

MOVABLE FLUID SOURCES IN *FLOW-3D*[®]

Flow Science, Inc.
March, 2000

OVERVIEW

There are many examples of fluid flow simulation where it would be useful to have specified fluid sources located inside the computational grid. By “fluid sources” we mean a source of fluid mass and momentum. Even more useful would be a capability where the location, flow rate, and flow direction of the fluid sources could be specified.

In this Technical Note, we describe a scheme that meets these goals in the *FLOW-3D*[®] program. In the next section we describe the basic approach that was taken to add this capability and following that there are several examples illustrating the method.

MODELING APPROACH FOR GENERAL FLUID SOURCES

A recent addition to *FLOW-3D*[®] is the ability to have full momentum coupling between a continuum fluid and discrete mass particles. This addition is described in the Flow Science Technical Note FSI-99-TN50, “Particle-Fluid Coupling.” It is this capability that forms the basis of our addition for general fluid sources.

For purposes of discussion, suppose that we want to model fluid exiting the end of a pipe placed somewhere inside a computational grid. Fluid exits the end of the pipe with a specified flow rate and flow velocity, i.e., both magnitude and direction are specified. In the most general case, the velocity distribution over the cross section of the pipe exit could be non-uniform. The location of the pipe end should be arbitrary, so that moving the pipe by relocating it every time step in a transient computation would be possible.

To meet these goals, we imagine covering the end of the pipe with a set of particles. Each particle is assigned a mass source rate and a velocity corresponding to the fluid flow direction at its location. We may think of these particles as representing average values of the flow for small areas surrounding them. In this sense, summing up the particles is equivalent to a numerical integration of the flow over the cross section of the pipe. The particles simply represent the discrete elements of the integration area.

Having a set of particles covering the source cross section is what makes it possible to have different cross section shapes and different flow distributions, e.g., a Poiseuille profile, over the cross section. Of course, these particles would not be free to move like real mass particles. This limitation is easy to specify in *FLOW-3D*[®] because mass particles are assigned an integer flag that tells the solver whether or not a particle needs special treatment, e.g., to ignore them if they

are stuck on a wall. In the present case, we use this flag to skip the motion part of the particle routines but to still include their momentum effects on the fluid.

Using specially designated particles has another advantage in that the definition of a source is not explicitly dependent on the computational grid. In particular, the particles can be easily moved to any location within a grid. This fact alone makes the concept of using a distribution of particles highly attractive.

For this approach to work, we must show that a particle can be made to induce the proper mass and momentum into a computation. The mass part is easy to accomplish because *FLOW-3D*[®] already has a mass source array in its formulation. All that must be done is to add the particle mass source rates to this array, which only requires us to determine in which grid cell a particle resides.

Momentum addition is more difficult because it involves a vector quantity. However, the implicit particle-fluid coupling in *FLOW-3D*[®] solves this problem in a very nice way. To see how this happens, consider a single particle having assigned velocity \mathbf{u} . Particles are assumed to exchange momentum with a surrounding fluid through a drag expression proportional to relative velocity (or velocity squared). A discrete approximation for advancing the fluid velocity \mathbf{U} (in a control volume containing the particle) from time-step n to step $n+1$ would look like,

$$M_f \mathbf{U}^{n+1} = M_f \tilde{\mathbf{U}} + m_p \int dt k (\mathbf{u}_p^{n+1} - \mathbf{U}^{n+1})$$

The second term on the right is the negative of the change in momentum of the particle during the time step, which must be absorbed by a change in fluid momentum. Here δt is the time-step size, while M_f is the mass of the fluid in the control volume and m_p is the mass of the particle. The drag coefficient representing the interaction strength between the fluid and particle is denoted by k . A tilde over the \mathbf{U} in first term on the right side means that this velocity is the sum of the value at time-step n plus all accelerations coming from body forces, pressure gradients, etc.

