APENDICE C

Listado del módulo principal del simulador numérico GEO

INSTITUTO TECNOLOGICO DE ESTUDIOS SUPERIORES DE MONTERREY **CAMPUS MORELOS** **INSTITUTO DE INVESTIGACIONES ELECTRICAS** **DEPARTAMENTO DE GEOTERMIA** **MARZO DE 1998**		
PROGRAMA	:	GEO
DESCRIPCION	:	SIMULADOR DE YACIMIENTOS GEOTERMICOS EN UNA O DOS FASES, EN UNA, DOS Y TRES DIMENSIONES.
FLUIDO	:	AGUA PURA
ESTADO	:	TRANSITORIO
ELABORADO POR	:	I.Q. MARCO ANTONIO CRUZ CHAVEZ
TESIS	:	PARA LA OBTENCION DEL GRADO DE MAESTRO EN CIENCIAS COMPUTACIONALES, CON LA ESPECIALIDAD EN SISTEMAS ASISTIDOS POR COMPUTADORA
ASESOR	:	DRA. SARA LILIA MOYA ACOSTA.

MODULO 1. Programa que resuelve un sistema poroso de cualquier geometría mediante la técnica de diferencias finitas integrales, utilizando el método de Newton para linealizar el sistema de ecuaciones acopladas que representan el fenómeno (transporte de masa y transporte de energía), y creando la matriz partida e invirtiéndola para obtener el vector solución.

```
#define MOD1 1
#include"defmod.h"

void main(void)
{
 int nom,maxit,itm=1,pt=0,i=0; double t=0.0,dt,maxt,perte,pertd,tole,told;
 entrada(&dt,&maxt,&perte,&pertd,&maxit,&tole,&told,emant,dmant,dm,eim,&nom);
 resultados(dmant,emant,t,pt++,g,rei,rdm,nom);
 do{
  transitorio(&i,&t,&dt);
  tiempotot=time(0);
  for(;;){
```

```
  if(Ecflujo(t,itm,pt,perte,pertd,tole,told,dt,emant,dmant,dm,eim))break;
  if(BD_propfis(eim,dm)<=0 || (itm++)>=maxit){
   nuevo_tiempo(&t,&dt,itm,maxit);
   correccion(&itm,emant,dmant,eim,dm);
  }
 }
 tiempo_cpu();
 resultados(dm,eim,t,pt++,g,rei,rdm,nom);
correccion(&itm,eim,dm,emant,dmant);
}while(t<=maxt);
closegraph();
}

int Ecflujo(double t,int itm,int pt,double perte,double pertd,double tole,
            double told,double dt,double *emant,double *dmant, double
            *dm,double *eim){
lista_elemento *iper[10]; register int i; int ke,ipvt[TA],kd;
iper[0]=NULL;
flux(0,iper,dt,emant,dmant,eim,dm,fdm,MASA);
residuo_func(fdm,&rdm,&kd);
flux(0,iper,dt,emant,dmant,eim,dm,fei,ENERGIA);
residuo_func(fei,&rei,&ke);
if(fabs(rei)<tole && fabs(rdm)<told)return(1);
convergencia(t,rei,rdm,ke,kd,itm,pt);
tiempo=time(0);
inicia_matriz();
jac(FLUX1,FLUX2,FLUX3,FLUX4,fei,fdm,emant,dmant,eim,dm,dt,perte,pertd,d,e);
 solve_part(d,e,FLUX1,FLUX2,FLUX3,FLUX4);          /*da solución a la matriz partida*/
 for(i=1;i<No_el;i++){                             /*se aplica corrección a densidad y energía*/
  dm[i]=dm[i]+d[i];
  eim[i]=eim[i]+e[i];
 }
return 0;
}

void residuo_func(double *fun,double *residuo,int *k)
{
register int i;
*residuo=0.0;  *k=0;
for(i=1;i<No_el;i++)
 if(fabs(fun[i])>(*residuo)){
  *residuo=fabs(fun[i]);
  *k=i;
 }}
```

```
void flux(int mj,lista_elemento *iper[],double dt,double *emant,double
          *dmant,  double *eim,double *dm,double *dgei,int ENERG){
 lista_elemento *inf; lista_conex *im;  register int i,j=0;
 double ftot,fliq,fvap,voln,pn,pm,d,a,kab,klr,kvr,dl,dv,vl,vv,K,tn,tm,Tant,
        Pant,Sant,hl,hv,droc,cproc,cos,FLk,FLk1;
 inf=(!mj)?princ->sig:iper[j];
 for(;;){
  if(inf==NULL)break;
  i=inf->num_elem;
  ftot=0.0;                                   /*inicialización de contador*/
  im=inf->abajo;
