References


12) Rogers, R.C. "Mixing of Hydrogen Injected From Multiple Injectors Normal to a Supersonic Airstream", NASA TN D-6476. 1971


51) Smith, L. Development of Renormalization Group Analysis of Turbulence, Center for Turbulence Research Annual Research Briefs, 1989


Appendix A: Coding

Given below are sample subroutines added to GASP version 2.2 to create the RNG $K-\varepsilon$ turbulence models, along with the associated subroutine call statements. The comments added to the programming should be sufficient to allow the interested reader to understand the logic used in each routine.

For routines that must be written separately for each logical direction, only the 'k' routines are given below. The other routines are similar. The call statements shown are for the variation mixing RNG, but comments explain how the other variations are called.

* * *

Copyright (c) 1994 AeroSoft, Inc.
All rights reserved. The GASP v2.2 Software Licence Agreements specify the terms and conditions for modification and redistribution.
Please do not remove this notice

#include "gasp.h"

---

Within subroutine planek (or planei or planej)...

Visck is called. Within subroutine visck (or visci or viscj)...

After all the necessary type declarations and the usual temp space allotments are made...
The following subroutines are modifications made for RNG K-epsilon by Susan K. Cox-Stouffer.  

Subroutines \texttt{derivsi}, \texttt{derivsj}, and \texttt{derivsk} differ only in the logical direction of their plane of operation, and only \texttt{derivsk} is presented.  

Subroutines \texttt{lsttrmi}, \texttt{lsttrmj}, and \texttt{lsttrmk} are similarly related, and only \texttt{lsttrmk} is presented.  

First the RNG terms in the Reynolds stress equation must be calculated.  

\begin{verbatim}
call derivsk (idim,jdim,nspec,neve,neqn,q1(1,k-1),si(1,1,k),
             sj(1,1,k),sk(1,1,k),t(idqaray),t(ik),
             t(ieps))
call fsttrm (idim,jdim,t(idqaray),t(ik),t(ieps),t(iprodtm),
             t(ijactm))
\end{verbatim}

The idea here is to calculate the first derivatives on the \(k-1\) and \(k+1\) planes for use in the higher order derivatives in \texttt{lsttrm}.  This is easier than
modifying derivs to do it all at once. Don't forget to pass all the necessary metrics.

The last two calls of derivsk and the call of lsttrmk are used only for the 'full' variation. For the others they are commented out.

call derivsk (idim,jdim,nspec,nnev,neqn,q(1,k-2),si(1,1,k-1),
  sj(1,1,k-1),sk(1,1,k-1),t(idqm1),t(itrash))
call derivsk (idim,jdim,nspec,nnev,neqn,q(1,k),si(1,1,k+1),
  sj(1,1,k+1),sk(1,1,k+1),t(idqp1),t(itrash))
call lsttrmk (idim,jdim,t(idqm1),t(idqaray),t(idqp1),t(ik),
  t(ieps),si(1,1,k-1),sj(1,1,k-1),sk(1,1,k-1),
  t(iprodtm),t(ijactm))

Before we add turbulent transport properties, we need to calculate an "effective" eddy viscosity rmutstr based on the RNG terms in the Reynolds stress.

Modmut is called only for the 'mixing' variation. Else it is commented out.

call modmut (idim,jdim,nspec,neqn,q(1,k),t(imut),t(iprodtm),
  t(idqaray),t(imutstr))

Add turbulent and laminar transport properties.

For the 'mixing' variation, addtrpt (a standard routine, not one written for RNG) is called using the new eddy viscosity, which is stored as t(imutstr). For other variations t(imutstr) is replaced by the standard eddy viscosity, which is stored as t(imut).

call addtrpt(idim,jdim,nspec,nnev,neqn,t(itrp),thp(1,1,k),
  t(imutstr),t(intermit),ivscoef,rvscoef,t(itmp))

Near the end of the routine, the RNG components to the residual and Jacobian are
added.  addjac is added just after addmsrc and
c before fillkesrc inside the Jacobian logic block.
c The Jacobian routines may need a small amount
c of additional work to make them fully
c compatible with most solution schemes
c (such as three-factor approximate factorization).

