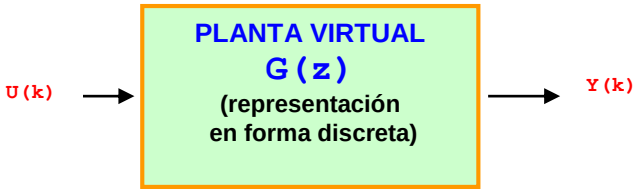
1. Respuesta de un sistema en Lazo Abierto.

Tenemos un modelo matemático de una planta; dicho modelo debe estar representada en **Transformada Z,** para luego llevarla a **Ecuaciones en Diferencias** y poder implementarla mediante un programa en C.



$$G(z) = \frac{0.4673z^{-1} - 0.3393z^{-2}}{1 - 1.5327z^{-1} + 0.6607z^{-2}}$$

El objetivo es hacer un programa en C que obtenga la respuesta del sistema frente a un escalón.

Solución:

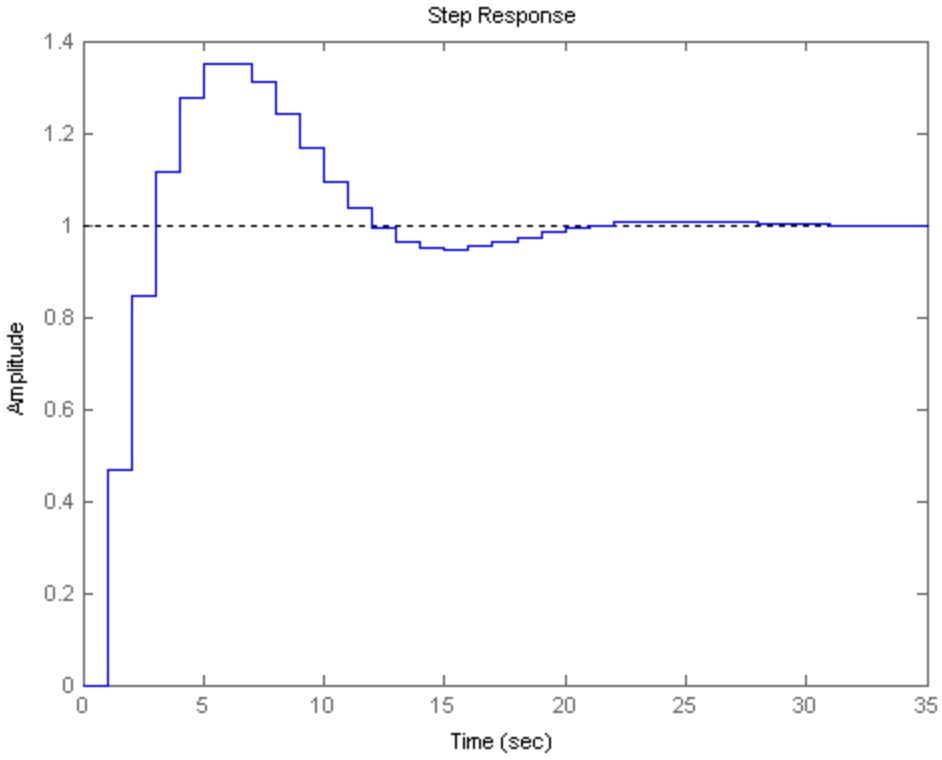
Para tener una idea de cuál sería la respuesta en el tiempo del sistema frente a un escalón, analicémoslo desde matlab:

```
close all; clear all; clc;

%usando dstep
num =[0.4673  -0.3393];
den=[1 -1.5327 0.6607];
dstep(num,den,35)

%usando dlisim
U=ones(1,35);
figure
dlsim(num,den,U)
```

La respuesta en el tiempo es la que se muestra. De allí podemos observar que el sistema es estable y que en un tiempo de 35 unidades es suficiente para la simulación.



