NCSX

Design Basis Analysis

EM Analysis of Modular Coil Leads

NCSX-CALC-14-011-00

8 Septmeber 2008

Prepared by:

David Williamson
2008.09.08 11:54:00 -04'00'

D. Williamson, ORNL

I have reviewed this calculation and, to my professional satisfaction, it is properly performed and correct. I concur with analysis methodology and inputs and with the reasonableness of the results and their interpretation.

Reviewed by:

Kevin Freudenberg

Digitally signed by Kevin Freudenberg
DN: cn=Kevin Freudenberg, c=US, o=Fusion Energy, ou=ORNL, email=fdk@ornl.gov
Date: 2008.09.08 11:58:29 -04'00'

K. Freudenberg, ORNL

EM and Structural Analysis of Leads

1. Executive summary

The purpose of this analysis is to determine the worst electromagnetic (EM) loads on the modular coil leads and evaluate them structurally. Previous analyses have looked at the seven reference operating scenarios and determined that the 2T scenario produces the largest modular coil forces [1,2]. This analysis assumes the same current scenario, but includes the coil leads geometry, modeled as filaments, in order to determine the maximum running load on the individual conductors. Results indicate that the peak load in the unsupported leads between the modular coil shell and the lug connection is 34-lb/in, which produces a stress of 30-ksi in the conductor. This stress level is lower than the allowable of 35-ksi and does not limit the performance of the coils.

2. Assumptions

The geometry of the modular coil leads is shown in Fig.-1. In this calculation, the coil geometry is represented by linear, line-type finite elements (Fig.-2) with a characteristic length of ~3-in. In the leads area, the discretization is ~6x greater.

Coil currents are specified for the 2-T, high beta scenario, as shown in Table-1 [3].

3. Analysis Methodology

Using the MAGFOR code [4] as a guide, a finite-element computer program has been developed that uses 2-node line-type current-carrying elements to represent the individual conductors. The source code for the field calculator is listed in Appendix-A and the force calculator in Appendix-B. The programs are written in Python, an open source language available from http://www.python.org.

The geometry is specified in ANSYS format using N, R, and EN commands [5].

Once the running load is determined, peak stress is calculated by treating the unsupported length of conductor as a simply supported beam.

4. Results

In order to verify the field/force calculator, a comparison has been made with MAGFOR results obtained during preparations for the C1 coil test [6]. The MAGFOR code represents current-carrying elements using 20-node isoparametric hexahedrons, but in this analysis the elements shapes are simple and sometimes overlapping. As shown in figures 3-5, the MAGFOR model produces a field at the coil center of 0.55-T and a running load in the winding pack that varies from 1,350- to 6,800-lb/in. The field/force calculator produces the same field at the coil center and the same average running load, but the peak value is 3,050-lb/in. In the leads area, the running loads are in good agreement. Since the MAGFOR model has overlapping elements in the areas with the highest discrepancy, it is believed that there is an overestimate in these areas, and that the filament-based calculator gives a reasonable result.

The analysis model was extended to include all modular, TF, and PF coils at the time steps described in Table-1. Results indicate that the maximum running load in the winding pack occurs for the Type-B coil at time=0.050-s. The poloidal variation of running load at that time step is shown in Fig.-6. At other time steps, the peak running load for the Type-B coil decreases by about 8%.

The peak load in the leads area at time=0.50-s is 91-, 92-, and 67-lb/in for the Type-A, -B, and –C coils respectively. Figure 7 shows the distribution along the conductor from the winding pack, which is supported by the lead blocks, to the terminal lug attachment, which is unsupported. In the 4.1-in long unsupported region, the maximum running load is 34-lb/in for Type-B coil. This corresponds to a conductor stress of 30-ksi, which is less than the allowable based on testing [7].

5. Summary

The results indicate that for the complete coil set operating at 2-T, the electromagnetic loads on the leads are no worse than for the C1 coil test, and the peak stress in the conductor is less than the allowable. If non-coaxial buswork is required to route the leads from the assembly shown on drawing SE142C-050, then the analysis should be extended to determine the required supports.

6. Attachments

Input files are located at:
ftp://ftp.pppl.gov//priv/bob-simmons+sig03/Williamson/Job1416

field3.py – source code for field calculator
force3.py – source code for force calculator
case_c1.prp – input file for coil C1, 36,580-A/turn
case_050.prp – input file for all mod coils, TF, and PF at time=0.050-s

7. References

[1] Modular Coils Design Description, May-2004
http://ncsx.pppl.gov/NCSX_Engineering/Technical_Data/SDDs/040519_FDR_SDDs/SDD_WBS14_040510.doc

[2] MCWF Final Design Review, May 19-20, 2004
http://ncsx.pppl.gov/Meetings/FDR_2004/FDR_htm/NCSX_Final_Design_Review.html

[3] NCSX Coil Technical Data
http://ncsx.pppl.gov/SystemsEngineering/Requirements/Specs/GRD/Rev4/C08R00_C8_TDS.pdf

[4] W.D. Cain, MAGFOR- A Magnetics Code to Calculate Fields and Forces in Twisted Helical Coils of Constant Cross Section, Sym Fusion Engr, 1984.

[5] ANSYS Ver 11.0 User Manual, ANSYS Inc, Canonsburg, PA

[6] G. Gettelfinger, C1 Coil Test Report, Jun-2006
http://ncsx.pppl.gov/NCSX_Engineering/R&D_Results/PPPL/C1%20Testing/Analysis%20results/C1%20Test%20Results_Final.pdf

[7] I. Zatz, Interim Material Properties, Dec-2003
http://ncsx.pppl.gov/NCSX_Engineering/R&D_Results/CTD/031212_MatProps_140_IJZ.doc

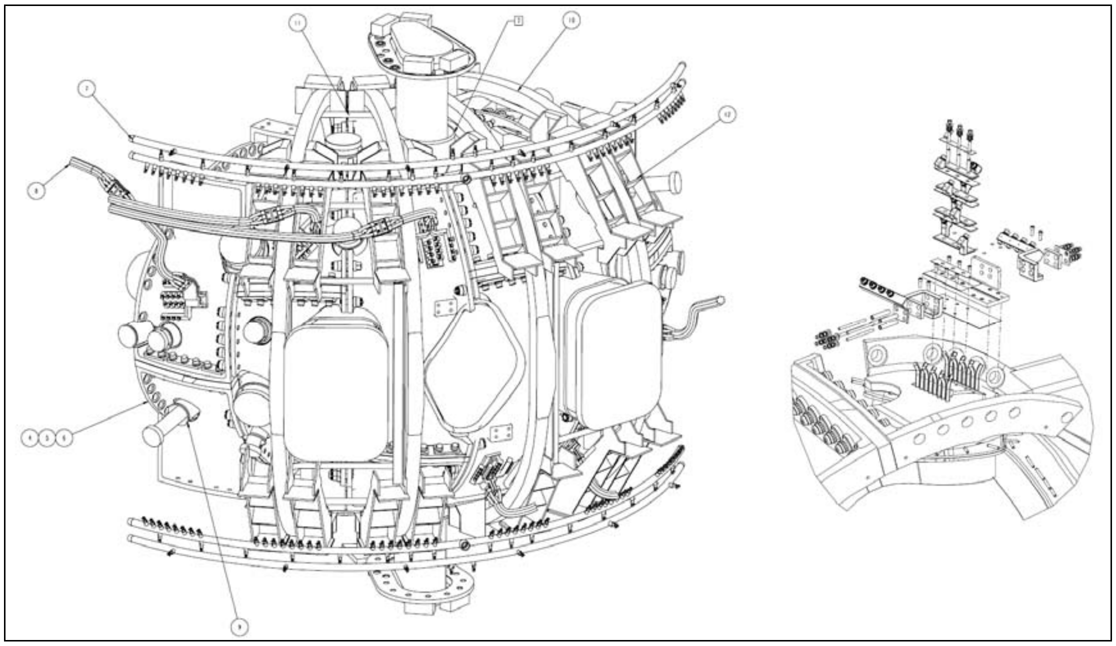
Fig.-1 Modular Coil Leads Assembly

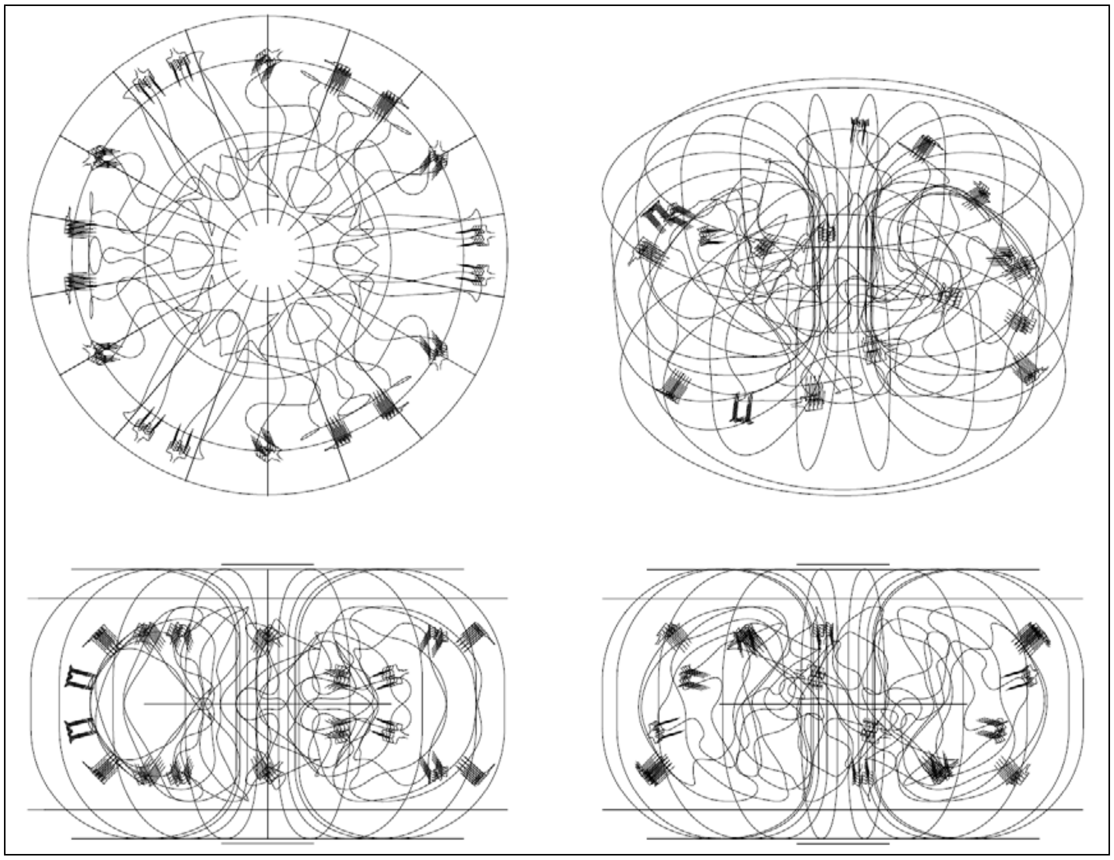
Fig.-2 Coil and Leads Finite Element Model

Table-1 Coil Currents for 2-T High Beta Scenario

	M1	M2	M3	PF1	PF2	PF3	PF4	PF5	PF6	TF	Plasma
Turns	22	22	20	72	72	72	80	24	14	12	1

2T High Beta Scenario	t(s)	M1	M2	M3	PF1	PF2	PF3	PF4	PF5	PF6	TF	Plasma
	-0.850	0	0	0	0	0	0	0	0	0	0	0
	0.000	37190	37783	36538	-14615	-14615	-9054	-7498	460	453	-1301	0
	0.050	37190	37783	36538	-14615	-14615	-9054	-7498	460	453	-1301	0
	0.097	37190	37783	36538	-6931	-6931	-2919	-5041	660	6114	-1301	-141238
	0.192	35075	34852	31783	-7540	-7540	-4319	-7595	2348	7300	4424	-209732
	0.197	35075	34852	31783	-7522	-7522	-4305	-7589	2349	7300	4424	-209732
Maximum		37190	37783	36538	0	0	0	0	2349	7300	4424	0
Minimum		0	0	0	-14615	-14615	-9054	-7595	0	0	-1301	-209732
I2t (A2-s)		1.28E+09	1.27E+09	1.15E+09	1.41E+08	1.40E+08	5.74E+07	5.20E+07	2.24E+06	1.20E+07	2.36E+08	
tESW (s)		0.93	0.89	0.86	0.66	0.65	0.70	0.90	0.41	0.23	12.06	

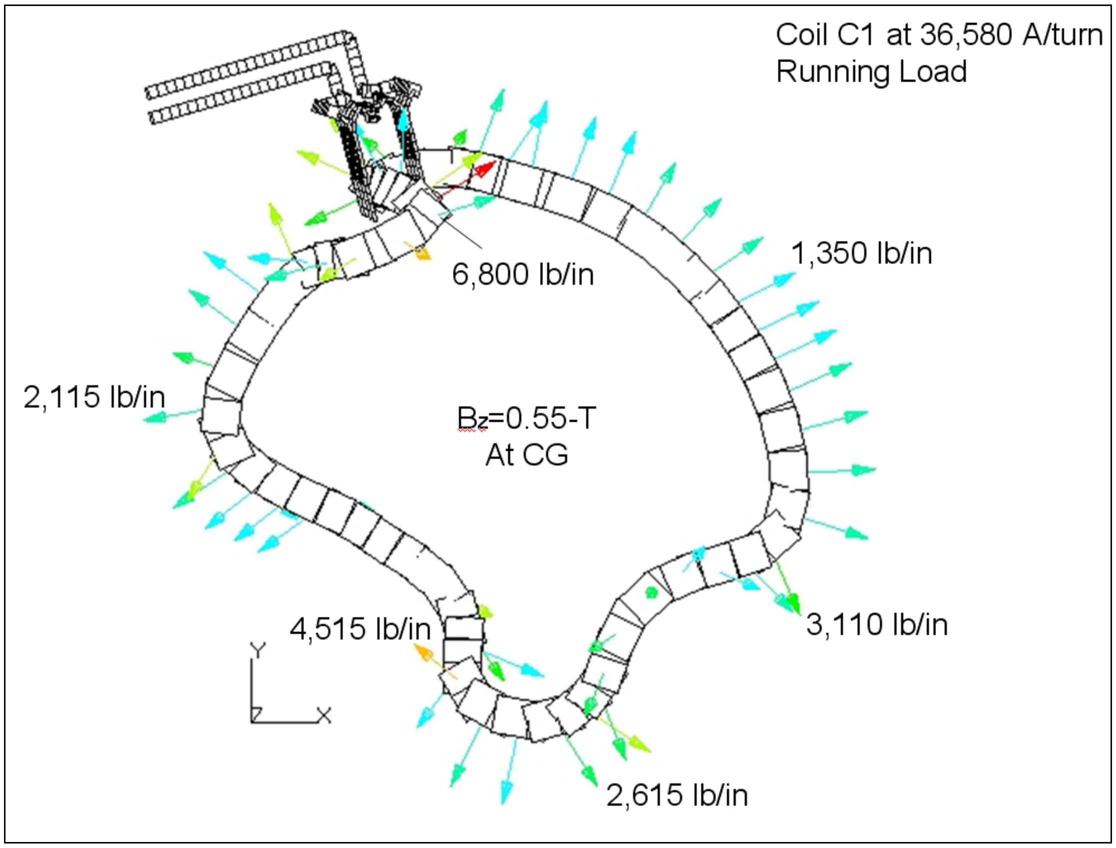

Fig.-3 EM Load for Single Type-C Coil (MAGFOR)

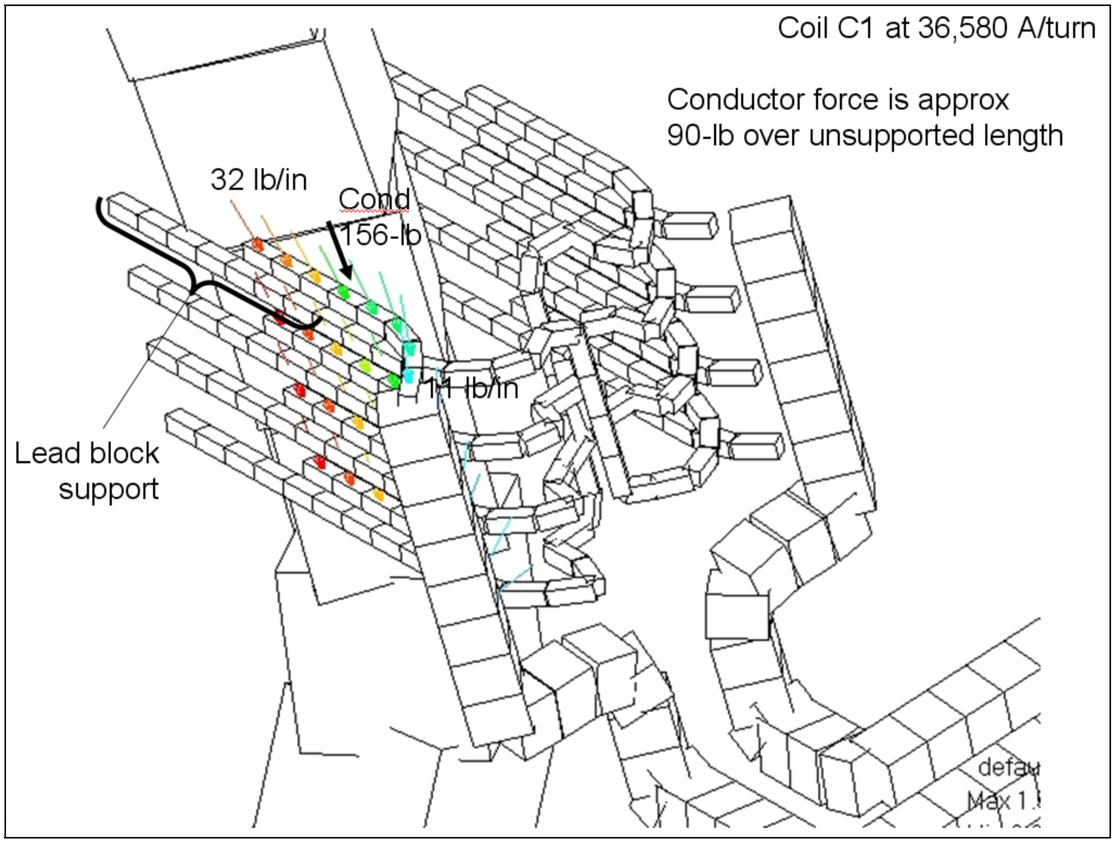

Fig.-4 Conductor EM Load (MAGFOR)