  temp_anterior(emant[i],dmant[i],&Tant,&Pant,&Sant);
  profgeo_elem(inf,&voln,&pn,&tn,&droc,&cproc);
  FLk=ENMA(0,ENERG,por,droc,cproc,tn,emant[i],dmant[i],Tant);
  FLk1=ENMA(1,ENERG,por,droc,cproc,tn,eim[i],dm[i],Tant);
  while(im!=NULL){
   icara(inf,im,&d,&a,&kab,&klr,&kvr,&dl,&dv,&vl,&vv,&pm,&K,&tm,&hl,&hv,&cos);
   fliq=(vl!=0.0)?(-(kab*klr*dl/vl)*((pn-pm)/d-dl*g*cos)):0.0;
   fvap=(vv!=0.0)?(-(kab*kvr*dv/vv)*((pn-pm)/d-dv*g*cos)):0.0;
   if(ENERG){
    fliq=hl*fliq;
    fvap=hv*fvap;
   }
   ftot=ftot+a*((ENERG)?(-K*(tn-tm)/d+(fliq+fvap)):(fliq+fvap));
   im=im->abajo;
  }
  dgei[i]=FLk1-FLk-(dt/voln)*(ftot+voln*((ENERG)?inf->Q:inf->q));
  inf=(!mj)?inf->sig:iper[++j];
 }
}

double ENMA(int actual,int ENER,double por,double d_roc,double
            cproc,double tn,double eim,double dm,double Tant){
 if(actual)return((ENER)?(por*eim*dm):(por*dm));
 else return((ENER)?(por*eim*dm-(1.0-por)*d_roc*cproc*(tn-Tant)):(por*dm));
}

void rectifica(int j,lista_elemento **inic)
{
 if(j!=(No_el-1))*inic=(*inic)->ant;
}
```

```
void jac(double (*M1)[TA],double (*M2)[TA],double (*M3)[TA],double
                (*M4)[TA],double *fei,double *fdm,double *ema,
                double *dma,double *eim,double *dm,double dt,double
                perte,double pertd,double *d,double *e){
 lista_elemento *inic=princ,*iper[10]; register int j;
 for(j=1;j<No_el;j++){
  salvdm[j]=fdm[j]; sd_m[j]=dm[j];
  salvei[j]=fei[j]; se_im[j]=eim[j];
 }inic=inic->sig;
 for(j=1;j<No_el;j++){          /*0 perturbo densidad 1 energía. 0 ec. masa 1 ec energía */
  direc_pert(iper,j);
  MJ(iper,&inic,M1,ema,dma,eim,dm,se_im,sd_m,salvdm,fdm,pertd,dt,j,0,0);
  rectifica(j,&inic);
  MJ(iper,&inic,M2,ema,dma,eim,dm,se_im,sd_m,salvdm,fdm,perte,dt,j,1,0);
  rectifica(j,&inic);
  MJ(iper,&inic,M3,ema,dma,eim,dm,se_im,sd_m,salvei,fei,pertd,dt,j,0,1);
  rectifica(j,&inic);
  MJ(iper,&inic,M4,ema,dma,eim,dm,se_im,sd_m,salvei,fei,perte,dt,j,1,1);
  d[j]=-salvdm[j];                              /*vector de términos independientes*/
  e[j]=-salvei[j];
 }
}

void direc_pert(lista_elemento *iper[],int j)
{
 int i;
 lista_elemento *inic=princ; lista_conex *pm;
 iper[0]=hallar(&inic,j);
 pm=iper[0]->abajo;
 inic=princ;
 for(i=1;pm;){
  if(pm->elem_conect!=0)iper[i++]=hallar(&inic,pm->elem_conect);
  pm=pm->abajo;
 }
 iper[i]=NULL;
}
```

```
void MJ(lista_elemento *iper[],lista_elemento **inic,double (*M)[TA],
        double *ema,double *dma,double *eim,double *dm,double
        *seim,double *sdm,double *salv,double *fun,double ped,double
        dt,int j,int pered,int FLUJO){
 register int i;
 lista_elemento *inicant;
 inicant=*inic;
 (pered)?(eim[j]=seim[j]+ped):(dm[j]=sdm[j]+ped);
 prop_elem_pert(inic,dm[j],eim[j],j);                    /*cálculo prop pert de un elem*/
 flux(1,iper,dt,ema,dma,eim,dm,fun,FLUJO);                          /*0 ec.masa*/
 for(i=0;iper[i];i++)                                  /*deriv df=d(f)/d(eim,dm)*/
  M[iper[i]->num_elem][j]=(fun[iper[i]->num_elem]-salv[iper[i]->num_elem])/ped;
 (pered)?(eim[j]=seim[j]):(dm[j]=sdm[j]);                          /*ret val sin pert*/
 prop_elem_pert(&inicant,dm[j],eim[j],j);              /*ret prop sin pert del elem*/