Ahora, el sistema G(z) deberemos llevarlo a ecuaciones en diferencias por tanto hay que realizar los siguiente:

Sea $G(z) = \frac{C(z)}{R(z)}$, entonces el sistema G(z) representado en ecuaciones en diferencias es:

```
Ck=1.5327*Ck_1 - 0.6607*Ck_2 + 0.4673*Rk_1 - 0.3393*Rk_2;
```

La implementación de esta ecuación en lenguaje C seria la siguiente:

```
#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#include <math.h>

FILE *hfiletxt;

//---------------- Inicio de Programa ----------------------
void main(void)
{
 int n;
 float Rk;                          //amplitud del escalon
 float respuestaLA[100];      //respuesta en el tiempo
 float Rk_1=0, Rk_2=0;
 float Ck=0;
 float Ck_1=0,Ck_2=0;

 int puntos=35;

 Rk=1;
 for(int t=0;t<puntos;t++){
      Ck=1.5327*Ck_1 - 0.6607*Ck_2 + 0.4673*Rk_1 - 0.3393*Rk_2;
      Rk_2 = Rk_1;      //se actualiza el mas pasado
      Rk_1 = Rk;        //se actualiza el pasado
      Ck_2 = Ck_1;      //se actualiza el mas pasado
      Ck_1 = Ck;        //se actualiza el pasado

      respuestaLA[t]=Ck;
 }

//--------------------escribir archivo--------------------

      hfiletxt= fopen((char*)("c:\\resultado.txt"),"wb+");
      if(hfiletxt!=NULL)
      {
            for(n=0;n<puntos;n++)
                  fprintf(hfiletxt,"%lf ",respuestaLA[n]);

       fclose(hfiletxt);
      }

}
```

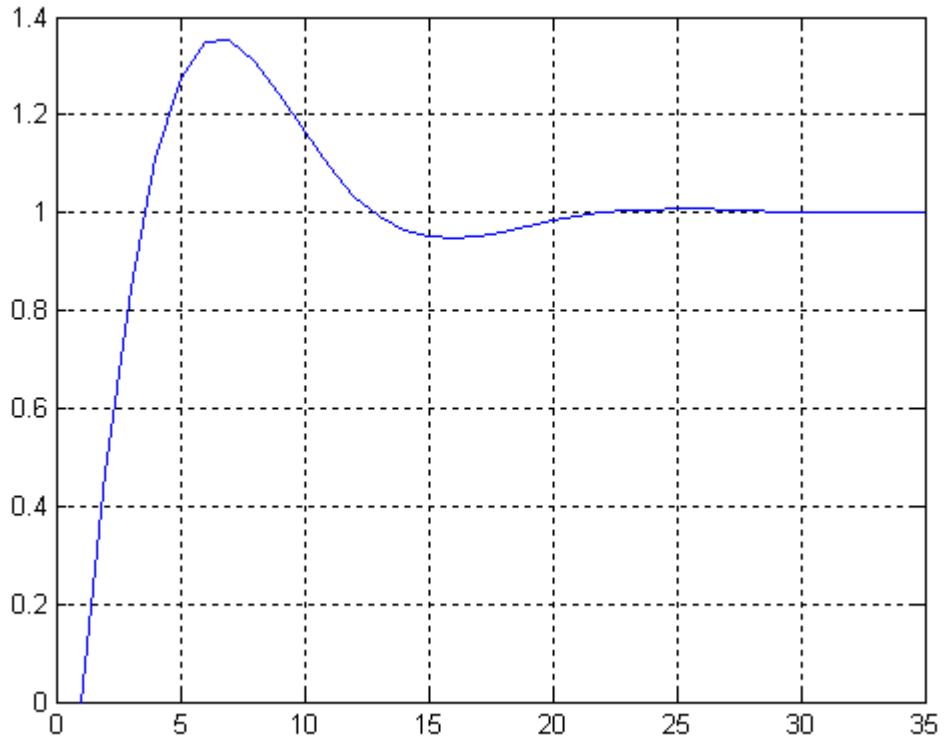
La respuesta ha sido almacenada en un array **`respuestaLA[100]`**. Luego para poder observar su resultado de manera gráfica, dicho buffer lo guardamos en un archivo para que posteriormente solamente sea dibujado desde matlab

Usamos un archivo script Matlab para plotear el resultado:

```
% lee un archivo binario enviado desde C++ par verificar
% si el procesamiento en c++ se realizo bien
% ----------------------------------------
clear all;close all

fid = fopen('c:\resultado.txt','r');
%lectura directa en A se almacena en una sola columna
A = fscanf(fid,'%f');
fclose(fid);
plot(A)
grid
```

El ploteo se muestra en la figura, por tanto podemos confirmar que nuestra simulación en Lenguaje C ha sido correcto.



