```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! Program to solve pipe network problems. Data input may include network
! equations as an alternative to 'number lists' specifying pipe/valve
! characteristics and pressure/flow conditions at nodes.
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
begin
!
!-------------------------------------------------------------------------
externalroutinespec pressures(longrealarrayname a,b,p,f,fo,c
integer nn,nf,y,integerarrayname pset,qset)
!
externalroutinespec set up a(longrealarrayname k,kb,p,tp,fn,ncap,c
a,b,den,ht,realarrayname qrterms,integerarrayname qterms,qtctr,c
u,d,pfix,longreal delta,tcon,integer qlctr,nn,nf,string(20) linmeth)
!
externalroutinespec nlink(integer nnodes,nlinks,c
integerarrayname in,out,ncode,link,cc,cp,longrealarrayname pp,fexx)
!
externalroutinespec fcheck(longrealarrayname q,c
fexx,integerarrayname pfix,link,in,cp,c
integer nn,longreal ftol,longrealname hftot,integername hfnod,check)
!
externalroutinespec flows(longrealarrayname p,kv,k,kb,f,fo,l,da,c
rk,ft,denav,den,ht,vis,cd,cv,cl,c2,temp,realarrayname qrterms,c
integerarrayname qtctr,qterms,ipbr,u,d,tlink,pform,integer c
pass,printit,nf,nn,npump,nfluid,string(20) linmeth)
!
externalroutinespec idenst(longrealarrayname cd,longrealname c
dens,longreal press,temp,integer nfluid)
!
externalroutinespec idfit(longrealarrayname d,p,t,longrealname c
cd1,cd2,cd3,integer nfluid)
!
externalroutinespec ipmpnet(longrealarrayname pchar,cl,c2,c
integerarrayname npts,pform,integer npump)
!
externalroutinespec ivfit(longrealarrayname v,p,t,longrealname c
cv1,cv2,cv3,integer nfluid)
!
externalroutinespec ivisco(longrealarrayname cv,longrealname c
visc,longreal press,temp,integer nfluid)
!
externalroutinespec ilnpump(longreal p1,p2,cl,c2,den,c
integer pform,n,longrealname q,a,b)
!
externalroutinespec setupm(longrealarrayname p,fnum,qn,k,kv,kb,a,rhs,c
realarrayname qrterms,rvalue,integerarrayname qterms,qtctr,pset,qset,c
pfix,ffix,qfix,in,out,tlink,ivalue,itype,integername qlctr,ierror,mrows,
integer pass,nn,nf,sum)
!
externalroutinespec set up rm(longrealarrayname ppi,p,k,kb,f,c
fn,qn,a,b,integerarrayname node,pset,qset,pfix,ffix,qfix,in,out,c
integer nn,nf,integername sum,mrows)
!
externalroutinespec getdata(longrealarrayname p,c
kv,l,da,rk,ft,temp,fn,ht,ndtp,tpres,tvisc,tden,ttemp,pchar,mu,ncap,sfp,c
```

```
integerarrayname node,ltno,in,out,ffix,pfix,npts,ipbr,tlink,ncode,c
itype,ivalue,u,d,realarrayname rvalue,integername c
nn,nf,npump,nfluid,ierror,longrealname ptot,rav,stringname filename)
!
externalroutinespec set up k(longrealarrayname f,fo,flst,k,kb,kv,p,plst,c
l,da,denav,temp,vis,ht,ft,rk,cv,cd,cl,c2,mu,sfp,integerarrayname c
ltno,in,out,tlink,pform,ipbr,integer nf,npump,c
nfluid,pass,string(20) linmeth)
!
externalroutinespec emas3prompt(stringname s)
!
externalroutinespec emas3(stringname comm,parms,c
integername flag)
!
!-------------------------------------------------------------------------
!      ***** <<<<<   MAIN PROGRAM   >>>>> *****
!
!  main arrays...
!   a - matrix for linearised equations
!   b - constant vector for  ..   ..
!   p - new pressures to be calculated
!   po- last pressures
!   kv - valve consts for flow=kv*sqrt(delta p)
!   k - linearised valve constants
!   f - new flows
!   fo - last flows
!   fn - node specified flows
!
!
!  'structure' arrays...
!   u(i) - number of node upstream on branch i
!   d(i) -   ..     ..   downstream ..  ..
!   pfix(i) - is 1 if pressure at node i is fixed specification,
!             0 if variable
!   ffix(i) - is 1 if flow at node i is fixed specification,
!             0 if variable
!              N.B. flow into node is +ve, out of node is -ve
!
longrealarray a(1:100,1:100),p,b,fo,f(1:100),po,ppi,ncap,tp,ht,c
plst,flst,l,da,rk,ft,qn,fn,den,nodtemp,temp,mu,sfp(1:40)
longrealarray k,kv,kb,denav,vis(1:40)
longrealarray tpres,tvisc,tden,ttemp(1:3),pchar(1:10,1:10)
longrealarray cl,c2(1:10)
longrealarray cv,cd(1:3)
realarray rvalue(1:40,1:10),qrts(1:10,1:5)
integerarray in,out,ffix,qfix,pfix,tlink,ncode(1:40),npts,ipbr(1:10)
integerarray itype,ivalue(1:40,1:10),node(1:40)
integerarray u,d(1:40)
integerarray cp,cc(1:40,1:6),link(1:40)
integerarray ltno,pset(1:40),qset(1:40)
integerarray pbr,pform(1:10)
integerarray qts(1:10,1:5),qtcount(1:10)
integer nn,nf,npump,ierror,i,j,mv
longreal ptot,rav,time,hftot,ftol,delta,tcon
integer hfnod,check,rcheck,qlcount
integer nfluid{-ve for gas, 0 or +ve for liquid}
integer ll, mm, ntotal, tc, y,HH,zz,mcc,qq,ks
integer sum, mrows, pass, eflag, pcount, zw
string(20) filename
string(40) outfile
```

```
!
ownstring (20) linmeth="newton" {initial solution method}
!
ftol=0.000001
!
!-------------------------------------------------------------------------
! CHECK P : This routine checks pressures for convergence
!           and returns 0 if sum of absolute changes is less
!           than specified limit. Also updates po().
!
integerfunction check p(longrealarrayname p,po,integer nn)
!
! in... p(),po(),nn
! out.. po()
integer i
longreal sum
sum=0
for i=1,1,nn cycle
  sum=sum+mod(p(i)-po(i))
  po(i)=p(i)
repeat
if sum<0.1 then result=0
printstring("press Error = ") ; printfl(sum,7) ; newline
result=1
!
end
!-------------------------------------------------------------------------
!
! initialise values which will be returned by the parser routine
for i=1,1,40 cycle
qset(i)=0
ppi(i)=0
for j=1,1,10 cycle
itype(i,j)=0 ; ivalue(i,j)=0
rvalue(i,j)=0
repeat
repeat
qtcount(i)=0 for i=1,1,10
for i=1,1,10 cycle
  for j=1,1,5 cycle
    qts(i,j)=0
    qrts(i,j)=0
  repeat
repeat
!
!
! Initialise pump parameter values
for i=1,1,10 cycle
pform(i)=0
c1(i)=0 ; c2(i)=0
repeat
!
qfix(i)=0 for i=1,1,40
!
! Get input data file
!
filename="name of file : "
emas3prompt(filename)
readstring(filename)
emas3("define","2,.out",eflag)
```

```
outfile="name of output file : "
emas3prompt(outfile)
! Get name of output file for results
!
readstring(outfile)
emas3("define","11,".outfile,eflag)
!
!set value of delta and tcon
tcon=0.0
delta=0.0
!
!initialise values of ncap (the capacity of each node in m**3)
!
ncap(zz)=0.0 for zz=1,1,40
!
! Call routine to read network data (and parse network equations if prese
getdata(p,kv,l,da,rk,ft,temp,fn,ht,nodtemp,tpres,tvisc,tden,c
ttemp,pchar,mu,ncap,sfp,node,ltno,in,out,ffix,pfix,npts,ipbr,tlink,c
ncode,itype,ivalue,u,d,rvalue,c
nn,nf,npump,nfluid,ierror,ptot,rav,filename)
!
!set value of tp
tp(zz)=0.0 for zz=1,1,nn
!
! assign values to elements of array ppi for nodes whose p's are fixed
!
for i=1,1,nn cycle
if pfix(i)=1 then ppi(i)=p(i)
repeat
!
! get no. of links to each node
!
nlink(nn,nf,in,out,ncode,link,cc,cp,p,fn)
!
! assign average node pressure (returned in 'ptot' by routine getdata)
! to nodes which have not been assigned inititial (fixed) pressures
!
      for y=1,1,nn cycle
      if p(y)=0 then start
      p(y)=ptot
      po(y)=ptot
      ppi(y)=ptot
      finish else start
      ppi(y)=p(y)
      po(y)=p(y)
      finish
      repeat
!
!     assign initial values to hfnod and hftot
!     hfnod is the node identifier of the node with the highest excess
!     inflow/outflow after each iteration
!     hftot is the value of the excess inflow/outflow.
!
      hfnod=0
      hftot=0
!
selectinput(0)
!
!get pump characteristics if there are pumps in the network
!
```

```
if npump>0 then start
ipmpnet(pchar,c1,c2,npts,pform,npump)
finish
!
!get viscosity and density fit details
        ivfit(tvisc,tpres,ttemp,cv(1),cv(2),cv(3),nfluid)
        idfit(tden,tpres,ttemp,cd(1),cd(2),cd(3),nfluid)
!
! initialise the constants in the pipe flow/pressure equations
!
for mv=1,1,nf cycle
k(mv)=0.0 ; kb(mv)=0.0
repeat
!
! call routine flows to get initial flow distribution in network
!
flows(p,kv,k,kb,f,fo,l,da,rk,ft,denav,den,ht,vis,cd,cv,cl,c2,temp,qrts,c
qtcount,qts,ipbr,u,d,tlink,pform,0,0,nf,nn,npump,nfluid,linmeth)
!
selectinput(0)
!
ntotal=nn+nf
!
for i=1,1,100 cycle ;! ---------------- start iteration ----------------
pass=1
if i=1 then start
pass=0
finish
!
! Get current pressure and flow values for input to routine set up k
!
plst(zw)=p(zw) for zw=1,1,nn
flst(zw)=f(zw) for zw=1,1,nf
!
set up k(f,fo,flst,k,kb,kv,p,plst,l,da,denav,temp,vis,ht,ft,rk,c
cv,cd,cl,c2,mu,sfp,ltno,u,d,tlink,pform,ipbr,nf,npump,nfluid,pass,linmeth)
!
set up rm(ppi,p,k,kb,f,c
fn,qn,a,b,node,pset,qset,pfix,ffix,qfix,u,d,nn,nf,sum,mrows)
!