Replacing the product $m_p \delta t k$ by the quantity δm , and solving for the new fluid velocity we obtain an interesting result,

$$(M_f + \delta m) \mathbf{U}^{n+1} = M_f \tilde{\mathbf{U}} + \delta m \mathbf{u}^{n+1}$$

This expression says that the velocity of the combined fluid mass and mass δm at the end of time-step $n+1$ is equal to the sum of the momenta of the initial fluid and δm quantities. This is exactly what is expected of a fluid mass source that adds mass δm in time δt .

What we want is that $\delta m = \delta t Q$, where Q is the rate of mass generated by the “source particle.” This means that the effective particle-fluid drag coefficient, k , should be equal to Q/m_p . Thus, it appears that we can use the machinery for coupling mass particles with a fluid to impart momentum to the fluid at a specified rate and with any desired velocity. We do this by setting

the particle velocity equal to the source velocity and define the particle drag coefficient in a way that gives the desired rate of momentum addition. Also, we must set the flag that prevents these “source” particles from moving, and we must add their mass source to the standard source array.

THE SOURCE MODEL

We have developed the above ideas into a general fluid source model that allows the specification of a mass and momentum source distribution over an arbitrary surface. This addition makes use of special “source” particles working with the coupled particle-fluid capabilities already in *FLOW-3D*[®].

The additions to the program consist of several modifications of existing routines. For instance, a contribution has been added to the mass source array coming from particle sources. Also, contributions to fluid momentum from the source particles have been added to the routine that does this for fluid-particle coupling. Finally, we have modified the d’Arcy-like drag contribution to the momentum equation to include a mass source contribution. This last modification is a general improvement in the momentum change arising from all types of mass sources. In the unmodified code these momentum changes were done explicitly, but now they are implicit and more consistent with the fluid momentum at the end of a time cycle.

When testing the source model as a source placed in a region that was otherwise void of fluid, it was discovered that some additional changes were needed in the program to keep the flow in the vicinity of a source from exhibiting unwanted fluctuations. Otherwise, the model was found to work quite well.

Source particles must be initialize by the user in a special subroutine that has been developed for this purpose. This routine, *parqsr.for*, has extensive comments to guide users in defining and locating sources. Several tasks are performed by the subroutine *parqsr.for*: first off, it contains a list of input data that is needed to define a source. This data is then used to initialize a source(s) during the first time cycle of computation. If the source is stationary then this subroutine is never called again. However, if the source is time dependent, then the routine destroys all source particles from the previous cycle so that the user can redefine them in a new location or with a new velocity distribution.

To aid users in establishing a source at an arbitrary location with any orientation, we have included a simple particle generator with a coordinate transformation within the *parqsr.for* routine. The assumption is that a source will be flat with a rectangular or circular cross section. A regular array of particles is defined over the cross section having its center at the origin of the grid coordinate system and with all particles in the $z=0$ plane. Then the particles are transformed by rotating the distribution successively about the x, then y, then z axes and finally translating the center to a new (x,y,z) location.

The flow direction of the source, as specified by velocity components assigned to the particles, is also transformed so the velocity should be initialize with respect to the initial particle location.

Finally, we have assumed that the user will specify a total mass flow rate, so that after particles have been generated a separate loop is performed to assign individual mass source rates to the particles such that the total rate is preserved. This assignment assumes a uniform mass flow rate over the cross section defined by the particles. We have also omitted the generation of source particles in obstacle regions since they cannot make a source contribution.

Of course, the user may introduce into *parqsr.for* any other specification for source particles that might be simpler for a particular application.

The number of particles that should be used to define a source must be enough to have at least one particle per mesh cell in the region of the source. That is, the spacing of the particles should be less than the minimum grid cell size. Also, when doing two-dimensional problems the user must remember that particle sources are inherently three dimensional. This means that the number of particles used in the third direction may be one but could be more.

USING THE MODEL

Aside from the user data required to define sources in routine *parqsr.for*, only two other input parameters must be set. One parameter in namelist XPUT is *iqsr*, which tells the program that there are sources and which type of fluid (i.e., fluid type one or two) is being generated by the source. The other parameter in namelist PARTS is *ipqsr*. This is a new parameter that is used to indicate when there are particles defining a source. There are three possible values for this parameter:

- ipqsr* = 0, no particle sources (the default)
- = 1, particle source(s) independent of time
- = 2, particle source(s) that depend on time (i.e., location and/or flow rate).

Source definitions are quite arbitrary, however, it should be remembered that the results can be no better than the resolution of the grid in which the source is placed.

Three types of sources can be defined:

1. A source consisting of an open pipe end would have a mass flow rate equal to the velocity of the flow times the pipe cross sectional area.
2. A showerhead or sparger has a flow area that is less than the cross sectional area, so the product of its flow velocity times total cross sectional area would be greater than the net mass flow rate. This type of source will generate a flow having a fluid fraction less than unity.
3. Finally, a source could have a mass flow rate that is greater than the product of flow velocity times cross sectional area. In this case the flow would ooze out and generate an expanding external flow to insure the incompressibility of the fluid.

SAMPLE TEST PROBLEMS

Several simple problems have been used to test the new source capability. For expediency most of the tests are two dimensional, but this does not affect conclusions drawn from the tests since the model is entirely three dimensional in its construction. Unless otherwise specified, there are no gravity or surface tension forces in these examples.

Uniform Downward Flow from Square Source

A square source region was initialized having a unit edge length. Ten particles were used in each direction for a total of 100 particles. Strictly speaking, only one particle is needed in the y direction because we are only considering a two-dimensional problem with variations in the x and z directions.

The velocity was normal to the plane of particles with a unit magnitude and the mass generation rate was equal to the product of the area times velocity (i.e., unity). A translation was used to move the source to $x=1.0$ and $z=3.0$. Figure 1a,b show the early flow development (a) and flow at a later time (b). Swelling of the front of fluid initially generated at the source, see bottom of jet in Fig. 1a, occurs because it takes a little time before the source velocity builds up to the proper value. The spread also occurs because fluid fraction advection at the front tends to flatten and broaden out the front a little.

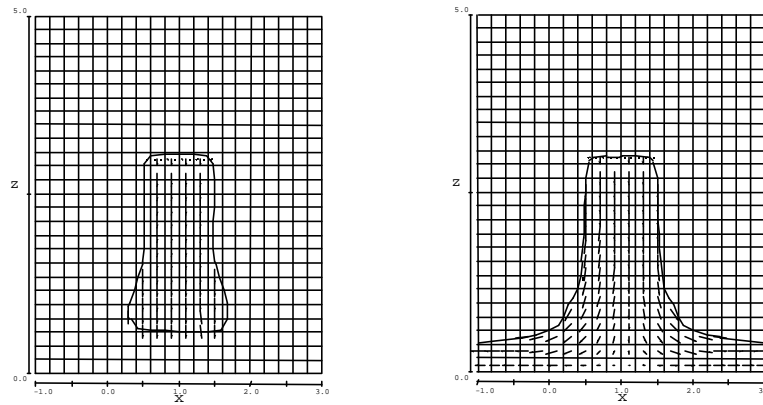


Figure 1. (a) Initial flow from horizontal source (left plot).
(b) Established flow from source spreading out on a horizontal plate (right plot).

Jet Source Tilted at 45 Degrees

As a check of how well the new source model might work when the flow is not aligned with the grid, we initialized the source at a 45° angle in the grid. Figure 2 shows that the source continues to work well in this configuration.

One effect we have observed is a recorded fluid volume error (loss) on the order of 10% or larger. This loss error seems excessive but is amplified in the test case at early times because the loss is reported as a percent error and there is very little fluid at early times. Also, fluid in the test problems is leaving the grid. When the loss is compared to the total amount generated it is quite small.

The origin of the volume loss appears to be an over filling of one or more grid cells containing source particles. It may be that there is some conflict between the requirement of a zero velocity divergence for incompressibility and a specified velocity corresponding to the source.

