OpenVSP-to-CHARM Automation

2022 OpenVSP Workshop

NASA LaRC & NIA, Hampton, VA

August 9-11, 2022

Brandon Litherland, AST

NASA Langley Research Center

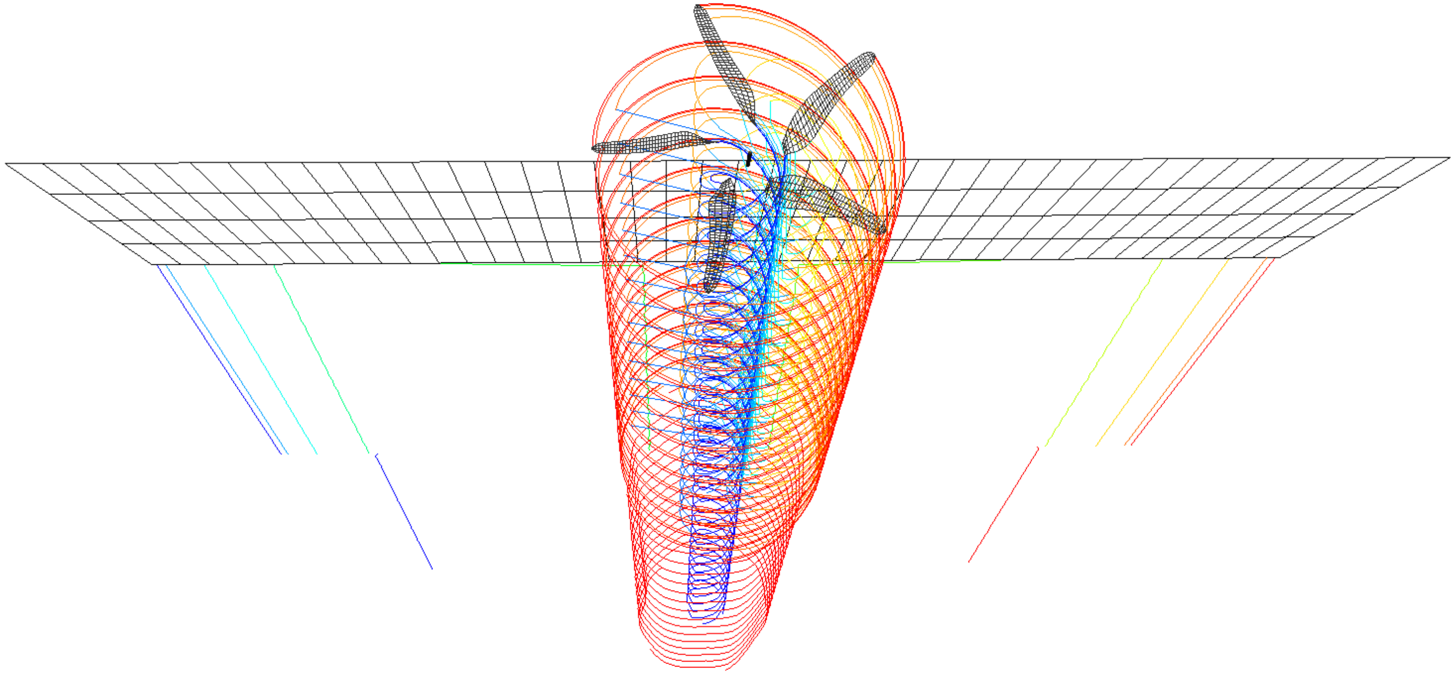
Aeronautics Systems Analysis Branch

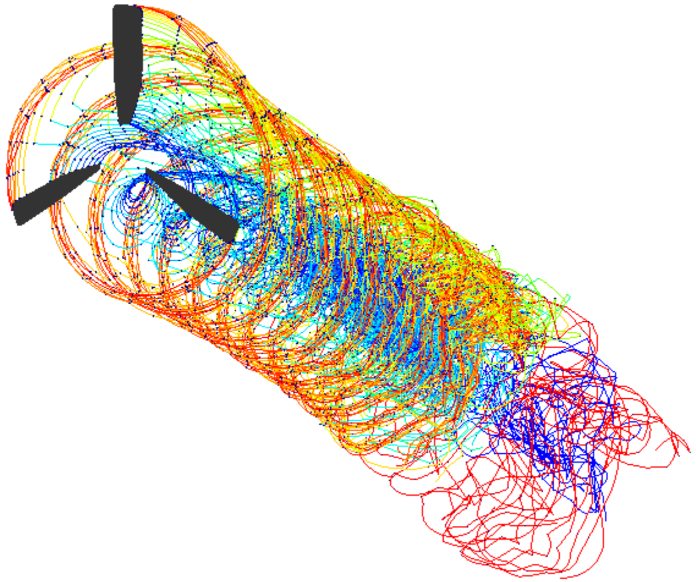
CHARM

Comprehensive Hierarchical Aeromechanics Rotorcraft Model

- Performance, aerostructural loads, trim, wake structure, blade dynamics and deformation, and surface pressure data for forward flight and hover.
- Thickness and loading noise including blade-vortex interaction by interfacing with WOPWOP/PSU-WOPWOP.

