7.2. MANUAL DEL USUARIO.

Q E N Q O

INTERPRETE PARA MODELOS EN DINAMICA DE SISTEMAS

Contenido del Manual:

- Introducción.
- Ambiente integrado.
- Funciones disponibles.
- Definición de funciones propias del usuario.
- Proposiciones.
- Definición de Subsistemas.

7.2.1. Introducción.

El intérprete que usted ha adquirido sirve para realizar las simulaciones de los modelos basados en la metodología de Dinámica de Sistema en un ordenador digital.

Puede ser utilizado en un ordenador compatible con la IBM/PC, con al menos 512 Kb de Memoria RAM, tarjeta gráfica CGA o HERCULES Monocromática, y un Diskette Drive.

Para empezar a utilizar el Intérprete cargue desde el Sistema Operativo MS/DOS lo siguiente :

A>QENQO [<Archivo>]

Con esta orden se ejecuta el intérprete Qenqo, donde <Archivo> es una opción para seleccionar un archivo de código ya escrito previamente. Si no se encuentra en el directorio o no se escribe el archivo se carga uno denominado SIN_NOMBRE.Q.

7.2.2. Ambiente Integrado.

El programa muestra un ambiente integrado que aparece en la figura siguiente. Mediante el uso de las flechas se seleccionan las opciones.

QENQO INTEPRETE PARA MODELOS EN DINAMICA DE SISTEMAS						
Archivo	Editar	Correr	Compilar	Globales	V.Sensibil	Salir
Fin de Compilación						
ARCHIVO:CINETICA.Q			()José Luis Segovia 1991			<-Mover->

- Seleccionar un archivo ya editado para su ejecución:

Mover el cursor hacia el menú de **Archivo,** luego pulsar <ENTER> y escribir el nombre del archivo incluyendo la extensión '.Q'.

- Editar un archivo.

Una vez seleccionado el nombre de un archivo, mover el cursor a la opción **Editar** y pulsar <ENTER>. El programa por defecto utiliza un editor de textos llamado NE.COM (Norton Editor).

- Compilar el Código fuente.

Seleccionando ésta opción, el archivo mostrado en la parte inferior de la pantalla se compila (producción del código intermedio) con la opción **Compilar,** para su posterior ejecución.

- Ejecución.

Mediante la opción **Correr** se generan los gráficos de los resultados. Previamente es posible cambiar las condiciones iniciales y/o seleccionar una variable para realizar su análisis de sensibilidad.

- Cambio de Condiciones Iniciales y Globales.

La opción **Globales** se utiliza para cambiar los valores iniciales de las variables de control y de nivel. Se

muestran en la ventana respectiva los valores de las variables de Control y de Nivel. Para seleccionar uno de ellos muévase con el cursor hasta ubicarse encima del valor y presione <Enter>, luego escriba el nuevo valor.

- Análisis de Sensibilidad.

Seleccionando la opción **VSensibil** se inicializan las condiciones para el estudio de Análisis de Sensibilidad de una variable sobre el modelo. Es decir los efectos de los cambios de esta variable (desde un valor inicial hasta otro final, con un incremento determinado) en el modelo. Todas las curvas de resultados se muestran en la pantalla, unas a continuación de otras. Ejemplo:

```
Nombre         :  POBLACION
Valor Inicial  :    1000.00
Valor Final    :   10000.00
Incremento     :    1000.00
```

Inmediatamente después de haber definido estas condiciones, ejecute la opción Correr para generar las gráficas de éste análisis.

- Salir.

Seleccionar esta opción para salir al Sistema Operativo.

7.2.3. Funciones disponibles.

Enseguida explicamos cada una de las funciones que dispone el intérprete.

Usted puede ejecutar las funciones en el programa. Las funciones retornan valores específicos de datos como SEN(). Las funciones pueden utilizarse en cualquier lugar de una expresión.

Para entender este documento:

Formato : Muestra la sintaxis propia para la instrucción.

Comentarios: Describe como se usa la instrucción.

Ver también: Muestra otras palabras clave para referencias.

Ejemplo : Muestra como usar la instrucción en programas.

Las funciones disponibles son:

ABS

Retorna el valor absoluto de una expresión X.
Formato :

```
v = ABS(X);
```

Comentario :
Puede especificar una expresión cualquiera para X. En v se carga el valor absoluto de esta expresión X.

Ejemplo :

```
Sistema Prueba;
Var a:AUX;
Control Desde_t = 1;
        Hasta_t = 10;
Inicio
  A = Tiempo + ABS(tiempo)/2;
  GRAFICA(A,1);
Fin.
```

COS
Retorna el coseno de un ángulo x, en radianes.

Formato:

```
v = COS(x);
```

Comentario:
El valor de x para ser calculado debe estar en radianes. Para convertir de grados a radianes, multiplique los grados por 180/π. Donde π=3.141593.

Ejemplo:

```
Sistema Prueba;
Var a,b : AUX;
Control dt = 0.01;
        Desde_t = 1;
        Hasta_t = 4;
Inicio
  b = tiempo;
  A = COS(b/100);
  Grafica(A,0);
Fin.
```

EXP
Retorna el valor de la función exponencial de la variable o expresión X.

Formato :
v = EXP(x)

Comentario:
Véase tipo de ejemplo de Cos.

LN
Logaritmo natural de una variable o Expresión X.

Formato:
v = LN(x)

Comentarios
El valor de x debe ser mayor que cero.

Ejemplo :

```
Sistema Prueba;
Var CP       :Nivel;
control P = 10;
Inicio
  P = ln(100);
  Gráfica(P);
Fin.
```

LOG
Logaritmo decimal de una variable o expresión X.

Formato
v= LOG(X)

Comentarios
El valor de x debe ser mayor que cero. Ejemplo véase LN.

RAIZC
Obtiene la raíz cuadrada de una expresión o variable x

Formato
v = RAIZC(X)

Comentarios
El valor especificado en x debe ser mayor o igual a cero.

Ejemplo:

```
Sistema Prueba;
Var P  : Aux;
control Desde_t=10;
        Hasta_t=20;
pon_grafica(tiempo,dt);
Inicio
       P = Raizc(tiempo);
       grafica(P,0);
Fin.
```

RND
Retorna un número aleatorio

Formato
v = RND;

Comentario
El número aleatorio está entre cero y uno.

SEN
Retorna el seno de un ángulo x, en radianes.

Formato:
v = SEN(x);

Comentario:
El valor de x para ser calculado debe estar en radianes. Para convertir de grados a radianes, multiplique los grados por 180/π. Donde π=3.141593.

Ejemplo:

```
Sistema Prueba;
Var a:AUX;
control
        dt = 0.01;
        desde_t = 0;
        hasta_t = 2;
pon_grafica(tiempo,dt);
Inicio
  A = SEN(tiempo);
  GRAFICA(A,1);
Fin.
```

7.2.4. Definición de Funciones propias del usuario.

El usuario puede definir sus funciones a ser utilizadas en un modelo. Debe hacerlo en lineas previas a la descripción del modelo donde se invoca la función. Ejemplo:

```
-------------------------------------------------------------
Sistema ejemplo;
var a, b, c:aux;
control dt = 0.001;
        desde_t = 1;
        hasta_t = 10;

{ ** Definición de Funcion Cubo de tipo auxiliar ** }
{ ** con parametro 'a' tipo auxiliar ** }

Funcion cubo(a:aux):aux;
var x:aux;                  { * Variable interna de función * }
inicio
   x = a;
   cubo = x*x*x;
fin;
inicio
   ...
   c = a+cubo(a); { ** Expresión de invocación a Cubo ** }
   ...
fin.
-------------------------------------------------------------
```

Como se aprecia en el ejemplo la función debe ser definida de algún tipo (Flujo, Nivel o Auxiliar). Si se va ha utilizar con fines operativos, es conveniente definirla de tipo Aux (Auxiliar) para ser usada en cualquier expresión.

7.2.5. Proposiciones.

Las proposiciones pueden ser usadas en el programa para realizar una acción como graficar, evaluar una ecuación, efectuar iteraciones y bifurcaciones.

```
-------------------------------------------------------------
GRAFICA
Gráfica de los estados de una variable X, frente al
tiempo u otra variable.
```

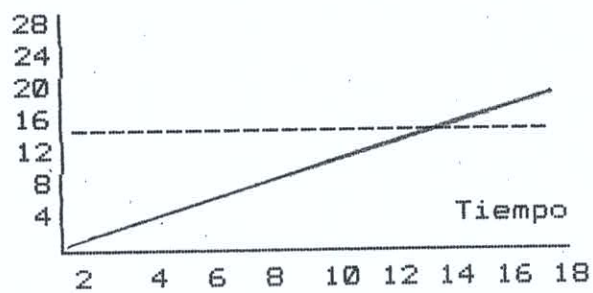
Formato:
Grafica(X, Tipo, [,Ni, Nf])

Comentario:
Trazado de la variable X desde el valor inicial Ni hasta el valor final Nf en el eje de las ordenadas. Si no se dan estos valores de intervalo, el intérprete hace la gráfica entre los valores máximos y mínimos hallados. El Tipo es un valor entero entre 0 y 4.

Ejemplo :

```
--------------------------
Sistema Prueba;
Var P,Q    : Aux;
control
    dt       = 1;
    Desde_t  = 00;
    Desde_t  = 18;
Pon_grafica(Tiempo,dt);  { Gráfica con intervalo DT}
Inicio
  P = 15;
  Q = tiempo;
  Grafica(P,1);
  grafica(Q,2);
Fin.
--------------------------
```

Resultados :



MIENTRAS ... HACER

Formato
MIENTRAS <Expresión Lógica> HACER
<Proposición>

Comentario
Ejecuta las instrucciones dentro <Proposición> siempre y cuando <Expresión Lógica> sea verdadera.

Ejemplo:

```
------------------------
Sistema Prueba;
Var    a, b :AUX;
control desde_t  = 1;
        hasta_t  = 40;
pon_grafica(tiempo,1);
Inicio
     B = 20
     a = 10;
  MIENTRAS b < a HACER
     Inicio
            B = a;
            a = a + 1;
     Fin;
     grafica(a,1);
Fin.
------------------------
```

SI .. ENTONCES .. SINO

Formato
SI <Expresión Lógica> ENTONCES <Proposición 1>
[SINO <Proposición 2>]

Comentario
Ejecuta la <Proposición 1> si es verdadera la <Expresión Lógica>, de otro modo procesa la <Proposición 2>.

Ejemplo:

```
------------------------
Sistema Prueba;
Var    a, b,c :AUX;
control  dt = 1;
       desde_t =1;
       hasta_t =10;
pon_grafica(tiempo,dt);
Inicio
  B = tiempo;
  a = 5;
  SI b < a ENTONCES
       Inicio
            c = a;
       Fin
    SINO c = b;
    grafica(c,1);
Fin.
------------------------
```

7.2.6. Definición de Subsistemas.

De acuerdo a la complejidad del sistema a estudiar y al criterio del analista, se puede descomponer un sistema en subsistemas que tienen sus propias variables y relaciones.

Ejemplo:

```
Sistema Ejemplo2;
var a,b,c:Aux;
control desde_t = 1;
        hasta_t = 10;
Pon_grafica(tiempo, dt);

 { ** Definición del subsistema Sub1, ** }
 { ** con parametros R, S auxiliares  ** }

SUBSISTEMA SUB1(R,S:AUX);
Inicio
      ....
      c = r*s;
      ....
fin;

Inicio              { ** Inicio del Sistema principal ** }
      ...
      Sub1(a,b);    {* Invocación al Subsistema *}
      ...
Fin;
```

Dentro de la definición de subsistemas se pueden definir constantes y variables internas. No es conveniente definir variables de control, ni proposiciones para gráficas dentro de un subsistema.