Tightening the convergence tolerance does not appear to have much of an effect. A reduced time-step size, however, reduces the size of the error. This phenomenon will be investigated in more detail, but it does not seem significant enough to delay the release of the new particle-source model.

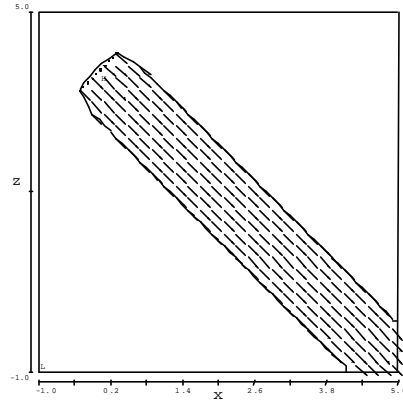


Figure 2. Jet from 45° inclined source.

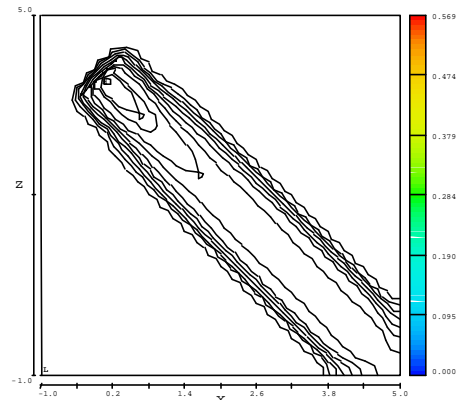


Figure 3. Jet from 45° inclined source as in Figure 2, but with twice the source velocity. Contours are of fluid fraction.

Jet with Excessive Mass Flow Rate

When the mass flow rate of the preceding example is larger than what the source velocity can remove, then fluid mass is pushed upward and outward from the source. This can be seen in Fig.4 for a case in which we left the mass rate unchanged but reduced the source velocity in half.

We see that the jet is somewhat wider because fluid has been forced outward from the ends of the source. Even more fluid is being pushed upward away from the source.

Angled Jet with Gravity Effect

Returning to the standard case corresponding to an open pipe end but adding the effect of a gravitational body force produces the result shown in Fig.5. The source flow is turned downward and an acceleration of the flow velocity causes a tapering of the jet.

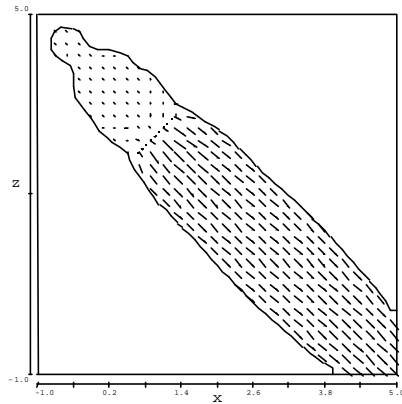


Figure 4. Result of source having an excessive mass flow rate compared to its velocity.

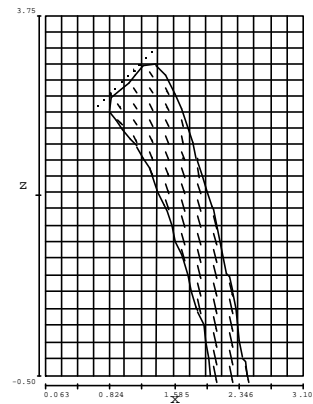


Figure 5. Jet source with effects of a gravitational body force.

Submerged Jet Source

As an example of a source that is surrounded by fluid rather than placed in a void region, we initialized the source in a pool of fluid with gravity acting downward. The source is again horizontal but has its velocity directed in the upward vertical direction, toward the free surface of the liquid pool. Figure 6 shows the pool to be filling as evidenced by the positive normal velocity components all across the surface of the pool. There is also a symmetric eddy structure in the flow beneath the surface. The source is clearly producing fluid volume because there is a jump in vertical velocity magnitude across the source.