! first time round call setupm to insert prespecified network equations ir
! full matrix of flow network equations
!
if pass=0 then setupm(ppi,fn,qn,k,kv,kb,a,b,qrts,rvalue,qts,qtcount,c
pset,qset,pfix,ffix,qfix,u,d,tlink,ivalue,itype,qlcount,ierror,mrows,c
pass,nn,nf,sum)
!
if i=1 then start
p(zz)=ppi(zz) for zz=1,1,nn
po(zz)=ppi(zz) for zz=1,1,nn
finish
!
selectoutput(2)
!
set up a(k,kb,p,tp,fn,ncap,a,b,denav,ht,qrts,qts,qtcount,u,d,pfix,c
delta,tcon,qlcount,nn,nf,linmeth)
!
pressures(a,b,p,f,fo,nn,nf,0,pset,qset)
!
flows(p,kv,k,kb,f,fo,l,da,rk,ft,denav,den,ht,vis,cd,cv,cl,c2,temp,qrts,c
```

```
qtcount,qts,ipbr,u,d,tlink,pform,pass,0,nf,nn,npump,nfluid,linmeth)
!     check for convergence
! check for flow convergence
!
fcheck(f,fn,pfix,link,u,cp,nn,ftol,hftot,hfnod,check)
!
newline; printstring("error = "); print(hftot,3,8)
printstring(" at node "); write(hfnod,3)
if (check = 0 and i>2) or i>40 then exit
!
repeat ;!              ---------------  next iteration ----------------
!
newline ; write(i,3)
flows(p,kv,k,kb,f,fo,l,da,rk,ft,denav,den,ht,vis,cd,cv,cl,c2,temp,qrts,c
qtcount,qts,ipbr,u,d,tlink,pform,l,l,nf,nn,npump,nfluid,linmeth)
!
selectoutput(2)
printstring("
  Used ") ; write(i,4) ; printstring(" iterations")
!
closestream(2)
selectoutput(11)
flows(p,kv,k,kb,f,fo,l,da,rk,ft,denav,den,ht,vis,cd,cv,cl,c2,temp,qrts,c
qtcount,qts,ipbr,u,d,tlink,pform,l,l,nf,nn,npump,nfluid,linmeth)
closestream(11)
!
endofprogram
```

```
!
!-----------------------------------------------------------------------
!
!  routine getdata : reads in data for a pipe network,presented as a list
!  pipe/pump/valve physical characteristics.
!
!  read in...
!  fixed pressures (pfix)
!  branches and associated kv's or piping data (u,d,kv,l,da,rk,ft,temp)
!  and set nn and nf
!
!-----------------------------------------------------------------------
!
externalroutine getdata(longrealarrayname p,kv,l,da,rk,ft,temp,fn,c
ht,nodtemp,tpres,tvisc,tden,ttemp,pchar,mu,ncap,sfp,c
integerarrayname node,ltno,in,out,ffix,pfix,npts,ipbr,tlink,ncode,c
itype,ivalue,u,d,realarrayname rvalue,integername c
nn,nf,npump,nfluid,ierror,longrealname ptot,rav,stringname filename)
!
externalroutinespec inoden(integer label,nnodes,integername index,c
integerarrayname node)
externalroutinespec indlis(integerarrayname lindex,node,instr,outstr,c
in,out,integername nnodes,npipes)
externalroutinespec eqparse(longrealarrayname kv,integerarrayname node,c
in,out,tlink,itype,ivalue,realarrayname rvalue,integername nn,nf,ierror)
externalroutinespec emas3(stringname comm,parms,integername flag)
!
integer i,f1,f2,f3,nnum,ii,jj,pcount,srflag,eflag,nnodes,qq,nlab
real nspec, pspec
integerarray uu,dd(1:40)
integerarray lindex,dlindex(1:80)
!
      ptot=0 {initialise sum of pressures}
      pcount=0 {initialise counter}
for i=1,1,40 cycle
      pfix(i)=0
      ffix(i)=0
      p(i)=0
      fn(i)=0
      ht(i)=0
      tlink(i)=-99
      ltno(i)=0
      mu(i)=0
      ncap(i)=0
      sfp(i)=0
!
repeat
!
!Enter branch details
!upstream node no., downstream node no.,
!if full piping details supplied answer > 0,
!if only constant is supplied answer 0 for non-linear, -1 for linear const
!if equation to be read in for this link later, enter <-1
!terminate with < 0
nnodes=0
nf=0
!
emas3("define","12,".filename,eflag)
selectinput(12)
```

```
for i=1,1,40 cycle
read(f1)
if f1<0 then exit
read(f2) ; if f2<0 thenexit
read(f3) ;
u(i)=(f1) ; d(i)=(f2)
uu(i)=u(i) ; dd(i)=d(i)
nf=nf+1
   if f3>0 then start
!signify that the link is either a pipe with full piping details
!supplied or it is a pump (the value of tlink is changed later
!in this routine if the link is a pump)
      tlink(i)=1
!read length (assumed to be in meters)
read(l(i))
!read pipe diameter (assumed to be in mm), and convert it to m
read(da(i)); da(i)=da(i)*1@-3
!read roughness ratio
read(rk(i))
!read fittings loss
read(ft(i))
!read temperature and convert it from celcius to kelvin
      read(temp(i)); temp(i)=temp(i)+273.0
!
   finish else if f3=0 or f3=-1 then start
!
!signify that link is a valve (valve const. supplied)
!
      tlink(i)=f3; {tlink=-1 signifies linear k, tlink=0 means non-linear
read(kv(i)); read(mu(i)) ; read(ltno(i)) ; read(sfp(i))
if ltno(i)>0 then sfp(i)=sfp(i)*100000
read(temp(i)) ; temp(i)=temp(i)+273.0
  finish else start
!
! the characteristics of this link are described in an equation later.
read(temp(i)) ; temp(i)=temp(i)+273.0
  kv(i)=0
  finish
repeat
!
!  get a list of all the nodes
  for qq=1,1,40 cycle
   in(qq)=0 ; out(qq)=0
   node(qq)=0
  repeat
   dlindex(i)=0 for i=1,1,40
   for qq=1,1,nf cycle
   lindex(qq*2-1)=u(qq) ; lindex(qq*2)=d(qq)
   repeat
   indlis(dlindex,node,uu,dd,in,out,nnodes,nf)
   for qq=1,1,40 cycle
   lindex(qq)=dlindex(qq)
   repeat
nn=nnodes
!Enter node conditions which are fixed
!             Answer >= 0 to prompt 'node spec ?' if specifying
!             a condition, else answer < 0
!
for i=1,1,40 cycle
read(nspec)
```

```
      if nspec < 0 thenexit
      read(nnum)
!  check that the node label is in the list
       inoden(nnum,nn,nlab,node)
!
!     the nodes are classified according to the following codes
!     code                  type of node
!     ----                  ------------
!      1                  fixed pressure node
!      2                  fixed flow node
!      3                  height specified
!
      read(ncode(nnum));!read the node code
      read(ncap(nnum));!   read the capacity of the node in m**3
      read(ht(nnum));!   read the height of the node in m
      if ncode(nnum)=2 then start; !fixed flow node
!
      pcount=pcount+1
read(fn(nnum)); ! note that flow in is +ve, flow out of node is -ve
      fn(nnum)=-fn(nnum)
         ffix(nnum)=1
      finish else start
      read(p(nnum))
      if p(nnum)>0 then start
      pcount=pcount+1
         p(nnum)=p(nnum)*1.0@5 ;! convert bar to n/m**2
         ptot=ptot+p(nnum)
      if ncode(nnum)=1 then pfix(nnum)=1
      finish
      finish
      read(nodtemp(nnum)) ; ! read the node temperature in C
      nodtemp(nnum)=nodtemp(nnum)+273.0 ; ! convert to kelvin
      repeat
!read in number of pumps in network
!
read(npump)
!
      if npump>0 thenstart
      for ii=1,1,npump cycle
!get the number of points on the characteristic.
      read(npts(ii))
      repeat
!
      for ii=1,1,npump cycle
      read(uu(ii)); read(dd(ii))
!
      for jj=1,1,nf cycle
!set value of tlink for appropriate link number.
!also set value of ipbr (link number of pump in network)
      if uu(ii)=u(jj) and dd(ii)=d(jj) then start
      tlink(jj)=2
!tlink(..) = 2 signifies a pump
      ipbr(ii)=jj
      finish
      repeat
!head/flow data
         for jj=1,1,npts(ii) cycle
!read head and flow values on pump characteristic
         read(pchar(ii,jj*2-1)); read(pchar(ii,jj*2))
         repeat
```

```
      repeat
      finish
!
      rav=0
!
!get the physical properties data
      for ii=1,1,3 cycle
      read(tpres(ii)); read(ttemp(ii))
      read(tden(ii)); read(tvisc(ii))
!convert bar to pascal, celcius to kelvin, cp to kg/ms
      tpres(ii)=tpres(ii)*1.0@5
      ttemp(ii)=ttemp(ii)+273.0
      tvisc(ii)=tvisc(ii)*1@-3
      rav=rav+(tpres(ii)/(tden(ii)*ttemp(ii)))
      repeat
!get value of nfluid (-ve for gas, 0 or +ve for liquid)
      read(nfluid)
!
! get the average value of the gas constant, rav.
!
      rav=rav/3
!
! see if should quit at this point ...
!
      read(f3)
!
if f3<0 then start
ptot=ptot/pcount
return
finish else start
eqparse(kv,node,in,out,tlink,itype,ivalue,rvalue,nn,nf,ierror)
finish
!
end
endoffile
```

```
!
!  ROUTINE EQPARSE : parses input equations describing network
!
externalroutinespec s to r(string(40) s,realname x,integername srflag)
externalroutinespec ucstrg(stringname s)
externalroutinespec inoden(integer label,nnodes,integername index,c
integerarrayname node)
!
externalroutine eqparse(longrealarrayname kv,integerarrayname node,c
in,out,tlink,itype,ivalue,realarrayname rvalue,integername nn,nf,ierror)
!
! For each atom, 4 entries are generated.
!      itype - indicates if item is P,Q,F or constant
!      ivalue - label indicating location of node/link in network
!      rvalue - coefficient of P,Q,F or value of constant
!