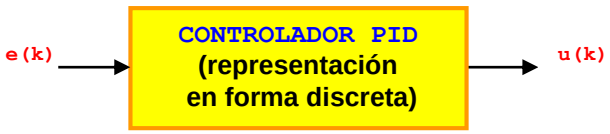
2. Control PID de una Planta Virtual

2.1 Implementación del algoritmo de control PID

Nuestro objetivo es implementar en lenguaje C un controlador PID o PI para la planta del ejercicio anterior. El algoritmo de control PID en tiempo continuo es el siguiente:

$$u(t) = Kp * e(t) + Ki \int e(t) + Kd \frac{de(t)}{dt}$$

Esta ecuación para ser implementada en una función en C debería ser discretizada y representada en ecuaciones en diferencias:



$$u(k) = u(k-1) + Kp[e(k) - e(k-1)] + Ki.T.e(k) + \frac{Kd}{T}[e(k) - 2e(k-1) + e(k-2)]$$

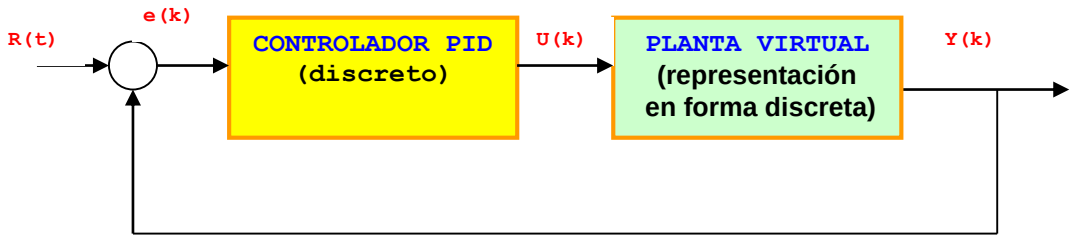
Note que hay distintas formas de discretizar un algoritmo PID. Trate de investigar una forma diferente a la dada aquí.
Una vez que se tiene el controlador en ecuaciones en diferencias, hay que implementarlo en lenguaje C. El algoritmo de control PI se muestra a continuación:

```
void algoritmo_pid(void)
{
e_act= data2-data1;
salida=salida1+kp*(e_act-e_ant)+ki*e_act*T;
e_ant=e_act;
salida1=salida;
}
```

```
Donde:
data2       : set point
data1       : valor del proceso
salida1     : señal de control anterior
salida      : señal de control actual
e_act       : error actual
e_ant       : error anterior
```

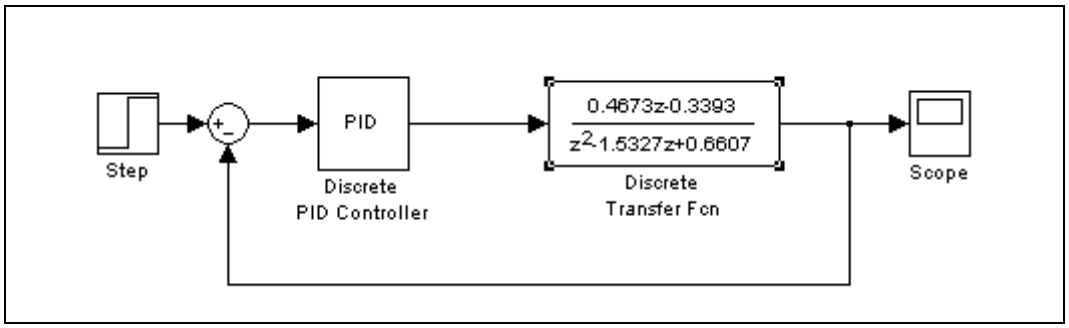
2.2 Control PI de la planta

Finalmente como ya tiene una planta virtual, y además se tiene el algoritmo de control; entonces tratemos de implementar en lenguaje C el control en lazo cerrado. El esquema que se simular es el que se muestra en la figura:

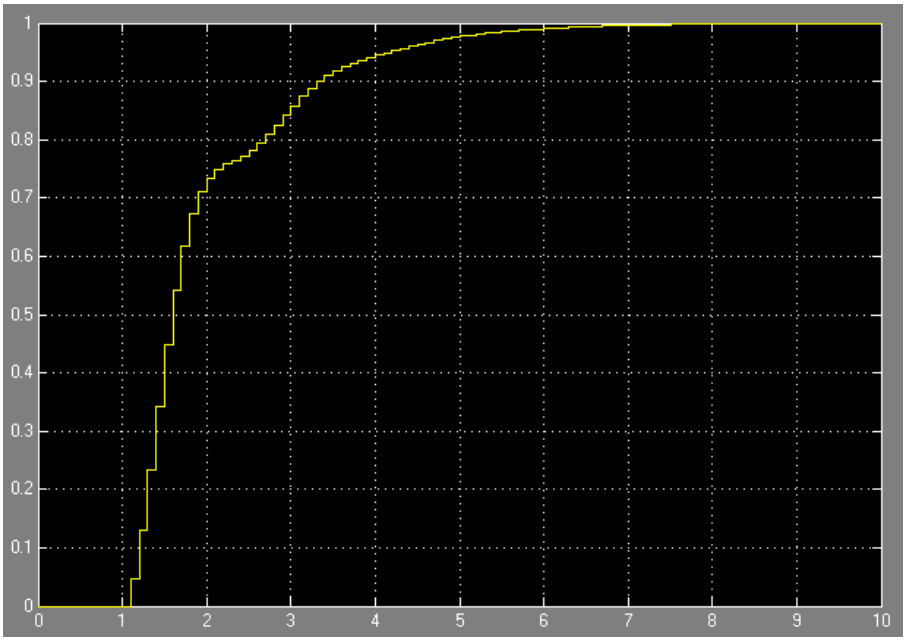


Solución:

Para tener una idea de cuál sería la respuesta en el tiempo del sistema en Lazo cerrado, analicémoslo desde simulink:



La respuesta en el tiempo es la que se muestra. De allí podemos observar que el sistema se logra controlar.



a continuación implementamos el programa de control en lenguaje C:

```
#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#include <math.h>

FILE *hfiletxt;

float planta(float Rk);
float algoritmo_pi(float data2, float data1);

float respuestaLC[100]; //respuesta en el tiempo
float T=100; //en milisegundos
//---------------- Inicio de Programa ----------------------
void main()
{
      float SP=1;
      float static salida_proceso1=0,u=0;
      float salida_proceso=0;
      float valor_sp,sensor;

      int puntos=90;


      //---------------- realizacion del control PI -----------
      printf("ingrese un valor de SP -->");
      scanf("%f",&SP);

      for(int t=0;t<puntos;t++){

            valor_sp= SP; //actualiza SP
            sensor = salida_proceso;
            u=algoritmo_pi(valor_sp,sensor);
            salida_proceso=planta(u);
            respuestaLC[t]= salida_proceso;

            }


      //----------------- escribir archivo--------------------

      hfiletxt= fopen((char*)("c:\\resultado.txt"),"wb+");
      if(hfiletxt!=NULL)
      {
            for(int n=0;n<puntos;n++)
                  fprintf(hfiletxt,"%lf ",respuestaLC[n]);
       fclose(hfiletxt);
      }
}


//.....continua
```

```
//-----------------------------------------------------
float planta(float Rk)
{
      static float Rk_1=0,Rk_2=0;
      static float Ck_1=0,Ck_2=0;
      float Ck=0;

      Ck=1.5327*Ck_1 - 0.6607*Ck_2 + 0.4673*Rk_1 - 0.3393*Rk_2;

      Rk_2 = Rk_1; //se actualiza el mas pasado
      Rk_1 = Rk;
      Ck_2 = Ck_1; //se actualiza el mas pasado
      Ck_1 = Ck;

      return Ck;   //retorna la salida del proceso en un instante
}
//-----------------------------------------------------
float algoritmo_pi(float data2, float data1)
{
    static float salida_ant=0,e_ant=0;
      static float Ck_1=0,Ck_2=0;
      float e_act,salida,kp,ki;

      kp = 0.1;   //actualiza Kp
      ki = 1.0;   //actualiza Ki

      e_act=data2-data1; //SP - Vp
      salida=salida_ant + kp*(e_act-e_ant)+ki*e_act*(T/1000.0);
      e_ant=e_act;
      salida_ant=salida;

      return salida;
}
```

Finalmente la salida seria la siguiente: