# Appendix A: 1 kt Nuclear Blast Standard

This appendix contains a FORTRAN compilable listing of the 1 kt nuclear blast standard. This standard has been adopted as the blast standard by both the American National Standards Institute (ANSI) and the International Standards Committee. The input and output units are MKS, with the yield in kilotons. Internally the program uses cgs units; all scaling is handled internally. Program main is included to provide a simple example of how the standard can be used. The reader is encouraged to write their own driver.

The program includes a subroutine for the 1962 standard atmosphere which is used for scaling. This is a stand alone routine which can be used for many purposes.

The program also contains a subroutine for the equation of state for air. This is the equation of state referenced in Chap. 3. It includes the effects of vibrational and rotational excitation of nitrogen and oxygen, dissociation and first ionization levels of nitrogen and oxygen. Heed the warnings on limitations given in Chap. 3.

The subroutine "well" represents the fireball. At late times the fireball will rise out of the center of the nearly spherical blast wave bubble. This routine attempts to account for the fireball motion.

The function "energyl" takes as input the density and the pressure and iteratively finds the specific internal energy for air.

```
      Program MAIN
       yield=1.
       height=100.
       tim=.25
       do 1 I=1,250
       ra=float(I)
       call shock(yield,height,tim,ra,opr,odr,vr)
       Print 950, tim,ra,opr,odr,vr
1      continue
950    format(1P8e12.4)
       end
```



C.E. Needham, *Blast Waves*, Shock Wave and High Pressure Phenomena,
DOI 10.1007/978-3-319-65382-2

```
      subroutine shock (yield,height,tim,ra,opr,odr,vr)
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      alt=height
c
      call scalkt (alt,yield,vsl,dsl,tsl,csl,psl)
c
      t=tim*tsl
      r=ra/csl
c
      call peak (t,r,prado,oppko,odpko,opro,odro,vpko,vro)
c
      opr=opro*psl
      odr=odro*dsl
      vr=vro*vsl
c
      return
      end
```

```
      subroutine peak  (t,ra,prado,oppko,odpko,opro,odro,vpko,vro)
c
      save
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      common /wfrt/ prad,oppk,odpk,vpk,opr,odr,vr,rzp,rzd,rzv,opmn,odmn
     c              ,vmn
c
      data told /0./
      data psca /0.1/, vsca /0.01/, rhosca /1000./
c
c   Convert distance to cm
c
      r=ra*100.
c
c   Calculate waveform radius at peak overpressure.
c
      if (t .ne. told) then
c
         rzp=wfzr(t)
         rzd=wfdzr(t)
         rzv=wfvzr(t)
         prad=wfpr(t)
c
         prado=prad*vsca
c
c   Calculate waveform peak overpressure.
c
         oppk=wfpkop(prad)
         oppko=oppk*psca
c
c   Calculate waveform peak overdensity.
c
         odpk=wfpkod(prad)
         odpko=odpk*rhosca
c
c   Calculate waveform peak velocity.
c
         vpk=wfpkv(prad)
         vpko=vpk*vsca
         told=t
      endif
c
      opr=0.
      odr=0.
      vr=0.
      opro=0.
      odro=0.
      vro=0.
c
      if (r .le. prad) then
c
c   Calculate overpressure at r.
c
         call wfprmt (t,r)
```

```
         opro=opr*psca
c
c   Calculate overdensity at r.
c
         call wfdrmt (t,r)
         call well (t,r,odw)
         odro=odw*rhosca
c
c   Calculate velocity at r.