! itype         meaning           ivalue     rvalue
! -----         -------           ------     ------
!   1        pressure term     node label    coeff
!   2        link flow term    link label    coeff
!   3        node flow term    node label    coeff
!   4        constant term          -1       value
!
!  The node identifiers are F (f) and P (p).
!  The link identifier is Q (q).
!  There are also identifiers for individual nodes and links.
!  Links are identified in terms of the nodes between which they run.
!
!
integer eqstat,j,ollen,stsign,lbctr,rbctr
integer natoms,sign,srflag,index1,index2,index,eqlen,ll,dpt,am
integer istr1,istr2,istr,qq,i,m,jj,errcnt
real x,xstr1,xstr2,xstr
string(1) lcr,ccr,lbr,rbr,nlcar
string(1) array oper(1:3),nos(1:10)
string(20) numstr,str,str1,str2
string(80) line,oline,nline,errmes
!
!  set the newline character
   nlcar="
"
!  set the array of operator values
   oper(1)="+"
   oper(2)="-"
   oper(3)="="
!
!  set the string values for brackets
   lbr="("
   rbr=")"
!
!  initialise the array of integers
   nos(1)="1" ; nos(2)="2" ; nos(3)="3"
   nos(4)="4" ; nos(5)="5" ; nos(6)="6"
   nos(7)="7" ; nos(8)="8" ; nos(9)="9"
   nos(10)="0"
!
!  start reading in equation lines (max of 20 lines is expected)
!.....................................................................
   for m=1,1,20 cycle
```

```
!
   ierror=0
   errmes=""
   line=""
   nline=""
   oline=""
   ccr=""
!
!  call routine ucstr to read the line
   ucstrg(line)
!  get the line length
   eqlen=length(line)
   if line="E" or line = "e" then return
!
   lbctr=0
   rbctr=0
   natoms=0
!
!  initialise eqstat at start of read. eqstat signifies whether
!  an "=" has been encountered yet in the line.
!
   eqstat=-1
!
!  cycle for the max no. of atoms expected in equation line.
!  initialise the string variable holding the previous character.
   lcr=""
   oline=line
!  initialise the sign to "+"
!
!  initialise the coefficient of P,Q,and F terms.
   x=1
!
   sign=1 ; stsign=1
   for i=1,1,10 cycle ; ! start of main cycle to read line
!.....................................................................
!
!  no blanks allowed
   if oline->(" ").oline  then ierror=-1 and -> errorl
!
!  see if line commences with '+' or '-'
!
   if i=1 then start
     if substring(oline,1,1)="+" then start
        oline -> ("+").oline ; sign=1 ; lcr="+"
     finish
     if substring(oline,1,1)="-" then start
        oline -> ("-").oline ; sign=-1 ; lcr="-"
     finish
   finish
!
numtest : !test for numbers
!  see if next char is a number
   if oline="" then exit
   ccr=substring(oline,1,1)
   for j=1,1,10 cycle
   if ccr=nos(j) then -> numlab
   repeat
   if ccr="." then start
     -> numlab
   finish
```

```
!
!  if the next character is not a number :
!  it may be an operator
!
   if ccr="+" or ccr="-" or ccr="=" then start
!
   if lcr="+" or lcr="-" or lcr="=" then ierror = -2 and -> errorl
   if lcr="+" or lcr="-" or lcr="=" then ierror = -2 and -> errorl
     if ccr="+" and lbctr>rbctr then stsign=sign else stsign=1
     if ccr="+" then sign=1 and lcr="+" and oline->(ccr).oline
     if ccr="-" and lbctr>rbctr then stsign=sign else stsign=1
     if ccr="-" then sign=-1 and lcr="-" and oline->(ccr).oline
     if ccr="=" then eqstat=1 and lcr="=" and oline->(ccr).oline
   if ccr="=" then sign=1 and x=1
   -> newlab
   finish
!
!  test if left bracket is present.
   if ccr=lbr then start
    oline -> (ccr).oline
    lbctr=lbctr+1
    -> newlab
   finish
!
!  test if right bracket is present.
   if ccr=rbr then start
    rbctr=rbctr+1
    if rbctr>lbctr then ierror=-3 and -> errorl
    oline -> (ccr).oline
    -> newlab
   finish
!  test whether the next char is an identifier
   if ccr="P" or ccr="Q" or ccr="F" then start
     if lcr="" or lcr="+" or lcr="-" or lcr="=" then start
      if ccr="P" then nline=oline and -> plab
      if ccr="Q" then nline=oline and -> qlab
      if ccr="F" then nline=oline and -> flab
     finish else ierror = -4 and -> errorl
   finish
!
!  if none of these things, then error
   ierror = -5
   -> errorl
!
numlab : !numbers
!-------------------------------------------------------------------------
!
!  initialise decimal point counter
   dpt=0
!  initialise exponent counter
   am=0
!
   numstr=""
!  look at the rest of the line
   ollen=length(oline)
!
!  has the line been completed ?
   if ollen=0 then -> newlabl
!
   for jj=1,1,ollen cycle
```

```
      ccr=substring(oline,jj,jj)
      for j=1,1,10 cycle
    if ccr=nos(j) and jj=ollen then numstr=numstr.ccr and -> numdec
    if ccr=nos(j) and jj#ollen then numstr=numstr.ccr and -> ncont
      repeat
      if ccr="." then start
        if dpt=1 then ierror=-6 and -> errorl  else start
          dpt=1
          numstr=numstr.ccr
          lcr="."
          -> ncont
        finish
      finish
      if ccr="@" then start
        if am=1 then ierror=-7 and -> errorl else start
          am=1
          numstr=numstr."@"
          lcr="@"
!  look at nline to see if a valid number follows the exponent
          -> ncont
        finish
      finish
!
   if am=1 then start
!  if the next char is a "+" or "-" after an "@".
     if lcr="@" and (ccr="+" or ccr="-") then start
      numstr=numstr.ccr
      lcr=ccr
      -> ncont
     finish
     if (lcr="+" or lcr="-") and (ccr="+" or ccr="-") c
     then ierror=-8 and -> errorl
   finish
!
!  if the next char is none of these things, then decode the number
   if jj=ollen and ccr=lbr then ierror=-9 and -> errorl
      lcr=ccr
      -> numdec
!
ncont : repeat
!-------------------------------------------------------------------------
numdec : !decode the number
!
   s to r(numstr,x,srflag)
!  look at rest of line following number.
!
   if lcr=lbr then start
    lbctr=lbctr+1
    nline=substring(oline,jj+1,ollen)
    -> plab
   finish
!
  if jj#ollen then c
   nline=substring(oline,jj+1,ollen) else nline=""
   ll=length(nline)
if ll#0 then ccr=substring(nline,1,1) else ccr=nlcar
!  is number a constant ?
!
 if ccr="+" or ccr="-" or ccr="=" or ccr=")" or ll=0  then start
   natoms=natoms+1
```

```
   itype(m,natoms)=4  ; ivalue(m,natoms)= -1
   rvalue(m,natoms)=stsign*x*sign*eqstat
  if ccr="-" then sign=-1 else sign=1
  if ccr="=" then eqstat=1 and x=1
   if jj=ollen then exit
   lcr=ccr
   oline=nline
!  atom finished. continue atom cycle
      -> newlab
   finish
!
!  the number is a coefficient
!
   if nline -> (lbr).nline then start
!
!.....................................................................
plab : ! pressure
!
   if nline -> ("P(").nline then start
   natoms=natoms+1
   itype(m,natoms)=1
   rvalue(m,natoms)=stsign*x*sign*eqstat
   nline -> str.(")").nline
!
   -> nodelab
   finish
!
!.....................................................................
!  may be link flow identifier
!
qlab : ! flow in link
   if nline -> ("Q(").nline then start
   natoms=natoms+1
   itype(m,natoms)=2
   rvalue(m,natoms)=stsign*x*sign*eqstat
   nline -> str.(")").nline
   -> nodelab
!
   finish
!.....................................................................
flab : !
!  may be node flow identifier
!
   if nline -> ("F(").nline then start
   natoms=natoms+1
   itype(m,natoms)=3
   rvalue(m,natoms)=stsign*x*sign*eqstat
   nline -> str.(")").nline
   finish
!
!.....................................................................
nodelab : !
!  at this point call the special routine for labels, inoden
!
!  if identifier is Q there are two labels to be matched
   if itype(m,natoms)=2 then start
   str -> str1.(",").str2
   s to r(str1,xstr1,srflag)
   s to r(str2,xstr2,srflag)
   istr1=int(xstr1) ; istr2=int(xstr2)
```

```
    inoden(istr1,nn,index1,node)
    inoden(istr2,nn,index2,node)
    for qq=1,1,nf cycle
    if (index1=in(qq) and index2=out(qq)) then c
    ivalue(m,natoms)=qq and exit
    if (index1=out(qq) and index1=in(qq)) then c
    ivalue(m,natoms)=-(qq) and exit
    repeat
   finish else if itype(m,natoms)=1 or itype(m,natoms)=3 then c
   start
    s to r(str,xstr,srflag)
    istr=int(xstr)
    inoden(istr,nn,index,node)
    ivalue(m,natoms)=node(index)
    finish
    lcr=")"
!
!   if it's another number.
!
!   see if next char is a number
    oline=nline
    finish
!.....................................................................
newlab : repeat
!
newlab1 : repeat
!
error1 : !error handling
     if ierror  < 0 then start
       if ierror = -1 then errmes="Blank character not allowed"
       if ierror = -2 then errmes="Invalid character after operator"
       if ierror = -3 then errmes="Brackets not matching"
       if ierror = -4 then errmes="Next character should be P, Q or F"
       if ierror = -5 then errmes="Invalid character"
       if ierror = -6 then errmes="Error in position of decimal point"
       if ierror = -7 then errmes="Error in position of exponent"
       if ierror = -8 then errmes="Invalid character after exponent"
       if ierror = -9 then errmes="Brackets not closed"
       newline ; printstring(errmes." in line "); write(m,3); newline
     finish
!
end
endoffile
```

```
external c
routine set up rm(longrealarrayname ppi,p,k,kb,f,fn,qn,a,b,c
integerarrayname node,pset,qset,pfix,ffix,qfix,in,out,c
integer nn,nf,integername sum,mrows)
!
!This routine solves for pressures and flows by setting up the equations
!for flows into nodes and flows into pipes as separate entities.
!
! in...
! nn, nf, p, k, f, fn, pfix, ffix, qfix, in, out
!
! out...