}

void solve_part(double *d,double *e,double (*FLUX1)[TA],double
                (*FLUX2)[TA],double (*FLUX3)[TA],double
                (*FLUX4)[TA]){
 double X0[TA],X1[TA],X2[TA];
 inversa(No_el-1,FLUX4,INV);
 mult_mat(No_el-1,FLUX2,INV,FLUX4);
 mult_mat(No_el-1,FLUX4,FLUX3,FLUX2);
 resta(No_el-1,FLUX1,FLUX2,FLUX2);
 mult_mat_vec(No_el-1,FLUX4,e,X0);
 resta_vec(No_el-1,d,X0,X0);
 inversa(No_el-1,FLUX2,FLUX1);
 mult_mat_vec(No_el-1,FLUX1,X0,X1);                  /*se obtienen x1 desconocidos*/
 mult_mat_vec(No_el-1,FLUX3,X1,X0);
 resta_vec(No_el-1,e,X0,X0);
 mult_mat_vec(No_el-1,INV,X0,X2);                    /*se obtienen x2 desconocidos*/
 intercambio(No_el-1,e,d,X1,X2);                  /*e sale con val de X2 y d con X1*/
}

int sat_vapor(double enintmez,double demez)
{ double P,T,Sv;
  transforma_propSint(demez,enintmez,&P,&T,&Sv);
  if(Sv<1.0 && Sv>0.0)return(2);                                    /*dos fases*/
  transforma_propS01(demez,enintmez,&P,&T,0);
  if(P<p_sat(T))return(2);                                          /*dos fases*/
  else if(fabs(Sv)<1.0)return(0);                                     /*líquido*/
  else if(fabs(Sv)>1.0)return(1);                                       /*vapor*/
  return 3; }
```

```
void elem_sincondini(int cont,int No_el,int *elem_cond,int *elem)
{
 register int i,j; int existe,aux=1;
 for(j=0;j<No_el;j++){
  existe=0;
  for(i=0;i<cont;i++)
    if(j==elem_cond[i])existe=1;
  if(!existe){
    elem[aux]=j;
    aux=aux+1;
  }
 }}

void intr_ci_def_el(double *eim,double *dm,double en,double den,int
                                    *elem,int el_sci){
 lista_elemento *inic=princ; register int i;
 inic=inic->sig;
 for(i=1;i<=el_sci;i++){
  propfis_elem(&inic,en,den,elem[i]);                    /*introd prop fis en lista*/
  eim[elem[i]]=en;
  dm[elem[i]]=den;
 }}

void condini_def_elemts(int sat_vap,int *elem,double *eim,double *dm,
                                    int el_sci){
 int i;
 if(sat_vap==0)                                          /*solo líquido*/
  intr_ci_def_el(eim,dm,227963.0,991.5,elem,el_sci);
 else if(sat_vap==2)                                     /*dos faces pendiente*/
  intr_ci_def_el(eim,dm,1128428.0,407.3451,elem,el_sci);
 else if(sat_vap==1)                                     /*solo vapor pendiente*/
  intr_ci_def_el(eim,dm,227963.0,991.5,elem,el_sci);
}
                                          /*prop físicas de todos los elementos*/
int BD_propfis(double *eim,double *dm)
{
 lista_elemento *inic=princ; int existe; register int i;
 inic=inic->sig;
 for(i=1;i<No_el;i++){
   existe=propfis_elem(&inic,eim[i],dm[i],i);
   if(!existe || existe==-1)break;
 }
 return existe;}
```

```
                                        /*propiedades físicas de la intercara por cada elemento*/
void intr_prop_lista(lista_elemento **inic,double P,double T,double denl,
                     double denv,double klrel,double kvrel,double viscl,
                     double viscv,double entl,double entv,double Sv,
                     double Sl,int elemento){
 lista_elemento *info;
 info=hallar(&*inic,elemento);
 info->sat_liqn=Sl;
 info->sat_vapn=Sv;