c
         call wfvrmt (t,r)
         vro=vr*vsca
      endif
c
      return
      end
```

```
      subroutine scalkt (hfpt,wb,vscale,dscale,tscale,cscale,pscale)
c
      save
c
c   This routine sets up the scale factors for each burst so the 1 KT
c   sea level data may be used
c
c   Inputs
c
c   hfpt - Height of field point above sea level in meters
c   wb   - Yield of the burst in KT
c
c   Output
c
c   tscale - scales the actual time to the 1 KT sea level time
c   vscale - scales the 1 KT sea level velocities to actual
c   pscale - scales the 1 KT sea level pressures to actual
c   cscale - scales the 1 KT sea level dimensions to actual
c   dscale - scales the 1 KT sea le vel densities to actual
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      dimension hn(4)
      data p1 /1.01325e5/, c1 /3.4029399e2/, r1 /1.225/, t1 /288.15/
c
      cubrt(x)=sign(exp(alog(abs(x))/3. ),x)
c
      call matm62 (hfpt,p3,c3,r3,t3)
      rp3=cubrt(p1/p3)
      wsw=cubrt(wb)
      tscale=c3/(wsw*rp3*c1)
      vscale=c3/c1
      pscale=p3/p1
      dscale=r3/r1
      cscale=rp3*wsw
c
      return
      end
```

```
      subroutine terror (t, name)
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      character*6 name
c
      write (6,2) t, name
      stop
c
    2 format (1x/' Time ERROR.'/' t = ',1pe13.5,' seconds in routine '
     c       ,a6/)
c
      end
```

```
      subroutine well (t,r,depth)
c
      save
c
c    This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      common /wfrt/ prad,oppk,odpk,vpk,opr,odr,vr,rzp,rzd,rzv,opmn,odmn
     c              ,vmn
c
      depth=odr
      rmsw=amin1(0.7*rzd,1.55e4)
c
      if (t .le. 1.2 .and. r .le. rmsw) then
c
         depth=amax1(-1.21e-3,-1.5e-3*exp(-8.e-13*r**3))
         if (t .gt. 1.) depth=(1.2-t)*depth*5.
         if (r .ge. 0.9*rmsw) depth=10.*(depth*(rmsw-r)
     c                             +odr*(r-0.9*rmsw))/rmsw
      endif
c
      return
      end
```

```
      function wfzr (t)
c
      save
c
c   Calculates the waveform radius at zero overpressure, (wfzr).
c   (i.e., the radius separating the negative and positive phase
c   portions of the waveform, at the specified time (t)).
c
c      wfzr is not defined for times less than approximately 0.1 sec
c      because for these times the waveform does not exhibit a
c      negative phase.
c
c      t    - Time (sec)
c      wfzr - Radius (cm)
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      data b /0.03291/, c /-1.086/, cz /33897./, bz /8490./
c
      wfzr=0.
c
      if (t .lt. 0.) then
         call terror (t, 'wfzr  ')
      else
         wfzr=(1.-b*t**c)*(cz*t+bz)
      endif
c
      return
      end
```

```
      function wfpr(t)
c
      save
c
c   This function also calculates the shock front radius at the
c   specified time.
c
      r1=24210.*t**0.371*(1.+(1.23*t+0.123)*(1.-exp(-26.25*t**0.79)))
      r2=(1.-0.03291*t**(-1.086))*(33897.*t+8490.)+8.36e3+2.5e3
     c   *alog(t)+800.*t**(-0.21)
c
      if (t .lt. 0.21) then
         wfpr=r1
      else if (t .gt. 0.28) then
         wfpr=r2
      else
         wfpr=(r2*(t-0.21)+r1*(0.28-t))/0.07
      endif
c
      return
      end
```

```
      function wfpkod (dummy)
c
      save
c
c   This routine computes the peak overdensity given the peak
c   overpressure using Rankine-Hugoniot relations.
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      common /wfrt/ prad,oppk,odpk,vpk,opr,odr,vr,rzp,rzd,rzv,opmn,odmn
     c              ,vmn
c
      data rhoz /1.225e-3/, opz /1.01325e6/, gamm1 /0.404574/
c
      op=oppk
      rtio=op/opz
      p=op+opz
      gmone=gamm1
      gamma=gamm1+1.
      gamra=gamma
c
      do 2 n=1,20
      rho1=rhoz*((2.*gamra+(gamra+1.)*rtio)/(2.*gamra+(gamra-1.)*rtio))
      ee=p/(gmone*rho1)
      call air (ee,rho1,gmone,dum1,dum2)
      gamrao=gamra
      gamra=2.*gmone/(2.5*gmone+1.)+1.
      if (abs(gamra-gamrao) .lt. 1.e-5) go to 3
    2 continue
c
    3 wfpkod=rho1-rhoz
c
      return
      end
```

```
      function wfpkv (dummy)
c
      save
c
c   This routine computes the particle velocity at peak overpressure
c   given the overdensity and overpressure using Rankine-Hugoniot
c   relations.
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      common /wfrt/ prad,oppk,odpk,vpk,opr,odr,vr,rzp,rzd,rzv,opmn,odmn
     c              ,vmn
c
      data rhoz /1.225e-3/
c
      wfpkv=sqrt(oppk*odpk/(rhoz*(rhoz+odpk)))
c
      return
      end
```

```
      subroutine wfdrmt (t,r)
c
      save
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      common /wfrt/ prad,oppk,odpk,vpk,opr,odr,vr,rzp,rzd,rzv,opmn,odmn
     c             ,vmn
c
      data ttold /0./
c
      rpk=prad
c
      if (t .ge. 0.2) then
c
         rpk=rpk*1.e-5
         r=r*1.e-5
c
         if (t .ne. ttold) then
            rz=rzd
            rmn=rz-9.7163e3*t**0.12115
            rz=rz*1.e-5
            rmn=rmn*1.e-5
            odmn=-0.5*odpk+2.2e-5*t**(-1.6026)
            rneg=rz-rmn
            rpls=rpk-rz
            rnp=rpk-rmn
            aln=odpk/rpls
            bln=odpk-aln*rpk
            odmnln=aln*rmn+bln
            fmlt=abs(odmn/odpk)
            odmhy=odmn+fmlt*(odmnln-odmn)
            alpha=(rnp/(odmhy-odpk)+rpls/odpk)/rneg
            beta=-rpls*(alpha+1./odpk)
            fngz=rnp/rpls
            dnom=alpha*rnp+beta
            bcrmn=1.-odmn/(rnp/dnom+odpk)
            crmnlb=alog(bcrmn)
            cgzl=(beta*(1./bcrmn-1.)/dnom)/((fngz*crmnlb
     c           *rnp**(fngz-1.))*(rnp+odpk*dnom))
            cgz=exp(cgzl)
            bgzl=crmnlb/cgz**(rnp**fngz)
            bgz=exp(bgzl)
         endif
c
         rbr=rpk-r
         gr=1.-bgz**(cgz**(rbr**fngz))
         if (r .gt. rz) gr=(rpk-r)/rpls*gr+(r-rz)/rpls
         hr=rbr/(alpha*rbr+beta)+odpk
         odr=gr*hr
         rpk=rpk*1.e5
         r=r*1.e5
      endif
c
      if (t .le. 1.) then
c
```

```
         if (t .ne. ttold) then
            a=-1.2e-3
            c=alog(max(1.e-20,-a/(odpk-a)))/(rzd-rpk)
            b=(odpk-a)*exp(-c*rpk)
         endif
c
         wflt=a+b*exp(c*r)
c
         if (t .le. 0.2) then
            odr=wflt
         else
            odr=(wflt*(1.-t)+odr*(t-0.2))*1.25
         endif
      endif
c
      ttold=t
c
      return
      end
```

```
      function wfdzr (t)
c
      save
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      data b /0.03499/, c /-1.068/, cz /33897./, bz /8490./
c
      wfdzr=0.
c
      if (t .lt. 0.) then
         call terror (t, 'wfdzr ')
      else if (t .gt. 0. .and. t .lt. 0.265) then
         wfdzr=2.568e4*t**0.395
      else
         wfdzr=(1.-b*t**c)*(cz*t+bz)+500.