! a, b, pset, fset
!
integer nl,i,j,s,ii,nfl,ln,flag,hh
integerarray flowval,flowdir(1:6),lmark(1:nf)
!
!get total number of equations (=no. of links + no. of nodes)
      nl=nn+nf
!
      flowval(i)=0 for i=1,1,6
      flowdir(i)=0 for i=1,1,6
      for i=1,1,nl cycle
      a(i,j)=0 for j=1,1,nl
      b(i)=0
      repeat
!
      pset(i)=0 for i=1,1,nn; qset(i)=0 for i=1,1,nf
!     lmark is a marker for each pipe. It is set to 1 once the flow
!     equation in that pipe has been inserted into the matrix. This
!     is in order that there can be no repetitions when the same
!     pipe is encountered again.
!
      lmark(i)=0 for i=1,1,nf
!
!     sum is the number of entities in the flow/pressure vector which
!     have been 'set' so far. ln is the line position marker.
!
      sum=0; ln=0
!
!     go through all the nodes
      cycle hh=1,1,nn
      ii=node(hh)
      if pfix(ii)=1 thenstart;! fixed pressure node
      ln=ln+1
!     first examine to see if this pressure is already in the
!     vector of flows and pressures (i.e. has it been 'set' yet)
!     if the entity is a new one then note its position in the
!     flow/pressure vector (as indicated by the value of 'sum')
      if pset(ii)=0 then sum=sum+1 and pset(ii)=sum
      a(ln,pset(ii))=1
      b(ln)=ppi(ii)
      finish
!
      if pfix(ii)=0 thenstart
      nfl=0
!     go through all links
      cycle i=1,1,nf
!     examine which links are connected to node ii
```

```
      if ii=out(i) or ii=in(i) thenstart
!     if pipe is previously 'unmarked' then mark it and set
!     flag to off.
   if lmark(i)=0 then lmark(i)=1 and flag=0 else flag=1
!  if flag is 'off' increase line no.
      if flag=0 and k(i)=0 and qfix(i)=1 then ln=ln+1
      if flag=0 and k(i)#0 then ln=ln+1
      nfl=nfl+1; flowval(nfl)=i
      if ii=out(i) then flowdir(nfl)=1 else flowdir(nfl)=-1
!     get b(ii)
      if flag=0 and k(i)#0 then b(ln)=-kb(i)
!
      if pset(ii)=0 then sum=sum+1 and pset(ii)=sum
!     is ii downstream or upstream node of pipe i?
        if ii=out(i) thenstart
         if pset(in(i))=0 then sum=sum+1 and pset(in(i))=sum
      if k(i)#0 then start
         if flag=0 then a(ln,pset(ii))=-k(i)
         if flag=0 then a(ln,pset(in(i)))=k(i)
      finish
        finish elsestart
         if pset(out(i))=0 then sum=sum+1 and pset(out(i))=sum
      if k(i)#0 then start
         if flag=0 then a(ln,pset(ii))=k(i)
         if flag=0 then a(ln,pset(out(i)))=-k(i)
      finish
        finish
      if qset(i)=0 then sum=sum+1 and qset(i)=sum
      if flag=0 and k(i)=0 and qfix(i)=1 then a(ln,qset(i))=1 c
      and b(ln)=qn(i)
      if flag=0 and k(i)#0 then a(ln,qset(i))=-1
      finish
!
      repeat
!     sum flows at a node if appropriate
      if nfl>=1 and ffix(ii)=1 then start
       ln=ln+1
       a(ln,qset(flowval(j)))=flowdir(j) for j=1,1,nfl
       b(ln)=fn(ii)
      finish
!
      finish
      repeat
!
   mrows=ln
!
      end
endoffile
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!        ROUTINE SETUPM
!        --------------
!        Adds equations in the data input file to the matrix of
!        network equations.
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
external routine setupm(longrealarrayname p,fnum,qn,k,kv,kb,a,rhs,c
realarrayname qrterms,rvalue,integerarrayname qterms,qtctr,pset,qset,c
pfix,ffix,qfix,in,out,tlink,ivalue,itype,integername qlctr,ierror,mrows,
integer pass,nn,nf,sum)
!
!This routine is a modified version of 'set up reqn', which solves
!for pressures and flows by setting up the equations for flows into
!nodes and flows in pipes as separate entries. In this routine,
!the matrix created from network data is added to, using additional
!network equations specified by the user.
!
! in...
! itype, ivalue, rvalue
!
! out...
! a, b, pset, qset
!
integer nl,i,j,s,mm,ii,nfl,ln,flag,nod1,nod2
integer inls,onls,nsp,modp,pp1,pp2,m,npts,nfts,nqts
integer ncts,mdim,jj,kk,natoms,index,cpindex,nodiq1,nodiq2
integer pcon,nindex,pindex,pinctr,fldir,nodiq
real cpincf,pincf,pp1cf,pp2cf
real cf1,cf2
longrealarray irhs(1:100)
integerarray flowval,flowdir(1:6),lmark(1:nf)
!
!  initialise the error flag, ierror
   ierror=0
!
!  initialise the number of link flows which are fixed
   qlctr=0
   qtctr(i)=0 for i=1,1,10
   for mm=1,1,10 cycle
     qterms(mm,i)=0 for i=1,1,5
     qrterms(mm,i)=0 for i=1,1,5
   repeat
!
!  the dimension of the (square) matrix = no. of links + no. of nodes.
   mdim = nn+nf
!
   for m=1,1,20 cycle ; !read each equation line
     if itype(m,1)=0 then -> nextcont
!
!increment no. of rows in matrix
   if pass=0 then mrows=mrows+1
!check that current no. of rows does not exceed mdim.
   if mrows>mdim then ierror=-1 and -> error1
!
   a(mrows,i)=0 for i=1,1,mdim
   irhs(mrows)=0
   npts=0 ; nqts=0 ; nfts=0 ; ncts=0 ; natoms=0
!
```

```
   for i=1,1,10 cycle
    if itype(m,i)=0 then exit
    natoms=natoms+1
    if itype(m,i)=1 then npts=npts+1
    if itype(m,i)=2 then nqts=nqts+1
    if itype(m,i)=3 then nfts=nfts+1
    if itype(m,i)=4 then ncts=ncts+1
   repeat
!
!-------------------------------------------------------------------------
!  is the equation a fixed pressure specification ; i.e. does it
!  contain only one pressure term and some constant terms ?
!
   if npts=1 and nqts=0 and nfts=0 and ncts>0 then start
!
      irhs(mrows)=0
      for j=1,1,natoms cycle
         if itype(m,j)=1 then start
!  look at the node index.
          index=ivalue(m,j)
          if pset(index)=0 then sum=sum+1 and pset(index)=sum
          a(mrows,pset(index))=rvalue(m,j)
          pfix(index)=1
         finish else start
          irhs(mrows)=irhs(mrows)+rvalue(m,j)
         finish
      repeat
!
   if irhs(mrows)>0 and a(mrows,pset(index))<0 then c
   a(mrows,pset(index))=0-a(mrows,pset(index))
   if irhs(mrows)<0 and a(mrows,pset(index))>0 then c
   irhs(mrows)=0-irhs(mrows)
   if irhs(mrows)<0 and a(mrows,pset(index))<0 then c
   ierror=-12 and -> error1
   p(index)=irhs(mrows)*1@5
   rhs(mrows)=irhs(mrows)*1@5
   continue
   finish
!-------------------------------------------------------------------------
!  is the equation a pseudo fixed pressure spec ; i.e. does it
!  contain two pressure terms and some constant terms ?
!
   if npts=2 and nqts=0 and nfts=0 and ncts>0 then start
!
      pinctr=0
      irhs(mrows)=0
      pp1=0 ; pp2=0; pp1cf=0 ; pp2cf=0
      for j=1,1,natoms cycle
         if itype(m,j)=1 then start
         pindex=0 ; pincf=0
         pinctr=pinctr+1
!  look at the node index.
          index=ivalue(m,j)
          if pset(index)=0 then sum=sum+1 and pset(index)=sum
          a(mrows,pset(index))=-rvalue(m,j)
          if pp1=0 and pfix(index)=0 then start
          pp1=-(index)
           pp1cf=rvalue(m,j)
          if pinctr=1 then pindex=index and pincf=rvalue(m,j)
          continue
```

```
            finish
            if ppl=0 and pfix(index)=1 then start
            cpindex=index ; cpincf=rvalue(m,j)
            finish
            if ppl#0 and pfix(index)=0 then pp2=-(index) and c
            pp2cf=rvalue(m,j)
          finish else start
!
            irhs(mrows)=irhs(mrows)+rvalue(m,j)
      finish
      repeat
!
!  see if both nodes are of unspecified pressure
!
   if ppl<0 and pp2<0 then start
     index=-(ppl) ; pfix(index)=pp2
     index=-(pp2) ; pfix(index)=ppl
   finish else start
     if ppl<0 then start
       nindex=-(ppl) ; pfix(nindex)=1
       if pindex>0 then start
        if pplcf<0 and rvalue(m,j)>0 then p(nindex)=c
        p(index)+irhs(mrows)
        if pplcf>0 and rvalue(m,j)<0 then p(nindex)=c
        p(index)-irhs(mrows)*1@5
       finish else start
        if pplcf<0 and cpincf>0 then p(nindex)=c
        p(cpindex)+irhs(mrows)
        if pplcf>0 and cpincf<0 then p(nindex)=c
        p(cpindex)-irhs(mrows)*1@5
       finish
     finish
   finish
!
   rhs(mrows)=irhs(mrows)*1@5
   -> contline
   finish
!-------------------------------------------------------------------------
! is the equation a flow specification for a pipe ?
!
   if npts=2 and nqts=1 and nfts=0 and ncts>=0 then start
!
!  get the identifiers of the end nodes.
!
   rhs(mrows)=0
   nod1=0 ; nod2=0 ; cf1=0 ; cf2=0
!
   for i=1,1,10 cycle
   if itype(m,i)=1 then start
     if nod1=0 then nod1=ivalue(m,i) and cf1=rvalue(m,i)else c
     nod2=ivalue(m,i) and cf2=rvalue(m,i)
   finish else if itype(m,i)=2 then start
   index=ivalue(m,i) ; if index<0 then index=-(index)
   ppl=in(index) ; pp2=out(index)
   finish else rhs(mrows)=rhs(mrows)+rvalue(m,i)
   repeat
!
!  check that the two nodes are connected by the given pipe no.
       if (nod1=ppl and nod2=pp2) or c
       (nod1=pp2 and nod2=ppl) then start
```

```
        if cf1#-(cf2) then ierror=-7 and -> error1
finish else start
!
!
ierror=-8
-> error1
finish
!
!  check that a 'k' value has not been assigned to this pipe ; if so
!  then assume that the coefficient (cf1,cf2) is the
!  required 'k' and set up the standard equation for that pipe.
!
          if k(index)#0 then ierror=-9 and -> error1
          if pset(nod1)=0 then sum=sum+1 and pset(nod1)=sum
          k(index)=cf2
          tlink(index)=-1 ; kv(index)=k(index)
          kb(index)=0
          a(mrows,pset(nod1))=-cf1
          a(mrows,pset(nod2))=-cf2
          if qset(index)=0 then sum=sum+1 and qset(index)=sum
          a(mrows,qset(index))=-1
!
continue
   finish
!
!-------------------------------------------------------------------------
! Q terms only
!
   if nqts>=1 and ncts>=0 and nfts=0 and npts=0 then start
   inls=0
   onls=0
   rhs(mrows)=0
!
     if nqts=1 then start ; !look at the end nodes
      for jj=1,1,natoms cycle
       if itype(m,jj)=2 then start
! see if the end nodes of the pipe are pendant nodes
        modp=ivalue(m,jj)
        fldir=1
        if modp<0 then modp=-(modp) and fldir=-1
        for kk=1,1,nf cycle
         if in(modp)=in(kk) or c
         in(modp)=out(kk) then inls=inls+1
         if out(modp)=in(kk) or c
         out(modp)=out(kk) then onls=onls+1
        repeat
        if inls=1 and onls=1 then ierror=-9 and -> error1
        if inls=1 or onls=1 then start
         if inls=1 then nsp=in(modp) else c
         nsp=out(modp)
         if pfix(nsp)=1 or ffix(nsp)=1 then ierror=-10 and -> error1
!
!  put the line in the coeff matrix
!
         if qset(modp)=0 then sum=sum+1 and qset(modp)=sum
         a(mrows,qset(modp))=-rvalue(m,jj)
         ffix(nsp)=1
        finish
       if inls>1 or onls>1 then start
!
```

```
! put the line in the coeff matrix
!
      if qset(modp)=0 then sum=sum+1 and qset(modp)=sum
      a(mrows,qset(modp))=-rvalue(m,jj)
      qfix(modp)=1
       finish
       finish else rhs(mrows)=rhs(mrows)+rvalue(m,jj)
      repeat
     if a(mrows,qset(modp))>0 then start
     if fldir=1 and onls=1 then fnum(nsp)=rhs(mrows)
     if fldir=-1 and onls=1 then fnum(nsp)=-rhs(mrows) and c
     rhs(mrows)=-rhs(mrows)
     if fldir=1 and inls=1 then fnum(nsp)=-rhs(mrows) c
     and rhs(mrows)=-rhs(mrows)
     if fldir=-1 and inls=1 then fnum(nsp)=rhs(mrows)
     if fldir=1 and (inls>1 or onls>1) then qn(modp)=rhs(mrows)
     if fldir=-1 and (inls>1 or onls>1) then qn(modp)=-rhs(mrows)
     finish
     finish
!
!if there are two 'q' terms.
!
   if nqts>1 then start
!
!  increment the counter for the number of fixed flows
!
     qlctr=qlctr+1
   nodiq=0 ; nodiq1=0 ; nodiq2=0
   mm=0
   for jj=1,1,natoms cycle
    if itype(m,jj)=2 then start
     mm=mm+1
     modp=ivalue(m,jj)
     fldir=1
     if modp<0 then modp=-(modp) and fldir=-1
!get the identifier of flow which is defined in terms of other network f
     if mm=1 then qtctr(qlctr)=modp
     if mm>1 then qterms(qlctr,mm-1)=modp
!
!put the line in the coefficient matrix
!
     if qset(modp)=0 then sum=sum+1 and qset(modp)=sum
     a(mrows,qset(modp))=-rvalue(m,jj)
     if mm>1 then qrterms(qlctr,mm-1)=-rvalue(m,jj)
!    if pfix(in(modp))=0 and (in(modp)=nodiq or c
!    ffix(in(modp))=0) then c
!    ffix(in(modp))=1 and nodiq=in(modp)
!    if pfix(out(modp))=0 and (out(modp)=nodiq or c
!    ffix(out(modp))=0) then c
!    ffix(out(modp))=1 and nodiq=out(modp)
     if nodiq1=0 then nodiq1=nodiq else nodiq2=nodiq
    finish else rhs(mrows)=rhs(mrows)+rvalue(m,jj)
   repeat
   if nodiq#0 and nodiq1=nodiq2 then fnum(nodiq)=rhs(mrows)
   finish
   finish
contline : !continue
!-------------------------------------------------------------------------
      repeat ; !finish reading each equation line
!
```

```
nextcont : !next
!   check that all k's have been assigned.
      for i=1,1,nf cycle
         if k(i)=0 then ierror=-1 and exit
      repeat
!
   for ii=1,1,nn cycle
!
!  check that flow balances have been set up for all nodes
!  where the pressure has not been assigned.
!
     if pfix(ii)=0 and ffix(ii)=0 then start
      mrows=mrows+1
      rhs(mrows)=0
      for jj=1,1,nf cycle
       if ii=in(jj) or ii=out(jj) then start
        if ii=in(jj) then a(mrows,qset(jj))=-1 c
        else a(mrows,qset(jj))=1
        ffix(ii)=1
       finish
      repeat
     finish
!
!  check that, for any node which has pressure specified in terms
!  of pressure at another node, that the pressure specification is
!  now specific
!
     if pfix(ii)<0 then start
       pcon=-(pfix(ii))
       if pfix(pcon)=1 then pfix(ii)=1 else c
       ierror=-6 and -> error1
     finish
   repeat
!
error1 : !error label
end
endoffile
```

```
!
!
!              ***** DYNAMIC NETWORK PROGRAM *****
!
begin
!
!-----------------------------------------------------------------------
externalroutinespec flprint(longrealarrayname p,k,kb,f,c
integerarrayname u,d,integer nn,nf)
!
externalroutinespec pressures(longrealarrayname a,b,p,f,fo,c
integer nn,nf,y,integerarrayname pset,qset)
!
externalroutinespec set up a(longrealarrayname k,kb,p,tp,fn,ncap,c
a,b,den,ht,realarrayname qrterms,integerarrayname qterms,qtctr,u,d,pfix,c
longreal delta,tcon,integer qlctr,nn,nf,string(20) linmeth)
!
externalroutinespec emas3cputime(longrealname time)
!
externallongrealfnspec logten(longreal x)
!
externalroutinespec nlink(integer nnodes,nlinks,c
integerarrayname in,out,pfix,link,cc,cp,longrealarrayname pp,fexx)
!
externalroutinespec fcheck(longrealarrayname q,c
fexx,integerarrayname pfix,link,in,cp,c
integer nn,longreal ftol,longrealname hftot,integername hfnod,check)
!
externalroutinespec flows(longrealarrayname p,kv,k,kb,f,fo,l,da,rk,ft,c
denav,den,ht,vis,cd,cv,cl,c2,temp,realarrayname qrterms,c
integerarrayname qtctr,qterms,ipbr,u,d,tlink,pform,integer pass,printit,c
nf,nn,npump,nfluid,string(20) linmeth)
!
externalroutinespec iaux(longrealname a,rhs,pp,integername nn,nz,c
nm,licn,lirn,icn,irn,ikeep,ivect,jvect,iw,idisp,c
rpt,longrealname anag,w)
!
externalroutinespec idenst(longrealarrayname cd,longrealname c
dens,longreal press,temp,integer nfluid)
!
externalroutinespec idfit(longrealarrayname d,p,t,longrealname c
cd1,cd2,cd3,integer nfluid)
!
externalroutinespec ipmpnet(longrealarrayname pchar,c1,c2,c
integerarrayname npts,pform,integer npump)
!
externalroutinespec ivfit(longrealarrayname v,p,t,longrealname c
cv1,cv2,cv3,integer nfluid)
!
externalroutinespec ivisco(longrealarrayname cv,longrealname c
visc,longreal press,temp,integer nfluid)
!
externalroutinespec ilnpump(longreal p1,p2,c1,c2,den,c
integer pform,n,longrealname q,a,b)
!
externalroutinespec set up rm(longrealarrayname ppi,p,k,kb,f,c
fn,qn,a,b,integerarrayname node,pset,qset,pfix,ffix,qfix,in,out,c
integer nn,nf,integername sum,mrows)
!
```

```
externalroutinespec getdata(longrealarrayname p,kv,l,da,rk,ft,c
temp,fn,ht,ntemp,tpres,tvisc,tden,ttemp,pchar,mu,ncap,sfp,c
integerarrayname node,ltno,in,out,ffix,pfix,npts,ipbr,tlink,ncode,c
itype,ivalue,u,d,realarrayname rvalue,integername c
nn,nf,npump,nfluid,ierror,longrealname ptot,rav,stringname filename)
!
externalroutinespec set up k(longrealarrayname f,fo,flst,k,kb,kv,p,c
plst,l,da,denav,temp,vis,ht,ft,rk,cv,cd,cl,c2,mu,sfp,integerarrayname c
ltno,in,out,tlink,pform,ipbr,integer nf,npump,c
nfluid,pass,string(20) linmeth)
!
externalroutinespec rsolex eqn(longrealarrayname a,b,p,f,fo,anag,c
w,integer nn,nf,y,nm,integername nz,licn,lirn,integerarrayname c
u,d,node,pset,qset,icn,irn,ikeep,iw,idisp,ivect,jvect,string(20) linmeth)
!
externalroutinespec emas3prompt(stringname s)
!
externalroutinespec emas3(stringname comm,parms,c
integername flag)
!
externalroutinespec opensq(integer m)
!
externalrealfnspec random(integername i, integer n)
!
!-------------------------------------------------------------------------
!      ***** Main program starts here *****
!            ------------------------
!
!      ***** beginning of declarations *****
!
!  main arrays...
!   a - matrix for linearised equations
!   b - constant vector for  ..   ..
!   p - new pressures to be calculated, or delta p's in Newton solution
!   po- last pressures
!   kv - valve consts for flow=kv*sqrt(delta p)
!   k - linearised valve constants
!   f - new flows.  NB max 2* no.of nodes
!   fo - last flows
!   fn - node specified flows
!
!
!  'structure' arrays...
!   u(i) - number of node upstream on branch i
!   d(i) -   ..     ..   downstream ..  ..
!   pfix(i) - is 1 if pressure at node i is fixed specification,
!              0 if variable
!   ffix(i) - is 1 if flow at node i is fixed specification,
!              0 if variable
!               N.B. flow into node is +ve, out of node is -ve
!
longrealarray a(1:40,1:40),p,plst,b(1:40),po,ppi,ncap,ptp,tp,ht,c
l,da,rk,ft,fn,den,temp,ntemp,mu,sfp(1:40)
longrealarray k,kv,kb,fo,f,flst,denav,vis(1:40)
longrealarray tpres,tvisc,tden,ttemp(1:3),pchar(1:10,1:10)
longrealarray cl,c2(1:10)
longrealarray cv,cd(1:3)
realarray rvalue(1:40,1:10),qrts(1:10,1:5)
integerarray ltno,pset(1:40),qset(1:40)
integerarray pbr,pform(1:10),qts(1:10,1:5)
```

```
integerarray cp,cc(1:40,1:6),link(1:40),qtcount(1:10)
integerarray in,out,ffix,pfix,tlink,ncode(1:40),npts,ipbr(1:10)
integerarray itype,ivalue(1:40,1:10),node(1:40)
integerarray u,d(1:40)
longreal ptot
longreal time2, time, tcon, hftot, ftol, rav
real delta,deltac,tnext,xmax,xrandom,deltatime,deltaclast,deltaco
real tmax
string(40) outfile
integer nm,hfnod,check,rcheck,oflag,nodemax,ntstep
integer nfluid{-ve for gas, 0 or +ve for liquid}
integer ll,mm,tc,y,zz,mcc,ks,irandom,nrandom
integer sum, mrows,pass,eflag,pcount,pcset,pc2s
integer nn,nf,npump,ierror,i,j,zw,zy,qlcount,ff
string(20) filename
string(1) ans
!
!
ownintegerseed1=1234567
ownintegerseed2=7654321
ownreal switch=10 {..after ? iterations switch to Newton}
ownreal eqset=0 {..solve full set or short set of eqns}
ownreal eps=0.001 {small number for 'compressibility'}
ownstring (20) linmeth="hutchison" {initial solution method}
ownstring (20) solmeth="shortset" {solve for pressures only}
!
!-------------------------------------------------------------------------
! CHECK P : This routine checks pressures for convergence
!           and returns 0 if sum of absolute changes is less
!           than specified limit. Also updates po().
!
integerfunction check p(longrealarrayname p,po,integer nn)
!
! in... p(),po(),nn
! out.. po()
integer i
longreal sum
sum=0
for i=1,1,nn cycle
  sum=sum+mod(p(i)-po(i))
  po(i)=p(i)
repeat
!
! current limit is 0.00001 N/m**2
!
if sum<0.00001 then result=0
!newline
!printstring("press Error = ") ; !printfl(sum,7) ; !newline
result=1
!
end
!-------------------------------------------------------------------------
!
nm=10
ftol=0.0000001
!
!
qlcount=0
for i=1,1,10 cycle
  qtcount(i)=0
```

```
  for j=1,1,5 cycle
    qts(i,j)=0; qrts(i,j)=0
  repeat
repeat
!
for i=1,1,40 cycle
for j=1,1,10 cycle
itype(i,j)=0 ; ivalue(i,j)=0
rvalue(i,j)=0
repeat
p(i)=0
f(i)=0
ppi(i)=0
repeat
!
for i=1,1,10 cycle
pform(i)=0
c1(i)=0 ; c2(i)=0
repeat
!
!
filename="name of file : "
emas3prompt(filename)
readstring(filename)
emas3("define","2,.out",eflag)
outfile="name of output file : "
emas3prompt(outfile)
! Get name of output file for results
!
readstring(outfile)
emas3("define","11,".outfile,eflag)
emas3("define","20,cfkout",eflag)
emas3("define","21,cfout",eflag)
emas3("define","22,cpout",eflag)
emas3("define","23,cpkout",eflag)
emas3("define","25,dy2list",eflag)
!
getdata(p,kv,l,da,rk,ft,temp,fn,ht,ntemp,tpres,tvisc,tden,c
ttemp,pchar,mu,ncap,sfp,node,ltno,in,out,ffix,pfix,npts,ipbr,tlink,c
ncode,itype,ivalue,u,d,rvalue,c
nn,nf,npump,nfluid,ierror,ptot,rav,filename)
!get no. of links to each node
nlink(nn,nf,in,out,ncode,link,cc,cp,p,fn)
!
      for y=1,1,nn cycle
      if ncap(y)>0 then ncap(y)=ncap(y)/(rav*ntemp(y))
      if p(y)=0 then start
      p(y)=ptot
      po(y)=ptot
      ppi(y)=ptot
      finish else start
      ppi(y)=p(y)
      po(y)=p(y)
      finish
      repeat
!
!     assign initial value to hfnod
      hfnod=0
      hftot=0
!
```

```
!get pump characteristics if there are pumps in the network
if npump>0 then start
ipmpnet(pchar,c1,c2,npts,pform,npump)
finish
!
!get viscosity and density fit details
        ivfit(tvisc,tpres,ttemp,cv(1),cv(2),cv(3),nfluid)
        idfit(tden,tpres,ttemp,cd(1),cd(2),cd(3),nfluid)
!
for y=1,1,nf cycle
k(y)=0.0 ; kb(y)=0.0
repeat
flows(p,kv,k,kb,f,fo,l,da,rk,ft,denav,den,ht,vis,cd,cv,cl,c2,temp,c
qrts,qtcount,qts,ipbr,u,d,tlink,pform,0,0,nf,nn,npump,nfluid,linmeth)
!
!      ****** start of program run with defined data set *****
!
cycle
emas3cputime(time)
!
selectinput(0)
emas3prompt("Switch to Newton:") ; read(switch)
emas3prompt("Label of node for pressure control:"); read(nodemax)
if nodemax>0 then start
emas3prompt("Maximum pressure variation at this node (Newtons):")
read(xmax);xmax=xmax*2
finish
emas3prompt("time step : "); read(delta)
      deltatime = 0
      deltac=0
      deltaco=0
if nodemax>0 then start
 emas3prompt("non-disturbance time(secs) : "); read(deltatime)
finish
emas3prompt("no of time steps : "); read(ntstep)
emas3prompt("value of tmax : "); read(tmax)
      tcon=0; ! initialise time counter
      tp(zz)=0 for zz=1,1,nn
      oflag=1;! set convergence flag to off
!
!
!-------------------------------------------------------------------------
!      ***** read data into file for plotting with "EASYGRAPH" *****
!
!     only print data for node whose pressure is being controlled.
      for j=1,1,nf cycle
      if ltno(j)>0 then start
      selectoutput(22)
!     write in time and pressure as data pairs
      newline
      print(deltac,5,2); print(p(ltno(j))*0.00001,4,10)
      newline
      finish
      repeat
!
!     closestream(22)
      selectoutput(21)
!
!     only print data for link through which flow is controlled
      for j=1,1,nf cycle
```

```
      if ltno(j)<0 then start
      ltno(j)=0-ltno(j)
!     write in time and flow as data pairs
      print(deltac,5,2); print(f(ltno(j)),4,5)
      ltno(j)=0-ltno(j)
      newline
      finish
      repeat
!
!      closestream(21)
      selectoutput(20)
!
      for j=1,1,nf cycle
      if ltno(j)<0 then start
!     write in time and valve constant as data pairs
      print(deltac,5,2); print(kv(j),2,10)
      newline
      finish
      repeat
!
!      closestream(20)
      selectoutput(23)
!
      for j=1,1,nf cycle
      if ltno(j)>0 then start
!     write in time and valve constant as data pairs
      print(deltac,5,2); print(kv(j),2,10)
      newline
      finish
      repeat
!
!      closestream(23)
!
!*** set values of flst and plst for next time round
flst(zw)=f(zw) for zw=1,1,nf
plst(zy)=p(zy) for zy=1,1,nn
!***
!
      deltaclast=0
      tnext=deltatime
      for tc=1,1,ntstep cycle;! start of time cycle
      deltac=deltac+delta
! see if time value has exceeded tnext
      if nodemax>0 then start
      if (deltac-deltaclast)>tnext then start
        deltaclast=deltac
        tnext=random(seed1,0)*tmax
newline ; printstring("time = ");print(deltac,5,2);printstring("secs")
newline ; printstring("new generated time interval = ");print(tnext,4,2)
        xrandom=random(seed2,0)*xmax
! if xrandom >= 200 then xrandom -> +ve fluctuation in pressure
! if xrandom < 200 then xrandom -> -ve fluctuation in pressure
        xrandom = xrandom - 200
        printstring("  x = ");print(xrandom*1@-5,2,3)
        if nodemax>0 then start
newline; printstring("old pressure was ");print(p(nodemax),7,3)
          p(nodemax)=p(nodemax)+xrandom
newline; printstring("new pressure is ");print(p(nodemax),7,3)
        finish
      finish
```

```
      finish
      for j=1,1,nn cycle
      if ncap(j)>0 then ptp(j)=tp(j) and tp(j)=p(j)
      repeat
!     newline;!printstring("iteration no. ");!write(tc,3)
!     newline;! printstring("time step = ");!print(deltac,5,3)
      tcon=delta
!     if tc=1 then tcon=0
!
for i=1,1,100 cycle ;! ---------------- start iteration ----------------
!
!  set the value of 'pass' for routines 'setupk' and 'flows'
!
pass=1
if tc=1 and i=1 then pass=0
if tc>1 and i=1 then pass=0
!
!
selectoutput(2)
set up k(f,fo,flst,k,kb,kv,p,plst,l,da,denav,temp,vis,ht,ft,rk,c
cv,cd,cl,c2,mu,sfp,ltno,u,d,tlink,pform,ipbr,nf,npump,nfluid,pass,linmeth
selectoutput(2)
!
set up a(k,kb,p,tp,fn,ncap,a,b,denav,ht,qrts,qts,qtcount,u,d,pfix,c
delta,tcon,qlcount,nn,nf,linmeth)
!
pressures(a,b,p,f,fo,nn,nf,0,pset,qset)
   if i>switch then linmeth="newton"
flows(p,kv,k,kb,f,fo,l,da,rk,ft,denav,den,ht,vis,cd,cv,cl,c2,temp,c
qrts,qtcount,qts,ipbr,u,d,tlink,pform,pass,0,nf,nn,npump,nfluid,linmeth)
!
!     check for convergence
fcheck(f,fn,pfix,link,u,cp,nn,ftol,hftot,hfnod,check)
!
if (check = 0 and i>2) or i>100 then start
!newline ;! printstring("check=0")
!newline;! printstring("error = ");! print(hftot,3,5)
!printstring(" at node ");! write(hfnod,3)
!newline
if (check = 0 and i>2) or i>100 then exit
!if check p(p,po,nn) = 0 or i>100  thenexit
finish
!
repeat ;!              ---------------  next iteration ----------------
!
! set up k(f,fo,flst,k,kb,kv,p,plst,l,da,denav,temp,vis,ht,ft,rk,c
!cv,cd,cl,c2,mu,sfp,ltno,u,d,tlink,pform,ipbr,nf,npump,nfluid,pass,linmetl
!
flows(p,kv,k,kb,f,fo,l,da,rk,ft,denav,den,ht,vis,cd,cv,cl,c2,temp,c
qrts,qtcount,qts,ipbr,u,d,tlink,pform,1,0,nf,nn,npump,nfluid,linmeth)
!
!*** set values of flst and plst for next time round
flst(zw)=f(zw) for zw=1,1,nf
plst(zy)=p(zy) for zy=1,1,nn
!***
emas3cputime(time2)
!printstring("cpu secs = ");!print(time2-time,5,3)
!printstring("
!  Used ") ;! write(i,4) ;! printstring(" iterations")
!newline
```

```
!
!-------------------------------------------------------------------------
!read data into file for plotting with "EASYGRAPH"
!
!       closestream(2)
!      only print data for node whose pressure is being controlled
       for j=1,1,nf cycle
       if ltno(j)>0 then start
       selectoutput(22)
!      write in time and pressure as data pairs
       print(deltac,5,2); print(p(ltno(j))*0.00001,4,10)
       newline
       finish
       repeat
!
!       closestream(22)
       selectoutput(21)
!
!      only print data for link through which flow is controlled
       for j=1,1,nf cycle
       if ltno(j)<0 then start
!      write in time and flow as data pairs
       ltno(j)=0-ltno(j)
       print(deltac,5,2); print(f(ltno(j)),4,5)
       ltno(j)=0-ltno(j)
       newline
       finish
       repeat
!
!       closestream(21)
       selectoutput(20)
!
       for j=1,1,nf cycle
       if ltno(j)<0 then start
!      write in time and valve constant as data pairs
       print(deltac,5,2); print(k(j),2,10)
       newline
       finish
       repeat
!
!       closestream(20)
       selectoutput(23)
!
       for j=1,1,nf cycle
       if ltno(j)>0 then start
!      write in time and valve constant as data pairs
       print(deltac,5,2); print(k(j),2,10)
       newline
       finish
       repeat
!
!       closestream(23)
       selectoutput(2)
!-------------------------------------------------------------------------
!
       for zz=1,1,nn cycle
       if ncap(zz)>0 then start
       if mod(ptp(zz)-tp(zz))<0.001 then oflag=0
       if tc=1 then oflag=1
       finish
```

```
      repeat
      if oflag=0 then c
      start
      printstring("
      end of time-step cycle")
      newline
      exit
      finish
!
      repeat; ! repeat for time step cycle
selectinput(0)
emas3prompt("Continue (Y or N) ?"); skipsymbol ; readitem(ans)
         if ans ="n" or ans="N" then c
         printstring("         Finish") and -> printlabel
!
!        set all pressures to original values
         p(ll)=ppi(ll) for ll=1,1,nn
      newline
      emas3prompt("change parameters (Y or N) ?")
      skipsymbol;readitem(ans)
      newline
      if ans="y" or ans="Y" then start
      emas3prompt("link or node parameter?")
      read(kv(l)); read(mu(l)); read(ltno(l)) ; read(sfp(l))
      finish
!
printlabel:
!closestream(2)
selectoutput(ll)
flows(p,kv,k,kb,f,fo,l,da,rk,ft,denav,den,ht,vis,cd,cv,cl,c2,temp,c
qrts,qtcount,qts,ipbr,u,d,tlink,pform,l,l,nf,nn,npump,nfluid,linmeth)
!closestream(ll)
if ans ="n" or ans="N" then stop
!
repeat
!
endofprogram
```

```
!-----------------------------------------------
!
! SET UP K : Routine to get the 'k' values for each link.
!            The 'k' values are obtained from the nonlinear
!            'kv' values. The equation set up for each link
!            is ; Q = K * (P(in) - P(out)) + KB
!
external c
routine set up k(longrealarrayname f,fo,flst,k,kb,kv,p,plst,l,c
da,denav,temp,vis,ht,ft,rk,cv,cd,c1,c2,mu,sfp,integerarrayname ltno,in,c
out,tlink,pform,ipbr,integer nf,npump,nfluid,pass,string(20) linmeth)
!
externalroutinespec ilnpump(longreal p1,p2,c1,c2,den,c
integer pform,n,longrealname q,a,b)
!
externalroutinespec idenst(longrealarrayname cd,longrealname c
dens,longreal press,temp,integer nfluid)
!
externalroutinespec ivisco(longrealarrayname cv,longrealname c
visc,longreal press,temp,integer nfluid)
!
externallongrealfnspec logten(longreal x)
!
! in... kv(),p(),fo(),nf
! out.. k()
!
integer i ,j, vv,ks
longreal flow,dp,pi,re,ff,pav
longrealarray den(1:40)
pi=3.14159
!
! limit sets minimum pressure difference or flow below which
! linearisation is not attempted.
!
constreal limit=0.0001
for i=1,1,nf cycle
      pav=0.5*(p(in(i))+p(out(i)))
!
!     get average density in the pipe
      idenst(cd,denav(i),pav,temp(i),nfluid)
!     get density at either end of the pipe
      idenst(cd,den(in(i)),p(in(i)),temp(i),nfluid)
      idenst(cd,den(out(i)),p(out(i)),temp(i),nfluid)
!
!     get viscosity in the pipe
      ivisco(cv,vis(i),pav,temp(i),nfluid)
      dp=(p(in(i))+9.81*denav(i)*ht(in(i)))-(p(out(i))c
      +9.81*denav(i)*ht(out(i)))
!
      dp=mod(dp)
  if dp<limit then dp=limit
  flow=mod(fo(i))
  if mod(flow)<limit then flow=limit
!
!for tlink=-1, k does not have to be linearised
!
      if tlink(i)=-1 and kv(i)#0 then start
      kb(i)=0
      if ltno(i)>0 then start
```

```
! is upstream or downstream pressure being controlled ?
!
if sfp(i)>0 then start
!
       k(i)=kv(i)+mu(i)*(sfp(i)-plst(ltno(i)))
!
finish else if sfp(i)<0 then start
!
       k(i)=kv(i)+mu(i)*(plst(ltno(i))+sfp(i))
!
finish
!
       if pass=0 then k(i)=kv(i)
       finish else if ltno(i)=0 then start
       k(i)=kv(i)
       kb(i)=0
       if p(in(i))<p(out(i)) then kb(i)=-kb(i)
       finish else if ltno(i)<0 then start
       vv=0-ltno(i)
       k(i)=kv(i)+mu(i)*(sfp(i)-flst(vv))
       kb(i)=0
       if pass=0 then k(i)=kv(i)
       if flst(vv)<0.0000001 then k(i)=kv(i)
       finish
      finish else if tlink(i)=-1 and kv(i)=0 then start
        k(i)=0
        kb(i)=0
      finish
      if tlink(i)=0 thenstart
       if ltno(i)>0 then start
! is upstream or downstream pressure being controlled ?
!
if sfp(i)>0 then start
!
       k(i)=kv(i)+mu(i)*(sfp(i)-plst(ltno(i)))
!
finish else if sfp(i)<0 then start
!
       k(i)=kv(i)+mu(i)*(plst(ltno(i))+sfp(i))
!
finish
!
       if pass=0 then k(i)=kv(i)
       k(i)=k(i)/2/sqrt(dp)
       kb(i)=k(i)*(sqrt(dp)/2)
       if p(in(i))<p(out(i)) then kb(i)=-kb(i)
       finish else if ltno(i)=0 then start
       k(i)=kv(i)/2/sqrt(dp)
       kb(i)=kv(i)*(sqrt(dp)/2)
       if p(in(i))<p(out(i)) then kb(i)=-kb(i)
       finish else if ltno(i)<0 then start
       vv=0-ltno(i)
       k(i)=kv(i)+mu(i)*(sfp(i)-flst(vv))
       if pass=0 then k(i)=kv(i)
       k(i)=k(i)/2/sqrt(dp)
       kb(i)=k(i)*(sqrt(dp)/2)
       if p(in(i))<p(out(i)) then kb(i)=-kb(i)
       finish
      finish
!
```

```
      if tlink(i)=1 then start
       if pass=0 then start
!get laminar 'k'
      kv(i)=pi/(2*vis(i))*denav(i)*da(i)**4/((l(i)+50*ft(i)*da(i))*64)
      k(i)=kv(i)
      kb(i)=0
      finish else start
!     get Reynolds number
      re=(flow*4)/(pi*da(i)*vis(i))
!
!     laminar or turbulent flow?
      if re<2500 thenstart
!     get friction factor for laminar flow
      ff=64/re
      finish else start
!     get friction factor for turbulent flow
!     use Chen explicit equation
      ff=(rk(i)**1.1098)/2.8257+(5.8506/(re**0.8981))
ff=-2*logten((rk(i)/3.7065)-(5.0452/re)*logten(ff))
ff=(1/ff)**2
      finish
!     get k(i)
kv(i)=2*mod(log(den(in(i))/den(out(i))))
kv(i)=1/(ff*l(i)/da(i)+ft(i)+kv(i))
!!!
if denav(i)<0 then start
  newline;printstring("up");write(in(i),4);print(p(in(i)),7,3)
  newline;printstring("down");write(out(i),4);print(p(out(i)),7,3)
finish
!!!
kv(i)=(pi/2)*da(i)**2*sqrt(denav(i)/2)*sqrt(kv(i))
  if linmeth="newton" then start
 k(i)=kv(i)/(2*sqrt(dp))
kb(i)=mod((kv(i)/2)*sqrt(dp))
if p(out(i))>p(in(i)) then kb(i)=0-kb(i)
      finish
if linmeth="hutchison" then k(i)=(kv(i)**2)/flow and kb(i)=0
      finish
      finish
!