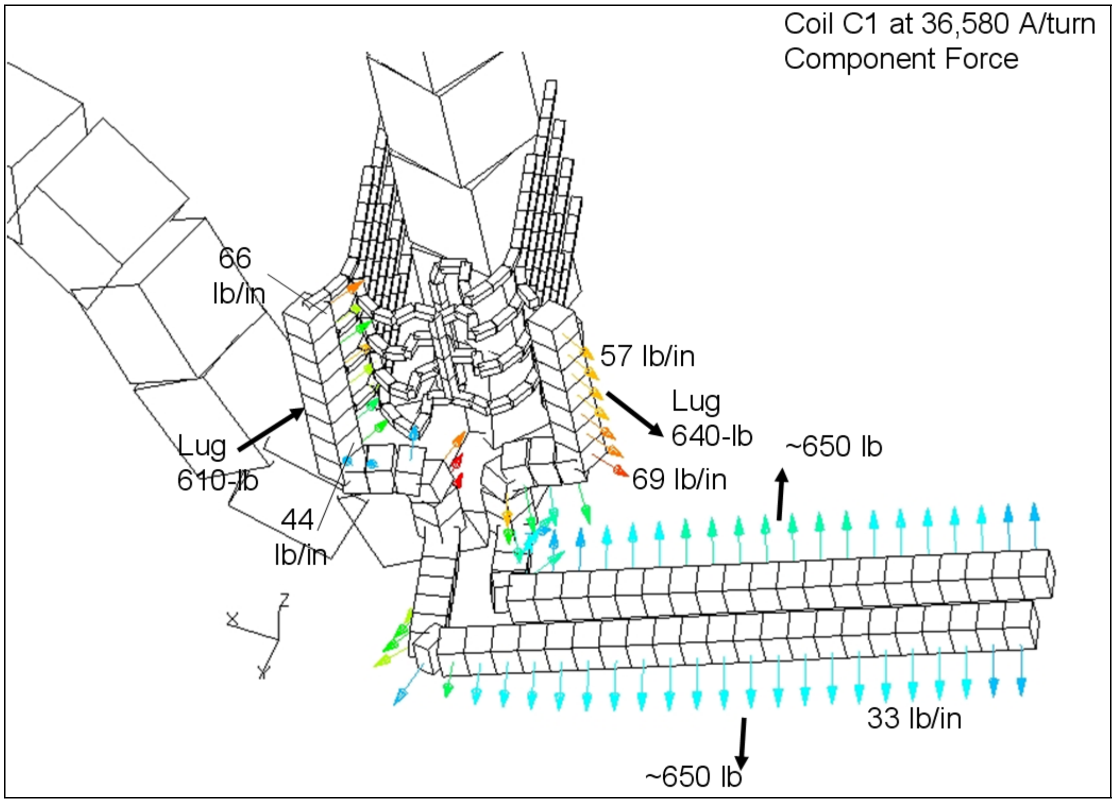

Fig.-5 Lugs EM Load (MAGFOR)

