Práctica 1

IBERO
Ciudad de México • Tijuana

Predicción de respuesta dinámica de sistemas lineales e invariables en tiempo (LTI)

Integrantes del equipo

César Mauricio Arellano Velásquez
Raúl González Portillo

Profesor

César Arturo Ángeles Ruiz

Materia

Taller de Desarrollo de Aplicaciones

Introducción:

En matemática y computación, el **método de Euler**, llamado así en honor a Leonhard Euler, es un procedimiento de integración numérica para resolver ecuaciones diferenciales ordinarias (EDO) a partir de un valor inicial dado. El método de Euler es el más simple de los métodos numéricos para resolver un problema de valor inicial, y el más simple de los Métodos de Runge-Kutta. El método de Euler es nombrado por Leonhard Euler, quien lo trató en su libro *Institutionum calculi integralis* (publicado en 1768-1770).

El método de Euler es un método de primer orden, lo que significa que el error local es proporcional al cuadrado del tamaño del paso, y el error global es proporcional al tamaño del paso. El método de Euler regularmente sirve como base para construir métodos más complejos.

Consiste en dividir los intervalos que va de x0 a xf en n subintervalos de ancho h; o sea:

$$h = \frac{x_f - x_0}{n}$$

de manera que se obtiene un conjunto discreto de n + 1 puntos: x0, x1, x2, …, xn del intervalo de interés [x0, xf]. Para cualquiera de estos puntos se cumple que:

$$x_i = x_0 + ih \qquad 0 \leq i \leq n$$

La condición inicial $y(x_0) = y_0$, representa el punto $P_0 = (x_0, y_0)$ por donde pasa la curva solución de la ecuación del planteamiento inicial, la cual se denotará como $F(x) = y$ Ya teniendo el punto P_0 se puede evaluar la primera derivada de $F(x)$ en ese punto; por lo tanto:

$$F'(x) = \left.\frac{dy}{dx}\right|_{P_0} = f(x_0, y_0)$$

Se resuelve para y_1

$$y_1 = y_0 + (x_1 - x_0) f(x_0, y_0) = y_0 + h f(x_0, y_0)$$

Es evidente que la ordenada y_1 calculada de esta manera no es igual a $F(x_1)$ pues existe un pequeño error. Sin embargo, el valor y_1 sirve para que se aproxime $F'(x)$ en el punto $P = (x_1, y_1)$ y repetir el procedimiento anterior a fin de generar la sucesión de aproximaciones siguiente:

$$y_1 = y_0 + hf(x_0, y_0)$$
$$y_2 = y_1 + hf(x_1, y_1)$$
$$\vdots$$
$$y_{i+1} = y_i + hf(x_i, y_i)$$
$$\vdots$$
$$y_n = y_{n-1} + hf(x_{n-1}, y_{n-1})$$

Objetivos:

- Comprender de manera básica el funcionamiento de algoritmos de predicción a través de métodos matemáticos para tener una mejor toma de decisiones.
- Determinar una correcta complejidad de tiempo y espacio para optimizar procesos y codificación.
- Mejorar habilidades de investigación para resolución de problemas.
- Comprender el método de Euler para obtener la curva solución para predecir el comportamiento de una función.

