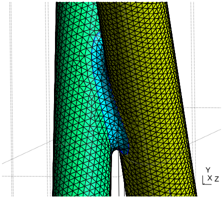
Uncertainty Quantification with Parsl in Composite Material Modeling

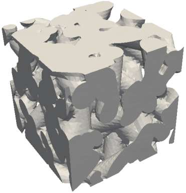
Kunkun Tang

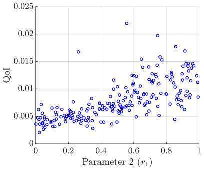
The Center for Exascale-enabled Scramjet Design (CEESD)

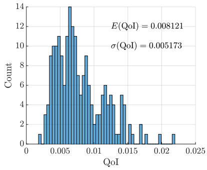
National Center for Supercomputing Applications (NCSA)

University of Illinois at Urbana–Champaign



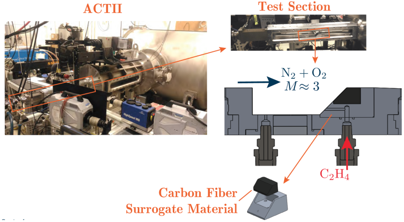


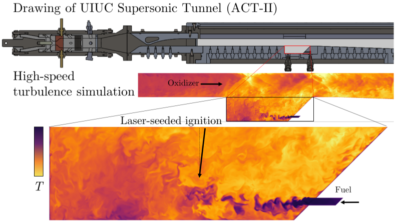




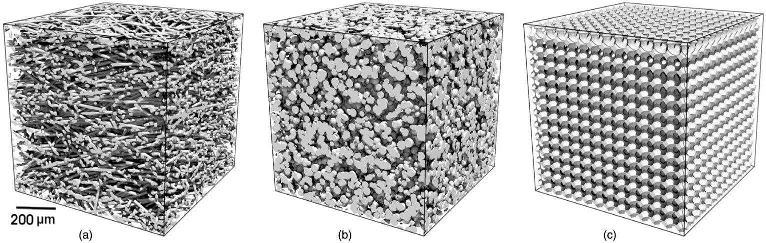
Some of CEESD Objectives

- New PSAAPIII Center (`http://ceesd.illinois.edu`) targeting scramjet design
- Establish predictive confidence using UQ-based integration of multi-scale/multi-physics models and exploiting HPC to resolve scales
- Advance a physics-based prediction capability for novel carbon-composites that will advance scramjet propulsion
- Initial studies focus on carbon-fiber microstructure, a common key feature of high-T composites
- Parsl will be employed to manage Workflow and provide Provenance





Uncertainties in Advanced Composite Materials



Joseph C. Ferguson, Francesco Panerai, Arnaud Borner, Nagi N. Mansour, "PuMA: the Porous Microstructure Analysis software", *SoftwareX* **7**, 81–87 (2018).

- Candidate models for material microstructure
  - Chemical composition
  - Random fibers — Simple cylinders, more complex geometry models
- Candidate models for material properties
  - Bilinear, Mises, Cohesive, Gurson, ...
  - Temperature-dependent, Strain rate-dependent
- Candidate models for crack models
  - Element death, Interface-Cohesive Elements, ...

PuMA $\longrightarrow$ Gmsh $\longrightarrow$ WARP3D Workflow

- **PuMA** (Porous Microstructure Analysis, `https://gitlab.com/jcfergus/PuMA_V3`) has been developed to compute effective material properties and perform material response simulations on digitized microstructures of porous media. [Ferguson et al. 2018]
- **Gmsh** (`https://gmsh.info`) is a three-dimensional finite element mesh generator. We use it to make STL $\longrightarrow$ Nastran/(Patran) conversion by creating a volume mesh (tet4).
- **WARP3D** (`http://www.warp3d.net`) is an open source code for 3D nonlinear finite element analysis of solids (static/dynamic).

PuMA & Surface Mesh Generation

- **INPUT** – test_projCEESD.cpp
  - This source cpp file contains all "hard-coded" parameters (for now), e.g. domain size, fiber diameter, etc.
- **OUTPUT** – RandomFibers_straightCircle.stl
  - A single STL surface mesh file

INPUT

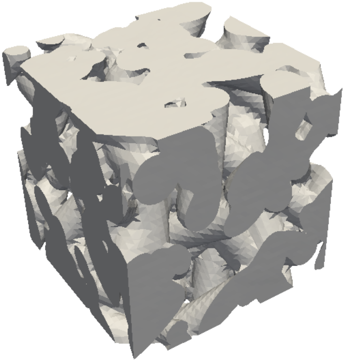
- test_projCEESD.cpp
  - "input.straightCircle(40, 40, 40, 3, 1, 40, 0, 90, 90, 60, true, 0.35, 999);"
  - Domain size
  - Fiber diameter + uncertainty
  - Fiber length + uncertainty
  - Orientation
  - Porosity

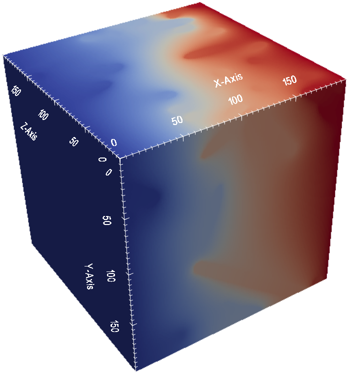
PuMA

OUTPUT

- STL surface mesh: RandomFibers_straightCircle.stl
  - "puma::export_STL"
  - An STL file describes a raw, unstructured **triangulated** surface by the unit normal and vertices (ordered by the right-hand rule) of the triangles using a three-dimensional Cartesian coordinate system.

Example: A Random Fiber Configuration (PuMA)





- QoI: Effective thermal conductivity (k_{xx}, k_{yy}, k_{zz})
- Goal: How microstructure properties and uncertainties affect QoI

Uncertain Parameters — Random-Fiber Model

	Symbol	Value	Uncertainty/Status	Description
Material topology [K. Tang et al.]	d_{fiber}	4	$\mathcal{U}(3, 5)$	Fiber diameter
	Δd_{fiber}	1	$\mathcal{U}(0, 2)$	Fiber diameter variation
	l_{fiber}	40	$\mathcal{U}(30, 50)$	Fiber length
	Δl_{fiber}	5	$\mathcal{U}(0, 10)$	Fiber length variation
	θ_x	45	$\mathcal{U}(0, 90)$	Orientation wrt x-normal plane
	θ_y	45	$\mathcal{U}(0, 90)$	Orientation wrt y-normal plane
	θ_z	15	$\mathcal{U}(0, 30)$	Orientation wrt z-normal plane
	ϕ	0.75	$\mathcal{U}(0.65, 0.85)$	Porosity
Material properties [K. Tang et al.]	k_a	12	$\mathcal{U}(10, 500)$	Fiber conductivity (axial)
	k_r	1.25	$\mathcal{U}(1, 50)$	Fiber conductivity (radial)
	k_{air}	0.0257	$\mathcal{U}(0.009, 5)$	Air conductivity (isotropic)

- 11 uncertain parameters typically require hundreds–thousands of simulations

Workflow by Parsl + Jupyter Notebook

- The starting point workflow is simple: Conductivity simulation + UQ analysis

jupyter puma_gsa Last Checkpoint: 09/21/2020 (autosaved)

File Edit View Insert Cell Kernel Help — Trusted — Python 3

In [2]:
```python
local_quartz = Config(
    executors=[
        HighThroughputExecutor(
            label="quartz_Local",
            worker_debug=True,
            address=address_by_hostname(),
            max_workers=1,
            cores_per_worker=1,
            provider=LocalProvider(
                channel=LocalChannel(),
                init_blocks=1,
                max_blocks=1,
            ),
        )
    ],
    monitoring=MonitoringHub(
        hub_address=address_by_hostname(),
        hub_port=6553,
        resource_monitoring_interval=1,
    )
)

parsl.load(local_quartz)
```

Out[2]:
```
<parsl.dataflow.dflow.DataFlowKernel at 0x7fdf311a6978>
```

In [3]:
```python
@bash_app #(cache = True)
def puma_sim(outputs=[], stdout="output/puma_sim.out", stderr="output/puma_sim.err"):
# def puma_sim(outputs=[]):
    """Call simulate and return results in the output file"""
    return 'cd {0}; ./run_puma.sh &> screen.log; cd -; echo ">>>>>>>>> {0} DONE"'.format(outputs[0].filepath)
```

In [4]:
```python
# loop to execute the simulation app 4 times
results = []
for i in range(3,7):
    out_file = "job_{0}/line1".format(i)
    results.append(puma_sim(outputs=[File(out_file)]))
```

Global Sensitivities – Ranking of Importances

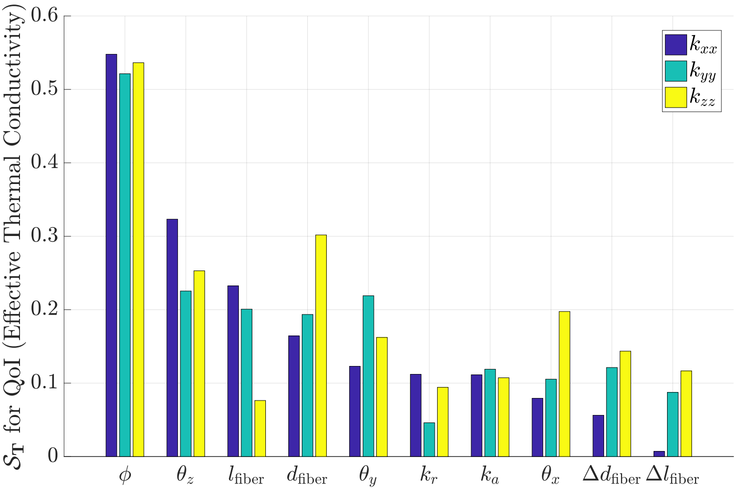
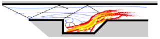
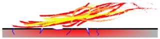
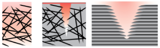
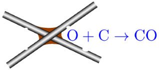
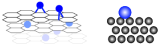


Figure: Parameters sorted in descending order wrt to values of k_{xx}.

Multi-scale/Multi-physics

	Physics/Scale	Code(s)	Essential Physics	Anticipated Physics	Potential Physics
Full	 ~ m	*MIRGE-Com* {*Nek5000-DG*} *Cantera* *Prometheus*	Turbulent mixing Shocks Combustion Complex geom.	Radiation Flexible wall Wall texture Wall transpiration	Particle trajectories Radicals
Needs: Wall conditions T, (maybe Y_i, geom.); **Provides:** Gas T, Y_i, (maybe $\boldsymbol{\sigma}$)					
Macro	 ~ m×cm	*WARP3D* {*RAPtor*}	Thermal conductivity	Fracture Fragmentation Recession Elastic response	Vibration
Needs: Local mechanical degradation, local Y_{O}, traction separation prms.; **Provides:** Cracking, regression, failure.					
Meso	 ~ mm	*PuMA* {*Cedar*}	Oxidation Transport	Micro-cracking Recession Detailed porous transport Porous material radiation	Sublimation Evaporation Wetting
Provides: Thermal conductivity, convective transport, local concentrations, microstructure geometry.					
Micro	 ~ μm	*SPARTA* *WARP3D* {*RAPtor*}	Surface kinetics	Stress-coupled reaction De-bonding	Grain-scale pitting
Provides: Local surface chemical kinetics.					
Nano	 ~ nm	*LAMMPS*		Solid-state diff. Traction-separation Phonon–kinetic models	Quantum (DFT) potentials
Provides: O diffusion, O-dependent traction separation.					