 info->den_liqn=denl;
 info->den_vapn=denv;
 info->k_liqreln=klrel;
 info->k_vapreln=kvrel;
 info->visc_liqn=viscl;
 info->visc_vapn=viscv;
 info->pn=P;
 info->tn=T;
 info->entp_liqn=entl;
 info->entp_vapn=entv;
}

int propfis_elem(lista_elemento **inic,double enintmez,double demez,int
                 elem){
 double P,T,Sv,psat;
 transforma_propSint(demez,enintmez,&P,&T,&Sv);
 if(Sv<1.0 && Sv>0.0){
  if(!transf_Intr_propSint(inic,enintmez,demez,elem))return -1;
  return 1;
 }
 transforma_propS01(demez,enintmez,&P,&T,0);
 psat=p_sat(T);
 if(P<psat){
  introduce_propPT_Sint(inic,psat,T,0.00001,elem);
  return 1;
 }
 else if(fabs(Sv)<1.0){
  if(!transf_Intr_propScero(inic,enintmez,demez,elem))return -1;
  return 1;
 }
 else if(fabs(Sv)>1.0){
  if(!transf_Intr_propSuno(inic,enintmez,demez,elem))return -1;
  return 1;
 }
 else return 0; }                        /*si no hay rango no hay convergencia*/
```

```
void icara(lista_elemento *inffo,lista_conex *infor,double *d,double *a,
           double *k_abs,double *k_liqrel,double *k_vaprel,double
           *den_liq,double *den_vap,double *visc_liq,double *visc_vap,
           double *pmm,double *cond_term,double *tmm,double
           *entp_liq,double *entp_vap,double *cosang){
 lista_elemento *inf,*inic=princ;
 double dnint,dmint;
 *cosang=infor->cos_ang;
 dnint=infor->dist_N_Int_M;
 dmint=infor->dist_M_Int_N;                    /*localizo conect m en lista princ n en inf*/
 *d=dnint+dmint;                              /*hay prop de conect m en lista princ en inf*/
 *a=infor->area_interf;                       /*tengo prop de n en lista principal en inffo*/
 inf=hallar(&inic,infor->elem_conect);              /*obtengo prom de propiedades físicas*/
          /*encuentro prop en interc por polinomios de interp de lagrange de 1er orden*/
 *k_abs   =lagrange(dnint,dmint,inffo->k_absn,inf->k_absn);
 *k_liqrel=lagrange(dnint,dmint,inffo->k_liqreln,inf->k_liqreln);
 *k_vaprel=lagrange(dnint,dmint,inffo->k_vapreln,inf->k_vapreln);
 *den_liq =lagrange(dnint,dmint,inffo->den_liqn,inf->den_liqn);
 *den_vap =lagrange(dnint,dmint,inffo->den_vapn,inf->den_vapn);
 *visc_liq=lagrange(dnint,dmint,inffo->visc_liqn,inf->visc_liqn);
 *visc_vap=lagrange(dnint,dmint,inffo->visc_vapn,inf->visc_vapn);
 *cond_term=lagrange(dnint,dmint,inffo->cond_termn,inf->cond_termn);
 *entp_liq=lagrange(dnint,dmint,inffo->entp_liqn,inf->entp_liqn);
 *entp_vap=lagrange(dnint,dmint,inffo->entp_vapn,inf->entp_vapn);
 *pmm=inf->pn;
 *tmm=inf->tn;
}                                          /*manda prop y geometría del elemento en cuestión*/

void profgeo_elem(lista_elemento *info,double *voln,double *pnn,double
                  *tnn,double *den_roc,double *cp_roc){
 *voln   =info->vol_elem;
 *pnn    =info->pn;
 *tnn    =info->tn;
 *den_roc=info->den_roca;
 *cp_roc =info->cp_roca;
}
                                        /*calcula prop pert de un elem y se introducen a la lista*/
                 /*calcula también prop de ese mismo elem sin pert y las retorna a la lista*/
void prop_elem_pert(lista_elemento **inic,double demez,double
                    enintmez,int elemento_n){
 propfis_elem(inic,enintmez,demez,elemento_n);
}
```

```
int transf_Intr_propSint(lista_elemento **inic,double enintmez,double
                                                demez,int elem){
 double P,T,Sv;
 if(!transforma_propSint(demez,enintmez,&P,&T,&Sv))return 0;
 introduce_propPT_Sint(inic,P,T,Sv,elem);
 return 1;
}

void introduce_propPT_Sint(lista_elemento **inic,double P,double
                                                T,double Sv,int elem){
 double denl,denv,klrel,kvrel,viscl,viscv,entl,entv,cond_term,Sl;
 Sl=1.0-Sv;
 propPT_Sint(P,T,Sl,&denl,&denv,&viscl,&viscv,&entl,&entv,&klrel,&kvrel);
 intr_prop_lista(inic,P,T,denl,denv,klrel,kvrel,viscl,viscv,entl,entv,Sv,Sl,elem);
}

int transf_Intr_propScero(lista_elemento **inic,double enintmez,double
                                                demez,int elem){
 double P,T,denl,denv,klrel,kvrel,viscl,viscv,entl,entv,cond_term,Sv,Sl;
 if(!transforma_propS01(demez,enintmez,&P,&T,0))return 0;
 Sv=0.0; Sl=1.0; klrel=1.0; kvrel=0.0;
 propPT_Scero(P,T,&denl,&denv,&viscl,&viscv,&entl,&entv);
 intr_prop_lista(inic,P,T,denl,denv,klrel,kvrel,viscl,viscv,entl,entv,Sv,Sl,elem);
 return 1;
}

int transf_Intr_propSuno(lista_elemento **inic,double enintmez,double
                                                demez,int elem){
 double P,T,denl,denv,klrel,kvrel,viscl,viscv,entl,entv,cond_term,Sv,Sl;
 if(!transforma_propS01(demez,enintmez,&P,&T,1))return 0;
 Sv=1.0; Sl=0.0; klrel=0.0; kvrel=1.0;
 propPT_Suno(P,T,&denl,&denv,&viscl,&viscv,&entl,&entv);
 intr_prop_lista(inic,P,T,denl,denv,klrel,kvrel,viscl,viscv,entl,entv,Sv,Sl,elem);
 return 1;
}

void correccion(int *itm,double *emant,double *dmant,double *eim,
                                                double *dm)
{
 *itm=1;
 BD_propfis(emant,dmant);                    /*da prop fis de cada elem e intr a list*/
 intercambio(No_el-1,eim,dm,dmant,emant);                    /*k se cambia por k+1*/
}
```

```
void temp_anterior(double enintmez,double demez,double *T,double *P,
                   double *Sv)
{
 double psat;
 transforma_propSint(demez,enintmez,&*P,&*T,&*Sv);
 if(*Sv<1.0 && *Sv>0.0)return;
 transforma_propS01(demez,enintmez,&*P,&*T,0);
 psat=p_sat(*T);
 if(*P<psat){
  *P=psat;
  *Sv=0.00001;
  return;
 }
 else if(fabs(*Sv)<1.0)*Sv=0.0;
 else if(fabs(*Sv)>1.0){transforma_propS01(demez,enintmez,&*P,&*T,1);*Sv=1.0;}
}

void entrada(double *dt,double *maxt,double *perte,double *pertd,
             int *maxit,double *gtol,double *ftol,double *emant,double
             *dmant,double *dm,double *eim,int *nombre){
 register int i;
 *nombre=ejemplos(maxt,dt,pertd,perte,maxit,gtol,ftol,emant,dmant,dm,eim);
 if(!(*nombre)){
  prop_sist(No_el);
  est_trans(maxt,dt);
  fuente_sumidero();
  tolerancia(pertd,perte,maxit,gtol,ftol);
  cond_inic(No_el,emant,dmant,dm,eim);
 }
}

void inicia_matriz(void)                    /*importante inicializar*/
{
 matriz01();matriz02();
}

void tiempo_cpu(void)
{
 ttot=time(0)-tiempotot;
 tpar=time(0)-tiempo;
}
```

```
void transitorio(int *i,double *t, double *dt)
{
 if(cambiodt){
  *t=(*t)+DT[*i];
  *dt=DT[(*i)++];
 }
  else *t=(*t)+*dt;
 if(((*i)==10 || DT[*i]==0.0)&& cambiodt){
  cambiodt=0;
  *dt=DT[--(*i)];
 }
}
```