      endif
c
      return
      end
```

```
      subroutine wfprmt (t,r)
c
      save
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      common /wfrt/ prad,oppk,odpk,vpk,opr,odr,vr,rzp,rzd,rzv,opmn,odmn
     c             ,vmn
c
      data ttold /0./
c
      rpk=prad
c
      if (t .ge. 0.1) then
c
         rpk=rpk*1.e-5
         r=r*1.e-5
c
         if (t .ne. ttold) then
            rz=rzp
            rmn=rz-9.7163e3*t**0.12115
            rz=rz*1.e-5
            rmn=rmn*1.e-5
            opmn=2.2e4/(t*sqrt(t))-0.5*oppk
            rneg=rz-rmn
            rpls=rpk-rz
            rnp=rpk-rmn
            aln=oppk/rpls
            bln=oppk-aln*rpk
            opmnln=aln*rmn+bln
            fmlt=abs(opmn/oppk)
            opmhy=opmn+fmlt*(opmnln-opmn)
            alpha=(rnp/(opmhy-oppk)+rpls/oppk)/rneg
            beta=-rpls*(alpha+1./oppk)
            fngz=rnp/rpls
            dnom=alpha*rnp+beta
            bcrmn=1.-opmn/(rnp/dnom+oppk)
            crmnlb=alog(bcrmn)
            cgzl=(beta*(1./bcrmn-1.)/dnom)/((fngz*crmnlb
     c           *rnp**(fngz-1.))*(rnp+oppk*dnom))
            cgz=exp(cgzl)
            bgzl=crmnlb/cgz**(rnp**fngz)
            bgz=exp(bgzl)
         endif
c
         rbr=rpk-r
         gr=1.-bgz**(cgz**(rbr**fngz))
         if (r .gt. rz) gr=(rpk-r)/rpls*gr+(r-rz)/rpls
         hr=rbr/(alpha*rbr+beta)+oppk
         opr=gr*hr
         rpk=rpk*1.e5
         r=r*1.e5
      endif
c
      if (t .lt. 0.95) then
c
```

```
         if (t .ne. ttold) then
            a=5.446e4*t**(-1.22)-7.135e5*(1.-t**2)
            if (t .lt. 0.2) c=7.41e-5*t**(-0.885)
            if (t .ge. 0.2) c=alog(max(1.e-20,-a/(oppk-a)))/(rzp-rpk)
            b=(oppk-a)*exp(-c*rpk)
         endif
c
         wflt=a+b*exp(c*r)
c
         if (t .lt. 0.1) then
            opr=wflt
         else
           opr=(wflt*(0.95-t)+opr*(t-0.1))/.85
         endif
      endif
c
      ttold=t
c
      return
      end
```

```
      function wfvzr (t)
c
      save
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      data b /0.08459/, c /-1.34/, cz /33897./, bz /8490./
c
      wfvzr=0.
c
      if (t .lt. 0.) then
         call terror (t, 'wfvzr ')
      else if (t .gt. 0. .and. t .lt. 0.263) then
         wfvzr=2.5e4*t**0.8
      else
         wfvzr=(1.-b*t**c)*(cz*t+bz)
      endif
c
      return
      end
```

```
      subroutine wfvrmt (t,r)
c
      save
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      common /wfrt/ prad,oppk,odpk,vpk,opr,odr,vr,rzp,rzd,rzv,opmn,odmn
     c              ,vmn
c
      data ttold /0./
c
      rpk=prad
c
      if (t .ge. 0.15) then
c
         rpk=rpk*1.e-5
         r=r*1.e-5
c
         if (t .ne. ttold) then
            rz=rzv
            rmn=rz-9.31e3*t**0.12115
            rz=rz*1.e-5
            rmn=rmn*1.e-5
            vmn=650./(t*sqrt(t))-0.5*vpk
            rneg=rz-rmn
            rpls=rpk-rz
            rnp=rpk-rmn
            aln=vpk/rpls
            bln=vpk-aln*rpk
            ovmnln=aln*rmn+bln
            fmlt=abs(vmn/vpk)
            ovmhy=vmn+fmlt*(ovmnln-vmn)
            alpha=(rnp/(ovmhy-vpk)+rpls/vpk)/rneg
            beta=-rpls*(alpha+1./vpk)
            fngz=rnp/rpls
            dnom=alpha*rnp+beta
            bcrmn=1.-vmn/(rnp/dnom+vpk)
            crmnlb=alog(bcrmn)
            cgzl=(beta*(1./bcrmn-1.)/dnom)/((fngz*crmnlb
     c           *rnp**(fngz-1.))*(rnp+vpk*dnom))
            cgz=exp(cgzl)
            bgzl=crmnlb/cgz**(rnp**fngz)
            bgz=exp(bgzl)
         endif
c
         rbr=rpk-r
         gr=1.-bgz**(cgz**(rbr**fngz))
         if (r .gt. rz) gr=(rpk-r)/rpls*gr+(r-rz)/rpls
         hr=rbr/(alpha*rbr+beta)+vpk
         vr=gr*hr
         rpk=rpk*1.e5
         r=r*1.e5
      endif
c
      if (t .le. 0.2) then
c
```

```
         rtio=r/prad
         wflt=vpk*rtio*sqrt(rtio)
c
         if (t .lt. 0.15) then
            vr=wflt
         else
            vr=(wflt*(0.20-t)+vr*(t-0.15))*20.
         endif
      endif
c
      ttold=t
c
      return
      end
```

```
      function wfpkop (r)
c
      save
c
c   Calculates the waveform peak overpressure (wfpkop) at the specified
c   radius (r), i.e., the overpressure at the shock front having the
c   specified radius (r).
c
c      r      - Radius(cm)
c      wfpkop - Peak overpressure (dynes/cm**2)
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      common /wfrt/ prad,oppk,odpk,vpk,opr,odr,vr,rzp,rzd,rzv,opmn,odmn
     c             ,vmn
c
      data ac /3.04e18/, aq /1.13e14/, astar /7.9e9/, rstar /4.454e4/
c
      rr=1./r
      rtio=2.24517e-5*r
      cf=sqrt(alog(rtio+3.*exp(-(sqrt(rtio)/3.))))
      wfpkop=((ac*rr+aq)*rr+astar/cf)*rr
c
      return
      end
```

```
      function speed (t)
c
      save
c
c   This routine is the time derivative of wfpr2.  It returns the
c   shock front speed as a function of time.
c
c   This routine is part of the AFWL 1KT Standard by Needham, et al.
c
      r1=24210.*t**0.371*(1.+(1.23*t+0.123)*(1.-exp(-26.25*t**.79)))
      r2=(1.-0.03291*t**(-1.086))*(33897.*t+8490.)+8.36e3+2.5e3*
     c   alog(t)+800.*t**(-0.21)
      v1=10086.685*t**(-0.629)+40826.049*t**0.371+exp(-26.25*t**0.79)*
     c   (617527.5*t**1.161-40826.049*t**0.371-1104.7749*t**(-0.629)+
     c    61752.75*t**0.161)
      v2=33897.+95.937326*t**(-1.086)+303.43481*t**(-2.086)+2.5e3/t-
     c   168.*t**(-1.21)
      v3=(v2*(t-0.21)+v1*(0.28-t))/0.07
c
      if (t .lt. 0.21) then
         speed=v1
      else if (t .gt. 0.28) then
         speed=v2
      else
         speed=v3
      endif
c
      return
      end
```

```
      subroutine air (eee,rrr,gmone,p,temp)
c
      e=eee*1.0e-10
      rholn=alog(773.39520495*rrr)
      e1=(8.5-e)*1.0256410256
      if (abs(e1) .lt. 5.) go to 20
      if (e1 .le. 0.) go to 10
      fo=exp(-0.22421524664*e)
      fon=0.
      ws=1.
      go to 30
c
   10 fo=0.
      fon=exp(-0.15082956259*e)
      ws=0.
      go to 30
c
   20 ws=(8.5+0.15504314*rholn-e)*exp(-0.05*rholn+0.02531780798)
      ws=1./(exp(-ws)+1.)
      fo=exp(-0.22421524664*e)*ws
      fon=exp(-0.15082956259*e)*(1.-ws)
c
   30 beta=0.
      if (e .gt. 1.) beta=(6.9487e-3*ws+1.38974e-2)*alog(e)
      e2=(e-40.)*0.33333333333333
      if (abs(e2) .lt. 5.) go to 50
      if (e2 .gt. 0.) go to 40
      fn=0.
      ws=0.
      go to 60
c
   40 fn=exp(-0.039215686275*e)
      ws=1.
      go to 60
c
   50 ws=(e-exp(0.0157*rholn+3.806662489))*exp(-0.085*rholn-1.38629436)
      ws=1.0/(exp(-ws)+1.0)
      fn=exp(-0.039215686275*e)*ws
c
   60 beta=beta+ws*(0.045-beta)
      ws=(e-160.)/amax1(30.+3.474356*rholn,6.)
      fe=0.
      if (ws .gt. -5.) fe=1./(exp(-ws)+1.)
      gm=(0.161+0.255*fo+0.28*fon+0.137*fn+0.05*fe)*exp(beta*rholn)
      gmone=gm
      p=gmone*eee*rrr
      rhols=rholn*rholn
      if (e .gt. 120.) go to 111
      f=9.72e-1-2.71434e-3*rholn+6.582549e-5*rhols
      g=2.645e-2-6.418873e-4*rholn-2.338785e-5*rhols
      h=-9.21e-5+5.971549e-6*rholn+3.923123e-7*rhols
      con1=3480.*gm*e
      con2=f+g*e+h*e*e
      temp=con1/con2
      go to 222
c
```

```
  111 alr=alog10(rrr)
      c1=1.69081e-5+2.99265e-7*alr
      c2=7.69813e-1+3.8618e-3*alr
      temp=c1*eee**c2+102.6275735
c
  222 continue
      beta=(e-3.)*1.5151515151
      if (beta .gt. 10.) return
      swit=1./(exp(beta)+1.)
      temp=con1/(con2*(1.-swit)+swit)
      if (temp .gt. 0.) return
      alr=alog10(rrr)
      c1=1.69081e-5+2.99265e-7*alr
      c2=7.69813e-1+3.8618e-3*alr
      temp=c1*eee**c2+102.6275735
c
      return
      end
```

```
      subroutine matm62 (ty,wsp,cs,wsr,wst)
c
      save
c
c   This routine supplies internal specific energy and density.
c
c   This routine is part of the DNA 1-Kt Standard by Needham, et al.
c
c   calculate atmosphere
c
c   tabat(1)=r, the gas constant in ergs/mole/degree
c   tabat(2)=radius of the earth in cm.
c   tabat(3)=acceleration due to gravity at sea level in cm/sec**2
c   tabat(4)=molecular weight of air at sea level
c
c   tabz is altitude in cm.
c   tabt is molecular scale temperature in degrees kelvin.
c   tabl is molecular scale temperature gradient in deg./cm.
c   tabp is pressure in dynes/cm**2.
c
c   nz is the number of altitudes.  tabz, tabt, and tabp are
c   dimensioned nz.  tabl is dimensioned nz-1.