c             add source-terms to mass matrix

call addmsrc...
call addjac (ncell,nspec,nnev,neqn,rmass,q(1,k),vol(1,k),
.                   t(ijactm))

c  The subroutine address comes just before the return and
c  end at the end of the subroutine.

call address (ncell,nspec,nnev,neqn,t(iprodtm),vol(1,k),
.             q(1,k),res)

return
end

subroutine derivsk: Calculates all first derivatives
of all velocity components
on a plane

subroutine derivsk(idim,jdim,nspec,nnev,q,s1,s2,s3,
.                      dqarray,rk,eps)

INTEGER idim,jdim,nspec,nnev
REAL q(-1:idim+1,-1:jdim+1,neqn,-1:1),s1(idim,jdim-1,5),
.     s2(idim-1,jdim,5),s3(idim-1,jdim-1,5,0:1),
.     dqarray(0:idim,0:jdim,3,3),
.     rk(idim-1,jdim-1),eps(idim-1,jdim-1)

locals
INTEGER ik, ieps, i, j, m, n, ilog, jlog
REAL dqdi, dqdj, dqdk, s1avg, s2avg, s3avg

ik = nspec+nnev+5
ieps = nspec+nnev+6

do 100 j = 0, jdim
  if (j.eq.jdim) then
    jlog = j -1
  elseif (j.eq.0) then
    jlog = 1
  else
    jlog = j
  endif

do 100 i = 0, idim
  if (i.eq.idim) then
    ilog = i -1
  elseif (i.eq.0) then
    ilog = 1
  else
    ilog = i
  endif

do 200 m = 1, 3
  if (i.eq.idim) then
    ilog = i -1
  elseif (i.eq.0) then
    ilog = 1
  else
    ilog = i
  endif

  do 200 m = 1, 3
    c The metrics are stored face-centered and non-
    c dimensional. Before we can work with them we must
    c multiply by their magnitudes (found in component
    c 4) and take averages to get cell-centered values.
    c Metrics are reflected across boundaries, so we
    c can calculate derivatives on boundary cells.
    c The metrics are stored nondimensionally as one-
    c dimensional arrays (m = 1-3 below) at each point
    c (i,j) in the three-dimensional arrays s1, s2, and s3.
    c The fourth element in one-dimensional arrays (m=4)
contains the magnitude of the metrics divided by 2.0. Hence the formulation given below and in other subroutines.

\[
\begin{align*}
\text{s1avg} &= s1(ilog+1,jlog,m)*s1(ilog+1,jlog,4) + \\
&\quad s1(ilog,jlog,m)*s1(ilog,jlog,4) \\
\text{s2avg} &= s2(ilog,jlog+1,m)*s2(ilog,jlog+1,4) + \\
&\quad s2(ilog,jlog,m)*s2(ilog,jlog,4) \\
\text{s3avg} &= s3(ilog,jlog,m,1)*s3(ilog,jlog,4,1) + \\
&\quad s3(ilog,jlog,m,0)*s3(ilog,jlog,4,0)
\end{align*}
\]

do 200 n = 1,3

Central differences in each direction for each velocity component n.

\[
\begin{align*}
\text{dudi} &= 0.5*(q(i+1,j,nspec+n,0)-q(i-1,j,nspec+n,0)) \\
\text{dudj} &= 0.5*(q(i,j+1,nspec+n,0)-q(i,j-1,nspec+n,0)) \\
\text{dudk} &= 0.5*(q(i,j,nspec+n,1)-q(i,j,nspec+n,-1)) \\
\end{align*}
\]

\[
\begin{align*}
\text{dqarray}(i,j,n,m) &= \text{dudi}*\text{s1avg} \\
&+ \text{dudj}*\text{s2avg} \\
&+ \text{dudk}*\text{s3avg}
\end{align*}
\]

200 continue

Separate out K and epsilon as different arrays, to be passed into fsttrm. That's cheaper than passing the whole q array.