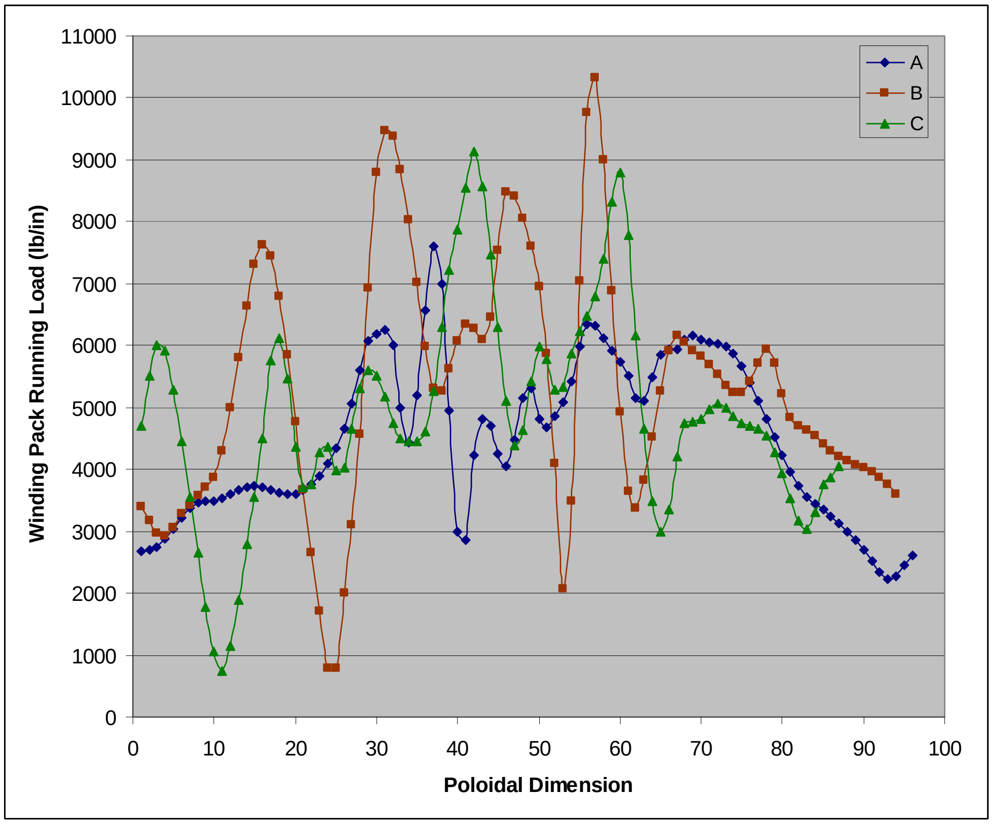

Fig.-6 Poloidal Variation of Running Load at Time=0.050-s

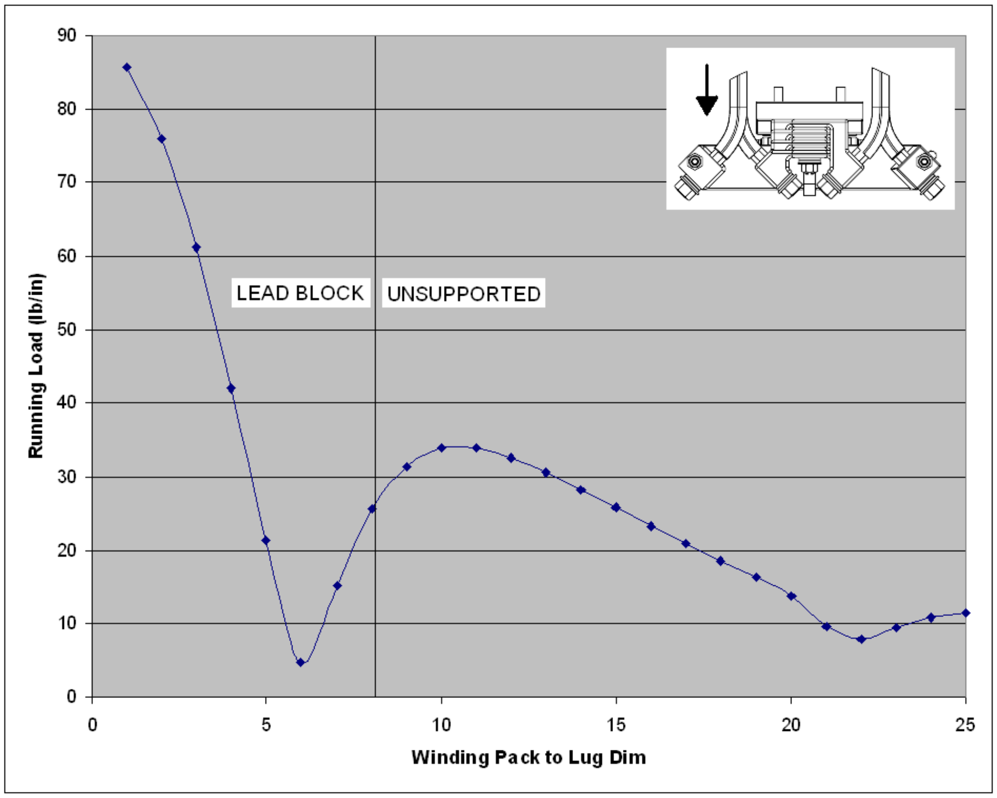

Fig.-7 Running Load

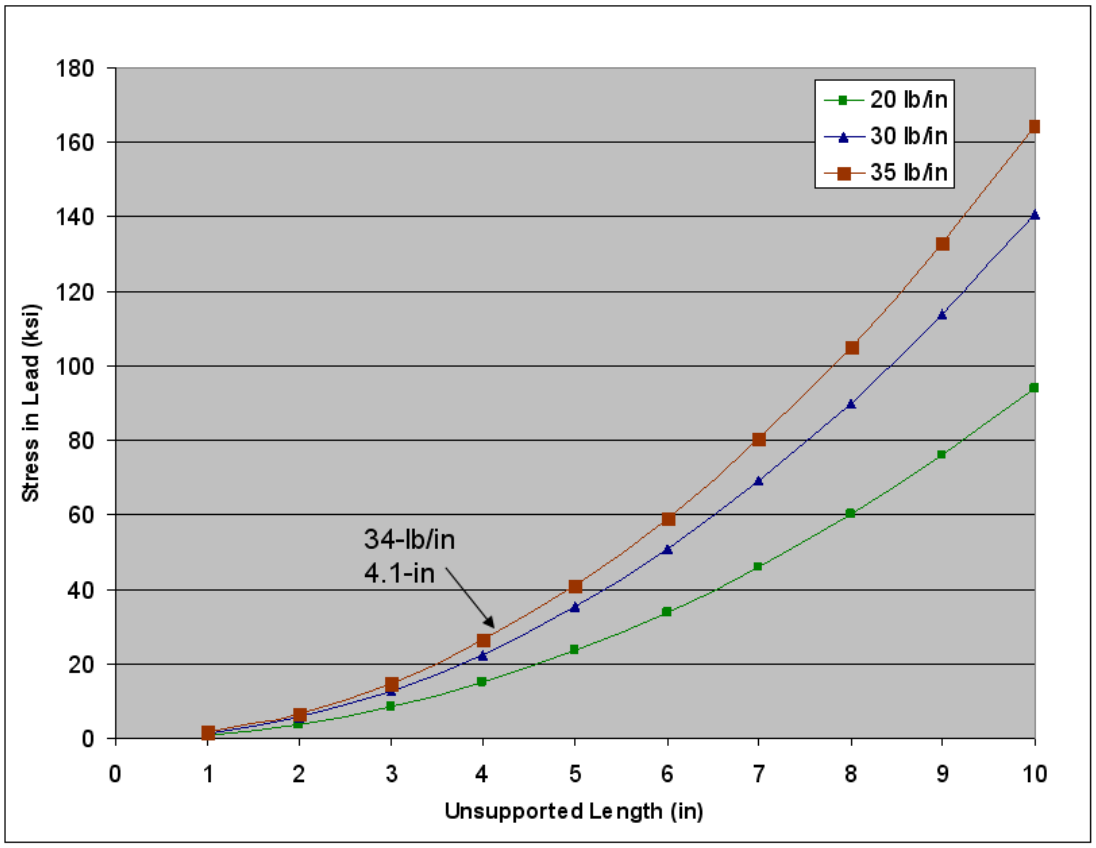

Fig.-8 Stress in Unsupported Conductor

Appendix A – Program for Field Calculation

```
#
# field.py - mag field due to line currents
# cmd: python field.py model.prp >field.out
import sys
from math import sqrt
#
# read ansys input
f=open(sys.argv[1],'r')
nodelist=[]
elemlist=[]
cur=0.
for line in f:
    if line[0:2]=='N,':
        cols=line.split(',')
        nodelist.append(int(cols[1]))
        nodelist.append(float(cols[2])*0.0254)
        nodelist.append(float(cols[3])*0.0254)
        nodelist.append(float(cols[4])*0.0254)
    elif line[0:2]=='R,':
        cols=line.split(',')
        cur=float(cols[2])
    elif line[0:2]=='EN':
        cols=line.split(',')
        elemlist.append(int(cols[1]))
        elemlist.append(int(cols[2]))
        elemlist.append(int(cols[3]))
        elemlist.append(cur)
f.close()
#
# set gauss points
w=[-0.3399810436,0.6521451548,0.3399810436,0.6521451548, \
   -0.8611363116,0.3478548541,0.8611363116,0.3478548451]
#
# set field to zero
bn=[0.]*len(nodelist)
#
# loop thru elements
for i in range(0,len(elemlist),4):
    #
    # end nodes
    x1=[]
    x2=[]
    for j in range(0,len(nodelist),4):
        if nodelist[j]==elemlist[i+1]:
            x1.append(nodelist[j+1])
            x1.append(nodelist[j+2])
