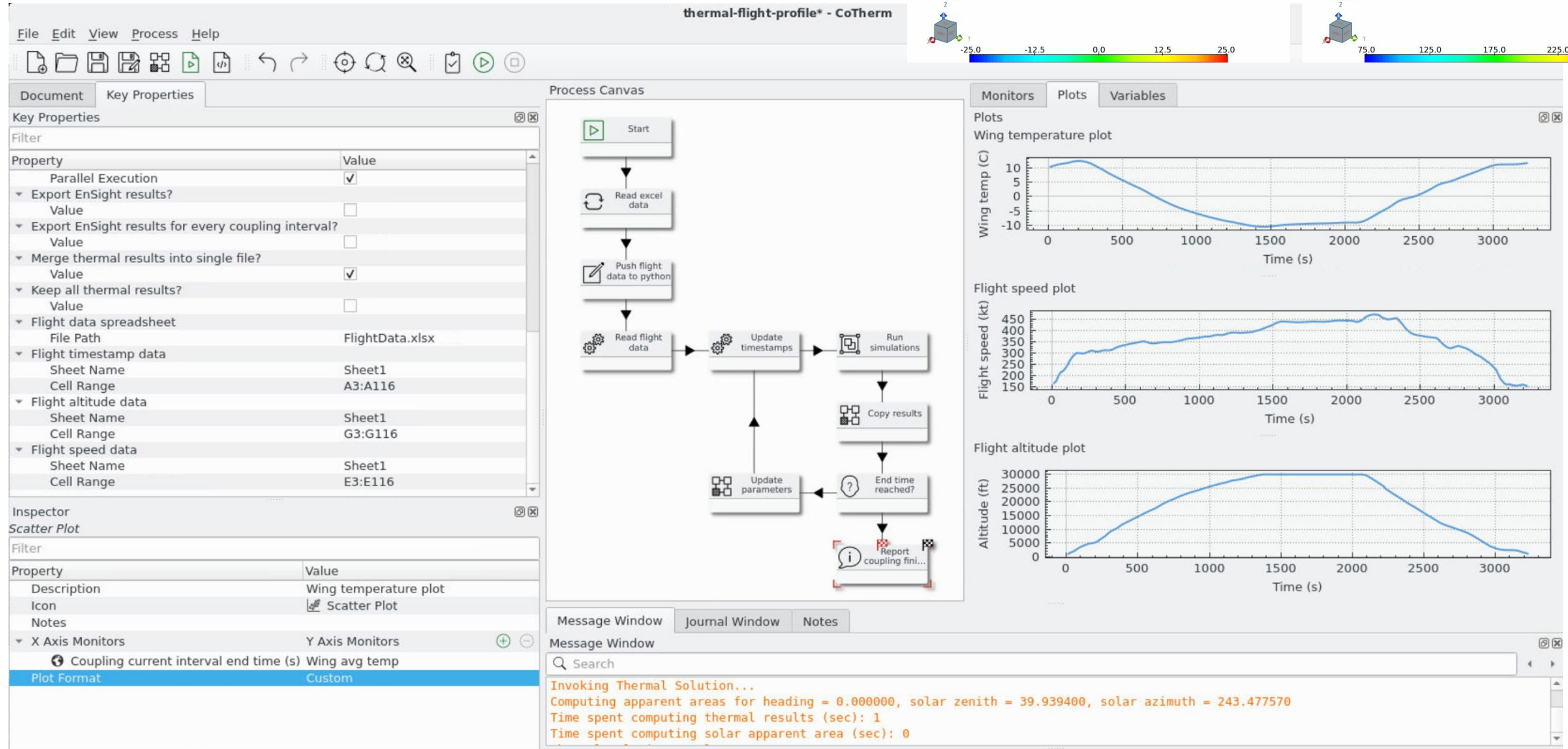
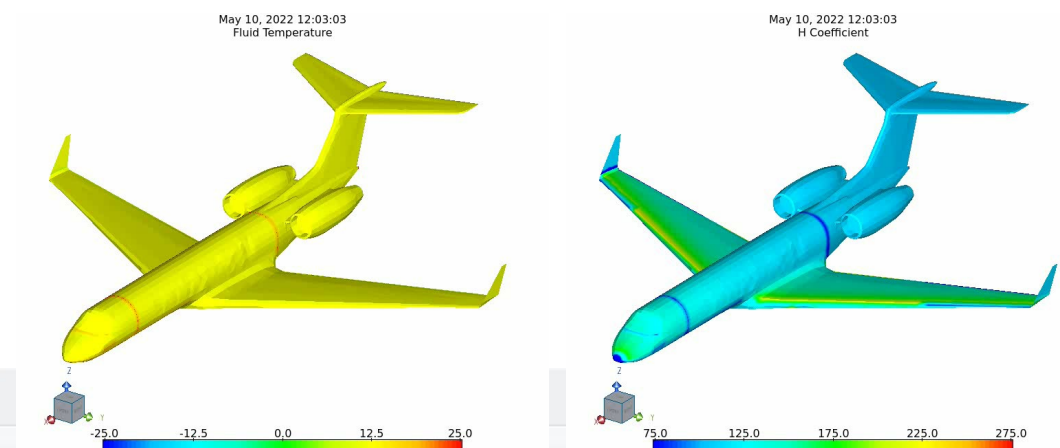
The screenshot displays the TAItherm software interface with several key windows open:

- TAItherm Model Altitude Environment Parameters:** This window is used to set the model's altitude and weather parameters. The **Altitude (m)** is set to 2831.08. The **Weather Data at model altitude** section has **Atmosphere Profile** selected. The **Wind at model altitude** section has **Use vehicle speed** selected. The **Model altitude weather file** section is empty.
- Part Selector:** This window shows the list of parts in the model. The **STR_Cabin_Skin** part is selected, and its properties are shown in the **Properties** tab. The **Conduction and Capacity** section shows the material as **Aluminum** and the thickness as **5.08**. The **Convection** section shows the **Aerodynamic** convection type.
- TAItherm Library Convection:** This window is used to configure the aerodynamic convection parameters. The **Fluid Temperature (°C)** is set to **Ambient Air**. The **Surface Type** is set to **Airfoil**. The **Flow Speed (m/s)** is set to **Velocity Value** with a value of 168.532. The **Altitude Above Sea Level (m)** is set to 2831.08. The **Multiplier (CAF)** is set to 1.0.

Arrows indicate the flow of data from the code to the software interface. The code defines functions for setting weather altitude and aerodynamic parameters, which are then called in the main simulation loop. The software interface shows the results of these calls, with the altitude and flow speed parameters being set correctly.

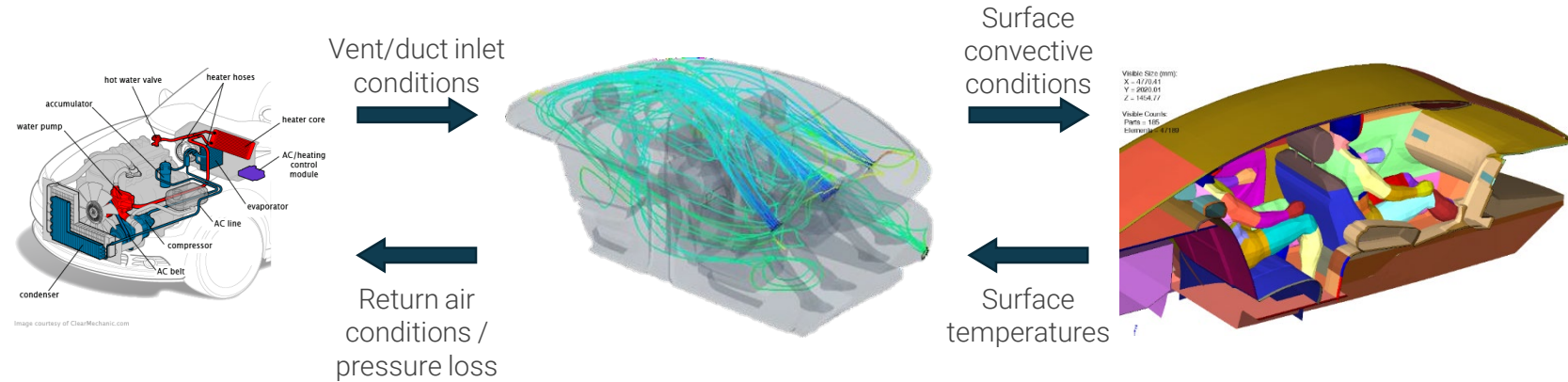
CoTherm + TdfIO (Py) Example: Transient flight plan simulation

CoTherm process plots flight scenario data and key thermal results



CoTherm 2022.1 Process Example Updates

- 3-code coupling with TAItherm, CFD (STAR-CCM+ or OpenFOAM), and system/1D models (FMU)
 - Full transient coupling at defined time intervals between all tools
- Pseudo-transient coupling processes support natural environment & human modeling



1D HVAC model

- Heating/AC component modeling
- Heating/cooling output & power consumption
- Fresh air / recirculation mix

CFD

- Cabin interior flow/temperature
- Convection to internal surfaces

TAItherm

- Heat transfer to/from/within cabin
- Human physiology & comfort
- Environment/solar load

3-code coupling with CoTherm: process demo



Thank you

US Location

ThermoAnalytics HQ
23440 Airpark Blvd.
Calumet, MI 49913

Joshua Pryor

jjp@thermoanalytics.com
www.thermoanalytics.com

<https://support.thermoanalytics.com>

Download new releases, access tutorials & FAQs, and more