if (j.ne.jdim.and.i.ne.idim.and.j.ne.0.and.i.ne.0) then
  \text{rk}(i,j) = q(i,j,ik,0) \\
  \text{eps}(i,j) = q(i,j,ieps,0)
endif

100 continue

return
end
subroutine fsttrm: Calculates production and Jacobian contributions to K and eps. equation due to first-order terms in RNG K-eps. model

subroutine fsttrm(idim,jdim,dqarray,rk,eps,prodtm,rjactm)

INTEGER idim,jdim
REAL dqarray(0:idim,0:jdim,3,3),rk(idim-1,jdim-1),
. eps(idim-1,jdim-1),
. prodtm(idim-1,jdim-1,2),rjactm(idim-1,jdim-1,2,2)

locals
INTEGER i,j,ii,ij,ia,ib
REAL prod,diveps,divk

constants
REAL cn1,c2,cn2,c3,c4,ceps1,cr1,ct1,ct2,ct3

  cn1 = -1.0d0
  c2 = 2.0d0
  cn2 = -2.0d0
  c3 = 3.0d0
  c4 = 4.0d0
  ceps1 = 1.44d0
  cr1 = 25.0d-3
  ct1 = 0.034d0
  ct2 = 0.104d0
  ct3 = -0.014d0

Prod is the summing variable that collects elements of the turbulence production from each component in each term of the equation at a given point.

   do 100 j = 1,jdim-1
      do 100 i = 1,idim-1
prod = 0.0d0
diveps = 1.0/eps(i,j)
divk = 1.0/rk(i,j)

do 200 ia = 1,3
do 200 ii = 1,3
do 200 ij = 1,3

if (ii.ne.ia.or.ij.ne.ia) then
  prod = prod + ct1*dqarray(i,j,ii,ia)*dqarray(i,j,ij,ia)*dqarray(i,j,ii,ij)
endif

200 continue

prodtm(i,j,l) = turbulence production term.
If l = 1, prodtm is K-equation term.
If l = 2, prodtm is epsilon-equation term at point (i,j)

prodtm(i,j,1) = cn1*rk(i,j)**c3*diveps*diveps*prod
prodtm(i,j,2) = ceps1*eps(i,j)*divk*prodtm(i,j,1)

rjac(i,j,m,n) = Jacobian components. m = 1 => K-eq'n.
m = 2 => eps-eq'n. n = 1 => K-derivative.
n = 2 => eps-derivative.

rjactm(i,j,1,1) = c3*prodtm(i,j,1)*divk
rjactm(i,j,1,2) = cn2*prodtm(i,j,1)*diveps
rjactm(i,j,2,1) = c2*prodtm(i,j,2)*divk
rjactm(i,j,2,2) = cn1*prodtm(i,j,2)*diveps
Reseting prod to use as another temporary
summing variable like it was used before.

\[ \text{prod} = 0.0 \text{d}0 \]

There are a heck of a lot of these summations over
multiple indices, and I'm not sure how to maximize
efficiency. I'm open to suggestions.

\[
\text{do 300 } \text{ia} = 1,3 \\
\quad \text{do 300 } \text{ib} = 1,3 \\
\quad \text{do 300 } \text{ii} = 1,3 \\
\quad \text{do 300 } \text{ij} = 1,3
\]

\[ \text{prod} = \text{prod} + \text{dqarray}(\text{i},\text{j},\text{ii},\text{ia}) \times \text{dqarray}(\text{i},\text{j},\text{ij},\text{ib}) \times \text{dqarray}(\text{i},\text{j},\text{ib},\text{ia}) \times \text{dqarray}(\text{i},\text{j},\text{ii},\text{ij}) \]

\[
\text{300 continue}
\]