            x1.append(nodelist[j+3])
        elif nodelist[j]==elemlist[i+2]:
            x2.append(nodelist[j+1])
            x2.append(nodelist[j+2])
            x2.append(nodelist[j+3])
    #
    # elem length
    lx=x2[0]-x1[0]
    ly=x2[1]-x1[1]
    lz=x2[2]-x1[2]
    le=sqrt(lx**2+ly**2+lz**2)
    #
    # current vector
    jx=elemlist[i+3]*lx/le
    jy=elemlist[i+3]*ly/le
    jz=elemlist[i+3]*lz/le
    #
    # loop thru integration pnts
    for j in range(0,len(w),2):
        #
        # gauss point coord
        x3=[]
        h1=0.5*(1.-w[j])
        h2=0.5*(1.+w[j])
        x3.append(h1*x1[0]+h2*x2[0])
        x3.append(h1*x1[1]+h2*x2[1])
        x3.append(h1*x1[2]+h2*x2[2])
        cst=-1.e-7*w[j+1]*le/2.
        #
        # loop thru nodes
        for k in range(0,len(nodelist),4):
            #
            # dist to gauss point
            dx=nodelist[k+1]-x3[0]
            dy=nodelist[k+2]-x3[1]
            dz=nodelist[k+3]-x3[2]
            rr=sqrt(dx**2+dy**2+dz**2)
            r3=rr*rr*rr
            #
            # field component
            bn[k]=nodelist[k]
            bn[k+1]+=cst*(jz*dy-jy*dz)/r3
            bn[k+2]+=cst*(jx*dz-jz*dx)/r3
            bn[k+3]+=cst*(jy*dx-jx*dy)/r3
#
# field results
nmx=1
bmx=0.
for i in range(0,len(nodelist),4):
    bmo=sqrt(bn[i+1]**2+bn[i+2]**2+bn[i+3]**2)
    print bn[i],bn[i+1],bn[i+2],bn[i+3],bmo
    if bmo >= bmx:
        nmx=bn[i]
        bmx=bmo
print 'node, bmax:',nmx,bmx
```