c
c
      dimension tabat(4), tabz(22), tabl(21), tabt(22), tabp(22)
c
      logical init
c
      data init/.false./
c
c   Data for temperate atmosphere
c
      data nz/21/,rhoz/1.22500000e-03/,tabat/8.31440000e+07,6.35670000e+
     108,9.80665000e+02,2.89644000e+01/
c
      data tabz/
     1 0.            , 1.10190000e+06, 2.00630000e+06, 3.21620000e+06,
     2 4.73500000e+06, 5.24290000e+06, 6.15910000e+06, 7.99940000e+06,
     3 9.00000000e+06, 1.00000000e+07, 1.10000000e+07, 1.20000000e+07,
     4 1.50000000e+07, 1.60000000e+07, 1.70000000e+07, 1.90000000e+07,
     5 2.20000000e+07, 3.00000000e+07, 4.00000000e+07, 5.00000000e+07,
     6 6.00000000e+07, 7.00000000e+07/
c
      data tabl/
     1-6.49291769e-05, 9.28018576e-08, 9.86255062e-06, 2.77080373e-05,
     2-1.72248474e-07,-1.96000240e-05,-3.91696897e-05, 1.60821507e-07,
     3 2.98166740e-05, 5.02020080e-05, 9.97762300e-05, 2.00108809e-04,
     4 1.49589024e-04, 1.00407490e-04, 6.97598500e-05, 6.68801467e-05,
     5 3.49035000e-05, 3.31099360e-05, 2.58868500e-05, 1.71252960e-05,
     6 1.09162420e-05/
c
      data tabt/
     1 2.88150000e+02, 2.16604540e+02, 2.16688470e+02, 2.28621170e+02,
     2 2.70704137e+02, 2.70616652e+02, 2.52659110e+02, 1.80575130e+02,
     3 1.80736048e+02, 2.10552722e+02, 2.60754730e+02, 3.60530960e+02,
     4 9.60857386e+02, 1.11044641e+03, 1.21085390e+03, 1.35037360e+03,
```

```
     5 1.55101404e+03, 1.83024204e+03, 2.16134140e+03, 2.42020990e+03,
     6 2.59146286e+03, 2.70062528e+03/
c
      data tabp/
     1 1.01325000e+06, 2.26320000e+05, 5.47486994e+04, 8.68013979e+03,
     2 1.10904998e+03, 5.90004987e+02, 1.82098959e+02, 1.03769924e+01,
     3 1.64379881e+00, 3.00749781e-01, 7.35439451e-02, 2.52169805e-02,
     4 5.06169601e-03, 3.69429709e-03, 2.79259780e-03, 1.68519867e-03,
     5 8.67381898e-04, 1.94317430e-04, 4.15743157e-05, 1.13023464e-05,
     6 3.55894454e-06, 1.22936355e-06/
c
c   Check if initialized.
c
    1 if (init) go to 2
c
c   Initialize.
c
      re=tabat(2)
      cons=re**2*tabat(3)*tabat(4)/tabat(1)
      trho=tabt(1)*rhoz*1.e3
      init=.true.
c
    2 tty=ty*100.
      j=1
      if (tty .le. 0.) go to 5
c
c   Find the altitude and index of point of interest.
c
      do 3 j=2,nz
      if (tty-tabz(j)) 4,5,3
    3 continue
c
      j=nz+1
c
    4 j=j-1
c
    5 dum2=(tabz(j)-tty)/((re+tty)*(re+tabz(j)))
      dum3=(re+tabz(j))/(re+tty)
      var1=(re+tabz(j))*tabl(j)-tabt(j)
      rvar1=1./var1
      var2=((tty-tabz(j))*tabl(j)+tabt(j))/tabt(j)
      fs=-cons*(((tabl(j)*alog(dum3*var2))*rvar1+dum2)*rvar1)
      wsp=tabp(j)*exp(fs)
      wst=tabt(j)+tabl(j)*(tty-tabz(j))
      wsr=trho*wsp/(wst*tabp(1))
      wsp=0.1*wsp
      cs=sqrt(1.4*wsp/wsr)
c
      return
      end
```

```
      function enrgyl (rho,pr)
c
      gm1=.404574
      dgamma=gm1
c
   10 e=pr/(gm1*rho)
      call air (e,rho,gmon1,p,temp)
      dp=p-pr
      dgamma=0.5*dgamma
      if (dp) 30,50,20
c
   20 gm1=gm1+dgamma
      go to 40
c
   30 gm1=gm1-dgamma
c
   40 if (abs(dp)/pr .gt. 1.e-3 .and. abs(dgamma) .gt. 1.e-3) go to 10
c
   50 enrgyl=e
c
      return
      end
```

# Index




C.E. Needham, *Blast Waves*, Shock Wave and High Pressure Phenomena,
DOI 10.1007/978-3-319-65382-2