\[ \text{prod} = \text{cr1} \times (\text{rk}(\text{i},\text{j}) \times \text{diveps}) \times \text{c4} \times \text{eps}(\text{i},\text{j}) \times \text{prod} \]
\[ \text{prodtm}(\text{i},\text{j},1) = \text{prodtm}(\text{i},\text{j},1) + \text{prod} \]
\[ \text{prodtm}(\text{i},\text{j},2) = \text{prodtm}(\text{i},\text{j},2) + \text{ceps1} \times \text{eps}(\text{i},\text{j}) \times \text{divk} \times \text{prod} \]
\[ \text{rjactm}(\text{i},\text{j},1,1) = \text{rjactm}(\text{i},\text{j},1,1) + \text{c4} \times \text{prod} \times \text{divk} \]
\[ \text{rjactm}(\text{i},\text{j},1,2) = \text{rjactm}(\text{i},\text{j},1,2) - \text{c3} \times \text{prod} \times \text{diveps} \]
\[ \text{rjactm}(\text{i},\text{j},2,1) = \text{rjactm}(\text{i},\text{j},2,1) + \text{ceps1} \times \text{c3} \times \text{eps}(\text{i},\text{j}) \times \text{divk} \times \text{divk} \times \text{prod} \]
\[ \text{rjactm}(\text{i},\text{j},2,2) = \text{rjactm}(\text{i},\text{j},2,2) + \text{cn2} \times \text{ceps1} \times \text{prod} \times \text{divk} \]

\[
\text{100 continue}
\]

\[ \text{return} \]
\[ \text{end} \]

subroutine lsttrmk: Calculates production and Jacobian
 contributions to K and eps. equation
due to higher-order terms in RNG K-eps.
model

 subroutine lsttrmk: Calculates production and Jacobian
 contributions to K and eps. equation
due to higher-order terms in RNG K-eps.
Subroutine lsttrmk(idim,jdim,dqm1,dqarray,dqp1,rk,eps,s1,s2,s3, prodtm,rjactm)

INTEGER idim,jdim
REAL dqm1(0:idim,0:jdim,3,3),dqarray(0:idim,0:jdim,3,3),
    dqp1(0:idim,0:jdim,3,3),rk(idim-1,jdim-1),eps(idim-1,jdim-1),
    s1(idim,jdim-1,5,-1:1),s2(idim-1,jdim,5,-1:1),
    s3(idim-1,jdim-1,5,-1:1),
    prodtm(idim-1,jdim-1,2),rjactm(idim-1,jdim-1,2,2)

LOCALS
PROGRAMS
REAL prod,diveps,dudi,dudj,dudk,term(3,3),scnd(3,3,3)

CONSTANTS
REAL cr2,ceps1,c4,c5,c6,c7

THIRD DERIVATIVE VARIABLES
REAL duxsi,dueta,duzeta,duxsxs,duetet,duzeze,duxset,duxsze,
    duetze,xsixsi,etaxsi,zetxsi,xsieta,etaeta,zeteta,xsizet,
    etazet,zetzet,s1avga,s2avga,s3avga,s1avgj,s2avgj,s3avgj

CR2 = -0.0352e-3
CEPS1 = 1.44
C7 = 7.0
C6 = 6.0
C5 = 5.0
C4 = 4.0

DO 100 J = 1,jdim-1
  DO 100 I = 1,idim-1
    DIVEPS = 1.0/eps(i,j)
    SCND(ib,ij,ig) = d^2 U(ib)/dx(ij)dx(ig)
    SECOND DERIVATIVES CALCULATED AS DERIVATIVES OF
    FIRST DERIVATIVES.
c account for all directions in first derivatives

   do 200 ig = 1,3

   c account for all directions in second derivative

   do 200 ij = 1,3

       s1avg = s1(i+1,j,ij,0)*s1(i+1,j,4,0) + s1(i,j,ij,0)*s1(i,j,4,0) +
       s2avg = s2(i,j+1,ij,0)*s2(i,j+1,4,0) + s2(i,j,ij,0)*s2(i,j,4,0) +
       s3avg = s3(i,j,ij,1)*s3(i,j,4,1) + s3(i,j,ij,0)*s3(i,j,4,0)

   c account for various U components

   do 200 ib = 1,3

       dudi = 0.5*(dqarray(i+1,j,ib,ig) - dqarray(i-1,j,ib,ig))
       dudj = 0.5*(dqarray(i,j+1,ib,ig) - dqarray(i,j-1,ib,ig))
       dudk = 0.5*(dqp1(i,j,ib,ig) - dqm1(i,j,ib,ig))

       scnd(ib,ij,ig) = dudi*s1avg + dudj*s2avg + dudk*s3avg

   200 continue

c term(ij,ia) = d^2U(ib)/dx(ij)dx(ig)*d^2U(ig)/dx(ia)*dx(ib)

   do 300 ia = 1,3

   do 300 ij = 1,3

       term(ij,ia) = 0.0

   do 300 ig = 1,3
\[ \text{term}(ij,ia) = \text{term}(ij,ia) + \]
\[ \text{scnd}(ib,ij,ig) \times \text{scnd}(ig,ia,ib) \]

300 continue

c calc \( d^3 U(ig)/dx(ij)dx(ia)dx(ib) \), multiply it by

\[ dU(ib)/dx(ig) \]

and add it to term

\[ \text{do 500 ig = 1,3} \]

\[ \text{do 500 ib = 1,3} \]

c calculate various derivatives of \( dUig/dxib \)

c first derivatives of \( dUig/dxib \)

\[ duxsi = (dqarray(i+1,j,ig,ib)-dqarray(i-1,j,ig,ib)) \times 0.5 \]

\[ dueta = (dqarray(i,j+1,ig,ib)-dqarray(i,j-1,ig,ib)) \times 0.5 \]

\[ duzeta = (dqp1(i,j,ig,ib)-dqm1(i,j,ig,ib)) \times 0.5 \]

c second derivatives of \( dUig/dxib \)

\[ duxsxs = dqarray(i+1,j,ig,ib)-2.0*dqarray(i,j,ig,ib) +dqarray(i-1,j,ig,ib) \]

\[ duetet = dqarray(i,j+1,ig,ib)-2.0*dqarray(i,j,ig,ib) +dqarray(i,j-1,ig,ib) \]

\[ duzeze = dqp1(i,j,ig,ib)-2.0*dqarray(i,j,ig,ib) +dqm1(i,j,ig,ib) \]

\[ duxset = 0.25*(dqarray(i+1,j+1,ig,ib)
\]  
\[ -dqarray(i+1,j-1,ig,ib) \]

\[ -dqarray(i-1,j+1,ig,ib) \]

\[ +dqarray(i-1,j-1,ig,ib)) \]
\[
duxsze = 0.25 \times (dqp1(i+1,j,ig,ib) - dqm1(i+1,j,ig,ib) - dqp1(i-1,j,ig,ib) + dqm1(i-1,j,ig,ib))
\]
\[
duetze = 0.25 \times (dqp1(i,j+1,ig,ib) - dqm1(i,j+1,ig,ib) - dqp1(i,j-1,ig,ib) + dqm1(i,j-1,ig,ib))
\]

Now loop over \(ia\) and \(ij\), to form third derivative in x-coordinates. Note that for this third derivative, third(\(ia,ij\)) = third(\(ij,ia\)). Also multiply the third derivative by \(dUib/dxig\).

\[
\text{do 500 } ia = 1,3
\]
\[
\text{do 500 } ij = ia,3
\]

Derivatives of metrics

Metrics are stored on cell faces, so we need to take averages to get quantities at cell centers before we do derivatives. Don't have to worry about "straight" derivatives; we can difference across the cell and it works out right.

\[
xsixsi = (s1(i+1,j,ij,0) \times s1(i+1,j,4,0) - s1(i,j,ij,0) \times s1(i,j,4,0)) \times 2.0
\]
\[
etaeta = (s2(i,j+1,ij,0) \times s2(i,j+1,4,0) - s2(i,j,ij,0) \times s2(i,j,4,0)) \times 2.0
\]
\[
zetzet = (s3(i,j,ij,1) \times s3(i,j,4,1) - s3(i,j,ij,0) \times s3(i,j,4,0)) \times 2.0
\]

Reflecting across boundaries:

\[
\text{if (i.eq.idim-1) then}
\]
\[
\text{ip1 = i}
\]
\[
\text{else}
\]
\[
\text{ip1 = i+1}
\]
\[
\text{endif}
\]
\[
\text{if (i.eq.1) then}
\]
im1 = i
else
  im1 = i-1
endif

if (j.eq.jdim-1) then
  jp1 = j
else
  jp1 = j+1
endif

if (j.eq.1) then
  jm1 = j
else
  jm1 = j-1
endif

etaxsi = (s2(ip1,j,ij,0)*s2(ip1,j,4,0) +
           s2(ip1,jp1,ij,0)*s2(ip1,jp1,4,0) -
           s2(im1,j,ij,0)*s2(im1,j,4,0) -
           s2(im1,jp1,ij,0)*s2(im1,jp1,4,0)) * 0.5

zetxsi = (s3(ip1,j,ij,0)*s3(ip1,j,4,0) +
           s3(ip1,j,ij,1)*s3(ip1,j,4,1) -
           s3(im1,j,ij,0)*s3(im1,j,4,0) -
           s3(im1,j,ij,1)*s3(im1,j,4,1))*0.5

xsieta = (s1(i,jp1,ij,0)*s1(i,jp1,4,0) +
           s1(ip1,jp1,ij,0)*s1(ip1,jp1,4,0) -
           s1(i,jm1,ij,0)*s1(i,jm1,4,0) -
           s1(ip1,jm1,ij,0)*s1(ip1,jm1,4,0)) * 0.5

zeteta = (s3(i,jp1,ij,0)*s3(i,jp1,4,0) +
           s3(i,jp1,ij,1)*s3(i,jp1,4,1) -
           s3(i,jm1,ij,0)*s3(i,jm1,4,0) -
           s3(i,jm1,ij,1)*s3(i,jm1,4,1))*0.5

xsizet = (s1(i,j,ij,1)*s1(i,j,4,1) +
           s1(i,jm1,ij,1)*s1(i,jm1,4,1) -
           s1(i,j,ij,0)*s1(i,j,4,0) -
           s1(i,jm1,ij,0)*s1(i,jm1,4,0)) * 0.5

etaxsi = (s2(ip1,j,ij,0)*s2(ip1,j,4,0) +
           s2(ip1,jp1,ij,0)*s2(ip1,jp1,4,0) -
           s2(im1,j,ij,0)*s2(im1,j,4,0) -
           s2(im1,jp1,ij,0)*s2(im1,jp1,4,0)) * 0.5
\begin{align*}
\text{etazet} &= (s2(i,j,ij,1)*s2(i,j,4,1) + \\
&\quad s2(i,jp1,ij,1)*s2(i,jp1,4,1) - \\
&\quad s2(i,j,ij,-1)*s2(i,j,4,-1) - \\
&\quad s2(i,jp1,ij,1)*s2(i,jp1,4,1))*0.5 \\
\text{Averages of metrics} \\
\text{Interior points only, so no boundaries.} \\
\end{align*}

\begin{align*}
s1avgj &= s1(i+1,j,ij,0)*s1(i+1,j,4,0) + \\
&\quad s1(i,j,ij,0)*s1(i,j,4,0) \\
\text{etc.} \\
s2avgj &= s2(i,j+1,ij,0)*s2(i,j+1,4,0) + \\
&\quad s2(i,j,ij,0)*s2(i,j,4,0) \\
\text{etc.} \\
s3avgj &= s3(i,j,ij,1)*s3(i,j,4,1) + \\
&\quad s3(i,j,ij,0)*s3(i,j,4,0) \\
\text{etc.} \\
s1avga &= s1(i+1,j,ia,0)*s1(i+1,j,4,0) + \\
&\quad s1(i,j,ia,0)*s1(i,j,4,0) \\
\text{etc.} \\
s2avga &= s2(i,j+1,ia,0)*s2(i,j+1,4,0) + \\
&\quad s2(i,j,ia,0)*s2(i,j,4,0) \\
\text{etc.} \\
s3avga &= s3(i,j,ia,1)*s3(i,j,4,1) + \\
&\quad s3(i,j,ia,0)*s3(i,j,4,0) \\
\text{etc.} \\
\end{align*}

