Additional Abaqus information

in silico Plants
OXFORD

A conserved cellular mechanism for cotton fiber diameter and length control

https://doi.org/10.1093/insilicoplants/diac004

Yanagisawa, M.[1,4,*], Keynia, S.[3,*], Belteton, S.[2,5], Turner, J. A.[3], Szymanski, D. B.[1,2*]

[1]Dept. of Botany and Plant Pathology, [2]Dept. of Biological Sciences, Purdue University, West Lafayette, Indiana, U.S.A.
[3]Mechanical and Materials Engineering, University of Nebraska-Lincoln, Lincoln, Nebraska, U.S.A
[4]Present address: Division of Cell Fate Dynamics and Therapeutics, Department of Biosystems Science, Institute for Frontier Life and Medical Sciences, Kyoto University, Kyoto, Japan
[5]Microscopy Imaging Core Suite, New Mexico State University, Las Cruces, New Mexico, U.S.A

[*]These authors contributed equally to this work.

in silico Plants

Sedighe Keynia, a PhD student at UNL created the Abaqus files

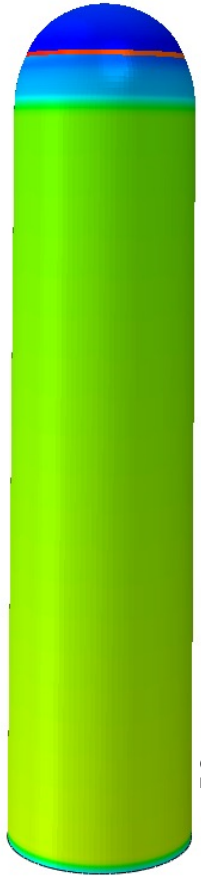
Abaqus 2019 was used for the FE analysis.

The Abaqus CAE files includes one of the cycles of the growth model.

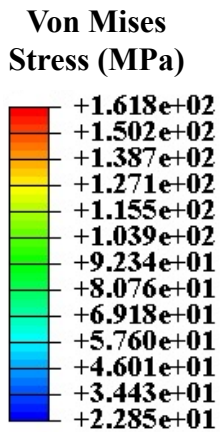
For questions, please contact J. Turner (jaturner@unl.edu)

Mechanical properties in FEM

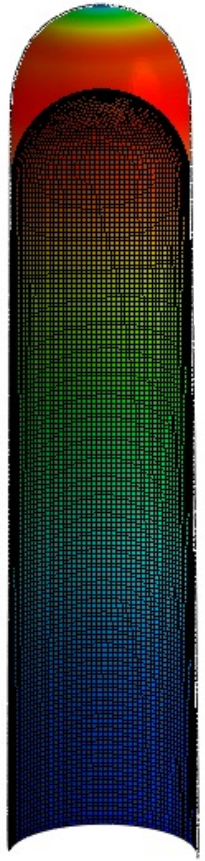
In all simulations:
Turgor pressure: 0.5 MPa
E1 (Along the fibers)=70GPa, E2 (perpendicular to the fibers)=E-matrix=100MPa, E isotropic zone=400MPa
G=45 Mpa, ν=0.45
Relaxation time (τ_i)= 6.88s, g_i= 0.125



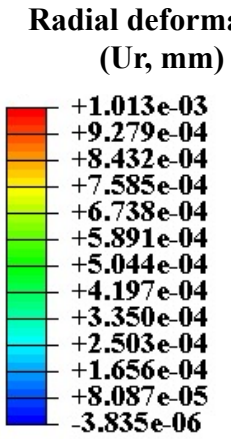
Length =70 µm
Transverse fiber orientation
Radius of curvature=7 µm