Plans and Challenges

- We plan to use Parsl to manage **complex workflow** involving UQ-based integration of multi-scale/multi-physics models
- **Primary Goals**
  - Coupled material modeling codes including mesh generation capability: PuMA, WARP3D, etc.
  - End-to-end UQ analysis: Sampling, surrogate polynomial approximation, sensitivity analysis, parameter estimation.
- **Challenge** encountered
  - Multi-cores performance issues have been observed in the coupling between Parsl and certain application code (e.g., PuMA to compute porous microstructure properties); reported on GitLab.

This material is based in part upon work supported by the Department of Energy, National Nuclear Security Administration, under Award Number DE-NA0003963.

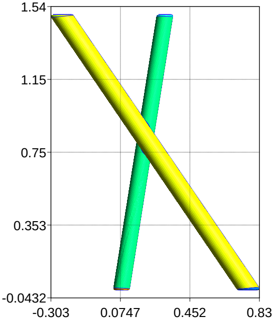
Acknowledgements

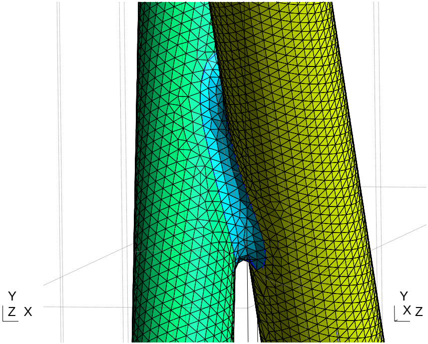
- Daniel Katz
- Kyle Chard
- Kelly Stephani
- Francesco Panerai
- Joseph Ferguson
- Harley Johnson
- Marco Panesi
- Jonathan Freund

Example 2:

Gmsh + WARP3D

Example: A Cross-Fiber Model Configuration





- **Gmsh** + **WARP3D**
- Only mechanical loading applied
- Bottom ends fixed
- Space and time (load steps) distributed force on top ends $(+x, -y)$
- This crack model (element death) requires a fracture threshold of the plastic strain value
- Currently learning to use interface-cohesive elements model