\begin{align*}
\text{Now assemble.} \\
\text{third} &= ((duxsxs*s1avgj + duxsi*xsixsi) \\
&\quad + duxset*s2avgj + dueta*etaxisi \\
&\quad + duxsze*s3avgj + duzeta*zetaxisi) \\
&\quad + (duxsxs*s1avga + duxsi*xsieta) \\
&\quad + duetet*s2avga + dueta*etaeta \\
&\quad + duetze*s3avga + duzeta*zeteta) \\
&\quad + (duxsze*s1avgj + duxsi*xsiset) \\
&\quad + duetze*s2avga + dueta*etazet \\
&\quad + duzeze*s3avga + duzeta*zetzet) \\
&\quad *s3avga ) *dqarray(i,j,ib,ig)
\end{align*}
\[
\begin{align*}
term(ij,ia) &= term(ij,ia) + \text{third} \\
\text{term(ia,ij)} &= \text{term(ia,ij)} + \text{third}
\end{align*}
\]

\textbf{500} \quad \text{continue}

c \quad \text{multiply it all together so we're ready for the}

c \quad \text{source-term and Jacobian calculations.}

c \quad \text{Prod is a dummy variable again.}

\text{prod} = 0.0

do 600 \, ia = 1,3

do 600 \, ij = 1,3

do 600 \, ii = 1,3

\quad \text{prod} = \text{prod} + (\text{term(ij,ia)}*\text{dqarray(i,j,ii,ia)}

\quad \quad + \text{term(ii,ia)}*\text{dqarray(i,j,ij,ia)})

\quad \quad \times \, \text{dqarray(i,j,ii,ij)}*cr2

\textbf{600} \quad \text{continue}

\text{prodtm(i,j,1)} = \text{prodtm(i,j,1)} + \text{prod}*(\text{rk(i,j)}*\text{diveps})

\quad \quad **5*\text{rk(i,j)}*\text{rk(i,j)}

\text{prodtm(i,j,2)} = \text{prodtm(i,j,2)} + \text{prod}*(\text{rk(i,j)}*\text{diveps})

\quad \quad **4*\text{rk(i,j)}*\text{rk(i,j)}*\text{ceps1}

\text{rjactm(i,j,1,1)} = \text{rjactm(i,j,1,1)} + \text{prod}**7*(\text{rk(i,j)}*

\quad \quad \text{diveps})**5*\text{rk(i,j)}

\text{rjactm(i,j,1,2)} = \text{rjactm(i,j,1,2)} - \text{c5}*\text{prod}**6*\text{rk(i,j)}

\text{rjactm(i,j,2,1)} = \text{rjactm(i,j,2,1)} + \text{ceps1}**6*\text{prod}**6*\text{rk(i,j)}

\text{rjactm(i,j,2,2)} = \text{rjactm(i,j,2,2)} - \text{ceps1}**5*\text{prod}**5*\text{rk(i,j)}

\textbf{100} \quad \text{continue}

\text{return}
end

subroutine addres(nlen,nspec,nnev,neqn,prodtm,vol,q,res)
  \begin{verbatim}
  INTEGER nlen,nspec,nnev,neqn
  REAL prodtm(nlen,2), vol(nlen),res(nlen,neqn),q(nlen,neqn)
  \end{verbatim}

  locals
  INTEGER nl,ik,ieps,n
  REAL rho

  ik = nspec+nnev+5
  ieps = nspec+nnev+6

  do 100 nl = 1,nlen

  So far we've calculated $P/rho$, the production term without the density. We need to calculate the density and multiply it in.

  rho = 0.0

  do 200 n = 1,nspec
    rho = rho + q(nl,n)
  200 continue

  Now add the production term to the proper residual.