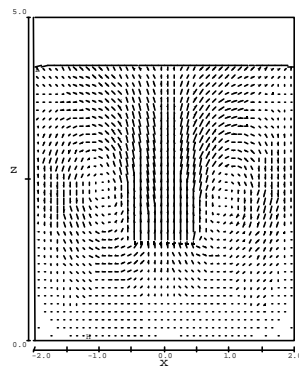


Figure 6. Example of submerged jet source in a pool of liquid.

Moving Jet Source

As an example of a moving source, the source particles were specified to move on the arc of a circle with a constant speed. Gravity was directed downward. In this case the source velocity should include not only the velocity of the fluid directed out the end of the source pipe but also the velocity of the source itself since fluid particles in the pipe are constrained to move with it.

A copy of the subroutine *parqsr.for* that was used to initialize and move the source is attached to this report as Appendix A. Several dummy input parameters were used so that the source could be changed through input data. We have also made use of the built-in coordinate transformation in routine *parqsr.for*.

The results of this simulation are shown in Fig. 7 in the form of four snapshots of the flow. The last two frames have source locations symmetric to those in the first two frames. We see some fluctuations along the surface of the jet, depending where it is located along the moving path. These fluctuations cannot be completely avoided because of the way in which source particles lay in the grid. The number of source particles in grid cells at the edges of the source must vary in number. This introduces a variable source amount and consequently some fluctuations in the surface of the source flow.

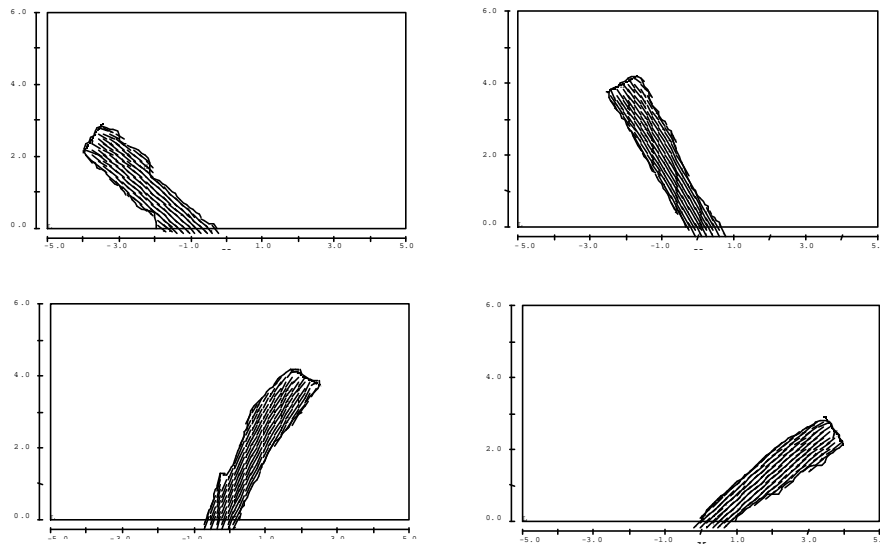


Figure 7. Source moving along a circular arc from left to right.

Three-Dimensional Source

Finally, a circular source was set up as an example of a fully three-dimensional problem. The source plane was made vertical with its velocity directed in the x direction. Gravity was downwards. The source was stationary. Results at the end of the simulation are shown in Fig.8.

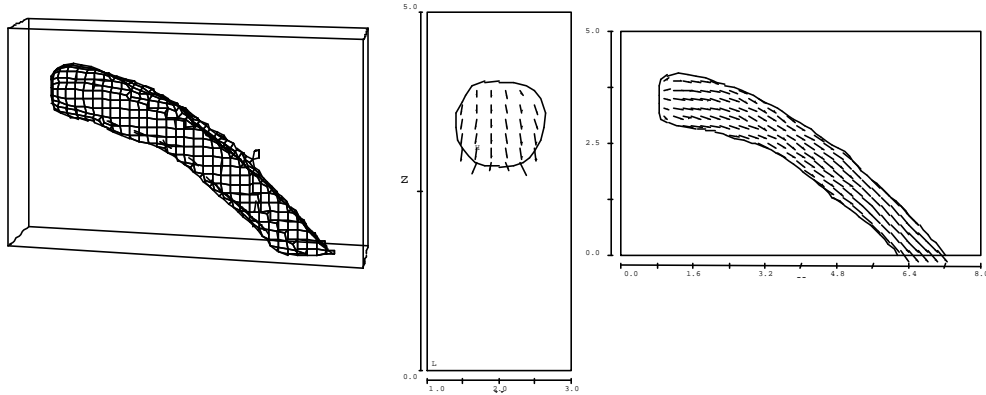


Figure 8. Example of a three-dimensional circular jet source.

SUMMARY COMMENT

A new type of mass and momentum source for inclusion in **FLOW-3D**[®] has been described and illustrated. The new model employs particles to define the cross section of the source and to assign mass rates and velocities at each location in the cross section.

The advantage of using particles to define a source is that it can be placed anywhere in a grid, and can even be moved through a grid, without having to worry about grid cell sizes (except for the question of resolution accuracy). Particles automatically serve as a type of discrete integration mechanism that allows the source to be spread over one or more grid cells.

A new subroutine called *parqsr.for* has been added to the program that allows users to customize sources for their individual needs.

APPENDICES

A. Source program for subroutine *parqsr.for*. This routine is used to initialize source particles and/or define moving sources. Data in the routine is for the moving source example described in the sample problems.

B. Prepin.inp file for the moving source example problem.

```

subroutine parqsr
C
C *****
C **          notice          **
C **  this subprogram contains flow science, inc. proprietary  **
C **    trade secret and confidential information.            **
C **          **
C **          unauthorized use prohibited                    **
C **    copyright 1985-1999 flow science, inc.                **
C *****
C
C use arrays_module
C
C use meshcb_module
C
C use avgco_module
C
C include '..\comdeck\precis.for'
C include '..\comdeck\params.for'
C include '..\comdeck\phiou.for'
C include '..\comdeck\arrayp.for'
C include '..\comdeck\comcom.for'
C include '..\comdeck\nocomp.for'
C include '..\comdeck\cntrl.for'
C include '..\comdeck\const.for'
C include '..\comdeck\diag.for'
C include '..\comdeck\edit.for'
C include '..\comdeck\grfdat.for'
C include '..\comdeck\index.for'
C include '..\comdeck\state.for'
C include '..\comdeck\acctrm.for'
C include '..\comdeck\pardat.for'
C include '..\comdeck\scala.for'
C include '..\comdeck\dumn.for'
C integer iexcd
C save iexcd
C include '..\comdeck\func.for'
C data iexcd /0/
C
C * * Define Particle Mass & Momentum Sources
C
C   itnps = current number of particles
C   istore = maximum number of particles possible (a parameter)
C   ipcmp(i)= 0 indicates non-moving (fixed) particle i
C             = 1 moving mass or marker particle i
C             = 2 mass & momentum source particle i
C   psize(i)= diameter of particle i if ipflag=1 (only used in
C             inertia model)
C             = density of particle i if ipflag=2 (only used in
C             inertia model)
C             = mass source rate for source particle i (ipcmp(i)=2)
C   xp(i),yp(i),zp(i) are particle coordinates, where integer part of
C                   coordinate is cell index and fractional part is
C                   fraction of cell width where particle is located.
C   up(i),vp(i),zp(i) are particle i velocities
C   dum1,dum2,...,dum9 = dummy variables that may be used for
C                   defining sources. Set them in XPUT,
C ===== USER SPECIFIED DATA =====
C
C * * * total mass flow rate per unit time
C   tmrate=dum1
C * * * number of particles and size of source
C   naddx=20
C   naddy=1
C   xladd=-0.5
C   xhadd=0.5
C   yladd=-0.5
C   yhadd=0.5
C * * * velocity of source flow
C   thsor=(20+dum3*t)*0.0174533
C   upin=dum2*fcos(thsor)+dum4*dum3*0.0174533*fcos(thsor)
C   vpin=0.0
C   wpin=-dum2*fsin(thsor)+dum4*dum3*0.0174533*fsin(thsor)
C * * * transformation rotations (degrees) and translations to locate source
C   protx=zero
C   proty=-70.0+dum3*t
C   protz=zero
C   ptrnx=-dum4*fcos(thsor)
C   ptrny=zero

```

```

      ptrnz=dum4*fsin(thscr)
c ===== END OF USER DATA =====
c
      if(itnps.gt.ispart) go to 5000
c
c ----- For moving sources, previous source particles must first be removed
      if(ipqsr.eq.2 .and. cycle.gt.1) then
        nppk=0
        kn=1
        do 100 k1=1,itnps
          if(ipcmp(k1).ne.2) then
            xp(kn)=xp(k1)
            yp(kn)=yp(k1)
            zp(kn)=zp(k1)
            up(kn)=up(k1)
            vp(kn)=vp(k1)
            wp(kn)=wp(k1)
            psize(kn)=psize(k1)
            ipcmp(kn)=ipcmp(k1)
            if(k1.le.nmvhis) then
              do 10 kkl=1,nmvhis
                if(ihsprt(kkl).eq.k1) then
                  ihsprt(kkl)=kn
                  goto 20
                endif
              10 continue
            endif
            20 continue
            kn=kn+1
            nppk=nppk+1
          endif
        100 continue
        itnps=nppk
      endif
c
c ----- Define initial mass & momentum source particles -----
c
      if(cycle.gt.1 .and. ipqsr.ne.2) return
c
c * * * Initialize source particles (every time step if time dependent)
c * * * Add particles about origin with z=0. Use regular array with uniform
c spacing covering xladd,xhadd,yladd,yhadd. This gives square and slot
c shaped source cross sections. Particles could be restricted to a circle,
c annulus, or other shape within the rectangular region.
c      naddx=(xhadd-xladd)/xsize
c      naddy=(yhadd-yladd)/ysize
c
c * * * User must set values for position xin,yin,zin, and flow velocity up,vp,wp
c
      ntot=0
      xsize=(xhadd-xladd)/naddx
      ysize=(yhadd-yladd)/naddy
c * * * Change degrees to radians
      protx=protx*0.0174533
      proty=proty*0.0174533
      protz=protz*0.0174533
      do 1000 ky=1,naddy
        do 1000 kx=1,naddx
          xin=xladd+(kx-0.5)*xsize
          yin=yladd+(ky-0.5)*ysize
          istore=itnps+1
          ipcmp(istore)=2
c * * * Include mask if desired. For example, for a circular source of
c radius R only process particle locations in the circle, i.e.,
c      if(xin*xin-yin*yin.gt.R*R) goto 1000
          zin=zero
          up(istore)=upin
          vp(istore)=vpin
          wp(istore)=wpin
c ----- Move particles to desired location.
c * * * A Cartesian coordinate transformation is used to relocate initial particle plane
c to its desired location and orientation. The transformation consists of three
c rotations, first protx degrees about the x axis, then proty about y axis and finally
c protz about the z axis. This is followed by a translation, ptrnx,ptrny,ptrnz
c of the center (origin) to its new location. The transformation assumes Cartesian
c coordinates.
c
c ----- User must input rotations and translations ptrn* -----
c
c * * * Compute sines/cosines

```

```

      csnx=fsin(protx)
      ccnx=fcos(protx)
      csny=fsin(protoy)
      ccny=fcos(protoy)
      csnz=fsin(protz)
      ccnz=fcos(protz)
c
c * * * Transform particle to desired location for source
      if(fabs(protx).gt.ztest) then
          yint=yin
          vin=yin*ccnx-zin*csnx
          zin=yint*csnx+zin*ccnx
      endif
      if(fabs(protoy).gt.ztest) then
          xint=xin
          xin=zin*csny+xint*ccny
          zin=zin*ccny-xint*csny
      endif
      if(fabs(protz).gt.ztest) then
          xint=xin
          xin=xin*ccnz-yin*csnz
          yin=xint*csnz+yin*ccnz
      endif
      xia=xin+ptrnx
      yia=yin+ptrny
      zia=zin+ptrnz
c
c ----- Particle locations in the program are stored in integer fraction
c          format. For example, the integer part of location xp(k) is the cell
c          number in the x direction and the fractional part of xp(k) locates the
c          particle at that fraction of delx from the left cell edge.
c ----- Convert coordinates xia,yin,zin to relative cell location.
      do 200 i=2,iml
          if(xin.lt.x(i-1) .or. xin.ge.x(i)) go to 200
          xw=xin-x(i-1)
          xp(istore)=float(i)+xw*rdx(i)
          go to 250
200    continue
          goto 1000
250    continue
          do 300 j=2,jml
              if(yin.lt.y(j-1) .or. yin.ge.y(j)) go to 300
              ylt=yin-y(j-1)
              yp(istore)=float(j)+ylt*rdy(j)
              go to 350
300    continue
              goto 1000
350    continue
              do 400 k=2,kml
                  if(zin.lt.z(k-1) .or. zin.ge.z(k)) go to 400
                  zh=zin-z(k-1)
                  zp(istore)=float(k)+zh*rdz(k)
                  go to 450
400    continue
450    continue
c
c * * * Omit particles in obstacles
      include '..\comdeck\ijk.for'
      if(vf(ijk).lt.em6) goto 1000
      ntot=ntot+1
      itnps=itnps+1
      if(itnps.eq.ispart) then
          if(iexcd.eq.0) then
              iexcd=iexcd+1
              write(ierrfl,5100) itnps,cycle
              write(iout,5100) itnps,cycle
          endif
          goto 2000
      endif
1000 continue
c
c ----- Compute and set source rates for particles to conform to a specified --
c          total mass source rate, tmrate.
      if (ntot.gt.0) then
          pmrate=tmrate/ntot
          kls=itnps-ntot+1
          do 1050 k1=kls,itnps
              psize(k1)=pmrate
1050    continue
          ntot=0

```

```

endif
c
c ===== ADD MORE SOURCES ? =====
c      Could add more sources by repeating above loops with new data.
c -----
c
2000 continue
c
c ----- error messages and abort
c
5000 continue
c
5100 format(/,2x,' **** particle warning ****',/,
1 4x,' number of particles equal to maximum (ispart) - ',
2 i5,', at cycle = ',i6,' -',/,
3 4x,' no more SOURCE PARTICLES can be generated',
4   ' until total number of particles decreases',/,
5   2x,' *****',/)
c
return
end

```

Sample problem for moving source

\$xput

```
remark='units are cgs',
twfin=28.0,
itb=1,
gz=-2.0, ifvis=0, epsadj=1.0,
iqsr=1,
remark='dummy parameters for particle source definition',
dum1=4.0, dum2=4.0, dum3=5.0,
dum4=4.5,
```

APPENDIX B

\$end

\$limits

\$end

\$props

```
rhof=1.0,
mul=0.01, units='cgs',
```

\$end

\$scalar

\$end

\$bcdata

```
wl=3, wr=3,
wb=5, fbc(5)=0.0, pbc(5)=0.0,
```

\$end

\$mesh

```
nxcelt=60, px(1)=-5.0, px(2)=5.0,
nycelt=1, py(1)=-0.5, py(2)=0.5,
nzcelt=40, pz(1)=0.0, pz(2)=6.0,
```

\$end

\$obs

```
avrck=-3.1,
```

\$end

\$fl

\$end

\$bf

```
nbafs=0,
bz(1)=4.0, bxl(1)=1.0, bxh(1)=3.0,
```

\$end

\$temp

\$end

\$motn

\$end

\$gratic

\$end

\$parts

```
ipqsr=2,
```

\$end

Documentation: