

alfap = Constante ecuación de calentamiento de resistencias eléctricas, 0.01.
xvp = Fracción de volátiles, 0.455.
xwp = Fracción de agua, 0.1.
xinfp = Fracción de volátiles retirados en tiempo infinito, 0.38.
rpp = Radio de la partícula de carbón, 0.001 m.
asupp = Área superficial de una partícula de carbón, $3.14 \cdot 10^{-6}$ m².
numparp = Numero de partículas en el lecho.
ko = Factor de frecuencia de la ec. cinética para la Volatilización.
E = Energía de activación de la reacción de volatilización, 83700 J/kg.
R = Constante universal de los gases, 8.31.
Do = Difusividad referencia de compuestos gaseoso en sólidos, $1 \cdot 10^{-7}$ m²/s.
Ed = Energía de activación de la difusión, 61380 J/kg.

INCOGNITAS

fmch = Flujo másico de Char kg/s.
xich = Fracción inmediata de materia volátil en el lecho.
fmmvp = Flujo másico de materia volátil kg./s.
mp = Masa actual de sólidos en el pirolizador kg.
hsrp = Altura sobre el reboce de salida de Char m.
vch = Velocidad del Char en el tubo de salida m/s.
volact = Volumen actual que ocupa el sólido.
qep = Potencia eléctrica de calentamiento wats.
qpp = Potencia eléctrica total perdida al ambiente wats.
qconvp = Potencia elect. perdida al ambiente por convección wats.
qradp = Potencia elect. perdida al ambiente por radiación wats.
qkondp = Potencia elect. perdida al ambiente por conducción wats.
cppip = Cp Volatilización (calores reacción y sensibles) J/kg^oK.
mop = Masa que tendría el volumen actual si fuera carbón mineral kg.
Dmv,ch = Difusividad del material volátil en el Char m²/s.
k = Coeficiente cinético de la reacción de Volatilización.
x = Fracción materia volátil con base en masa hipotética inicial mo.
denmixp = Densidad de la mezcla de sólidos kg./m³.
denapp = Densidad aparente de los sólidos contenidos en el lecho kg/m³.
denapca = Densidad aparente del carbón mineral en el lecho kg./m³.

VALORES PROPIOS DEL SIMULADOR

tiempocalen = Tiempo transcurrido desde ultima variación de potencia.
incremento = Incremento en el valor del tiempo en cada paso s.

A1.2 COMPORTAMIENTO DEL SIMULADOR

Las Figuras A1 y A2 muestran el comportamiento del pirolizador sin control, lazo abierto, cuando se inicia la simulación desde dos condiciones diferentes: valores de las variables

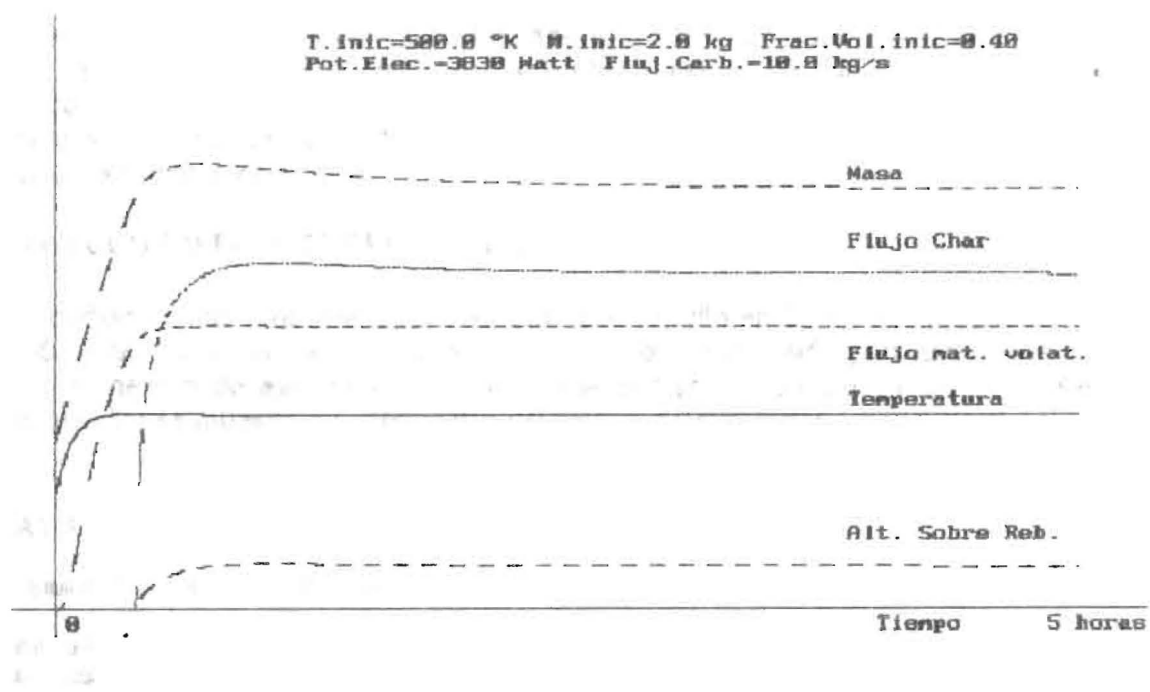


FIGURA A1. Pirolizador en lazo abierto (valores iniciales inferiores al estado estable).

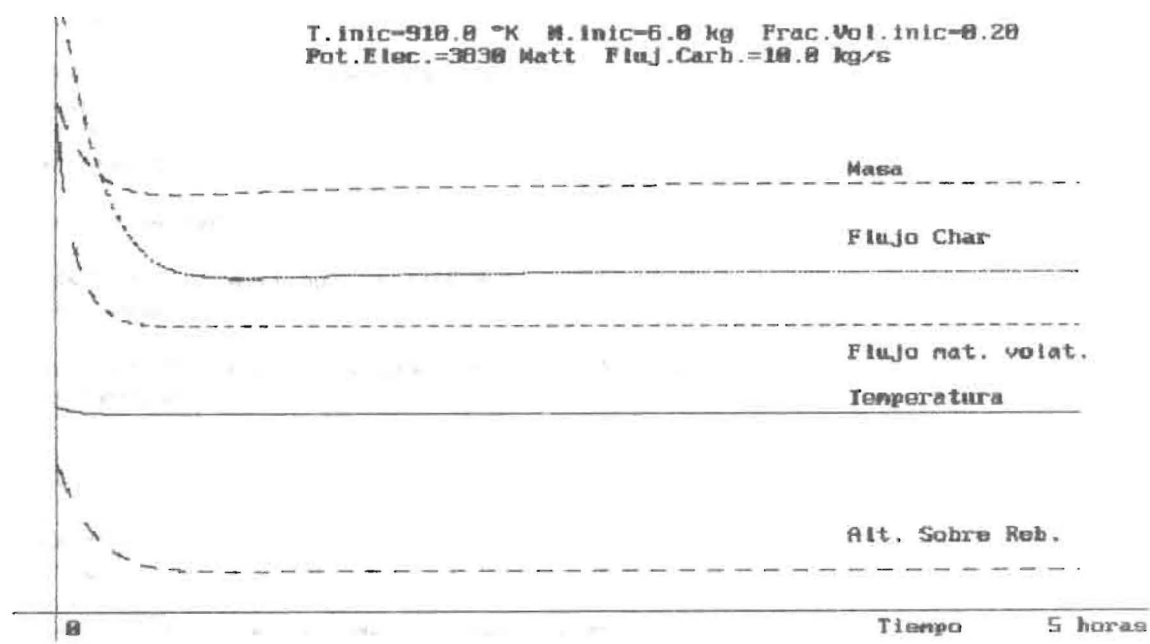


FIGURA A2. Pirolizador en lazo abierto (valores iniciales superiores al estado estable).

inferiores a los correspondientes al estado estable considerado, Figura A1, y valores de las variables superiores a los correspondientes al estado estable considerado, Figura A2. El comportamiento de las diversas variables y estado estable que se alcanza, corresponde muy bien con los datos reportados por el grupo de investigación en carbón activado en su informe (Aguirre y col., 1991).

A1.3 CODIGO FUENTE DEL SIMULADOR

El código que se presenta a continuación, esta escrito en C++ y se compiló en la versión 7.0 de Microsoft para este lenguaje. Los archivos de inclusión particulares, se presentan en el Anexo 5 de este trabajo, mientras que el archivo de inclusión que define el objeto pirolizador se muestra a continuación del código del simulador.

A1.3.1 Código Fuente del Simulador para el Pirolizador.

```
//SIMULADOR DEL PIROLIZADOR
```

```
#include<graph.h>
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<float.h>
#include<iostream.h>
#include<ejesgraf.h> //archivo particular.
#include<simpliro.h> //archivo del objeto pirolizador.

void main()
{
    int locx, locy, tiempo, incremento=1, tmuestreo=1;
    int sw1=0, sw2=0, sw3=0, sw4=0, sw5=0, sw6=0, sw7=0, sw8=0;
    float temppiro, mp, fmch, hsrp, fmmvp;
    float *semillas;

    semillas=new float[5];

    cout<<"\n\nEscriba la temperatura del PIROLIZADOR en K= ";
    cin>>semillas[0];

    cout<<"\n\nEscriba la masa actual en el PIROLIZADOR en kg= ";
    cin>>semillas[1];

    cout<<"\n\nEscriba la fraccion actual de volatiles del char= ";
    cin>>semillas[2];

    cout<<"\n\nEscriba la potencia electrica de estado estable en J/s = ";
    cin>>semillas[3];

    cout<<"\n\nEscriba el flujo masico de carbon crudo en kg/s = ";
    cin>>semillas[4];

    pirolizador pirolizadoruno(semillas);
    graphics_mode();
    draw_lines();
```

```

for(tiempo=0;tiempo<18000;tiempo+=incremento)
{
    temppiro=pirolizadoruno.predice_tp(tmuestreo, 0);

    mp=pirolizadoruno.muestramp();
    fmch=pirolizadoruno.muestrafmch();
    hsrp=pirolizadoruno.muestrahsrp();
    fmmvp=pirolizadoruno.muestrafmmvp();

    sw1++;
    sw3++;
    sw5++;
    sw7++;

    locx=newx(0.05*tiempo);
    locy=newy(-temppiro/3);
    _setpixel(locx,locy);

    if(sw1<50)
    {
        locx=newx(0.05*tiempo);
        locy=newy(-fmch*3e5);
        _setpixel(locx,locy);
    }
    else
    {
        sw2++;
        if(sw2>49)
        {
            sw1=0;
            sw2=0;
        }
    }

    if(sw3<150)
    {
        locx=newx(0.05*tiempo);
        locy=newy(-fmmvp*5e5);
        _setpixel(locx,locy);
    }
    else
    {
        sw4++;
        if(sw4>149)
        {
            sw3=0;
            sw4=0;
        }
    }

    if(sw5<200)
    {
        locx=newx(0.05*tiempo);
        locy=newy(-mp*125);
        _setpixel(locx,locy);
    }
    else

```



```

{
sw6++;
if(sw6>199)
{
sw5=0;
sw6=0;
}
}

if(sw7<250)
{
locx=newx(0.05*tiempo);
locy=newy(-hsrp*1100);
_setpixel(locx,locy);
}
else
{
sw8++;
if(sw8>249)
{
sw7=0;
sw8=0;
}
}
}

_settextposition(2,22);
printf("COMPORTAMIENTO DE VARIABLES EN EL PIROLIZADOR");

_settextposition(4,22);
printf("T.inic=%2.1f °K M.inic=%2.1f kg Frac.Vol.inic=%2.2f", semillas[0], semillas[1], semillas[2]);
_settextposition(5,22);
printf("Pot.Elec.=%1.0f Watt Flujo.Carb.=%2.1f kg/s", semillas[3], semillas[4]);

_settextposition(30,5);
printf("0");

_settextposition(30,62);
printf("Tiempo 5 horas");

cout<<"\n\n\n\nCONDICIONES FINALES EN EL PIROLIZADOR";
cout<<"\n\nLa Temperatura es = "<<temppiro<<" °K";
cout<<"\nLa Masa es = "<<mp<<" kg";
cout<<"\nEl Flujo de Char es = "<<fmch<<" kg/s";
cout<<"\nLa Altura Sobre el Rebosadero de salida es = "<<hsrp<<" metros";
cout<<"\nEl Flujo de Material Volatil es = "<<fmmvp<<" kg/s";

end_program();

delete [] semillas;
}

```

A.1.3.2 Código Fuente de Inclusión del Objeto Piroizador (simpipro.h).

```

//OBJETO PIROLIZADOR.
#include<stdlib.h>
#include<math.h>

```

```
#include<float.h>
```

```
class pirolizador
```

```
{
```

```
public:
```

```
    pirolizador(float *semilla_piro);  
    void reiniciopiro(float *semilla_piro);  
    float predice_tp(int deltatiempo, float deltacalor);  
    void cambiofmca(float nuevoflujo);  
    float muestraqep(){return(qep)};  
    float muestrafmch(){return(fmch)};  
    float muestramp(){return(mp)};  
    float muestrafmmvp(){return(fmmvp)};  
    float muestrahsrp(){return(hsrp)};  
    ~pirolizador();
```

```
private:
```

```
    float alfap, g, gc, ltubop, kfricp, xica;  
    float atubop, volutilp, aopro, xinfo;  
    float xirop, xirrp, rparp, asupp, dmvcao;  
    float edifp, runiv, cpch, cpgfp;  
    float fmgfp, cpmvp, cpca, epsilonp, aextexp;  
    float aextrp, aexpkp, xvp, xwp, ko, e;  
    float dench, denca, kcondp, hextp, uextp;  
    float text, tsupp, tgfp, volparp, xvha;  
  
    float fmch, qep, hsrp, cplrp, xich, xch, mp, k, fmmvprx, fmmvpdi;  
    float mmvp, mptp, denmixp, denapp, volactp, denapca, denmix, mop;  
    float dmvca, numparp, fmmvp, qpp, qkondp, qconvp, qradp, qrp, tp;  
    float mptpdelta, vchdelta, mpdelta, mmvpdelta, fmca, vca;  
    float qepo, vch;
```

```
};
```

```
pirolizador::pirolizador(float *semilla_piro)
```

```
{
```

```
    alfap=0.01;  
    g=9.8;  
    gc=1;  
    ltubop=0.31;  
    kfricp=29.1;  
    xica=0.455;  
    atubop=0.00114;  
    volutilp=0.00963;  
    aopro=0.0178;  
    xinfo=0.38;  
    xirop=0.46;  
    xirrp=0;  
    rparp=0.5e-3;  
    asupp=3.14e-6;  
    dmvcao=1e-7;  
    edifp=6.138e4;  
    runiv=8.31;  
    cpch=1464;  
    cpgfp=1196;  
    fmgfp=4.35e-3;  
    cpmvp=1520;  
    cpca=836;
```

```

epsilon=0.5;
aextep=1.24;
aextrp=0.615;
aexpkp=0.077;
xvp=0.455;
xwp=0.1;
ko=1e11;
e=8.37e4;
dench=800;
denca=1350;
kcondp=1.038;
hextp=15;
uextp=3.01e-9;
text=298;
tsupp=312;
tgflp=770;
volparp=5.236e-10;
xvhca=0.1;
tp=semilla_piro[0];
mp=semilla_piro[1];
xich=semilla_piro[2];
qep=semilla_piro[3];
fmca=semilla_piro[4];
vca=0.0005;
mtp=mp*tp;
mmvp=xich*mp;
mop=mp;
fmca=fmca/3600;
}

```

```

void piroizador::reiniciopiro(float *semilla_piro)

```

```

{
alfap=0.01;
g=9.8;
gc=1;
ltubop=0.31;
ktrcp=29.1;
xica=0.455;
atubop=0.00114;
volutilp=0.00963;
apiro=0.0178;
xinfp=0.38;
xirop=0.46;
xirp=0;
rparp=0.5e-3;
asupp=3.14e-6;
dmvco=1e-7;
edlfp=6.138e4;
runiv=8.31;
cpch=1464;
cpgfp=1196;
fmgfp=4.35e-3;
cpmvp=1520;
cpca=836;
epsilon=0.5;
aextep=1.24;
aextrp=0.615;
aexpkp=0.077;

```

```

xvp=0.455;
xwp=0.1;
ko=1e11;
a=8.37e4;
dench=800;
denca=1350;
kcondp=1.038;
hextp=15;
uextp=3.01e-9;
text=298;
tsupp=312;
tgflp=770;
volparp=5.236e-10;
xvhca=0.1;
tp=semilla_piro[0];
mp=semilla_piro[1];
xich=semilla_piro[2];
qep=semilla_piro[3];
fmca=semilla_piro[4];
vca=0.0005;
mtp=mp*tp;
mmvp=xich*mp;
mop=mp;
fmca=fmca/3600;
)

```

```

float piroizador::predice_tp(int deltat tiempo, float deltacolor)
{
    int tiempo, incremento=1, tiempocalen=0;
    float qepdelta;

    qepo=qep;
    qepdelta=deltacolor;

    for(tiempo=0; tiempo<deltatiempo; (tiempo+=incremento))
    {
        xich=mmvp/mp;

        denmixp=dench*(1-xich)+denca*xich;
        denapp=(1-epsilonp)*denmixp;

        volactp=mp/denapp;

        hsrp=(volactp-volutilp)/apiro;
        if(hsrp<0)
            vch=0;
        fmch=vch*dench*atubop;

        denapca=(1-epsilonp)*denca;
        mop=volactp*denapca;
        xch=(mop-mp)/mop;
        k=ko*exp(-e/(runiv*tp));
        fmmvprx=mop*k*(xinfp-xch);
        if(fmmvprx<0)
            fmmvprx=0;

        dmvcas=dmvcao*exp(-edfip/(runiv*tp));
    }
}

```

```

numparp=mp/(volparp*denmixp);
fmmvpci=dmvca*((xirop-xirrp)/rparp)*asupp*numparp*denca;

if(fmmvpci<fmmvpci)
  fmmvpci=fmmvpci;
else
  fmmvpci=fmmvpci;

tiempocalen+=incremento;
qep=qepo+qepdelta*(1-exp(-alfap*tiempocalen));

qconvp=hexp*aextexp*(tsupp-text);
qradp=uexp*aextrp*(pow(tp,4)-pow(text,4));
qkondp=kcondp*aexpkp*(tp-text);
qpp=qconvp+qradp+qkondp;

cpipr=(3.44e-5*(tp-273)+(0.33+5.49e-4*(tp-273))*(xvp/(1-xwp)))^4.1833e3;
qrp=fmmvpci*cpipr*(tp-text);

mptdelta=(fmca*cpca*tp-fmgfp*cpgfp*(tp-tgfp)-fmmvpci*cpmvpci*tp-fmch*cpch*tp
+qep-qpp-qrp)/(cpch);

mpdelta=fmca-fmch-fmmvpci-fmca*xvhca;

mmvpci=fmca*xica-fmch*xich-fmmvpci;

mpt=mpt+mptdelta*incremento;

mp=mp+mpdelta*incremento;

mmvpci=mmvpci+mmvpci*incremento;
if(mmvpci<0)
  mmvpci=0;
vchdelta=(hsrp/600)-kfricp*pow(vch,2);
vch=vch+vchdelta*incremento;
if(vch<0)
  vch=0;
tp=mpt/mp;
}
return(tp);
}

void pirolizador::cambiofmca(float nuevoflujo)
{
  fmca=nuevoflujo;
}

pirolizador::~pirolizador()
{
}

```