Wachspress, et al., "CHARM User's Manual (Version 6.6)," Continuum Dynamics, Inc., Dec. 2020

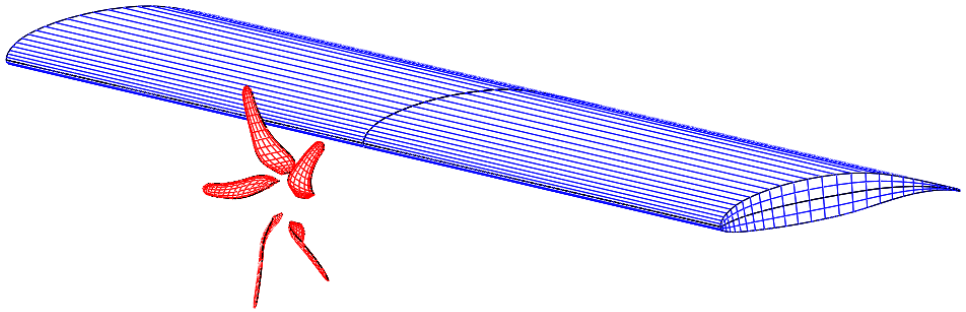


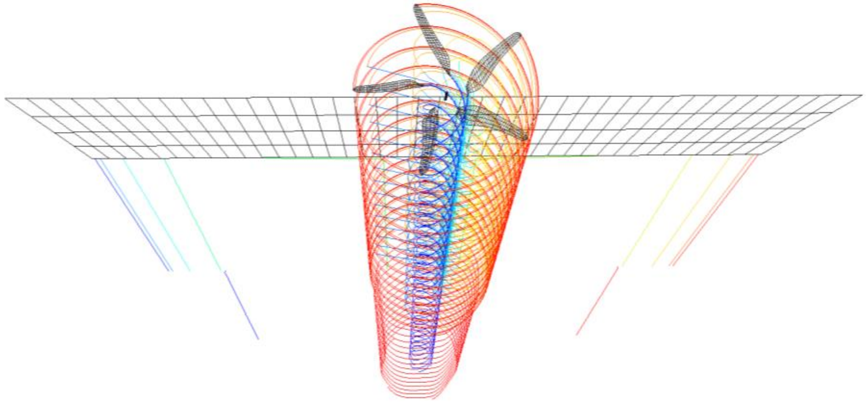


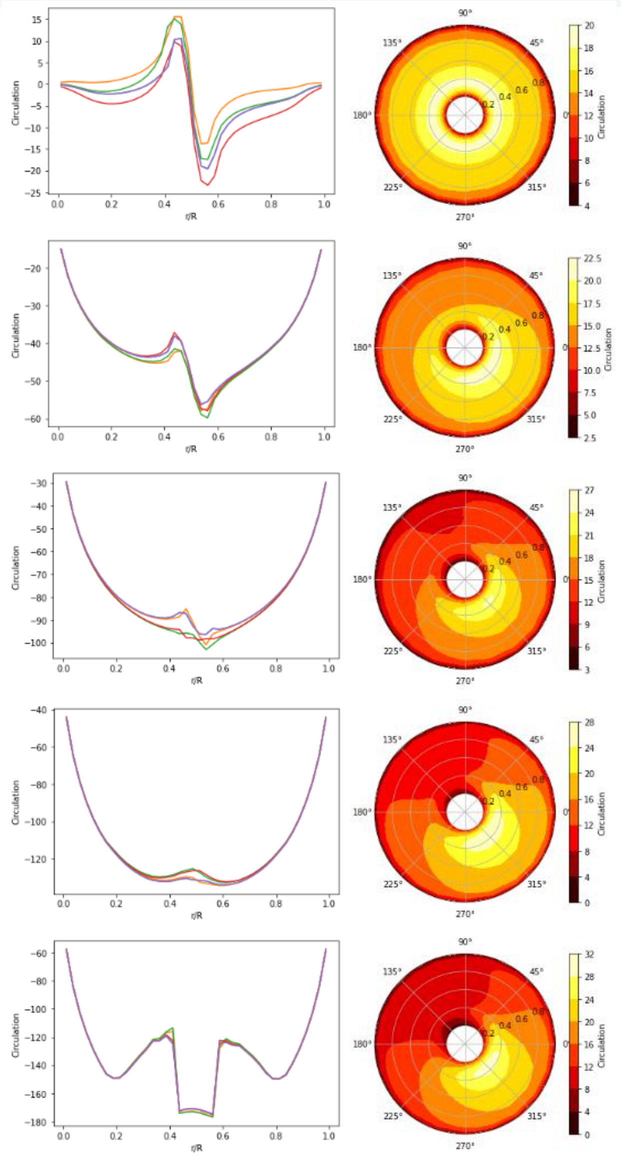
OpenVSP-to-CHARM Automation

Originally created by Alex Gary (Uber) and developed by Jason Welstead (NASA LaRC), the CHARM Automation tool makes it quick and easy to:

1. Generate CHARM input files from OpenVSP geometry.
    - Rotor and Wing Objects built from OpenVSP parameters.
    - OpenVSP model interaction enabled by the OpenVSP Python API.
    - Template files modified to generate Objects and inputs.
    - Mostly working from computer memory rather than multiple file I/O.





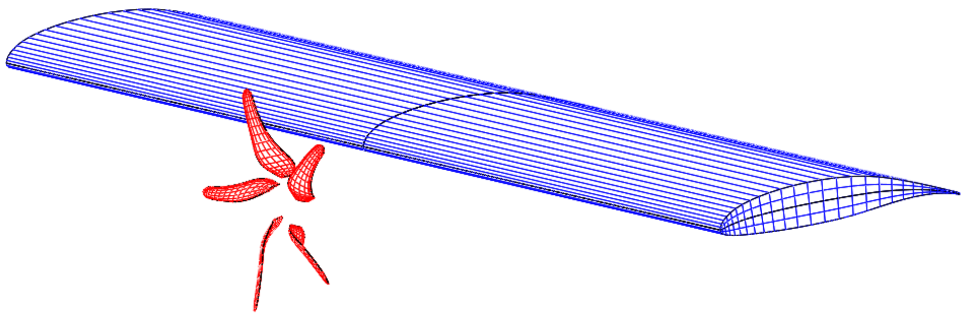


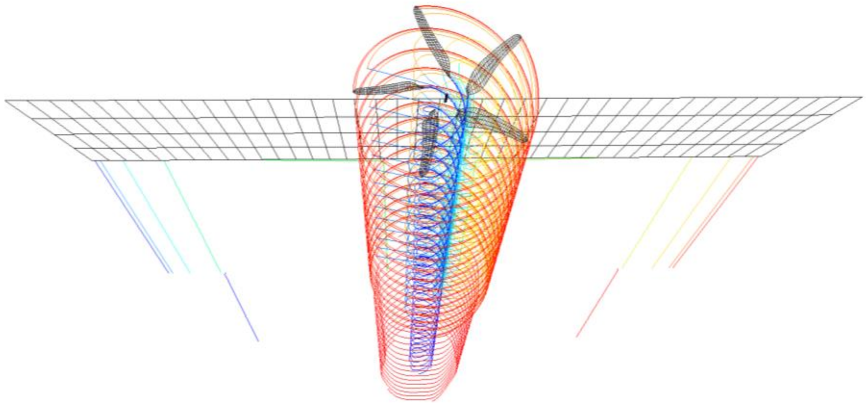
OpenVSP-to-CHARM Automation

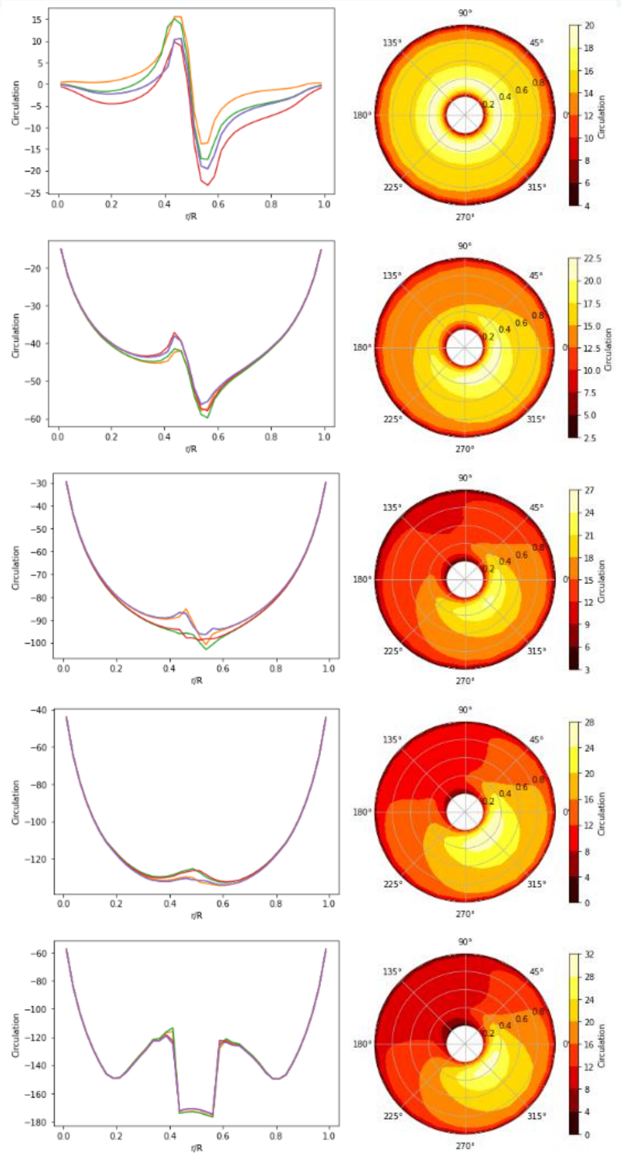
Originally created by Alex Gary (Uber) and developed by Jason Welstead (NASA LaRC), the CHARM Automation tool makes it quick and easy to:

2. Enable broad parallelization of CHARM trade studies.

- Runner utilities send multiple, single-case runs to processors.
- Array-of-arrays defines the design space. Relatively unlimited.
- OpenVSP model and CHARM inputs may be altered for exploration.
- Create your own optimizations within the scripts themselves
  - Example: Genetic algorithms





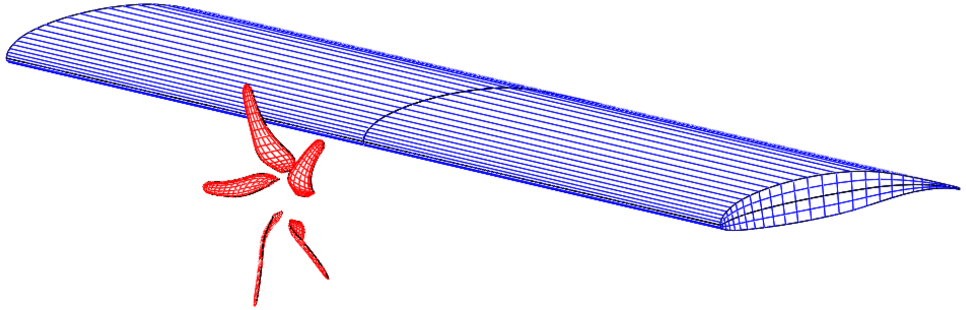


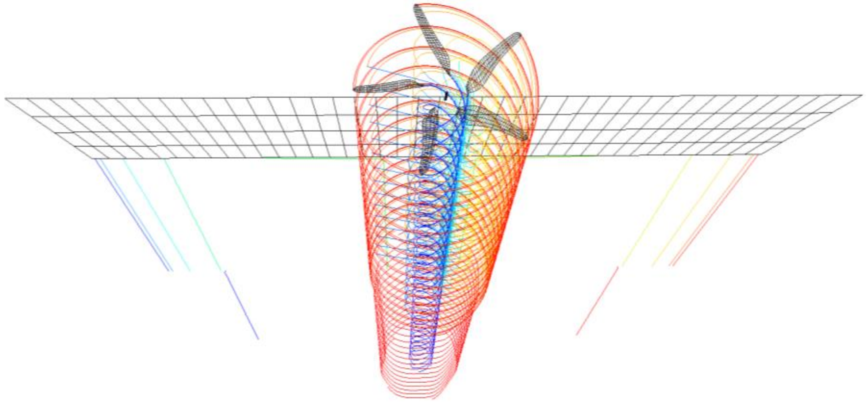
OpenVSP-to-CHARM Automation

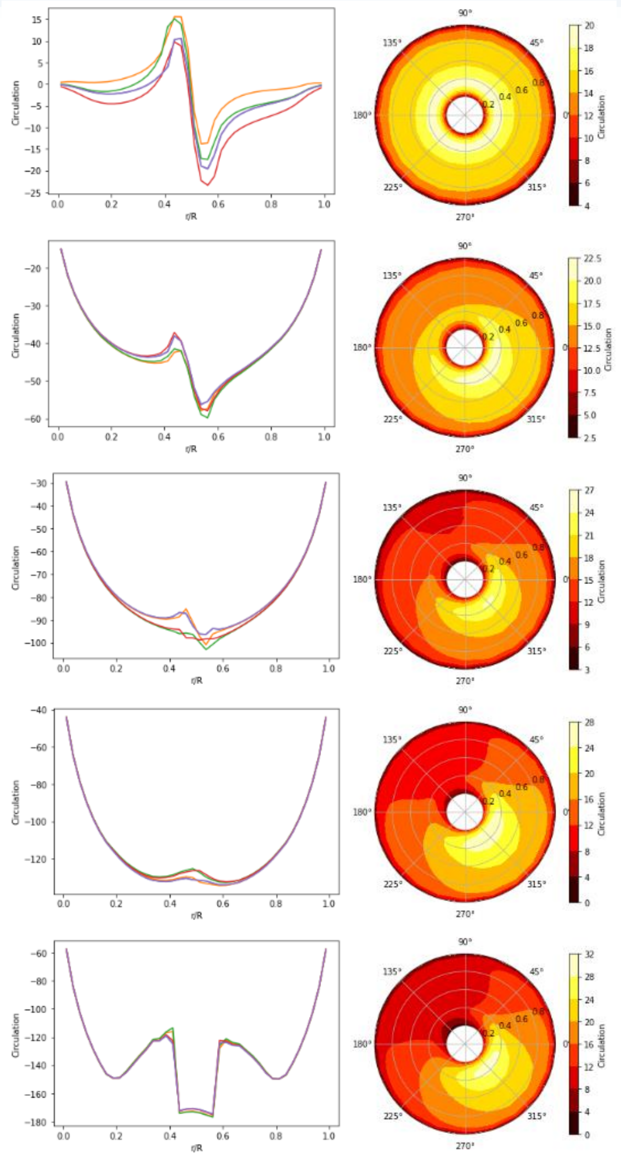
Originally created by Alex Gary (Uber) and developed by Jason Welstead (NASA LaRC), the CHARM Automation tool makes it quick and easy to:

3. Post-process CHARM results:

- Assemble single or multiple runs into data structure for processing.
- Line and contour plots of rotor and wing variables.
- Aircraft, rotor, performance, pressure/velocity, force/moment data, etc.
- Examine and plot rotor performance results.