!     get the pump number corresponding to this link number
      if tlink(i)=2 then start
      for j=1,1,npump cycle
      if ipbr(j)=i then start
      ilnpump(p(in(i)),p(out(i)),c1(j),c2(j),denav(i),c
      pform(j),pass,f(i),k(i),kb(i))
      finish
      repeat
      finish
repeat
!
end
endoffile
```

```
externalroutine set up a(longrealarrayname k,kb,p,tp,fn,ncap,a,b,c
den,ht,realarrayname qrterms,integerarrayname qterms,qtctr,u,d,pfix,c
longreal delta,tcon,integer qlctr,nn,nf,string(20) linmeth)
!
!  Create the a matrix of linearised equations and its vector b
!  from the linearised flow/pressure relations involving k.
!
integer i,j,f1,f2,fd1,fd2,vv
integer fflag,q2def,jj,mm,uu
!
for i=1,1,nn cycle
  a(i,j)=0 for j=1,1,nn
  b(i)=fn(i)
repeat
!
for i=1,1,nf cycle
  fflag=0
  if qlctr>0 then start
    for jj=1,1,qlctr cycle
      if i=qtctr(jj) then fflag=jj
    repeat
  finish
  if fflag=0 then start
    f1=u(i) ; f2=d(i)
    a(f1,f1)=a(f1,f1)-k(i)
    a(f2,f2)=a(f2,f2)-k(i)
    a(f1,f2)=a(f1,f2)+k(i)
    a(f2,f1)=a(f2,f1)+k(i)
        b(f1)=b(f1)+kb(i)
        b(f2)=b(f2)-kb(i)
        b(f1)=b(f1)-(k(i)*den(i)*9.81*(ht(f2)-ht(f1)))
        b(f2)=b(f2)-(k(i)*den(i)*9.81*(ht(f1)-ht(f2)))
  finish else start
!look at the related flows
!*************************************************************************
    for mm=1,1,5 cycle
      if qterms(fflag,mm)>0 then start
!examine whether these flows themselves are defined in terms
!of other flows
        q2def=0
        for vv=1,1,qlctr cycle
          if qterms(fflag,mm)=qtctr(vv) then q2def=vv
        repeat
        if q2def=0 then start
          if k(qterms(fflag,mm))>0 then start
            f1=u(qtctr(fflag)) ; f2=d(qtctr(fflag))
            fd1=u(qterms(fflag,mm)) ; fd2=d(qterms(fflag,mm))
            a(f1,fd1)=a(f1,fd1)-(-1)*qrterms(fflag,mm)*k(qterms(fflag,mm))
            a(f1,fd2)=a(f1,fd2)+(-1)*qrterms(fflag,mm)*k(qterms(fflag,mm))
            a(f2,fd2)=a(f2,fd2)-(-1)*qrterms(fflag,mm)*k(qterms(fflag,mm))
            a(f2,fd1)=a(f2,fd1)+(-1)*qrterms(fflag,mm)*k(qterms(fflag,mm))
            b(f1)=b(f1)+(-1)*qrterms(fflag,mm)*kb(qterms(fflag,mm))
            b(f2)=b(f2)-(-1)*qrterms(fflag,mm)*kb(qterms(fflag,mm))
          finish
        finish else start
          for uu=1,1,5 cycle
            if qterms(vv,uu)>0 then start
              if k(qterms(vv,uu))>0 then start
                f1=u(qtctr(vv)) ; f2=d(qtctr(vv))
```

```
                fd1=u(qterms(vv,uu)) ; fd2=d(qterms(vv,uu))
        a(f1,fd1)=a(f1,fd1)-qrterms(fflag,mm)*qrterms(vv,uu)*k(qterms(vv,u
        a(f1,fd2)=a(f1,fd2)+qrterms(fflag,mm)*qrterms(vv,uu)*k(qterms(vv,u
        a(f2,fd2)=a(f2,fd2)-qrterms(fflag,mm)*qrterms(vv,uu)*k(qterms(vv,u
        a(f2,fd1)=a(f2,fd1)+qrterms(fflag,mm)*qrterms(vv,uu)*k(qterms(vv,u
        b(f1)=b(f1)+qrterms(fflag,mm)*qrterms(vv,uu)*kb(qterms(vv,uu))
        b(f2)=b(f2)-qrterms(fflag,mm)*qrterms(vv,uu)*kb(qterms(vv,uu))
              finish
            finish
          repeat
        finish
      finish
    repeat
!*************************************************************************
  finish
repeat
!
for i=1,1,nn cycle
  if pfix(i)=1 thenstart
    a(i,j)=0 for j=1,1,nn
    b(i)=p(i) ; a(i,i)=1
  finish
  if ncap(i)#0 and tcon>0 then start
    a(i,i)=a(i,i) - ncap(i)/delta
!
    b(i)=b(i)-tp(i)*ncap(i)/delta
  finish
repeat
end
endoffile
```

```
external routine flows(longrealarrayname p,kv,k,kb,f,fo,l,da,rk,ft,c
denav,den,ht,vis,cd,cv,cl,c2,temp,realarrayname qrterms,c
integerarrayname qtctr,qterms,ipbr,u,d,tlink,pform,integer c
pass,printit,nf,nn,npump,nfluid,string(20) linmeth)
!
externalroutinespec ilnpump(longreal p1,p2,c1,c2,den,c
integer pform,n,longrealname q,a,b)
externalroutinespec idenst(longrealarrayname cd,longrealname c
dens,longreal press,temp,integer nfluid)
externalroutinespec ivisco(longrealarrayname cv,longrealname c
visc,longreal press,temp,integer nfluid)
! calculate flows in branches once pressures are known
!
! in.. p() and pointers u(),d(), kv(),k(),den(),
! denav(),l(),da(),rk(),ft()
! and scalars pass, nf, printit
!
! out... f() and fo() when pass=0
!
integerarray iflow(1:10)
integer i,j,ii,jj,ifctr,lkflo,lkrflo,uu,vv,yy,zz,nflo,qflo
longreal flow,dp,s,ff,fp,re,pi,pav,kreal
!
owninteger op=0
op=printit
pi=3.14159
!
if op=1 then printstring("
node      pressure
")
if op=1 then start
for i=1,1,nn cycle
!
write(i,5)
print(p(i),4,4)
newline
repeat
finish
if op=1 then printstring("
Branch    from    to        flow            k               kb

")
ifctr=0
iflow(i)=0 for i=1,1,10
qflo=0
for i=1,1,nf cycle
!
      pav=0.5*(p(u(i))+p(d(i)))
!     get average density in the pipe
      idenst(cd,denav(i),pav,temp(i),nfluid)
!     get viscosity in the pipe
      ivisco(cv,vis(i),pav,temp(i),nfluid)
!
dp=(p(u(i))+9.81*denav(i)*ht(u(i)))-(p(d(i))+9.81*denav(i)*ht(d(i)))
if dp>=0 then s=1 else s=-1
dp=mod(dp)
!
!pipe data or valve const only ?
!
```

```
if tlink(i)=-99 then start
!
!flow is specified in terms of flow in another link
!
  for ii=1,1,10 cycle
    if qtctr(ii)=i then start
      ifctr=ifctr+1
      iflow(ifctr)=ii
    finish
  repeat
finish
!
if tlink(i)=1 thenstart
!pipe data supplied
!
!is it first time round ?
      if pass=0 thenstart
!     calculate laminar flow in each pipe
flow=(pi/(2*vis(i)))*denav(i)*da(i)**4/((l(i)+50*ft(i)*da(i))*64)*dp*s
      finish elsestart
!     calculate Reynolds number
      re=mod(f(i))*4/(pi*da(i)*vis(i))
!     laminar or turbulent flow ?
         if re<2500 thenstart
!        get laminar flow
flow=(pi/(2*vis(i)))*denav(i)*da(i)**4/((l(i)+50*ft(i)*da(i))*64)*dp*s
         finish elsestart
!        get flow
flow=kv(i)*sqrt(dp)*s
!if linmeth="newton" then flow=k(i)*dp*s + kb(i)
!if linmeth="hutchison" then flow=k(i)*dp*s
         finish
      finish
finish else if tlink(i)=0 thenstart
! only value for kv supplied
if pass=0 then flow=kv(i)*s*sqrt(dp)
if linmeth="hutchison" and pass=1 then flow=k(i)*dp*s + kb(i)
if linmeth="newton" and pass=1 then flow=k(i)*dp*s + kb(i)
      finish else if tlink(i)=-1 then start
      if pass=0 then flow=kv(i)*s*dp else c
      flow=k(i)*s*dp;{linear kv supplied}
      finish else if tlink(i)=2 then start
!     get the pump number corresponding to this link number
      for j=1,1,npump cycle
      if ipbr(j)=i then start
      ilnpump(p(u(i)),p(d(i)),c1(j),c2(j),denav(i),c
      pform(j),pass,flow,k(i),kb(i))
      finish
      repeat
      finish
if pass=0 then start
  if tlink(i)=-99 then flow=0
  if tlink(i)#-99 then fo(i)=mod(flow) else fo(i)=0
finish else start
  if linmeth="hutchison" then c
fo(i)=0.5*mod(fo(i)+mod(flow)) else fo(i)=flow
finish
f(i)=flow
repeat
!examine flows in links where tlink = -99 (signifies flow is
```

```
!specified in terms of other network flows)
!
if ifctr > 0 then start
  for ii=1,1,ifctr cycle
!  set flag to indicate that recursive flow definition has not (yet) occu
!
    nflo=1
    lkflo=iflow(ii)
    qflo=qtctr(lkflo)
! initialise value of flow in this link
    f(qflo)=0
    for zz=1,1,5 cycle
      lkrflo=0; yy=0; vv=0
      if qterms(lkflo,zz)>0 then lkrflo=qterms(lkflo,zz) and yy=zz
! is related flow expressed in terms of other flows ?
      if lkrflo>0 then start
        for uu=1,1,10 cycle
          if lkrflo=qtctr(uu) then vv=uu and nflo=0
        repeat
        if vv>0 then start
          for uu=1,1,5 cycle
            if qterms(vv,uu)>0 then start
              f(qflo)=f(qflo)+qrterms(lkflo,yy)*qrterms(vv,uu)*f(qterms(v
            finish
          repeat
        finish else start
          f(qflo)=-(1)*qrterms(lkflo,yy)*f(lkrflo)
        finish
      finish
    repeat
  repeat
finish
!
for i=1,1,nf cycle
  if op=1 then start
    write(i,5); write(u(i),7); write(d(i),5)
    print(f(i),8,9)
    print(k(i),5,9) and print(kb(i),6,6)
    newlines(1)
  finish
repeat
!
end
endoffile
```