Finding the spring constant of a helical spring

Biswajit Banerjee, Jan 2012

1 Introduction

Is there an easy way of find the spring stiffness of a helical spring without having to use an accurate load and displacement measuring system? How close are the results that we can get from finite element calculations to closed form solutions? What if the deformations are large?

Let us assume that the spring can be described by

$$f = k\,u$$

where f is the applied force, u is the displacement, and k is the spring stiffness.

2 Closed form solution

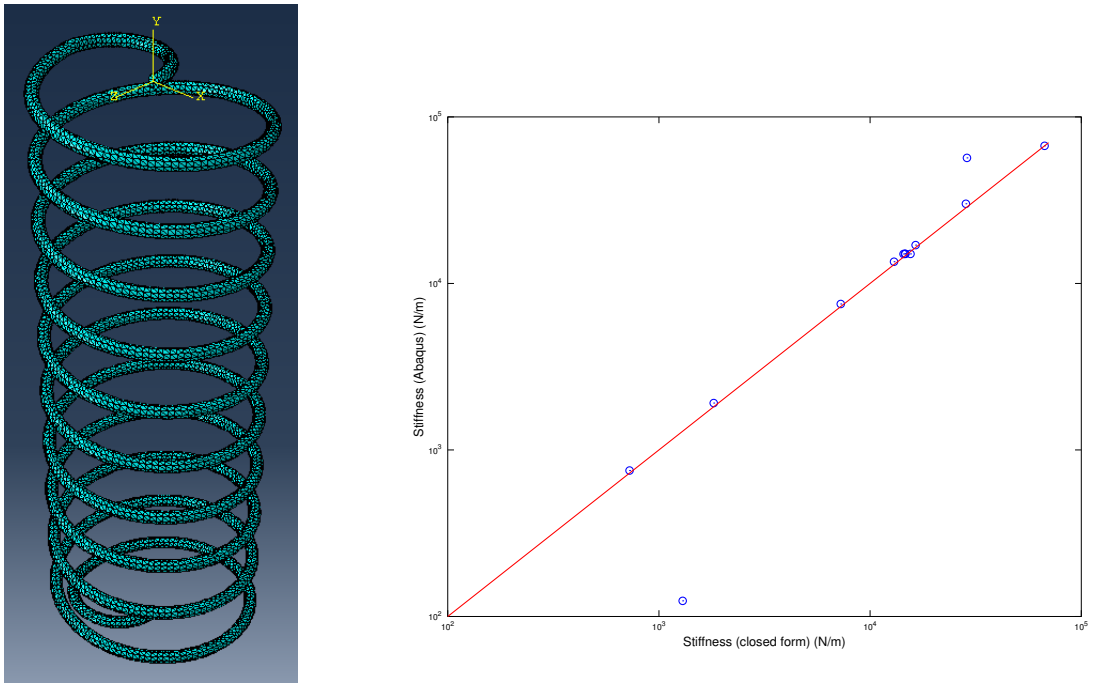
A closed form solution for the extensional (compressional) stiffness of a spring is [1]

$$\frac{1}{k} = \frac{n\pi D^3}{4\,G\,J\,\cos\alpha}\left[\left(1+\frac{d^2}{2D^2}\right)\cos^2\alpha + \left(1+\frac{d^2}{4D^2}\right)\frac{\sin^2\alpha}{(1+\nu)}\right]$$

where n is the number of loops, D is the diameter of the spring, d is the diameter of the wire, $G = 0.5\,E/(1+\nu)$ is the shear modulus of the spring material, E is the Young's modulus of the spring material, ν is the Poisson's ratio of the spring material, $J = \pi d^4/32$ is the polar moment of inertial of the cross-section of the wire, and the angle α is related to the pitch p of the spring by the relation $p = \pi\,D\,\tan\alpha$.

3 Finite element solutions

A parametrized Abaqus script that can be used to simulate the extension or compression of a spring is given in Appendix A. Quadratic triangles with reduced integration are used to mesh the spring as shown in Figure 1(a) . The displacement is computed for a few variations of load and geometry and the spring stiffness is computed. A comparison of the computed stiffness and the closed form solution in given in Figure 1(b). The input data used in



(a) Spring model. (b) Comparison of simulated and analytical solutions.

Figure 1 – *A finite element model of a helical spring and comparison of simulated and analytical solutions.*

the comparisons are given in Table 1. The derived quantities and results are given in Table 2.

Table 1 – *Input parameters for spring calculations.*

Test ID	d (mm)	D (mm)	p (mm)	n	E (GPa)	ν	f (N)
A	2	10	3	10	200	0.3	-100
B	2	10	3	10	200	0.3	100
C	2	10	3	10	200	0.15	-100
D	2	10	3	10	200	0.45	-100
E	2	10	3	10	10	0.3	-1
F	2	10	3	10	100	0.3	-1
G	2	10	3	5	200	0.3	-10
H	2	10	2.1	10	200	0.3	10
I	2	10	4	10	200	0.3	-10
J	2	6	3	10	200	0.3	-10
K	2	20	3	10	200	0.3	-10
L	0.6	10	3	10	200	0.3	-10
M	2.8	10	3	10	200	0.3	10

Table 2 – *Results and derived parameters for spring calculations.*

Test ID	G (GPa)	J (mm^4)	α (deg.)	u (mm)	k (N/m)	k_{Abaqus} (N/m)
A	76.923	1.5708	5.4548	-6.8050	1.4695e+04	1.5047e+04
B	76.923	1.5708	5.4548	6.4407	1.5526e+04	1.5047e+04
C	86.957	1.5708	5.4548	-6.0869	1.6429e+04	1.6994e+04
D	68.966	1.5708	5.4548	-7.7051	1.2978e+04	1.3500e+04
E	38.462	1.5708	5.4548	-1.3775	7.2597e+02	7.5235e+02
F	38.462	1.5708	5.4548	-1.3775	7.2597e+03	7.5235e+03
G	76.923	1.5708	5.4548	-0.35192	2.8416e+04	3.0094e+04
H	76.923	1.5708	3.8243	0.68392	1.4622e+04	1.5065e+04
I	76.923	1.5708	7.2561	-0.69453	1.4398e+04	1.5019e+04
J	76.923	1.5708	9.0431	-0.14920	6.7023e+04	6.7053e+04
K	76.923	1.5708	2.7336	-5.5028	1.8173e+03	1.9123e+03
L	76.923	0.0127	5.4548	-7.7173	1.2958e+03	1.2409e+02
M	76.923	6.0344	5.4548	0.34779	2.8753e+04	5.6740e+04

4 Discussion

The finite element solutions are close to the analytical solutions except for a couple of cases. The first discrepancy involves a soft spring where nonlinear effects become important. The second shows the difference between tension and compression when a relatively large load is applied. Care must be exercised when using the closed form solution.

References

[1] C. L. Dym. Consistent derivations of spring rates for helical springs. *ASME Journal of Mechanical Design*, 131:071004, 2009.