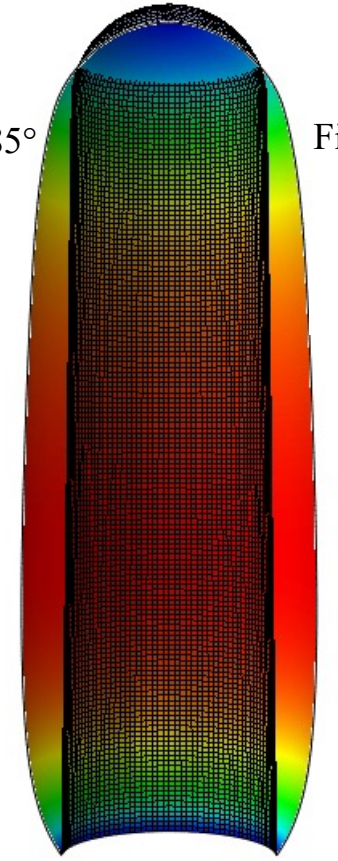
Simulation results
(Fig. 9A)



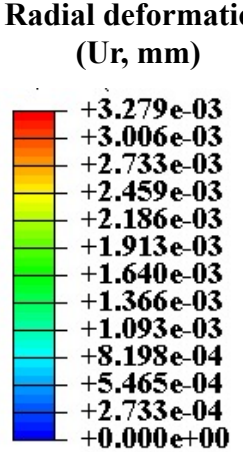
Length =75 µm
Fiber orientation angle=85°



Simulation results
(Fig. 10A)

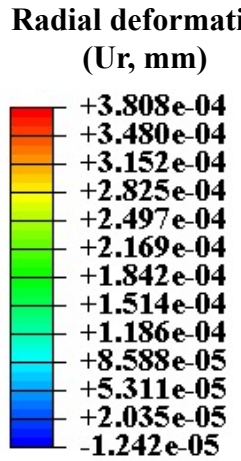


Length =75 um
Fiber orientation angle=10 °



Simulation results
(Fig. 10A)

Length =164 um
Transverse fiber orientation



Simulation results
(Fig. 10C)

Partial Python script code

```
# create a job
    mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF,
        explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
        memory=90, memoryUnits=PERCENTAGE, model=
        'cotton_'+str(cycle), modelPrint=OFF,
        multiprocessingMode=DEFAULT, name=jobname, nodalOutputPrecision=SINGLE,
        numCpus=5, numDomains=5, numGPUs=0, queue=None, resultsFormat=ODB, scratch=
        '', type=ANALYSIS, userSubroutine='', waitHours=0, waitMinutes=0)
# submit a job and wait until completion
    mdb.jobs[jobname].submit(consistencyChecking=OFF)
    mdb.jobs[jobname].waitForCompletion()
## #-*- END runABAQUS() -*-

# read odb and return diplacement of the tip.
def extractDisp(jobname):
    pathname = jobname + '.odb'
    odb = openOdb(path=pathname)
    probe = odb.rootAssembly.instances['COTTON_'+str(cycle)+'-1'].nodeSets['SET-1']
    disp = odb.steps['Step-1'].frames
    dispmentValue_X = []
    dispmentValue_Y = []
# read disp
    for d in disp :
        UT1value = d.fieldOutputs['UT'].values[-1].data[0]
        dispmentValue.append(UT1value)
        UT2value = d.fieldOutputs['UT'].values[-1].data[1]
        dispmentValue_X.append(UT1value)
        dispmentValue_X.append(UT2value)


    dispX = np.array(dispmentValue_X)
    dispY = np.array(dispmentValue_Y)
    slope = np.polyfit(dispX, dispY,1)
    ROC = slope[0]
        |

    for d in disp :
        UT3value = d150.fieldOutputs['UT'].getSubset(region=probe).values[-1].data[2]
        dispment150Value.append(UT3value)
    dispX = np.array(dispmentValue_X)
    dispY = np.array(dispmentValue_Y)
    slope = np.polyfit(dispX, dispY,1)
```

After the first simulation is completed, the deformed coordinate of the tip region is extracted to calculate tip radius of curvature and compare with the measurements.