Requirements

- Active CHARM license(s) to run cases. Post-processing results does not require license.
- OpenVSP API and OpenVSP-to-CHARM Automation Python packages installed.
  - Both packages are distributed with OpenVSP.
- Examples in this presentation are assembled in Jupyter notebooks. Similar Integrated Development Environment (IDE) may be used.

OpenVSP/python/CHARM/charm/README.md

CHARM Automation Python Package (using Python 3.9)

Setup Instructions

1. Follow the installation instructions in README.md located in the `python` directory of the OpenVSP distribution. You will know you are in the correct location if you see the files environment.yml, requirements-dev.txt, and requirements-uninstall.txt. A link to these instructions can be found in the section below.

   NOTE: `...` in a directory path refers to the `python` directory location.

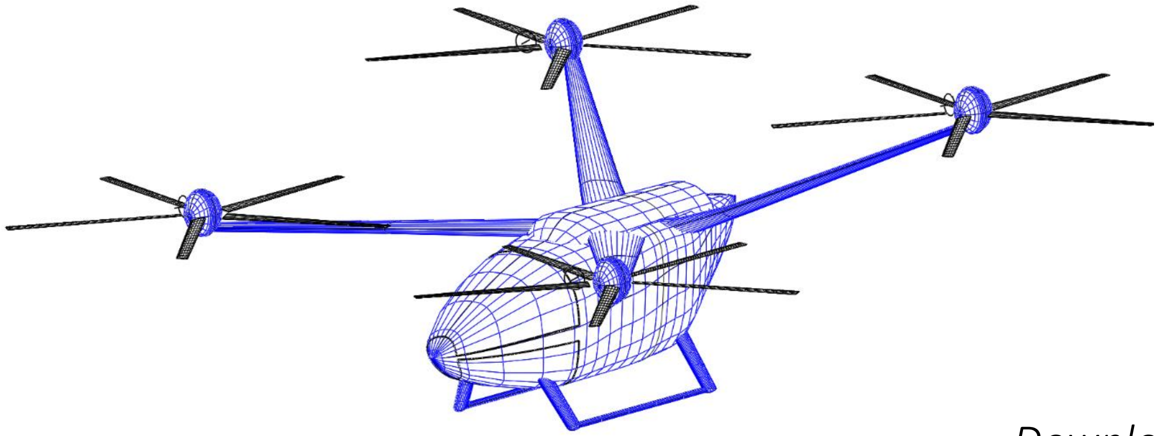
2. After completing Step 1, navigate to `.../charm/charm/charm_fortran_utilities`

OpenVSP Model Setup

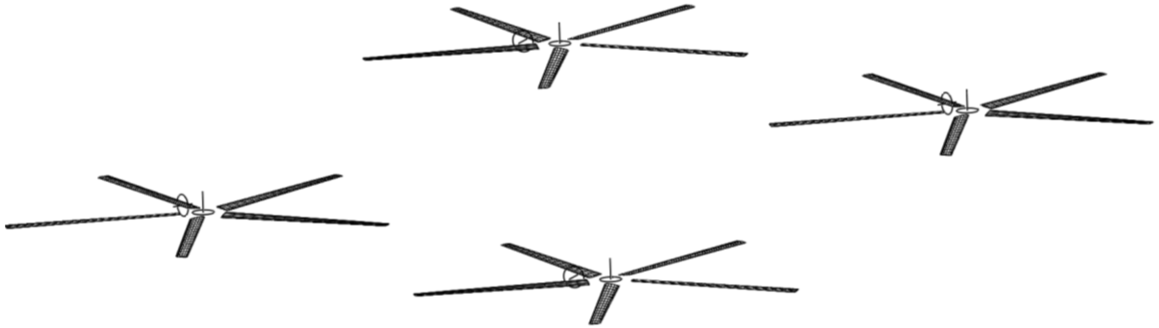
- Choose components and place in separate Set. Recommend “CHARM” set.
  - Structural or non-lifting/thrusting components are generally excluded.
  - Best practice to reduce sectional resolution as low as feasible for wings/rotors.
- Save the model when complete to update VSP3 file.