A Abaqus script

```
from abaqus import *
from abaqusConstants import *
from mesh import *
import regionToolset

# Geometry constants
referenceLengthScale = 1.0e-3                       # m
referenceLengthScale = referenceLengthScale*1.0e3   # mm
wireRad = referenceLengthScale
springRad = 5*referenceLengthScale
pitch = 3*referenceLengthScale
```

```
noOfLoops = 10
angle = 360*noOfLoops
extAngle = 180

# Material constants
# Ex = 200e9                                        # N/m^2
Ex = 200e9*1.0e-6                                   # N/mm^2
nuxy = 0.3

# Traction (Pa) - load (N)
load = -10
area = pi*referenceLengthScale*referenceLengthScale
traction = load/area

# Element size
referenceLengthScale = wireRad
#elSize = 0.5*referenceLengthScale
elSize = referenceLengthScale

# Computed variables
inRad = springRad
outRad = inRad+referenceLengthScale
height = pitch*noOfLoops
sheet = 5*height

#--------------------------------------------------------------------
# Create model
#--------------------------------------------------------------------
tmpModel = mdb.Model(name='tmp')
if mdb.models.has_key('Model-1'):
  del mdb.models['Model-1']
if mdb.models.has_key('SpringModel'):
  del mdb.models['SpringModel']

springModel = mdb.Model(name='SpringModel')
springAssem = springModel.rootAssembly
del mdb.models['tmp']

#--------------------------------------------------------------------
# Create sketch of spring
#--------------------------------------------------------------------
springSketch = springModel.ConstrainedSketch(name='__profile__',
            sheetSize=sheet)
springGeom = springSketch.geometry
springVert = springSketch.vertices
springDim = springSketch.dimensions
springConstraints = springSketch.constraints
springSketch.setPrimaryObject(option=STANDALONE)
springSketch.ConstructionLine(point1=(0.0, -0.5*sheet), point2=(0.0, 0.5*sheet))
springSketch.FixedConstraint(entity=springGeom[2])

centerPt = (-inRad, 0.0)
perimPt = (-outRad, 0.0)
springSketch.CircleByCenterPerimeter(center=centerPt, point1=perimPt)

#--------------------------------------------------------------------
# Create part from sketch of spring
#--------------------------------------------------------------------
springPart = springModel.Part(dimensionality=THREE_D, name='Spring', type=DEFORMABLE_BODY)
springPart.BaseSolidRevolve(sketch=springSketch, angle=angle, flipRevolveDirection=OFF, pitch=pitch,
flipPitchDirection=OFF, moveSketchNormalToPath=OFF)
springSketch.unsetPrimaryObject()
del springSketch

#--------------------------------------------------------------------
# Create sketch of extension of spring
#--------------------------------------------------------------------
extSketch = springModel.ConstrainedSketch(name='__profile__',
            sheetSize=sheet)
extGeom = extSketch.geometry
extVert = extSketch.vertices
extDim = extSketch.dimensions
extConstraints = extSketch.constraints
extSketch.setPrimaryObject(option=STANDALONE)
```

```
extSketch.ConstructionLine(point1=(-0.5*inRad, -0.5*sheet), point2=(-0.5*inRad, 0.5*sheet))
extSketch.FixedConstraint(entity=extGeom[2])

centerPt = (-inRad, 0.0)
perimPt = (-outRad, 0.0)
extSketch.CircleByCenterPerimeter(center=centerPt, point1=perimPt)

#--------------------------------------------------------------
# Create part from sketch of extension of spring
#--------------------------------------------------------------
extPart = springModel.Part(dimensionality=THREE_D, name='Extension', type=DEFORMABLE_BODY)
extPart.BaseSolidRevolve(sketch=extSketch, angle=extAngle, flipRevolveDirection=ON, pitch=0,
flipPitchDirection=OFF, moveSketchNormalToPath=OFF)
extSketch.unsetPrimaryObject()
del extSketch

#--------------------------------------------------------------
# Create datum planes for the spring extension
#--------------------------------------------------------------
# theta = (180 - extAngle)*pi/180
# xx = -inRad*0.5*(1-cos(theta))
# zz = -inRad*0.5*sin(theta)
# locRad = sqrt(xx*xx+zz*zz)
# extPart.DatumPlaneByThreePoints(point1=(0,0,0), point2=(0,1,0), point3=(xx,0,zz))

#--------------------------------------------------------------
# Create vertical part of the spring extension
#--------------------------------------------------------------
# faces, edges, datums = extPart.faces, extPart.edges, extPart.datums
# transform = extPart.MakeSketchTransform(sketchPlane=datums[2], sketchUpEdge=edges[0],
#  sketchPlaneSide=SIDE1, sketchOrientation=RIGHT,
#  origin=(0, 0, 0))
# sketch = springModel.ConstrainedSketch(name='__sweep__',
#  sheetSize=sheet, gridSpacing = 0.0005, transform=transform)
# geom = sketch.geometry
# vert = sketch.vertices
# dim = sketch.dimensions
# constraints = sketch.constraints
# sketch.setPrimaryObject(option=SUPERIMPOSE)
# extPart.projectReferencesOntoSketch(sketch=sketch, filter=COPLANAR_EDGES)
# sketch.Line(point1=(0,inRad), point2=(0,0))
# sketch.Line(point1=(0,0), point2=(locRad,0))
# sketch.unsetPrimaryObject()
# extPart.SolidSweep(pathPlane=datums[2], pathUpEdge=edges[0], profile=faces[1],
#  pathOrientation=RIGHT, path=sketch)
# del sketch

#--------------------------------------------------------------
# Create material
#--------------------------------------------------------------
springMat = springModel.Material(name='Steel')
springMat.Elastic(table=((Ex, nuxy), ))

#--------------------------------------------------------------
# Create section
#--------------------------------------------------------------
springSec = springModel.HomogeneousSolidSection(name='Section-1',
    material='Steel', thickness=None)

#--------------------------------------------------------------
# Assign section
#--------------------------------------------------------------
cellList = springPart.cells
cells = cellList.getSequenceFromMask(mask=('[#1 ]', ), )
region = regionToolset.Region(cells=cells)
springPart.SectionAssignment(region=region, sectionName='Section-1', offset=0.0,
    offsetType=MIDDLE_SURFACE, offsetField='',
    thicknessAssignment=FROM_SECTION)
cellList = extPart.cells
cells = cellList.getSequenceFromMask(mask=('[#1 ]', ), )
region = regionToolset.Region(cells=cells)
extPart.SectionAssignment(region=region, sectionName='Section-1', offset=0.0,
    offsetType=MIDDLE_SURFACE, offsetField='',
    thicknessAssignment=FROM_SECTION)
```

```
#-----------------------------------------------------------------
# Create assembly
#-----------------------------------------------------------------
springAssem.DatumCsysByDefault(CARTESIAN)
springInst = springAssem.Instance(name='Spring-1', part=springPart, dependent=ON)
extInst1 = springAssem.Instance(name='Ext-1', part=extPart, dependent=ON)
extInst2 = springAssem.Instance(name='Ext-2', part=extPart, dependent=ON)
springAssem.translate(instanceList=('Ext-2',), vector=(0.0, -height, 0.0))
springAssem.rotate(instanceList=('Ext-2',), axisPoint=(-0.5*inRad, 0.0, 0.0),
axisDirection=(0.0, -height, 0.0), angle=180)
springAssem.InstanceFromBooleanMerge(name='SpringPart',
instances=(extInst1, springInst, extInst2, ),
originalInstances=DELETE, domain=GEOMETRY)
springInst = springAssem.instances['SpringPart-1']
springAssem.makeIndependent(instances=(springInst,))
springPart = springModel.parts['SpringPart']

#-----------------------------------------------------------------
# Create mesh
#-----------------------------------------------------------------
#springPart.seedPart(size=0.5*referenceLengthScale, deviationFactor=0.1)
#springPart.generateMesh()
#extPart.seedPart(size=0.5*referenceLengthScale, deviationFactor=0.1)
#extPart.generateMesh()

cells = springInst.cells
pickedRegions = cells.getSequenceFromMask(mask=('[#1 ]', ), )
pickedSet = springAssem.Set(cells=pickedRegions, name='Set-1')
elemType = ElemType(elemCode=C3D10M, elemLibrary=STANDARD,
distortionControl=DEFAULT)
springAssem.setElementType(regions=pickedSet, elemTypes=(elemType,))
springAssem.setMeshControls(regions=pickedRegions, elemShape=TET,
technique=FREE)
partInstances =(springInst, )
springAssem.seedPartInstance(regions=partInstances, size=elSize,
deviationFactor=0.1)
springAssem.generateMesh(regions=partInstances)

#-----------------------------------------------------------------
# Create step
#-----------------------------------------------------------------
springModel.StaticStep(name='Step-1', previous='Initial',
    maxNumInc=500, initialInc=1e-02, maxInc=1,
    stabilizationMagnitude=0.0002,
    stabilizationMethod=DISSIPATED_ENERGY_FRACTION,
    continueDampingFactors=False, adaptiveDampingRatio=0.05, nlgeom=ON)

#-----------------------------------------------------------------
# Create output request
#-----------------------------------------------------------------
springModel.FieldOutputRequest(name='F-Output-2',
    createStepName='Step-1', variables=PRESELECT)

#-----------------------------------------------------------------
# Apply BC
#-----------------------------------------------------------------
# Shear Load at the top
#
loadPoint1 = (0, 0, 0)
loadPoint2 = (0, 1, 0)
faces = springInst.faces
side1Face = faces.findAt((loadPoint1,))
region = regionToolset.Region(side1Faces=side1Face)
springModel.SurfaceTraction(name='Load',
createStepName='Step-1', region = region, magnitude=traction,
directionVector=(loadPoint1, loadPoint2),
        distributionType=UNIFORM, field='', localCsys=None, resultant=ON)

#
# Fixed displacement at the bottom
#
edges = springInst.edges
dispPoint = (0, -height, 0)
```

```
dispFace = faces.findAt((dispPoint,))
region = regionToolset.Region(faces=dispFace)
springModel.PinnedBC(name='DispBC', createStepName='Step-1', region=region)

#-------------------------------------------------------------------
# Create self contact interactions  (not tested)
#-------------------------------------------------------------------
#springModel.ContactProperty('IntProp-1')
#springModel.interactionProperties['IntProp-1'].NormalBehavior(
#    pressureOverclosure=HARD, allowSeparation=ON,
#    constraintEnforcementMethod=DEFAULT)
#s1 = springInst.faces
#side1Faces1 = s1.getSequenceFromMask(mask=('[#fffffffd #3ff ]', ), )
#region=regionToolset.Region(side1Faces=side1Faces1)
#springModel.SelfContactStd(name='Int-1', createStepName='Step-1',
#    surface=region, interactionProperty='IntProp-1', thickness=ON,
#    contactTracking=ONE_CONFIG)

#-------------------------------------------------------------------
# Create job
#-------------------------------------------------------------------
mdb.Job(name='SpringJob', model='SpringModel', description='', type=ANALYSIS,
    atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
    memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
    explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE, echoPrint=OFF,
    modelPrint=OFF, contactPrint=OFF, historyPrint=OFF, userSubroutine='',
    scratch='', multiprocessingMode=DEFAULT, numCpus=1)

#-------------------------------------------------------------------
# Submit job
#-------------------------------------------------------------------
mdb.jobs['SpringJob'].submit(consistencyChecking=OFF)
```