Appendix B – Program for Force Calculation

```
#
# force.py - force due to line current
# cmd: python force.py model.prp field.inp >force.out
# note: remove "node, bmax:" line from field output
import sys
from math import sqrt
#
# read ansys input
f=open(sys.argv[1],'r')
nodelist=[]
elemlist=[]
cur=0.
for line in f:
    if line[0:2]=='N,':
        cols=line.split(',')
        nodelist.append(int(cols[1]))
        nodelist.append(float(cols[2])*0.0254)
        nodelist.append(float(cols[3])*0.0254)
        nodelist.append(float(cols[4])*0.0254)
    elif line[0:2]=='R,':
        cols=line.split(',')
        cur=float(cols[2])
    elif line[0:2]=='EN':
        cols=line.split(',')
        elemlist.append(int(cols[1]))
        elemlist.append(int(cols[2]))
        elemlist.append(int(cols[3]))
        elemlist.append(cur)
f.close()
#
# read field at nodes
f=open(sys.argv[2],'r')
bn=[]
for line in f:
    cols=line.split()
    bn.append(int(cols[0]))
    bn.append(float(cols[1]))
    bn.append(float(cols[2]))
    bn.append(float(cols[3]))
f.close()
#
# set gauss points
w=[-0.3399810436,0.6521451548,0.3399810436,0.6521451548, \
   -0.8611363116,0.3478548541,0.8611363116,0.3478548451]
#
# set force to zero
fe=[0.]*len(elemlist)
#
# loop thru elements
for i in range(0,len(elemlist),4):
    # fe[i]=elemlist[i]
    #
    # end nodes
    x1=[]
    x2=[]
    for j in range(0,len(nodelist),4):
        if nodelist[j]==elemlist[i+1]:
            x1.append(nodelist[j+1])
            x1.append(nodelist[j+2])
            x1.append(nodelist[j+3])
        elif nodelist[j]==elemlist[i+2]:
            x2.append(nodelist[j+1])
            x2.append(nodelist[j+2])
            x2.append(nodelist[j+3])
    #
    # field at end nodes
    b1=[]
    b2=[]
    for j in range(0,len(nodelist),4):
        if bn[j]==elemlist[i+1]:
            b1.append(bn[j+1])
            b1.append(bn[j+2])
            b1.append(bn[j+3])
        elif bn[j]==elemlist[i+2]:
            b2.append(bn[j+1])
            b2.append(bn[j+2])
            b2.append(bn[j+3])
    #
    # elem length
    lx=x2[0]-x1[0]
    ly=x2[1]-x1[1]
    lz=x2[2]-x1[2]
    le=sqrt(lx**2+ly**2+lz**2)
    fe[i]=le
    #
    # current vector
    jx=elemlist[i+3]*lx/le
    jy=elemlist[i+3]*ly/le
    jz=elemlist[i+3]*lz/le
    #
    # loop thru integration pnts
    for j in range(0,len(w),2):
        #
        # field at gauss pnt
        h1=0.5*(1.-w[j])
        h2=0.5*(1.+w[j])
        bx=h1*b1[0]+h2*b2[0]
        by=h1*b1[1]+h2*b2[1]
        bz=h1*b1[2]+h2*b2[2]
        cst=w[j+1]*le/2.
        #
        # force component
        fe[i+1]+=cst*(jy*bz-jz*by)
        fe[i+2]+=cst*(jz*bx-jx*bz)
        fe[i+3]+=cst*(jx*by-jy*bx)
#
# force results
emx=1
fmx=0.
for i in range(0,len(elemlist),4):
    fm=sqrt(fe[i+1]**2+fe[i+2]**2+fe[i+3]**2)
    xl=fe[i]*4.448*39.370
    print elemlist[i],fe[i+1],fe[i+2],fe[i+3],fm/xl
    if fm/xl >= fmx:
        emx=elemlist[i]
        fmx=fm/xl
print 'elem, fmax:',emx,fmx
```