Example: 6-pax Quadrotor UAM Reference Vehicle (quadcoilflap.vsp3)

OpenVSP Outer Mold Line



OpenVSP CHARM Set

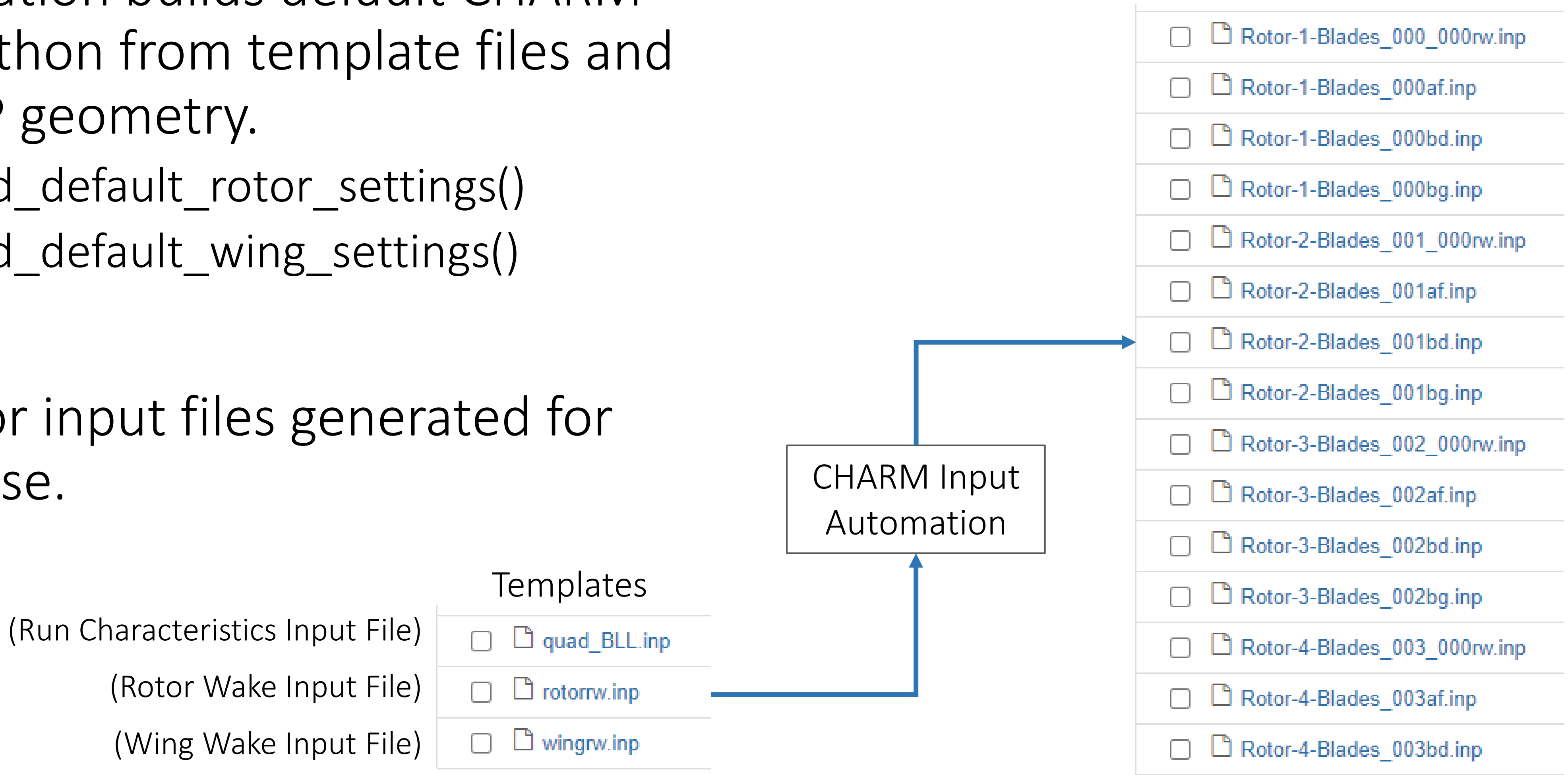


Download NASA UAM reference vehicles: https://sacd.larc.nasa.gov/uam

Building Default Rotor/Wing Objects

- Input automation builds default CHARM objects in Python from template files and the OpenVSP geometry.
  - charm.build_default_rotor_settings()
  - charm.build_default_wing_settings()

- Example rotor input files generated for quadrotor case.



Templates may be included with CHARM or in the /charm/test/ folder from automation.

Modifying Rotor/Wing Objects

Settings may be altered for all rotors and wings, paired symmetric copies of rotors and wings, or individual instances of rotors or wings.

```
rotor_rw = []
with open(os.path.join(path, base_directory, templates_directory, rotor_rw_filename), "r") as f:
    rotor_rw=f.readlines()

rotor_settings = charm.build_default_rotor_settings(degen_mgr=degen_mgr,
                                                    default_rpm=prop_rpm,
                                                    default_template=rotor_rw)
rotor_settings.icoll = 0 # 0 = fixed, 1 = varied to target CT
rotor_settings.initial_collective = prop_collective
rotor_settings.rpm = prop_rpm
prop_ct = src.ct(thrust_lb=800.0,
                 dens_alt_ft=0.0,
                 rpm=prop_rpm,
                 radius_ft=prop_diam_ft/2)
#print(prop_ct)
rotor_settings.ct = prop_ct
rotor_settings.nchord = 4
rotor_settings.nspan_override = 40 # to reduce the nspan 100 setting.
```

Read and build defaults

Simple Rotor Calculation package

Alter default rotor settings

Running Cases

- Single case may be run, if desired, however…
- Parallel submission of multiple cases enables rapid explorations.
  - Perform sweeps of flight conditions or configurations.
  - Alter the OpenVSP geometry parameters and examine the design space.
- Runner utilities assemble the variables and execute functions.
  - Leverages array-of-array format of inputs.
  - Full Factorial, Random, and Latin Hypercube available.
  - Or program your own algorithms.
- Multiple cases take the same time as one to complete.
  - Processors are each running a single case.
  - HPC clusters enable *massive* parallelization of these studies/optimizations.

```
alphas=np.arange(0,10.1,5) # se

print(alphas)
ff=FullFactorial((alphas,)) #o
ff.cases
ff.run(f=mycharmrun,nCores=4)
```

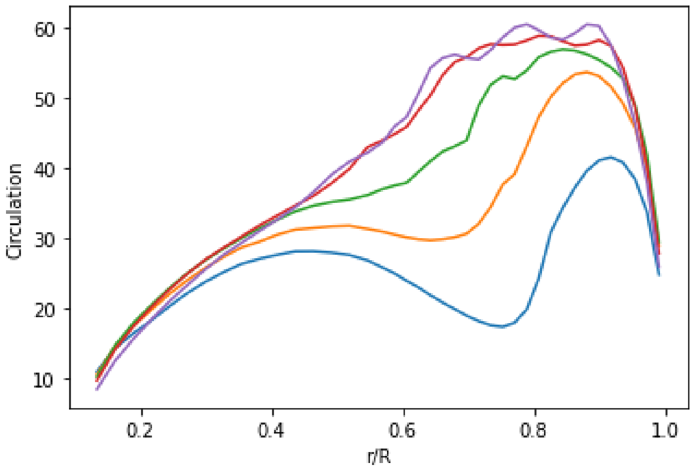
```
[ 0.   5. 10.]
Running case quad_v001_alpha05
Running case quad_v001_alpha10
```

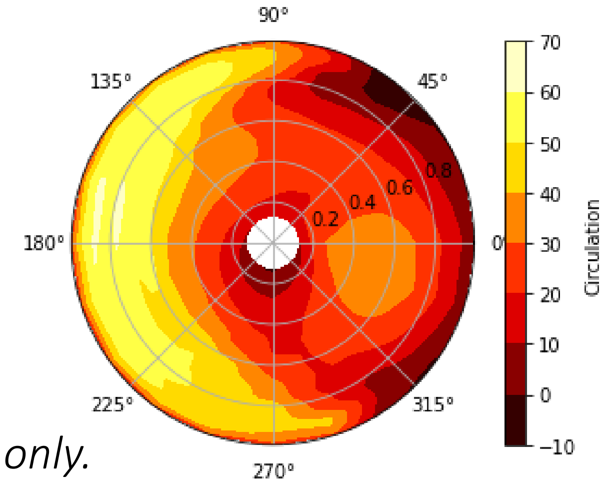
Visualizing Run Data

- Line and contour plots may be generated for any run case.
  - The cases may be from any CHARM run, not just the automation.
  - line_plot() and polar_plot() functions or create your own.
  - Multiple plots may be generated at once for comparison.

```python
# create line plot of performance data for rotor
# check hlp.inp for which surface is rotors N, here FR rotor = 1.
for i in range(len(results)):
    results[i].perf_data.line_plot(rotors=[1],
                                   revolutions=-1,
                                   variable='Circulation',
                                   npsi=[1,2,3,4,5],
                                   )
```

```python
# use cmap for colormap option
# use vmin and vmax to set contour limits
for i in range(len(results)):
    results[i].perf_data.polar_plot(rotor=1,
                                    revolution=-1,
                                    variable='Circulation',
                                    cmap='hot',
                                    #vmin = -20,
                                    #vmax = 60,
                                   )
```





For demonstration only.

Line and contour plots follow matplotlib kwargs.

Visualizing Run Data

- Rotor data shown is one example of the available data from a CHARM run.
- Some rotor results exist as Pandas DataFrame "row_data" that may be queried for post-processing and visualization. Over 20 variables available for comparison in this data alone.
- Refer to the CHARM Automation User Guide (HTML) documentation on Output Workflow.
  - …/charm/doc/html/userguide.html#output-workflow

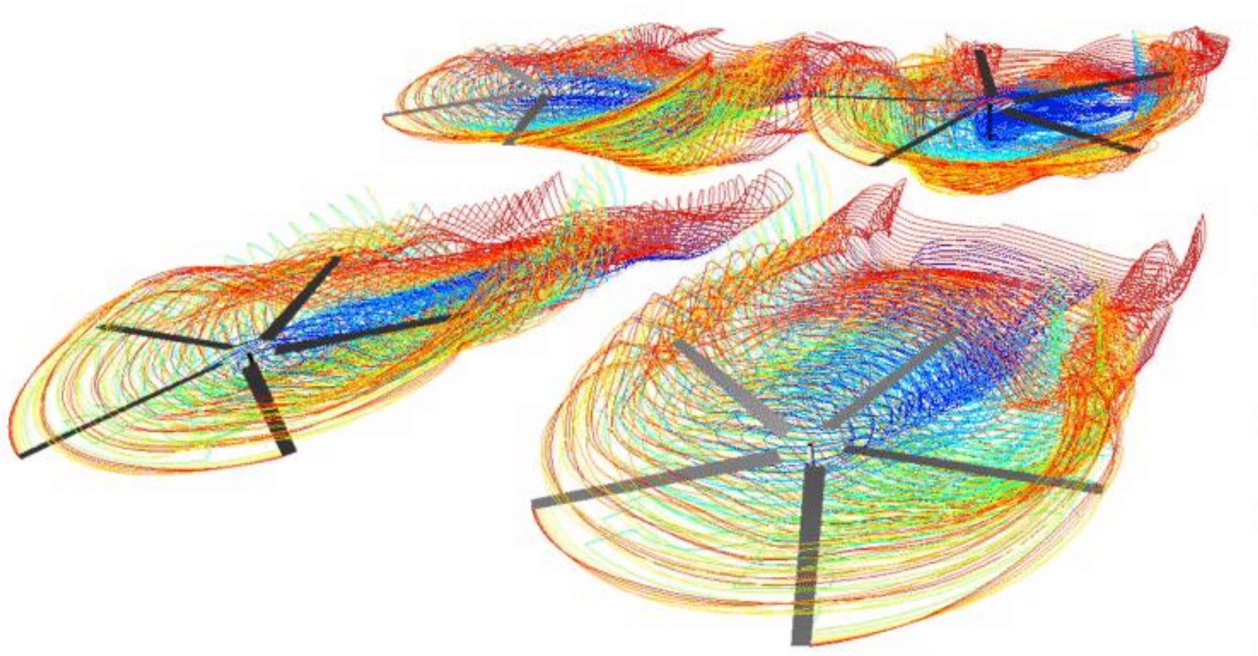
Row Data Variables

rotor	revolution	psi_ind	psi	x=r/R	dx	dCT/dx	dCQI/dx	dCQP/dx	X-force
CL2D	CD2D	CM2D	AOA2D	MACH2D	U-radial	V-aft	W-down	W-induced	Circulation

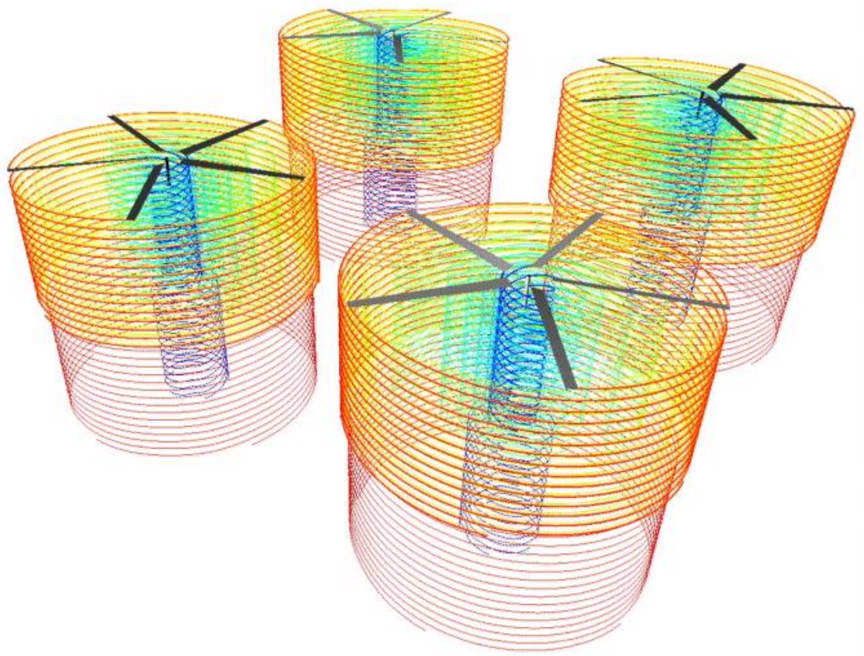
Visualizing Run Data

- Vortex-X Visualization Code
  - Generate *.sgp and *.sgp.graphics visualization files with “runv6p <path> <casename>”

50 ft/s Forward Speed Case w/ Free Wakes



Hover Case w/ Fixed Wakes



Thank you!

Questions?

Demonstration to follow…

Brandon Litherland, AST
NASA Langley Research Center
Aeronautics Systems Analysis Branch