7.3. ALGUNOS EJEMPLOS.

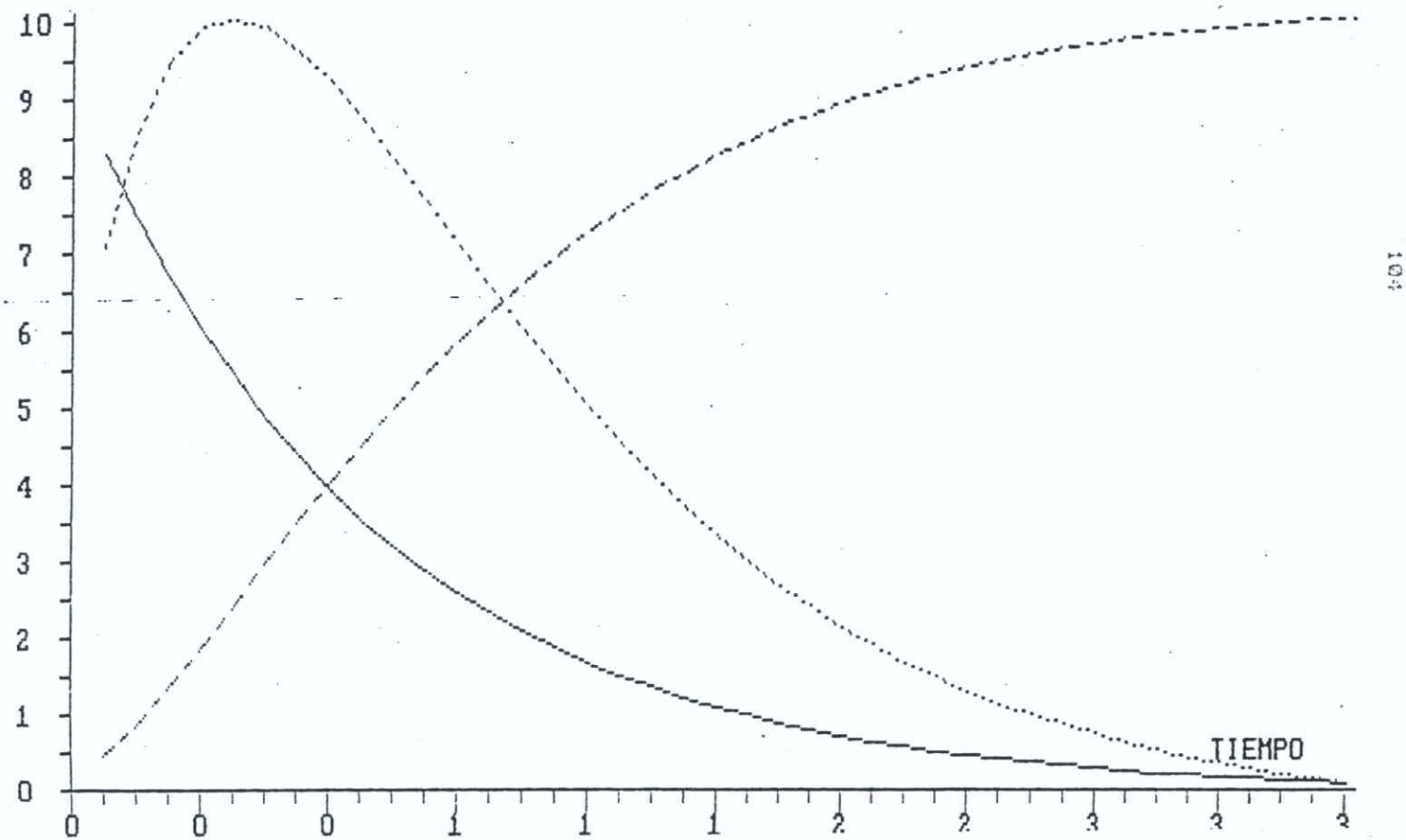
Enseguida se muestran dos ejemplos de modelos basados en Dinámica de Sistemas. Se incluye el código fuente y la gráfica de resultados.