  $\begin{array}{cl}
  \text{res(nl,ik)} &= \text{res(nl,ik)} - \text{vol(nl) prodtm(nl,1) rho} \\
  \text{res(nl,ieps)} &= \text{res(nl,ieps)} - \text{vol(nl) prodtm(nl,2) rho}
  \end{array}$

  100 continue

  return
end

subroutine addjac: Adds RNG contribution to jacobian

subroutine addjac(nlen,nspec,nnev,neqn,rmass,q,vol,rjactm)

INTEGER nlen,nspec,nnev,neqn
REAL rjactm(nlen,2,2),rmass(nlen,neqn,neqn),q(nlen,neqn)
REAL vol(nlen)

locals
INTEGER ispace,ne1,ne2,nl,n
REAL rho

ispacespace = nspec+nnev+4

So far we've calculated the Jacobian term without the density. We need to calculate the density and multiply it in with the Jacobian and volume.

do 200 nl = 1,nlen
   rho = 0.0
   do 200 n = 1,nspec
      rho = rho + q(nl,n)
200      continue

do 100 ne2 = 1,2
   do 100 ne1 = 1,2
       do 100 nl = 1,nlen

And add to the Jacobian.

rmass(nl,ne1+ispace,ne2+ispace) =
   .     rmass(nl,ne1+ispace,ne2+ispace) - rjactm(nl,ne1,ne2)
*rho*vol(nl)

100 continue

    return
    end

subroutine modmut: Calculates modified eddy viscosity
for use in turbulent transport
properties. (mixing variation)

subroutine modmut(idim,jdim,nspec,neqn,q,rmut,prodtm,
dqarray,rmutstr)

INTEGER idim,jdim,nspec,neqn

REAL q(-1:idim+1,-1:jdim+1,neqn),rmut(idim-1,jdim-1),
prodtm(idim-1,jdim-1),dqarray(0:idim,0:jdim,3,3),
rmutstr(idim-1,jdim-1)

    locals
    REAL div,rho,trace

    do 100 j = 1,jdim-1
       do 100 i = 1,idim-1

          rho is the total density.  div is the
denominator of the expression.  Prodtm
is the RNG turbulence production
contribution (without the density
included), and div is the denominator
of the second term in Equation 3.40.

          div = 0.0
          rho = 0.0
          do 200 n = 1,nspec
rho = rho + q(i,j,nspec)
continue

trace = dqarray(i,j,1,1) + dqarray(i,j,2,2) + 
       dqarray(i,j,3,3)

do 300 ij = 1,3
   do 300 ii = 1,3
      if (ij.ne.ii) then
         div = div + (dqarray(i,j,ii,ij) + 
                    dqarray(i,j,ij,ii))*dqarray(i,j,ii,ij)
      else
         div = div + (dqarray(i,j,ii,ij) + 
                    dqarray(i,j,ij,ii)-2.0/3.0*trace)*dqarray(i,j,ii,ij)
      endif
   300 continue
rmutstr(i,j) = rmut(i,j)
if (div.ne.0.0) then
   rmutstr(i,j) = rmut(i,j) + prodtm(i,j)*rho/div
endif

100 continue
return
end
Susan K. Cox-Stouffer

Susan Kay Cox-Stouffer was born on February 12, 1969, in Kingsport, Tennessee and grew up on her family’s small dairy farm in Fort Blackmore, Virginia, some twenty miles away. In 1986 she participated in Virginia’s Governor’s School for the Gifted, through which program she had the opportunity to intern at NASA’s Langley Research Center under Dennis Bushnell. It was Mr. Bushnell’s inspiration which solidified and focused her long-standing interest in mathematics, science, and discovery into a desire for a career in aerospace engineering. She graduated from Twin Springs High School in 1987 and enrolled in Virginia Polytechnic Institute and State University. While an undergraduate at that institution, she expanded her NASA experience by participating in the cooperative education program with Wallops Flight Facility. In 1992 she received her Bachelor of Science degree in Aerospace Engineering from VPI&SU and enrolled in the doctoral program in the same department. After completing her coursework on campus, she married Scott Stouffer in 1994 and spent roughly the next year and a half engaged in research as a part of NASA-Langley’s Hypersonics Vehicles Office in Hampton, Virginia. In 1996 she and her husband moved to Dayton, Ohio and became affiliated with Wright-Patterson Air Force Base. She is employed by Taitech, Incorporated, a small contracting firm, as a research scientist involved in hypersonic combustion research.