Limiting Cases for Parameters $\tan\theta_1$ & d_2

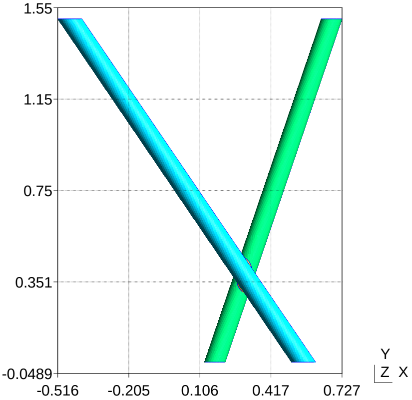


Figure: $\tan\theta_{1,\max}$ & $d_{2,\min}$

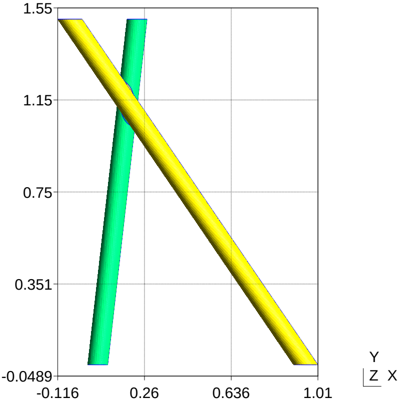


Figure: $\tan\theta_{1,\min}$ & $d_{2,\max}$

10 Uncertain Parameters — Cross-Fiber Model

	Symbol	Value	Uncertainty/Status	Description
Material topology [K. Tang et al.]	$\tan\theta_1$	$\Delta x_1/\Delta y_1$	$\mathcal{U}(\frac{0.25}{2.2}, \frac{0.75}{2.2})$	Fiber1 orientation angle
	r_1	0.05	$\mathcal{U}(0.04, 0.06)$	Fiber1 radius
	d_2	$d((x_1,0),(x_2,0))$	$\mathcal{U}(0.90, 1.30)$	Distance between center bases
	r_2	0.05	$\mathcal{U}(0.04, 0.06)$	Fiber2 radius
Material properties (Fillet) [K. Tang et al.]	E	10000	$\mathcal{U}(5000, 25000)$	Young's modulus
	ν	0.3	$\mathcal{U}(0.20, 0.50)$	Poisson's ratio
	(ρ)	0.00		Mass density
	(α)	0.0001		Thermal expansion coefficient
	σ_y	30	$\mathcal{U}(10, 60)$	Yield stress
	(E_T)	100		Hardening modulus
	(Curve param.)	Multiple		Stress–plastic strain curve
	(k)	$\{12, 1.2\}$		Thermal conductivity
Material properties (Fiber) [K. Tang et al.]	E	30000	$\mathcal{U}(25000, 35000)$	Young's modulus
	ν	0.3	$\mathcal{U}(0.20, 0.50)$	Poisson's ratio
	(ρ)	0.00		Mass density
	(α)	0.0001		Thermal expansion coefficient
	σ_y	70	$\mathcal{U}(60, 90)$	Yield stress
	(E_T)	100		Hardening modulus
	(Curve param.)	Multiple		Stress–plastic strain curve
	(k)	$\{12, 1.2\}$		Thermal conductivity

- **QoI: Fracture toughness (K_c)**

Input–Output (K_c) Sampling Data Examples

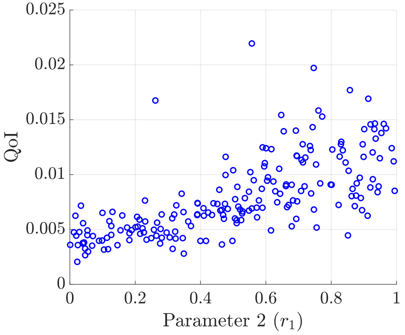


Figure: **Important** param. r_1

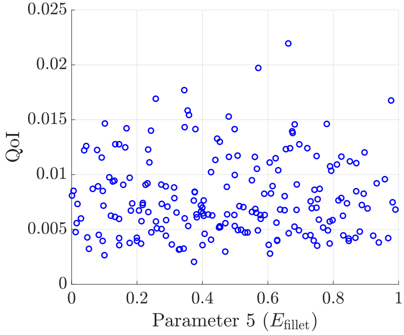
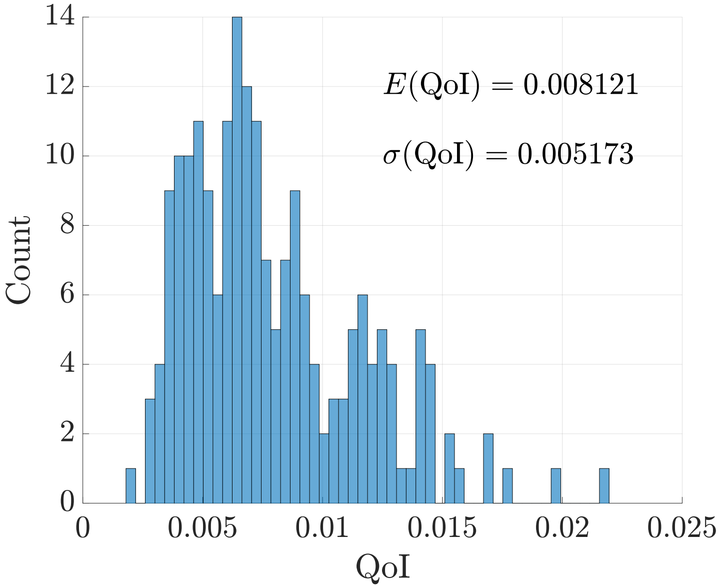
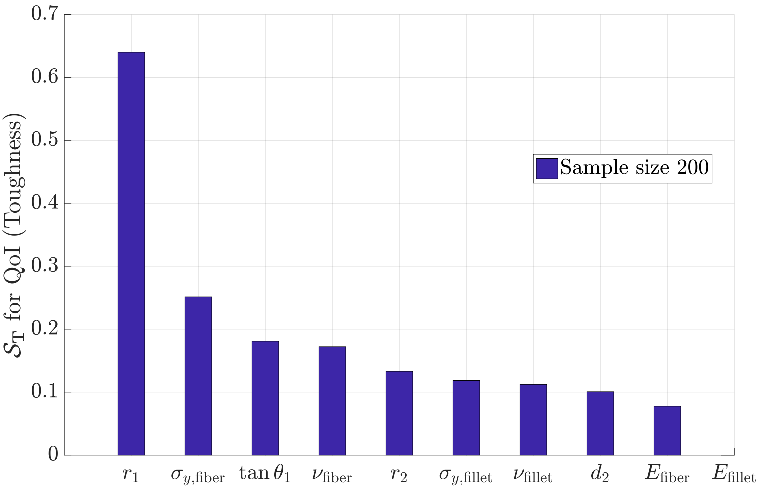


Figure: **Unimportant** param. E_{fillet}

Distribution and Moments of QoI (K_c)



Global Sensitivities – Ranking of Importances



PuMA $\longrightarrow$ Pointwise $\longrightarrow$ WARP3D Workflow

PuMA

- PuMA (Porous Microstructure Analysis) has been developed to compute effective material properties and perform material response simulations on digitized microstructures of porous media. [Ferguson et al. 2018]
- NASA software under a US & Foreign release
- Free research code: https://gitlab.com/jcfergus/PuMA_V3 (**Access needs to be granted**)
- UNIX operating systems required
- Typically runs remotely on NNSA clusters
- GCC version 4.4.7 or later
- Default Installation Option: Full installation of PuMA C++ Library
- Recommended Computer Specifications: 8gb of ram for small simulations (600^3 or smaller) 16-32gb of ram for medium simulations (800^3 range) 32+gb of ram for large simulations (above 1000^3)
- Simulation time varies
  - It takes ~ secs - mins to generate a medium-sized random fiber structure (single processor)
  - Anticipate much longer time for large simulations

PuMA & surface mesh generation

- **INPUT** - `test_projCEESD.cpp`
  - This source cpp file contains all "hard-coded" parameters (for now), e.g. domain size, fiber diameter, etc.
- **OUTPUT** - `RandomFibers_straightCircle.stl`
  - A single STL surface mesh file

INPUT

- test_projCEESD.cpp
  - "input.straightCircle (40, 40, 40, 3, 1, 40, 0, 90, 90, 60, true, 0.35, 999);"
  - Domain size
  - Fiber diameter + uncertainty
  - Fiber length + uncertainty
  - Orientation
  - Porosity

PuMA

OUTPUT

- STL surface mesh: RandomFibers_straightCircle.stl
  - "puma::export_STL"
  - An STL file describes a raw, unstructured **triangulated** surface by the unit normal and vertices (ordered by the right-hand rule) of the triangles using a three-dimensional Cartesian coordinate system.

Pointwise

- Mesh generation software
- Commercial: https://www.pointwise.com
- We use it to make STL->Patran conversion by creating a volume mesh (tet4)
- Available for Windows, Linux, and Mac
- Typically runs locally on laptops using GUI
- Scriptable?
- Processing time can be long
  - It takes me ~ an hour to just import an STL file with 3—4M nodes
- Gmsh is faster (open source and scriptable). However, does not support Patran as export format
  - (I used Gmsh to create/optimize a tet4 volume mesh and export it in a Nastran format, then used Pointwise to convert from Nastran to Patran)
- We are currently looking for (and will ultimately need) a scriptable meshing tool that can be used remotely as part of automated UQ analyses on clusters

Pointwise & volume mesh generation

- **INPUT** - `RandomFibers_straightCircle.stl`
  - A single STL surface mesh file
- **OUTPUT** - `RandomFibers_straightCircle.pat`
  - A single Patran volume mesh file (ASCII)

INPUT

- `STL surface mesh: RandomFibers_straightCircle.stl`

Pointwise/ Gmsh

OUTPUT

- `Patran volume mesh: RandomFibers_straightCircle.pat (tet4)`

WARP3D

- 3D Nonlinear Finite Element Analysis of Solids (Static/Dynamic)
- Open source code: https://github.com/rhdodds/warp3d
- Available for Windows, Linux, and Mac
- Typically runs remotely on NNSA clusters
  - To recompile the threads-only version from source, it requires
    - Intel Fortran 18.0.# OR Intel Fortran 19.0.2 (or newer),** OR **
    - GNU gfortran 7.3 (or newer)
  - To recompile the MPI + threads (hybrid) version, it requires
    - Intel Fortran 19.0.3 (or newer) ** AND ** the (free) Intel MPI 19.0.3 (or newer)
- Simulation time varies
  - It takes mins to simulate 1k loading steps for a 145477-element problem (single processor)
  - Anticipate much longer time for large simulations
- WARP3D contains
  - Patran neutral file-to-WARP3D translator program (pre-processing via “patwarp.go”, only works with Intel compilers)
  - WARP3D results-to-ParaView program (post-processing via “python warp3d2exii”)

WARP3D pre-processing

- **INPUT** - `RandomFibers_straightCircle.pat`
  - A single ASCII Patran volume mesh file (**limited to 4M nodes**)
- **OUTPUT** - All necessary ASCII input files of a WARP3D simulation
  - warp3d_input – Main input file (we will need to specify material definitions, loading patterns, finite element analysis parameters, etc.)
  - coords.inp – Mesh node ID & coordinates
  - incid.inp – Mesh element ID & related nodes ID
  - constaints.inp – Boundary conditions, e.g. fixed nodes, fixed planes

INPUT

- Patran volume mesh: RandomFibers_straightCircle.pat (tet4)
  - **Limited to 4M nodes**

WARP3D - patwarp.go

OUTPUT

- warp3d_input
  - *Add material*
  - *Add loading etc.*
- coords.inp
  - Node coordinates
- incid.inp
  - Element–NodeID
- constraints.inp
  - *Add boundary conditions*

WARP3D static/dynamic finite element analysis

INPUT

- warp3d_input
  - Material properties
  - Stress-strain curve
  - Element type (tet, hex, linear/nonlinear)
  - Element integration order
  - Initial conditions (T, stress, etc.)
  - Loading pattern (force, T, constraints, etc.)
  - crack growth parameters
  - nonlinear analysis parameters
- coords.inp
  - Node coordinates
- incid.inp
  - Element-NodeID
- constraints.inp
  - Boundary conditions

WARP3D

OUTPUT

- Series of stream files
  - wee0000100_stream (strain @ element)
  - wes0000100_stream (stress @ element)
  - wnd0000100_stream (displacement @ node)
  - wnt0000100_stream (temp @ node)
- RandomFibers_straightCircle.text (flat text file – mesh)

WARP3D post-processing

INPUT

- Series of stream files
  - wee0000100_stream (strain @ element)
  - wes0000100_stream (stress @ element)
  - wnd0000100_stream (displacement @ node)
  - wnt0000100_stream (temp @ node)
- RandomFibers_straightCircle.text (flat text file – mesh)

Python warp3d2exii

OUTPUT

- RandomFibers_straightCircle.exo (Exodus format)