7.3.1. MODELO DE CINETICA.

```
Sistema cinetica;
     {**  Tomado de Helfgott M., Vera E. 1989. Cap. 1  **}
     {**  Amaru Editores, Lima - Perú ** }
```

```
const
   k1 = 1 ;
   k2 = 2 ;


var fa, fb1, fb2, fc : flujo;
    a, b, c: nivel;

control
   dt = 0.02;              { * Incremento de Tiempo * }
   desde_t = 0 ;           {   Inicio de Tiempo     }
   hasta_t = 4;

   a = 200;                { Condiciones Iniciales    }
   b = 10;
   c = 0;

pon_grafica(tiempo,dt);
inicio
   fa = -k1*a;               { ** Ecuaciones de Flujo ** }
   fb1 = k1*a;
   fb2 = k2*b;
   fc  = k2*b;
   a := fa;                  { ** Ecuaciones de Nivel ** }
   b := fb1-fb2;
   c := fc;
   grafica(a,0);             { ** Ordenes para graficar ** }
   grafica(b,1);
   grafica(c,2);
fin.
```

* QENQO * RESULTADOS	LINEA	VARIABLE	FACTOR	V.MINIM	V.MAXIM
MODELO : CINETICA	———	A	19	3	196
	.....	B	5	3	56
	- - - - -	C	20	0	205
Presione <ENTER> para salir al menu.					



7.3.2. MODELO FIEBRE AMARILLA.

```
{** Este  modelo del  sistema de  transmision de la Fiebre
Amarilla  **}
 { ** está tomado de la referencia [9], p97. ** }
 { ** Los comentarios aparecen entre llaves ** }

sistema Fiebre_Amarilla;
const ptm = 500000;    { Población total de mosquitos }
      vmm = 18;        { Vida media de mosquito }
      dpmo  =  3;     {  Duración  de  la  peligrosidad del
mosquito }
      mip = 1;        { Mosquitos incubando por picadura }
      peim = 12;      { Período de incubación del mosquito }
      dim = 3;    { Duración de la infección en mosquitos }
      pifp = 1;         { Población infectada por picadura }
      dihu = 4.5;       { Duración de incubacion en hombre }
      dpc  = 4.5;       { Duración período de contagio }
      tmm =  0.1;       { Tasa de mortalidad }
      denf = 2.5;       { Duración de enfermedad }
      pdm   = 0.2;       { Picaduras por mosquito y por día}


var mpp, min, mif, pv, pin, pco, pen, pne:nivel;

      { ** significado de las variables de nivel ** }
      { mpp = Mosquitos potencialmente peligrosos }
      { min = Mosquitos en incubacion }
      { pv  = Población humana vulnerable }
      { pin = Población incubando }
      { pco = Población contagiada }
      { pen = Población enferma }
      { pne = Población inmune }

    nm,  mei,  msi,  mmi,  pif,  psi,pfc,  pse,  mnp, mor,
fpv:flujo;

      { ** significado de las variables de flujo ** }
      { nm   = Nacimientos de mosquitos }
      { mei  = Mosquitos que entran en incubación }
      { msi  = Mosquitos que salen de incubación }
      { mmi  = MOrtalidad de mosquitos infectados }
      { pif  = población infectada }
      { psi  = Población que sale de incubación }
      { pfc  = Población que termina el contagio }
      { pse  = Población que supera la enfermedad }
      { mnp  = Mosquitos que dejan de ser peligrosos }
      { mor  = Mortalidad de población }

      tpc, tpv, pto  : aux;

      { **  significado de las variables auxiliares ** }
```

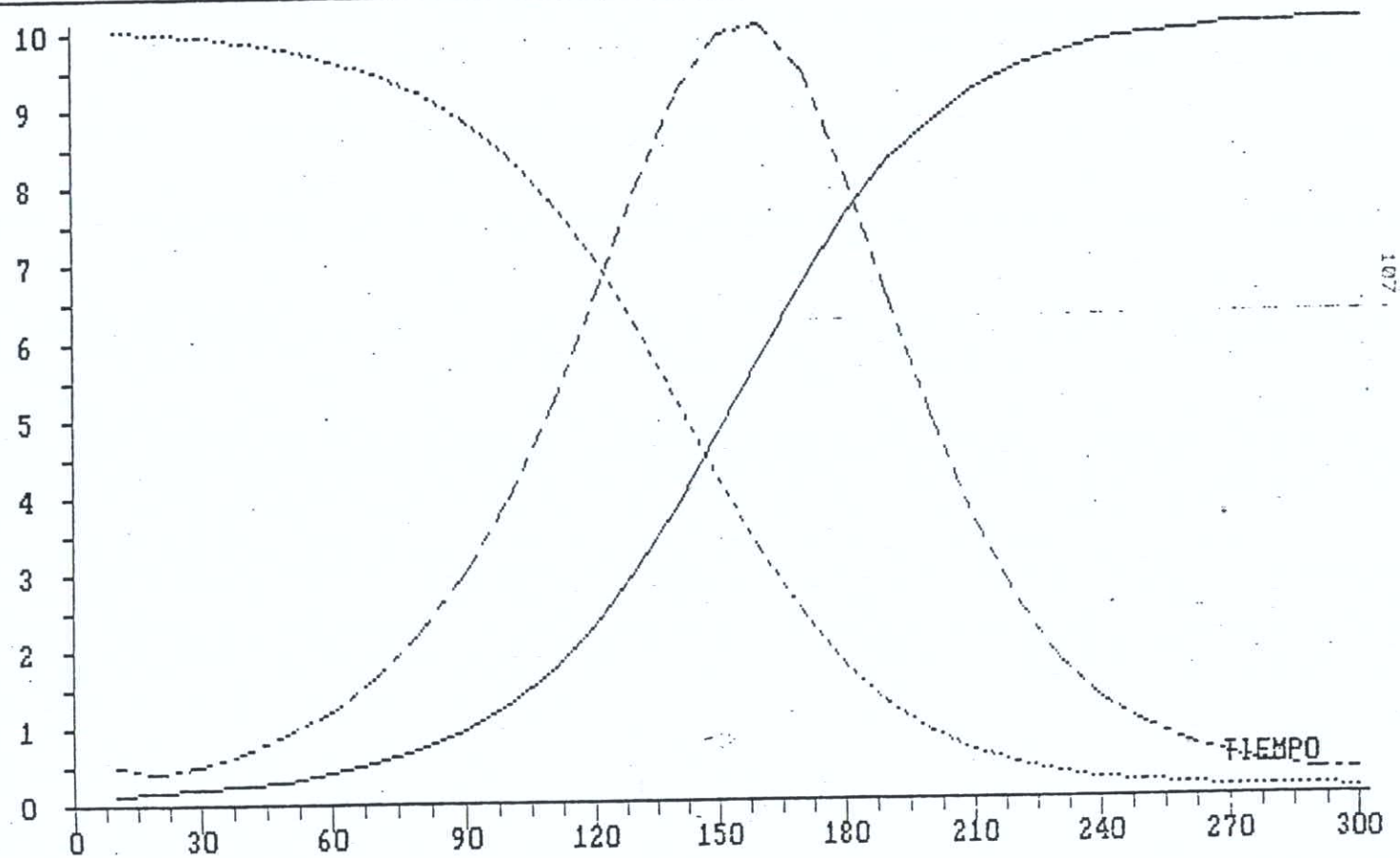
```
      { tpc = Personas contagiadas por picadura }
      { tpv = Personas vulnerables por picadura }
      { pto = Población total }

control
      dt = 1;                  { Incremento de un mes }
      desde_t = 0;           { Inicio del horizonte temporal }
      hasta_t = 300;           { Fin del horizonte temporal }
      mpp = 1;                 { Condiciones iniciales }
      min = 1;
      mif = 1;
      pv = 20000;
      pin = 100;
      pco = 1;
      pen = 1;
      pne = 1;

Pon_grafica(tiempo,10);    { Inicializacion de graficos }

inicio       { Descripción de las relaciones del sistema }
   nm  =  ptm/vmm;
   mei =  mpp*pdm*mip*tpc;
   mnp =  mpp/dpmo;
   msi =  min / peim;
   mmi =  mif / dim;
   pif = tpv*pdm*pifp*mif;
   psi = pin/dihu;
   pfc = pco/dpc;
   pse = (1-tmm)*pen/denf;
   mor = pen*tmm/denf;
   pto = pne+pen+pco+pin+pv;
   tpv = pv/pto;
   fpv = mif;
   tpc = pco/pto;
   mpp := ent(nm - mei - mnp);    { Ecuaciones de nivel }
   min := ent(mei - msi);
   mif := ent(msi - mmi);
   pv  := ent(- pif);
   pin := ent(pif - psi);
   pco := ent(psi - pfc);
   pen := ent(pfc - pse - mor);
   pne := ent(pse);
   grafica(pne,0);                { Orden para graficar }
   grafica(pv,1);
   grafica(mor,2);
fin.
```

* QENQO * RESULTADOS

MODELO : FIEBRE_AMARILLA

Presione <ENTER> para salir al menu.

LINEA	VARIABLE	FACTOR	V.MINIM	V.MAXIM
———	PNE	1616	1	16163
.....	PV	1785	2154	20000
- - - - - -	MOR	2	0	18



8. BIBLIOGRAFIA.

1. AHO, A., SETHI, R., ULLMAN, J. 1986 Compilers, principles, techniques, and tools. Reading Massachusetts, Addison-Wesley Publishing Company.

2. ARACIL, J. 1986 Introducción a la Dinámica de Sistemas. Madrid, Alianza Editorial Textos.

3. BORLAND INTERNATIONAL. 1987 Turbo Pascal Owner's Handbook. Version 4.0.

4. DAHL, O.J., DIJKSTRA, E.W. and. HOARE, C.A. 1972 Structured Programming. London, Acad. Press.

5. FORRESTER, J. 1962 Dinámica Industrial. Buenos Aires, El Ateneo.

6. FORRESTER, J. 1968 Principles of Systems. Cambridge Massachusetts, MIT Press.

7. HUNTER, R. 1985 Compilers. Their design and Construction Using Pascal. Singapore, John Wiley & Sons.

8. JOURDAN, C.P. 1972 Dynamine, Precompilateur pour Modeles de Type Dynamique des Systemes. Ecole Nationale Superieure des Mines de Saint-Etienne. Ministerio de L'Industrie, France.

9. MARTINEZ, S. y REQUENA, A. 1986 Dinámica de Sistemas, Simulación por Ordenador. Madrid, Alianza Editorial.

10. MEJIA PUENTE, M. 1988 Compilador de Pascal. Tesis, Pontificia Universidad Católica del Perú - Escuela de Graduados. Lima.

11. PICHLER, F. 1978 Decomposition Principles and Methods: An Overview. University of Linz. Linz, Austria.

12. PRATT, T. W. 1987 Lenguajes de Programación. Diseño e Implementación. (2da. Edición). México, Printice-Hall Hispanoamericana S.A.

13. SANCHIS-LLORCA, F.J. y GALAN-PASCUAL C. 1986 Compiladores. Teoría y Construcción. Madrid, Paraninfo S.A.

14. TENNENT, R.D. 1981 Principles of Programming Language. London, Prentice Hall International.