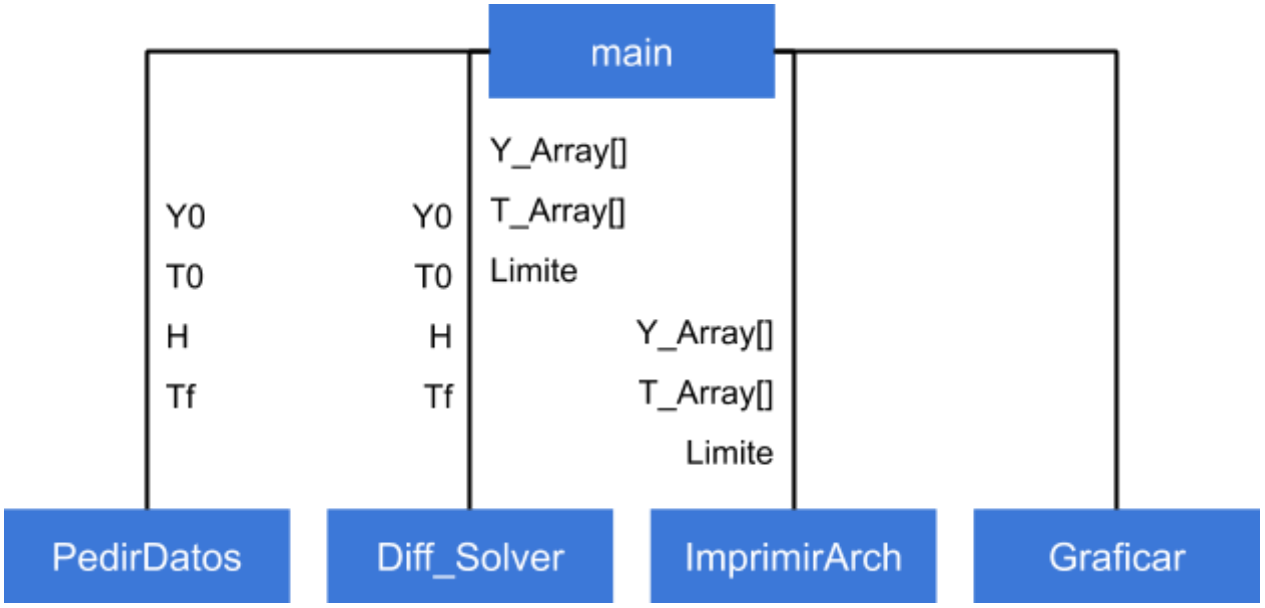
Análisis

Pseudocódigo:

```
#define return = retorna el valor de alguna operación o variable.
#define fopen = Abre el archivo (de texto o binario) que se le indica y junto al modo de ejecución.
#define fclose = Cierra el archivo que se le indique y que previamente esté abierto.
#define fprintf = Imprime en el archivo especificado.
#define wt = Abre el archivo en modo de escritura.
Principal ( | )
{
      PedirDatos ( | Y0, T0, H, Tf);
      Diff_Solver (Y_Array, T_Array, Y0, T0, H, Tf | Limite);
      ImprimirArch (Y_Array, T_Array, Limite | );
}

PedirDatos ( | Y0, T0, H, Tf)
{
      Imprimir (“Introduzca los siguientes datos:”);
      Imprimir (“T0:”);
```

```
      Leer (T0);
      Imprimir (“Y(T0)”);
      Leer (Y0);
      Imprimir (“H”);
      Leer (H);
      Imprimir (“Tf”);
      Leer (Tf);
}

Funcion (Y0, T0 | )
{
      return T0+Y0; //Función matemática establecida (x+y).
}

Diff_Solver (Y_Array, T_Array, Y0, T0, H, Tf | Limite)
{
      Desde i = 0; Hasta T0 <=Tf; i = i + 1
      {
            Y_Array[i] = Y0;
            T_Array[i] = T0;
            Y0 = Y0 + H * (Funcion(T0,Y0));
            T0 = H + T0;
      }
      ↑Limite = i;

}

ImprimirArch (Y_Array, T_Array, Limite | )
{
      Archivo = fopen(“LTI.txt”,”wt”);
      desde i=0;  hasta i < Limite; i = i + 1
      {
            fprintf (Archivo, T_Array[i],” ”, Y_Array[i]);
      }
      fclose(Archivo);
}
```

Diagrama IPO



Entradas Procesos y Salidas

Entradas	
Nombre	**Descripción**
Y0	Guarda la evaluación de la función Y en T0 (Y(T0))
T0	Guarda el valor inicial T0
h	Guarda el valor del step que dará la función por cada iteración
Tf	Indica el Tiempo donde terminará la gráfica
Procesos	
Nombre	**Descripción**
PedirDatos	Solicita los datos necesarios para realizar la predicción
Diff_Solver	Realiza una predicción en base a los datos del usuario utilizando el método de Euler y la guarda en dos arreglos.
ImprimirArch	Imprime los datos de los arreglos Y_Array[] y T_Array[] en un formato interpretable por

	GNUPlot
Graficar	Inicializa GNUPlot con los títulos de la gráfica y los ejes y le envía el archivo creado para graficarlo.

Salidas

Nombre	**Descripción**
Limite	Guarda cuantas iteraciones tuvo la función, sirve para determinar hasta donde se debe graficar
Y_Array[]	Un arreglo que guarda los resultados a graficar en el eje Y
T_Array[]	Un arreglo que guarda los resultados a gráficar en el eje X (T)

Código

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

void PedirDatos(float *Y0, float *T0, float *H, float *tf);
float Funcion(float T0,float Y0);
void Diff_Solver(float Y_Array[],float T_Array[], float Y0, float T0,float H ,float Tf,int *i);
void ImprimirArch(float Y_Array[],float T_Array[],int Limite);
void Graficar();

void Cargando (char Mensaje[])
{
  system ("clear");
  puts (Mensaje);
  printf ("\n");
  system ("sleep 0.15");
  system ("clear");
  puts (Mensaje);
  printf (".\n");
  system ("sleep 0.15");
  system ("clear");
  puts (Mensaje);
  printf ("..\n");
  system ("sleep 0.15");
  system ("clear");
  puts (Mensaje);
  printf ("...\n");
  system ("sleep 0.15");
}
```

```c
int main (void)
{
  float Y0,T0,H,Tf, Y_Array[100000], T_Array[100000];
  int Limite;
  PedirDatos(&Y0,&T0,&H,&Tf);
  Cargando("Resolviendo ecuación");
  Diff_Solver(Y_Array,T_Array,Y0,T0,H,Tf,&Limite);
  Cargando("Imprimiendo datos en archivo");
  ImprimirArch(Y_Array,T_Array,Limite);
  Cargando("Graficando con GNUPlot");
  Graficar();
  return 0;
}

void PedirDatos(float *Y0, float *T0, float *H, float *Tf)
{
  printf("Introduzca los siguientes datos:\n");
  printf("T0: ");
  scanf(" %f",T0);
  printf("Y0 (y(T0)): ");
  scanf(" %f",Y0);
  printf("Avance (h):");
  scanf(" %f",H);
  printf("Tf: ");
  scanf(" %f",Tf);
}

float Funcion(float T0,float Y0)
{
  return T0+Y0;
}
```

```c
void Diff_Solver(float Y_Array[],float T_Array[], float Y0, float T0,float H,float Tf,int *Limite)
{
  int i;
  for(i=0; T0<=Tf; i++)
    {
      Y_Array[i] = Y0;
      T_Array[i] = T0;
      Y0 = Y0+H*(Funcion(T0,Y0));
      T0=H+T0;
    }
  *Limite=i;
}
void ImprimirArch(float Y_Array[],float T_Array[],int Limite)
{
  FILE *Archivo;
  //printf("Limite: %d\n",Limite);
  Archivo = fopen("LTI.txt","wt");
  for(int i=0; i<Limite; i++)
      fprintf(Archivo,"%f %f\n",T_Array[i], Y_Array[i]);
  fclose(Archivo);
}
void Graficar()
{
  int i;
  char *AbrirGnuPlot[] = {"set title \"Método de Predición / Euler\"",
                          "set ylabel \"----Y--->\"",
                          "set xlabel \"----T--->\"",
                          "plot \"LTI.txt\" with lines"
  };
  FILE *VentanaGnuPlot = popen ("gnuplot -persist", "w");
  for (i=0; i<4; i++)
    fprintf(VentanaGnuPlot, "%s \n", AbrirGnuPlot[i]);
}
```

Ejecución del programa

```
cesar@Cesar-PC:~/Cursos/Taller de Apps/Prácticas/Práctica1$ ./p1.exe
Introduzca los siguientes datos:
T0: 0
Y0 (y(T0)): 1
Avance (h):0.01
Tf: 50
```

```
cesar@Cesar-PC:~/Cursos/Taller de Apps/Prácticas/Práctica1$ gnuplot

        G N U P L O T
        Version 5.2 patchlevel 6    last modified 2019-01-01

        Copyright (C) 1986-1993, 1998, 2004, 2007-2018
        Thomas Williams, Colin Kelley and many others

        gnuplot home:     http://www.gnuplot.info
        faq, bugs, etc:   type "help FAQ"
        immediate help:   type "help"  (plot window: hit 'h')

Terminal type is now 'wxt'
gnuplot> plot "LTI.txt" with lines, exp(x)*(-exp(-x)*x-exp(-x)+2) with lines
```

Demostración a través de gráfica.