UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO Y VALIDACIÓN DE ESTRATEGIAS DE CONTROL DE FORMACIONES EN SISTEMAS MULTIRROBOT EN UNA PLATAFORMA VIRTUAL BASADA EN EASY JAVA SIMULATIONS.

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Jaime Sancha Miguel

TUTORIZADO POR

Antonio González Sorribes

COTUTORIZADOADO POR

Pedro José García Gil

CURSO ACADÉMICO: 2020/2021

ÍNDICE DE DOCUMENTOS:

ÍNDICE DE LA MEMORIA:

ÍNDICE DE FIGURAS:

ÍNDICE DE SIGNOS:

Abreviaturas:

LDR	Lugar de las raíces.
FDT	Función de transferencia.
Ec. en dif.	Ecuación en diferencias.
MIMO	Multiple-input Multiple-output.
SISO	Single-input Single-output.
EJSS/EJS	Easy Java/Javascript Simulations.

Constantes:

C	Coeficiente de rozamiento.
M	Masa del robot.
F	Ganancia de la acción de los actuadores.
T	Periodo de muestreo.
m	Grado del polinomio del numerador de la planta.
n	Grado del polinomio del denominador de la planta.
μ	Grado del polinomio del numerador del regulador.
η	Grado del polinomio del denominador del regulador.
Kv	Ganancia de la velocidad lineal.
$K\omega$	Ganancia de la velocidad angular.
p1,c1	Polos y ceros de la planta.
Kbc	Ganancia en bucle cerrado.
g1,q1	Polos y ceros diseñados para el controlador.
b	Ganancia diseñada para el controlador.
$\alpha 1,2,3$	Posiciones de los ceros añadidos.

Variables:

xpos/ypos	Posición del agente en x e y.
xvel/yvel	Velocidad del agente en x e y.
xref/yref	Valores de las referencias en x e y.
v	Velocidad lineal
ω	Velocidad angular
ρ/*rho*	Distancia al objetivo/coordenadas.
Φ/*phi*	Orientación en radianes respecto a las coordenadas.
ux/uy	Acción de control en x e y.
uw/thw	Orientación en radianes del agente.
w	Orientación del agente en el caso no lineal.

1. Resumen.

En este proyecto se estudiará el comportamiento de distintos sistemas de control de formaciones en una maqueta modelada y programada mediante el programa Easy Java/Javascript Simulations en la que se deberá dirigir los robots a unas posiciones absolutas elegidas por el usuario.

En primer lugar, se diseñarán las leyes de control con el apoyo de MATLAB y las librerías rltool o Simulink. Se tendrá en cuenta distintos modelos dinámicos para describir la dinámica de cada uno de los agentes, incluyendo robots con restricciones no holonómicas. En segundo lugar, se programará la maqueta virtual mediante Easy Java Simulations para finalmente validar las estrategias de control propuestas.

1. Abstract

In this project, the behavior of different training control systems will be studied in a model modeled and programmed using the Easy Java / Javascript Simulations program in which the robots must be directed to absolute positions chosen by the user.

First, the control laws will be designed with the support of MATLAB and the rltool or Simulink libraries. Different kinematic models will be taken into account to describe the dynamics of each of the agents, including robots with non-holonomic restrictions. Second, the virtual model will be programmed using Easy Java Simulations to finally validate the proposed control strategies.

1. Resum

En aquest projecte s'estudiarà el comportament de diferents sistemes de control de formacions en una maqueta modelada i programada mitjançant el programa Easy Java / JavaScript Simulations en la qual s'haurà de dirigir els robots a unes posicions absolutes triades per l'usuari.

En primer lloc, es dissenyaran les lleis de control amb el suport de MATLAB i les llibreries rltool o Simulink. Es tindrà en compte diferents models cinemàtics per descriure la dinàmica de cada un dels agents, incloent robots amb restriccions no holonómicas. En segon lloc, es programarà la maqueta virtual mitjançant Easy Java Simulations per finalment validar les estratègies de control proposades.

2. Agradecimientos.

Este trabajo de fin de grado marca el mayor punto de inflexión de toda mi carrera estudiantil por ello tras todos estos años de esfuerzo y dedicación me gustaría agradecerles a mis padres todo lo que han hecho por mí sin los valores de responsabilidad y trabajo esto nunca habría sido posible. También agradecer a todos mis amigos y compañeros que he hecho durante esta carrera por tantos momentos juntos, tantas tardes estudiando, tantos trabajos, proyectos y horas compartidas sin ellos y su apoyo anímico y emocional llegar a este punto habría sido imposible.

Finalmente, a mi tutor Antonio que con su manera de enseñar me convenció para escogerlo para ser mi guía durante esta recta final. Gracias por ese desempeño, tiempo y paciencia invertidos en mí.

3. Introducción.

3.1. Control de formaciones en sistemas multirrobot.

En primer lugar, se define como un sistema multirrobot a aquel espacio en el que un grupo de agentes que interactúan entre sí y a su vez son capaces de resolver un problema planteado forma autónoma. Dentro de este campo el control de formaciones se encuentra en pleno auge gracias a la necesidad del desarrollo de una industria que cada vez tiende a una mayor automatización. Por ello está recibiendo una especial atención por parte de investigadores y desarrolladores del sector.

Para un mejor entendimiento de su funcionamiento se debe tener en cuenta que estos agentes son parcialmente autónomos por lo que es necesaria la creación y desarrollo de una serie de leyes de control que a su vez integren la posibilidad de modelar formaciones. Para estas leyes se pueden aplicar tres tipos de controles distintos tal y como se ha probado en [1]:

- Control por posición: los agentes son capaces de detectar su propia posición respecto al sistema de coordenadas global a su vez estos pueden controlarla para así lograr alcanzar una posición global prescrita.
- Control basado en el desplazamiento: consiste en que los agentes pueden controlar los desplazamientos de sus agentes vecinos para llegar a una formación final deseada. Para ello, estos agentes deben detectar la orientación respecto a la posición global a alcanzar de sus vecinos. Cabe destacar que ellos mismos no tienen la necesidad de conocer sus propias coordenadas ni del sistema de coordenadas global.
- Control basado en la distancia: se controlan activamente las distancias entre los agentes para lograr la formación que viene dada por la misma distancia deseada para estos. Los mismos agentes deberán de detectar las posiciones relativas de sus vecinos con respecto a sus mismos sistemas de referencia locales. En este caso no será necesario que las orientaciones de cada uno de los participantes estén necesariamente controladas por sus vecinos.

Hay que tener en cuenta que la diferenciación anterior es útil para caracterizar los esquemas de control de la formación en términos de los requisitos sobre la capacidad de detección y la topología de interacción. El control basado en la posición es beneficioso en términos de interacción entre los robots, no obstante, requiere que los agentes estén equipados con sensores más avanzados que en otros enfoques. Por otra parte, el control basado en la distancia es ventajoso en términos de la capacidad de detección, pero requiere más interacciones entre los participantes ya que como se ha comentado anteriormente es necesario conocer la orientación de los vecinos. Finalmente, el control basado en el desplazamiento es moderado en términos tanto de la capacidad de detección como de la topología de interacción en comparación con otros enfoques.

Primero, un esquema de control de formación puede clasificarse en centralizado o descentralizado esto se puede traducir en si es necesario el control independiente de los distintos subsistemas o no mediante la acción de un coordinador global. Sin embargo, tal categorización no es apropiada para una revisión de varios esquemas de control de formación. De hecho, bajo este criterio, encontramos que la mayor parte de la formación por un coordinador global, nos referimos a una entidad que recopila información de los agentes, toma una decisión y luego distribuye algún comando de coordinación a los agentes. Los esquemas de control de la capacidad de detección versus la topología de interacción que se encuentran en la práctica caen en el control descentralizado porque no requieren explícitamente un coordinador global.

En segundo lugar y volviendo a lo ya probado en [3], los significados de control de formación descentralizado no son exactamente los mismos en la práctica y son más bien subjetivos.

- Relativo: Todo esquema de control de formación requiere que los agentes detecten variables

como posiciones y actitudes con respecto a los sistemas de coordenadas locales asociados con los agentes individuales o un sistema de coordenadas globales asociado con el sistema de agentes múltiples. El término relativo se suele interpretar en el sentido de que una variable se detecta con respecto a un sistema de coordenadas local, no a uno global. Por el contrario, una variable que se percibe con respecto a un sistema de coordenadas global se llama absoluta. Uno puede ser asociado relativo con descentralizado. En este sentido, el control de formaciones basado en la distancia puede considerarse más descentralizado que el control basado en la posición y el desplazamiento. Sin embargo, esta caracterización puede causar confusión porque descentralizado tiene otros significados. Sin embargo, enfatizamos que el concepto de relativo se puede describir claramente en términos de la capacidad sensitiva de los agentes individuales.

- Local: El término local puede entenderse en varias formas. Primero, se puede asociar con interacciones entre los agentes. En segundo lugar, se puede considerar que local significa que una variable se detecta con respecto a un sistema de coordenadas local, en este caso, el concepto de local puede ser claramente descrito por la topología de detección. Finalmente, se trata de la falta de un coordinador global.

3.2. Formulación de objetivos.

Durante la ejecución de este trabajo se podrán distinguir entre tres principales objetivos: el desarrollo de una maqueta de un sistema multiagente mediante el programa Easy Java Simulations, estudio e implementación de un modelo dinámico lineal, el estudio e implementación de un modelo dinámico no lineal con y sin restricciones no holonómicas.

- Modelado de la maqueta mediante el programa Easy Java Simulations: mediante este programa de implementación de simulaciones destinadas a la docencia e investigación se pretende mediante la programación y la herramienta visual que brinda este programa crear un escenario ocupado por cuatro agentes cuyo objetivo es alcanzar unas posiciones absolutas escogidas por un usuario externo.

  Las leyes de control de dichos robots o agentes de cuya evolución depende el movimiento de estos serán modificadas en función de qué clase de modelo dinámico se desee estudiar.

- Modelo dinámico lineal: mediante un modelo planteado que definen las leyes de control de movimiento se pretende estudiar su comportamiento en bucle abierto mediante la herramienta Simulink de Matlab y teniendo en cuenta que es un modelo lineal obtener su función de transferencia.

  Una vez obtenida su función de transferencia en función de sus características estáticas y dinámicas se procederá con el diseño de distintos controladores discretos como son los del tipo PID y los algebraicos mediante los métodos de asignación de polos, síntesis directa y Dead-beat. Por último, cada uno de estos reguladores serán puestos a prueba en la maqueta desarrollada en EJS. Es importante que para validar cualquier regulador a lo largo de este trabajo se empleará Simulink como método de “troubleshooting”, para que en caso de error se sepa si se debe al diseño del controlador o a la implementación en la maqueta.

- Modelo dinámico no lineal: Al igual que con el lineal, se estudiará su comportamiento mediante Matlab Simulink. Finalmente se comprobará con la implementación y validación una segunda ley de control se puede estabilizar dicho sistema además de estudiar su convergencia con un robot no holonómico y se valorará empleando la maqueta desarrollada en este trabajo.

3.3. Alcance del trabajo.

Una vez se hayan cumplido todos los objetivos de forma satisfactoria, se deben de haber estudiado el comportamiento de todos los sistemas en bucle abierto mediante la aplicación Simulink y el comportamiento en bucle cerrado con los reguladores diseñados todos ellos estudiados como modelos discretos. Seguidamente se espera obtener distintas simulaciones para cada uno de los sistemas con sus correspondientes reguladores de la misma forma que con el modelo no lineal con una segunda ley de control. En último lugar se concluirá comparando cada una de las respuestas, sus características y lo que implicaría su implementación en un sistema real.

Finalmente, se pretende dar más visibilidad al uso del programa Easy Java Simulations para su empleo en la investigación y la docencia además de la ampliación de conocimientos en el estudio de la formación multiagente de robots con y sin restricciones no holonómicas.

4. Introducción a Easy Java Simulations.

4.1. Descripción y uso de la herramienta EJS.

Las simulaciones están cada vez más presentes en la forma en que se educa y se instruye en materia científica. Tanto en la educación previa a antes de los grados postobligatorios como en estos, donde los ordenadores se utilizan cada vez más como una forma de impartir la enseñanza para que las clases sean más atractivas para los estudiantes y que estos mismo sean capaces de visualizar y de entender mejor un determinado aspecto científico. No obstante, es cierto que muchos profesores se muestran renuentes a emplear nuevas técnicas o tecnologías enfocadas a la docencia debido, seguramente, a la falta de medios o a la carencia de conocimientos o comprensión de estas. El entorno de desarrollo de EJS está disponible de forma gratuita y on-line en el siguiente enlace:

https://www.um.es/fem/EjsWiki/Main/Download

Tal y como indican sus creadores en [2], Easy Java/ JavaScript Simulations es una herramienta concebida para el diseño y la creación de simulaciones y escenarios discretos cuya finalidad es reproducir con fines pedagógicos o científicos dicha escena a través de los distintos estados que se pueden suceder. Cada uno de estos estados vendrá dado por una serie de variables, restricciones y algoritmos que variarán durante el transcurso de la simulación. Como resultado, mediante EJS se pueden desarrollar aplicaciones compatibles con un navegador web, tanto mediante Applets de Java o leguaje Javascript. Como se comentan en el artículo [3] y en el libro [4] Javascript es el lenguaje de programación principal para los navegadores web y para apps modernas cuya sintaxis es muy similar al leguaje C. Regularmente se solía emplear en páginas web HTML para realizar operaciones sin embargo ahora se utiliza para el envío y recepción de información. Por otra parte tal y como se definen en [5] las applets son aplicaciones web escritas mediante el lenguaje Java que permiten a los usuarios realizar experimentos en una máquina virtual a través de una interfaz fácil de usar que no requiere la instalación de un software local más allá del navegador web y el sistema operativo.

De la misma manera y tal como se expone en el artículo [6], esta aplicación surge también con la intención de que tanto profesores, como alumnos o incluso investigadores con unos conocimientos básicos del lenguaje de programación C sean capaces de desarrollar sus propias maquetas y simulaciones de una manera rápida y sencilla. Lo que puede permitir a alumnos realizar ciertas prácticas sin necesidad de asistir físicamente a laboratorios. Si bien es cierto que el desarrollo de una interfaz gráfica interactiva y sofisticada dependiendo del modelo que se desee implementar puede resultar complejo. EJS brinda una serie de menús en la sección *View/ HTMLView* muy intuitivos que facilitan dicha tarea concediendo al usuario tiempo para centrarse en aspectos más importantes como depurar el código u optimizar los algoritmos.

En última instancia, la elección de Easy Java Simulations como programa de desarrollo de simulaciones también es justificable por su alta compatibilidad con sistemas operativos y con casi cualquier plataforma de software, además, las simulaciones pueden ser ejecutadas en cualquier navegador web, eso sí, debe estar habilitado para el lenguaje Java.

4.2. Programación previa al uso de la maqueta.

4.2.1. Modelo.

El panel de trabajo Modelo es donde se definirán todos los elementos que compondrán el modelo a realizar, como son, por ejemplo: las variables, la evolución, las relaciones fijas y elementos propios los cuales han sido empleados principalmente para la creación de funciones de donde se obtiene la actualización de ciertas variables.

4.2.1.1. Declaración de variables.

Las variables son los elementos más básicos en las leyes que determinan un modelo estas son utilizadas para representar valores o conceptos. Por esa razón es importante que al iniciar la implementación del modelo el primer paso fuera definir e inicializar las variables que definirán la futura dinámica del sistema. Dentro del programa empleado se pueden distinguir cuatro tipos:

- Double: Permite almacenar números con decimales.
- Int: Solo puede contener números enteros.
- Char: Con ellas se pueden crear cadenas de caracteres con valores numéricos.
- Booleanas: Solo pueden tomar dos valores posibles: True o False.

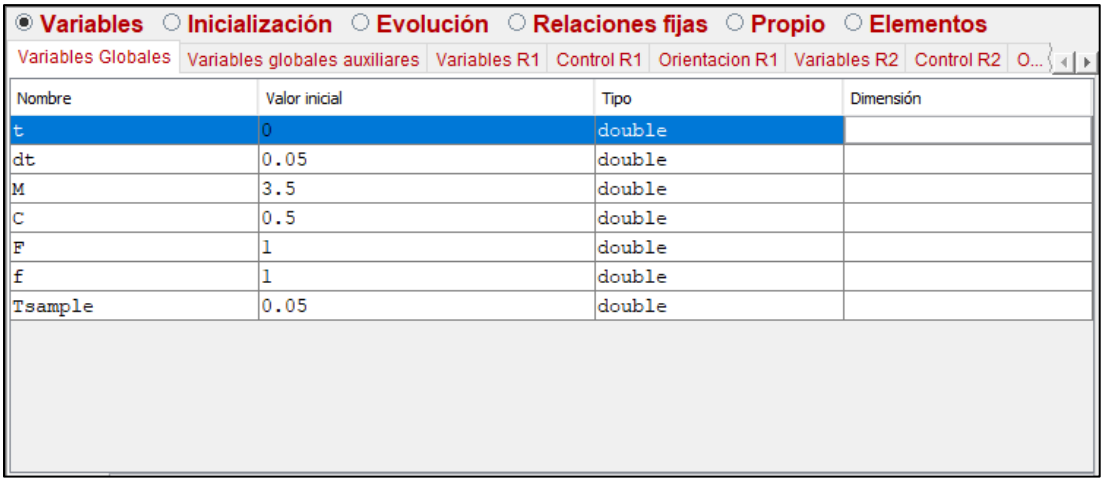


Figura 1. Variables del Modelo.

Como se puede observar en la figura 1 cuando las variables son creadas se les asigna un valor inicial. Siendo la mayoría de ellas del tipo "*double"* dado que la mayoría de los valores en un modelo físico requerirán decimales.

Hay que destacar que a diferencia de muchos programas Easy Java Simulations permite en todos los submenús del panel Modelo crear páginas distintas, gracias a ello, la información está mejor ordenada y trabajar es mucho más sencillo.

4.2.1.2. Evolución.

El submenú de evolución, representado por la figura 2, estará principalmente empleado para describir la dinámica del sistema mediante una serie de ecuaciones diferenciales. Es importante emplear las variables que se han definido como *"t"* y *"dt"* estas harán referencia al tiempo transcurrido y como avanzará este de forma discreta hacia la solución.

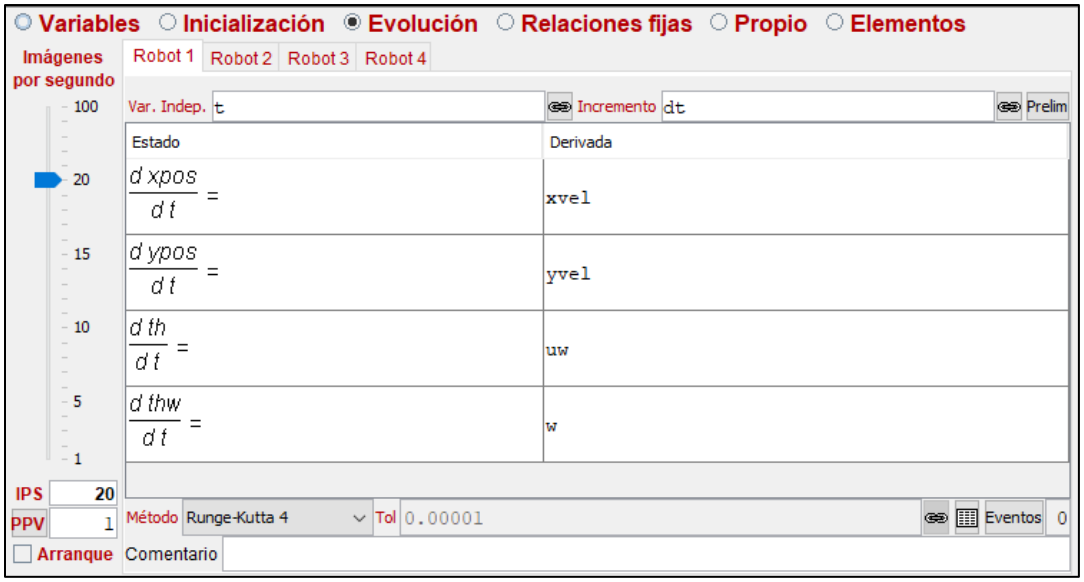


Figura 2. Submenú evolución.

4.2.1.3 Relaciones fijas.

Como es de entender no se trabaja ni sucede toda la actualización de las variables en el submenú Evolución. Por eso es importante diferenciar entre las variables que se calculan utilizando la evolución algoritmo como variables de estado o dinámicas y las variables que dependen de estas otras como auxiliares o de salida. Además, dichas variables también se pueden calcular después de que la evolución haya sido ejecutada.

Dentro de este mismo aparatado se ha programado la estructura necesaria para el movimiento de los agentes, basada principalmente en una estructura switch case, esta proporciona una forma sencilla de enviar la ejecución a diferentes partes del código en función de un valor. Por ejemplo, pasar de la fase de orientación a la de movimiento cambiando tan solo el valor de una variable. A continuación, se mostrará en la figura 3 un diagrama de flujo del comportamiento de esta estructura.

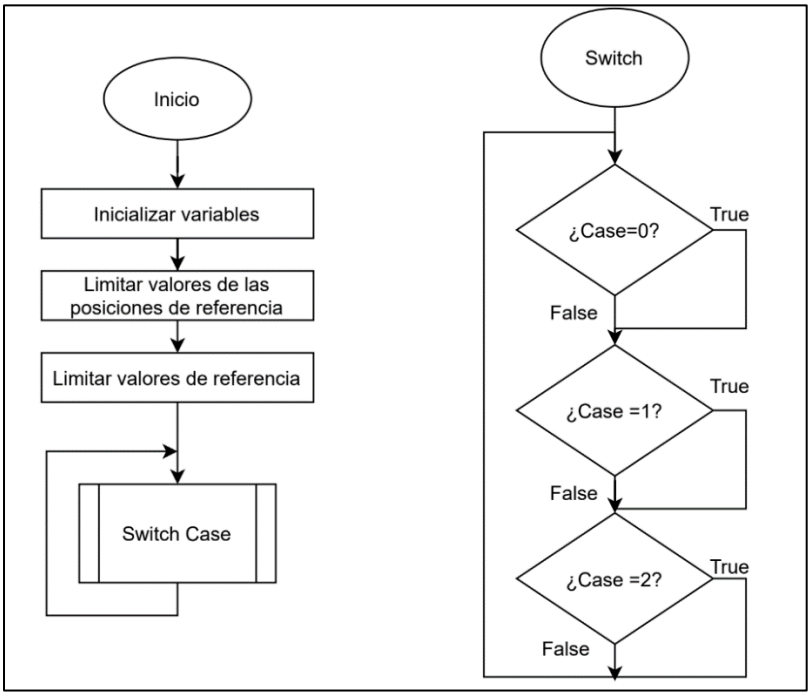


Figura 3. Flujograma Switch

Dentro del switch se distinguen tres estructuras distintas, teniendo cada una de ellas su función respectiva.

La primera o *"case 0"*, cuyo código corresponde con el de la figura 5, tiene la utilidad de realizar los cálculos referentes a la orientación y el giro del agente. Su flujograma correspondería con el siguiente de la figura 4:

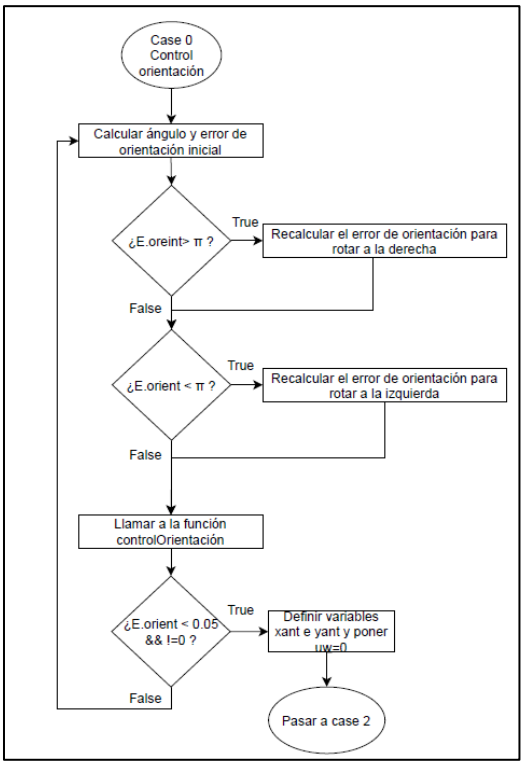


Figura 4. Control de orientación-Case 0.

```
switch (modo_R4){
  case 0:
    th_ref_R4 = Math.atan2(yref_R4-ypos_R4, xref_R4-xpos_R4);
    error_orientacion_R4 = th_ref_R4 - th_R4;
    if(error_orientacion_R4 > Math.PI){  //Giro a derechas negativo
          error_orientacion_R4 = -(2*Math.PI-error_orientacion_R4);
    }
    if(error_orientacion_R4 < -Math.PI){  //Giro a izquierdas positivo
          error_orientacion_R4 = (2*Math.PI+error_orientacion_R4);
    }
    uw_R4 = controlOrientacion(error_orientacion_R4);
    error_or_abs_R4 = Math.abs(error_orientacion_R4);
    if(error_or_abs_R4 < 0.05 && error_or_abs_R4 != 0){ //Condición paso siguiente etapa
      uw_R4 = 0;
      xant_R4 = xref_R4;
      yant_R4 = yref_R4;
      modo_R4 = 1;
    }
    break;
```

Figura 5. Control de orientación-Case 0.

Para la obtención del error de orientación o la posición respecto a una orientación se deberá realizar el cálculo de *"th_ref"* para ello se empleará la arcotangente cuadrada, esta nos dará el valor en radianes de la posición deseada.

Una vez obtenido dicho valor mediante dos estructuras del tipo *"if"* se buscará que tipo de rotación o giro debe ser el empleado. Por ejemplo, si tomamos como valor de referencia de 0 radianes y deseamos ir a una posición de más de π rad el agente rotará hacia la izquierda. Sucedería lo contrario si el valor a alcanzar se situara por debajo de los π rad. Tras haber seleccionada el tipo de rotación se obtendrá un nuevo valor de error de orientación al cual se le habrá sumado 2* π rad para que el agente de la vuelta y pueda estar orientado hacia el punto.

Seguidamente, se llamará a la función "controlOrientación" que nos permitirá reducir dicho error y de esta forma se empleará la variable *"uw"* cuya integral definida en el apartado de evolución representa la rotación del agente.

Finalmente se impone una condición antes de pasar a la siguiente etapa, esta consiste en que el error de orientación debe ser menor que 0.05 o igual a 0. Una vez se cumpla se igualará a 0 la variable *"uw"* para reiniciarla y se igualarán los valores de *"xant"* y *"yant"* a las posiciones de referencia, estas variables serán utilizadas más tarde para reiniciar otras variables. Esto se verá con más claridad en el siguiente *"case"*.

El "case 1" o modo 1, representado por el código de la figura 7, será el encargado de realizar los cálculos que darán valores a las variables que definen las ecuaciones de movimiento del subapartado de "Evolución". A su vez, en este modo se realizarán los cálculos de las variables auxiliares, si se requieren. Además de limitar la acción de control de los reguladores. Cuyo diagrama de bloques es el siguiente flujograma representado por la figura 6:

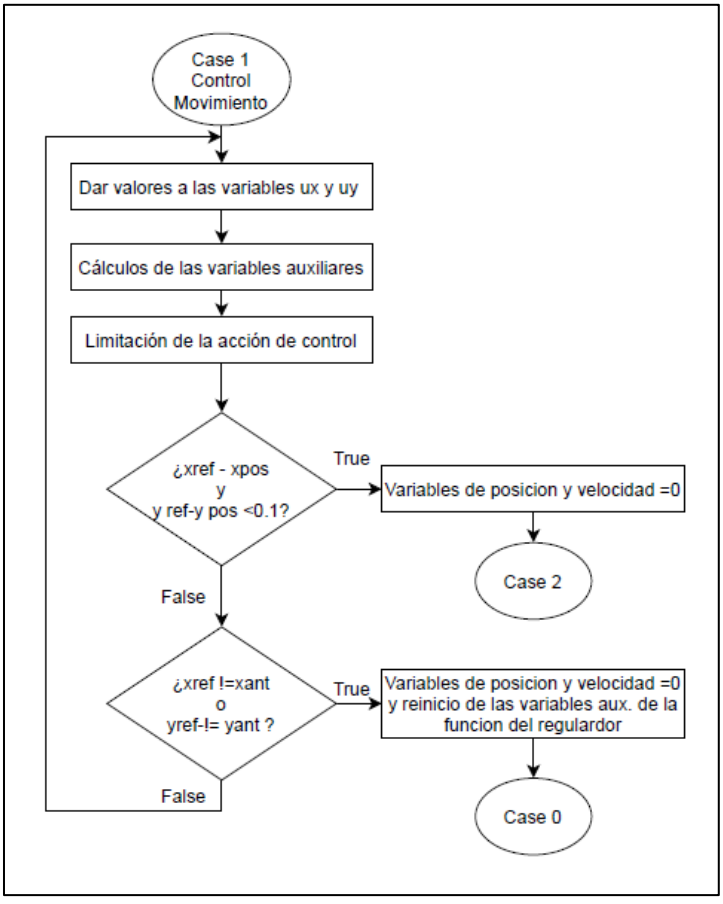


Figura 6. Control de movimiento-Case 1.

```
case 1:
   ux_R4 = controlSignalGenericxR4();
   uy_R4 = controlSignalGenericyR4();
   //Limitación accioón de control
   if(ux_R4>=12) ux_R4=12;
   if(ux_R4<=-12) ux_R4=-12;
   if(uy_R4>=12) uy_R4=12;
   if(uy_R4<=-12) uy_R4=-12;
   if(Math.abs(xref_R4 - xpos_R4) < 0.1  && Math.abs(yref_R4 - ypos_R4) < 0.1 ){
      ux_R4 = 0;
      uy_R4 = 0;
      xvel_R4 = 0;
      yvel_R4 = 0;
      modo_R4 = 2;
   }
   if(xref_R4!=xant_R4 || yref_R4!=yant_R4){
      ux_R4 = 0;
      uy_R4 = 0;
      xvel_R4 = 0;
      yvel_R4 = 0;
      ekx1_R4 = 0;
      ukx1_R4 = 0;
      eky1_R4 = 0;
      uky1_R4 = 0;
      modo_R4 = 0;
   }
```

Figura 7. Control de movimiento-Case 1

Para empezar las variables *"ux"* y *"uy"* forman parte de las ecuaciones del movimiento situadas en el subapartado de evolución, su obtención será mediante las funciones *"controlSignalGenericx/y"* cuya definición e implementación se encuentra en el subíndice *"Propio"* que se explicará más tarde en el siguiente apartado.

Como bien se ha comentado anteriormente se ha implementado una limitación de la acción de control ya que como más tarde se verá que debido a la distribución de los polos y ceros de la planta; los reguladores requerirán de una elevada acción de control para llegar a las especificaciones deseadas. A su vez, esta función es realmente útil ya que se evitaría que en la misma simulación desaparecieran los agentes por la acción de la saturación sobre estos.

Para que el agente pueda realizar la parada sobre el punto requerido la diferencia en valores absolutos de los valores de posición de referencia y los de la posición actual deben debe ser menor de 0.1 tanto en abscisas como en ordenadas. Si esto se cumpliera se reiniciarían las variables correspondientes con la velocidad lineal y las variables "ux" y "uy" y se pasará al siguiente "case".

Sin embargo, si la situación anterior no se da, ósea que el agente sigue desplazándose hacia la posición, se deben reiniciar continuamente las variables de velocidad y todas las relacionadas con la función *"controlSignalGenericx/y"* además de las ya mencionadas *"ux"* y *"uy"*.

El último modo, modo 2, cuyo código mostrado en la figura 9 estará destinado a realizar una espera hasta que los valores de *"xref"*y *"yref"* varíen. Esto significa que una vez el agente haya alcanzado los valores de referencia estos podrán ser cambiados por el usuario por otros completamente distintos para que los agentes completen una nueva trayectoria. Finalmente, su flujograma será el indicado en la figura 8.

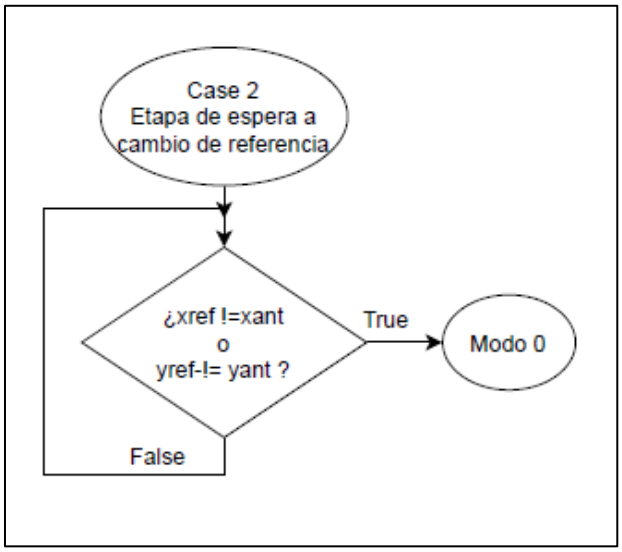


Figura 8. Espera a cambio de referencia-Case 2.

```
// Estapa de Espera a cambio de referencia
case 2:
    if(xref_R1!=xant_R1 || yref_R1!=yant_R1){
        modo_R1 = 0;
    }
    break;
}
```

Figura 9. Espera a cambio de referencia-Case 2.

4.2.1.4 Propio / definición de funciones.

Este submenú se ha dedicado específicamente al desarrollo de las funciones mencionadas en el apartado anterior cuyo principal uso es realizar los cálculos de la acción de control y realizar ajustes sobre el error de orientación. Dentro de las páginas dedicadas a la función de control se pueden distinguir entre dos tipos distintos de funciones.

```
function controlSignalGenericxR3(){
  var uux_R3;
  if(Tsample>0){
    if(t-n1_zoh_R3*Tsample>=Tsample){
      uux_R3 = controlDiscretoxR3();
      n1_zoh_R3=n1_zoh_R3+1;
    }
  }
  return uux_R3;
}
```

Figura 10. Función controlSignalGeneric

Como se puede observar en la figura 10, la estructura está compuesta por otras dos del tipo *"if"*, la primera marca una condición que siempre va a suceder ya que se define al tiempo de establecimiento o *"Tsample"* como 0.05 s y dicha variable permanecerá siempre constante. El segundo "if" está pensado para que se ejecute esa misma instrucción cada periodo de muestreo de forma que se ejecute

un control discreto y no continuo. Dentro de esta última estructura se llamará a la siguiente función cuyo valor es la salida de la función *"controlSignalGeneric"*.

```
function controlDiscretoxR3(){
  var a1=0.003597, a2=0, a3=0, a4=0;
  var b0=366.6, b1=-348.7, b2=0, b3=0;

  //Error de realimentacion
  var ekx_R3 = xref_R3 - xpos_R3;

  //Calculo accion de control
  var ukx_R3 = b0*ekx_R3 + b1*ekx1_R3 + b2*ekx2_R3 + b3*ekx3_R3 + a1*ukx1_R3 + a2*ukx2_R3 + a3*ukx3_R3 + a4*ukx4_R3;

  //Actualización
  ekx3_R3=ekx2_R3;
  ekx2_R3=ekx1_R3;
  ekx1_R3=ekx_R3;

  ukx4_R3=ukx3_R3;
  ukx3_R3=ukx2_R3;
  ukx2_R3=ukx1_R3;
  ukx1_R3=ukx_R3;

  return ukx_R3;
}
```

Figura 11. Función control Discreto.

El funcionamiento principal de esta función, representada en la figura 11, consistirá en resolver una ecuación en diferencias cuyos resultados se irán actualizando con cada periodo de muestreo. Para empezar, daremos los valores a "a" y "b" con los valores de las raíces del numerador y denominador de la fdt del regulador. Seguidamente, se calculará el error de realimentación cuyo valor será la diferencia entre el valor de referencia y el valor correspondiente con la realimentación, en este caso la posición actual del robot. Una vez calculado este error se procederá con el cálculo de la acción de control con los valores comentados anteriormente y con su actualización, esta última acción permitirá actualizar los valores manteniendo hasta los tres anteriores necesarios para el cálculo de una nueva acción de control.

4.2.2. View/ HTMLView (diseño gráfico de la maqueta).

Por último, el tercer menú es el HTML el cual permite crear y diseñar gráficamente los escenarios donde se desarrollarán las evoluciones de las leyes de control planteadas. Dentro de este menú se nos ofrecerá una gran cantidad de elementos que podrán ser incluidos en nuestras representaciones. Sin embargo, durante la realización de este trabajo se emplearán principalmente los submenús de interfaz y elementos de dibujo en 2D.

Dentro del árbol de elementos se pueden distinguir entre tres tipos de paneles distintos:

- El panel principal estará formado por la representación gráfica de los agentes y de los lugares hacia los que se tienen que dirigir. Para la formación de esta estructura se han empleados bloques del tipo panel, siendo en el primero de ellos donde se definirá el tamaño que ocupará esta interfaz en píxeles, un panel de dibujo, que podemos encontrar en el submenú de interfaz, donde se recogerán todos los elementos móviles.

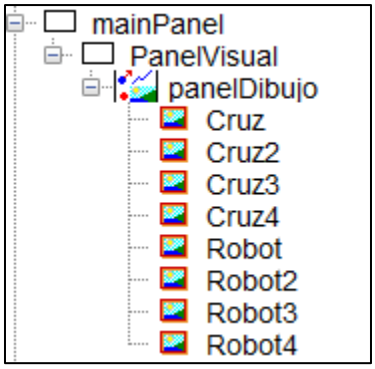


Figura 12. Panel visual.

Una vez definidos los agentes con sus posiciones, justo como se ejemplifica en la figura 12, hay que ligar estos con las variables que deben representar o que definirán su movimiento. En el caso de las cruces hay que incluir las variables *"xref_Rx" y "yref_Rx"* en el apartado de posición X e Y cuyo significado serán las coordenadas a alcanzar. Mientras que para los robots en el apartado de posición añadiremos las variables *"xpos_Rx"* y *"ypo_Ry"* y en el de rotación *"th_Rx"*.

- El siguiente panel tendrá la función de ofrecerle al usuario la posibilidad de elegir la localización de cada una de las cruces anteriormente mencionadas. Para la creación de este apartado al igual que para el anterior la estructura está basada en una serie de paneles que agruparan los elementos campo numérico, estos bloques se pueden encontrar en el submenú interfaz en el subapartado representado por el mismo bloque. Finalmente deberá quedar un esquema como el de la figura 13.

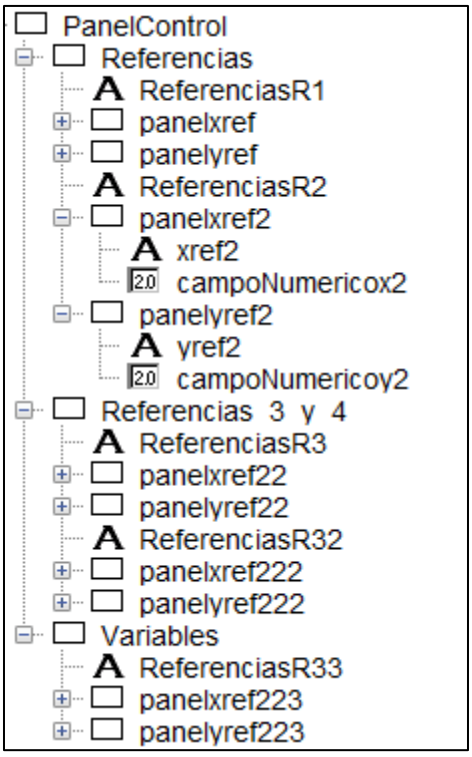


Figura 13. Panel de elección de coordenadas.

Dentro de estos bloques llamados campos numéricos deberemos incluir las variables correspondientes con las posiciones de referencia, de esta manera se podrán elegir las coordenadas destino de los agentes. Además, se decidió añadir otro panel para la elección de las constantes" kv*"* y *"kw "*que pertenecerán al modelo dinámico no lineal.

- Por último, se incluye un pequeño panel con tres funciones muy simples: inicio, pausa y reinicio (Figura 14).

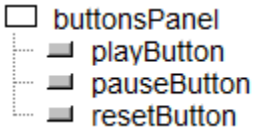


Figura 14. Panel control

A continuación, en la figura 15 se podrá apreciar el resultado final del diseño gráfico (las cruces son solapadas por los agentes porque tienen la misma posición inicial):

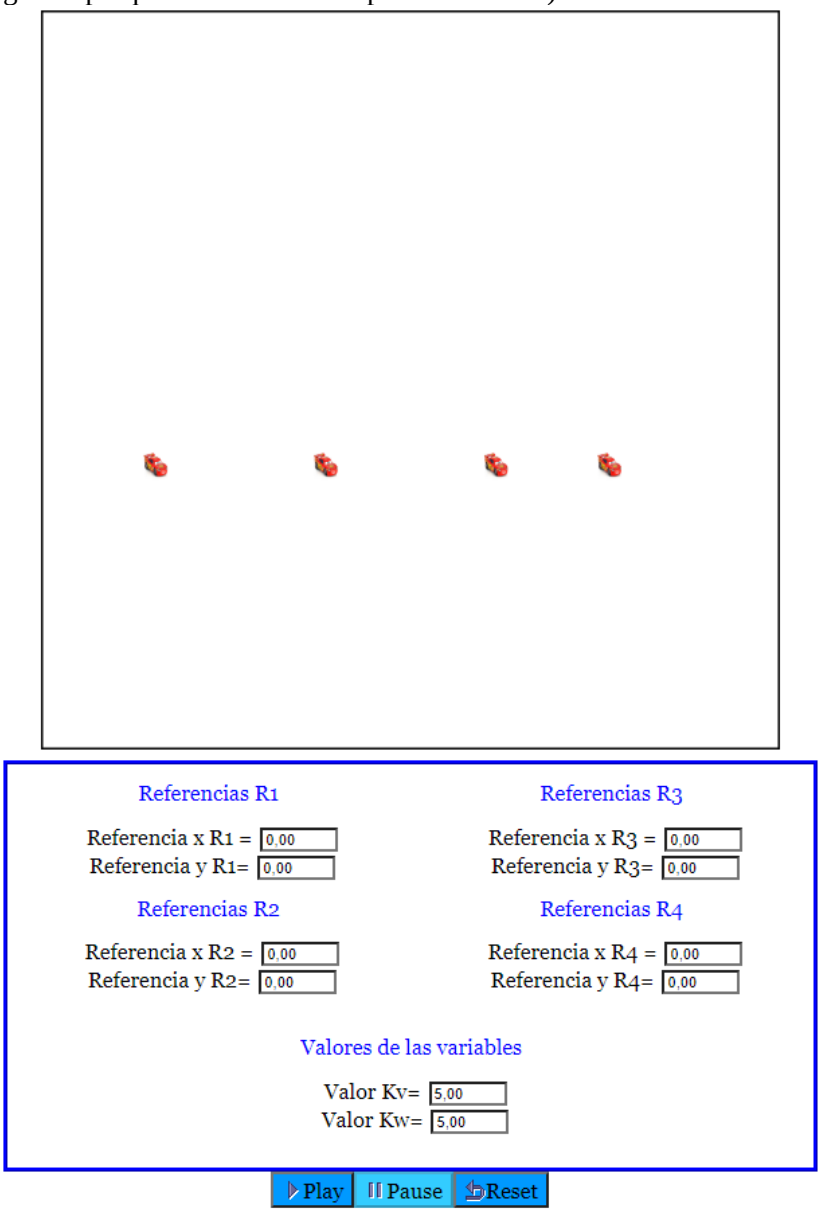


Figura 15. Vista final de la maqueta.

5. Control de formaciones del sistema moltirrobot.

5.1. Agentes con dinámica lineal.

5.1.1 Descripción del modelo.

Previamente al análisis del sistema hay que entender su funcionamiento, para empezar, se dispone de la igualdad en que la derivada de la posición es la velocidad, como es lógico. A continuación, se obtendrá la derivada de la velocidad, siendo esta la aceleración, despejando la segunda ley de newton (F= m·a). Estando la fuerza definida como el producto del coeficiente de rozamiento por la velocidad lineal más una constante de fuerza, F, por la acción de los actuadores, ux/uy.

$$\frac{dxpos}{dt} = xvel$$

$$\frac{dxvel}{dt} = \frac{1}{M} * (-C * xvel + F * ux)$$

Hay que recordar que estas ecuaciones son las correspondientes a un robot móvil para cuál se desea realizar un estudio de la posición en dos dimensiones por lo que deben existir ecuaciones para los dos ejes.

$$\frac{dypos}{dt} = yvel$$

$$\frac{dyvel}{dt} = \frac{1}{M} * (-C * yvel + F * uy)$$

Existiendo una última cuya utilidad será variar el ángulo de orientación del agente:

$$\frac{dth}{dt} = uw$$

A simple vista el sistema puede ser definido como un sistema MIMO (multiple input-multiple output) dado que para controlar las posiciones (salidas), se requieren los valores de las velocidades, que a su vez de calculan con la salida del regulador (entradas). No obstante, como se puede observar las salidas de las "x" no requieren ni de las entradas ni de las salidas de "y" y viceversa, esto significa que ambas son independientes y que por lo tanto el sistema puede ser estudiado como un sistema SISO (single input-single output).

Siempre que se desea modelar un sistema es importante conocer si sus ecuaciones son lineales o no. Para ello habrá que comprobar si cumplen las propiedades de homogeneidad y superposición. Cogiendo el ejemplo de [7], supongamos la ecuación de una recta:

$$y(t) = m * x(t) + n$$

Comprobamos su homogeneidad:

$$y_1(t) = m * x_1(t) + n$$

$$y_2(t) = K * [m * x_1(t)] + n = K * m * x_1 + n$$

$$y_2(t) \neq K * y_1 = K * m * x_1 + K * n$$

Por lo que la ecuación sería no lineal ya que incluye una constante, n, que no está multiplicando a la variable x. Destacar que todas las funciones trigonométricas (coseno, seno, tangente, etc…) junto a operadores como las raíces cuadradas hacen que las ecuaciones sean no lineales. Por otra parte, las derivadas son operaciones lineales. Observando las ecuaciones dadas tanto para la "x" como para la "y" se concluye que es un sistema lineal por lo que se procede a la obtención de la función de

transferencia tal y empleando la metodología de [8]:

1. Se realizada la transformada de Laplace en cada una de las ecuaciones diferenciales lineales mediante la expresión:

$$F(s) = L[f(t)] = \int_0^{\infty} f(t) * e^{-st} dt$$

$$L\left[\frac{d^n}{dt^n}\right] = s^n * F(s) - (s^{n-1} * f(0) + s^{n-2} * f^1(0) + \ldots f^{(n-1)}(0))$$

Teniendo la variable s parte real y parte compleja.

$$s * xpos = xvel$$
$$s * xvel = \frac{1}{M} * (-C * xvel + F * ux)$$

2. Seguidamente se despeja *"xvel"* para dejar la ecuación con las variables que representan las entradas y las salidas (*"xpos"* y *"ux"* respectivamente).

$$s^2 * xpos = \frac{1}{M} * (-C * s * xpos + F * ux)$$

3. Ahora se procede a sustituir las constantes por sus valores (M =3.5 Kg, C=0.5 μ y F=1) y se opera con estos.

$$s^2 * xpos = \frac{1}{3.5} * (-0.5 * s * xpos + 1 * ux)$$

$$s^2 * xpos \approx 0.29 * ux - 0.145 * (s * xpos)$$

4. Para lograr representar la función de transferencia se deberán despejar las variables de modo que se obtenga una función que muestre Salida/Entrada:

$$s^2 * xpos + 0.145 * (s * xpos) \approx 0.29 * ux$$

Sacando factor común:

$$xpos * (s^2 + 0.145 * s) = 0.29 * ux$$

$$\frac{Salida}{Entrada} = \frac{ux}{xpos} = \frac{0.29}{(s^2 + 0.145 * s)}$$

Dado que se desea trabajar con simulaciones donde se procesarán secuencias de valores en determinados instantes se discretiza la función de transferencia empleando el siguiente código de Matlab en la figura 16.

```
T=0.05;
z=tf('z',T)
M=3.5;
C=0.5
F=1
f=1

T=0.05;
z =tf('z',T)
s = tf('s');
num = [0,0,-(1/M)];
den = [-1,(1/C)*0.5,0];
Gs = tf(num,den)

Gz = c2d(Gs,T,'zoh');
Gz = minreal(Gz,0.001)
zpk(Gz)
sisotool(Gz)
```

Figura 16. Discretización del sistema lineal.

En primer lugar, se definirán las constantes y se definirán las "s" y "z" como funciones de transferencia. A continuación, se formará la función de transferencia del sistema en continuo para más tarde discretizarla con el comando "c2d". Este comando permite convertir los modelos continuos a discretos empleando varios métodos distintos:

- Zero-order hold o "zoh": Supone que las entradas de control son constantes por partes durante el periodo de muestreo. Se emplea para entradas de escalón.
- Firsorder hold: Similar al anterior, pero se emplea en entradas lineales por partes.
- Impulse: Igual a los anteriores, pero para entradas del tipo impulso.
- Tustin: Se desea una buena coincidencia en el dominio de la frecuencia entre los modelos de tiempo continuo y discreto. Además, el modelo tiene cierta dinámica importante en una frecuencia en particular.

Como se desea que la entrada del sistema discreto sea del tipo escalón el método empleado será el Zero-order hold. Con esto se obtiene la siguiente función de transferencia de la figura 17:

$$\frac{0.00036317\ (z+1.017)}{(z-1.051)\ (z-1)}$$

Figura 17. Función de transferencia del sistema lineal.

Como se puede apreciar esta función estará compuesta por un cero fuera del círculo unidad y por lo tanto inestable y por lo que sería casi un doble integrador. Debido a esta dinámica de doble integrador el sistema siempre sería inestable en bucle abierto ante una entrada del tipo escalón. Para comprobarlo en el command window de Matlab se puede emplear el comando "*step(Gx)*" para que se muestre en una Figura la respuesta temporal del sistema ante un escalón como se representa en la figura 18:

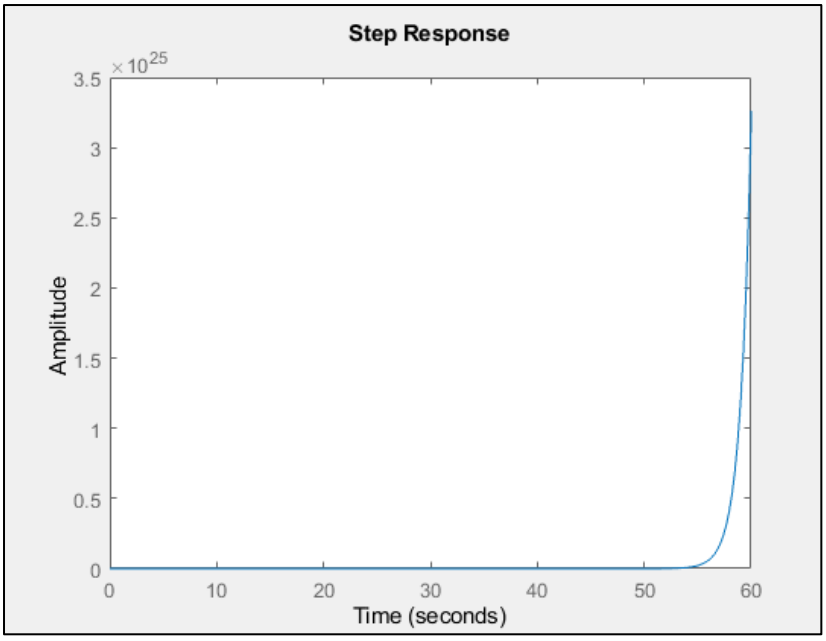


Figura 18. Respuesta en bucle abierto de Gz.

Por otra parte, se debería realizar un estudio del sistema en bucle cerrado dado que la implementación en la maqueta será ante una entrada en escalón en bucle cerrado. Con ello se realizó un estudio que consistía en implementar las ecuaciones mediante una estructura de bloques en Matlab Simulink. Cuyo diagrama de bloques mostrado en la figura 19 para una variable independiente tendría una forma similar a esta:

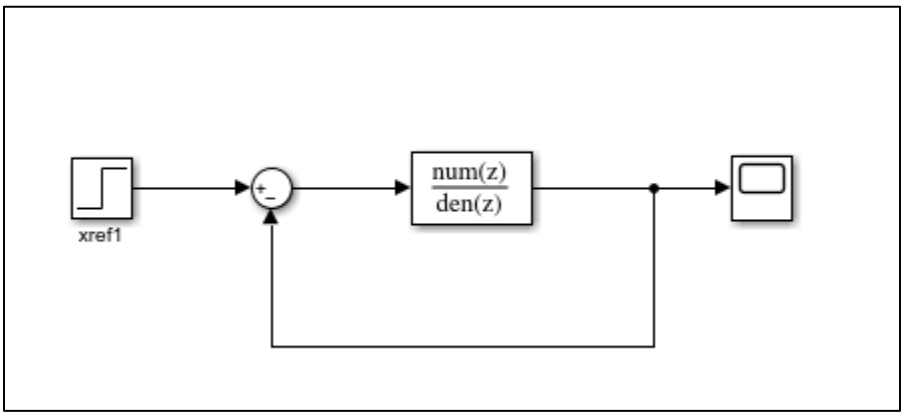


Figura 19. Diagrama de bloques sistema lineal en bucle cerrado

Por otro lado, se decidió realizar el estudio de ambas variables a la vez para demostrar su independencia y su comportamiento igual ante la misma referencia.

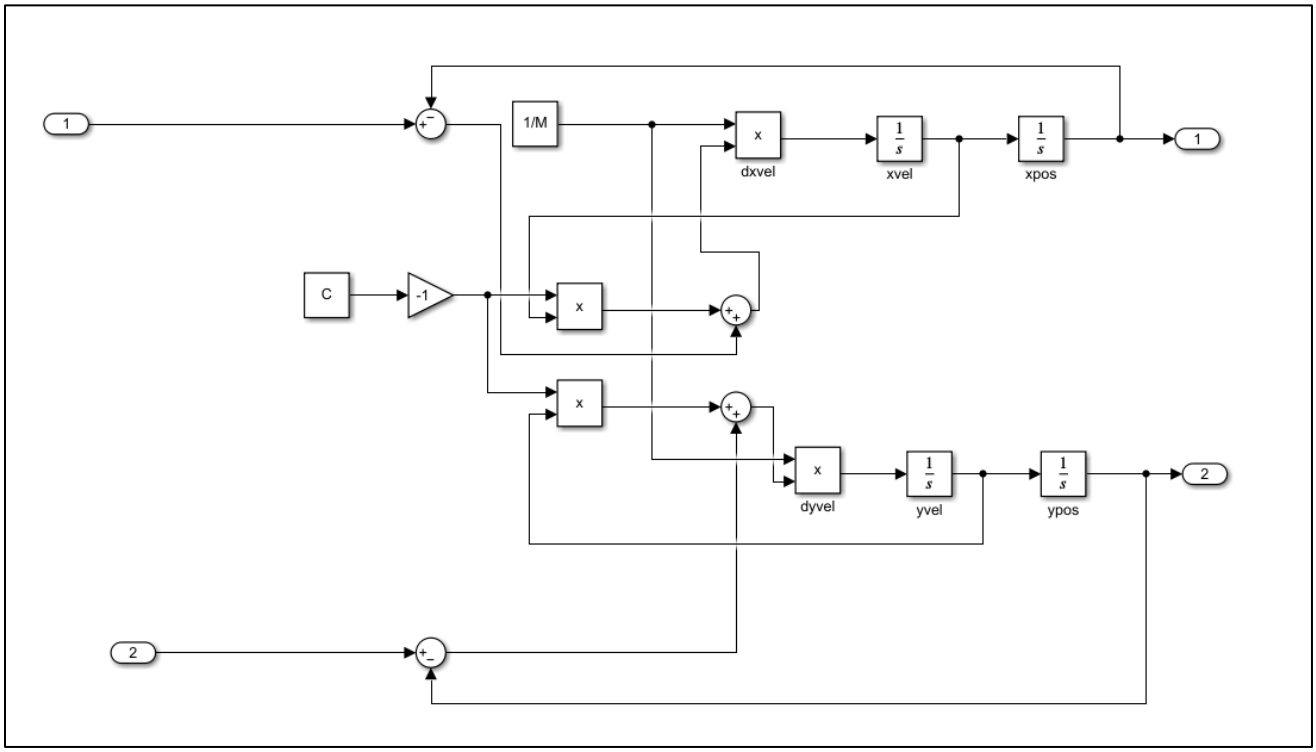


Figura 20. Diagrama de bloques de las ecuaciones del modelo.

Para el diseño de los diagramas de bloques mostrados en las figuras 20 y 21 se emplearon los siguientes bloques:

- Constant: Incluir las constantes previamente definidas.
- Gain: Principalmente empleado en añadir una ganancia de valor -1 para cambiar de signo una constante.
- Sum: Su uso consiste en restar la salida de la planta a la referencia y así formar la realimentación.
- Integrator: Integra la señal recibida para obtener los parámetros necesarios para seguir operando o dar la salida.
- Step: Estos bloques están programados para ofrecer una entrada del tipo escalón cuyos valores son escogidos por el propio usuario.
- Scope: Permiten mostrar el desarrollo temporal del sistema.
- XY Graph: muestra los valores que han alcanzado las variables.

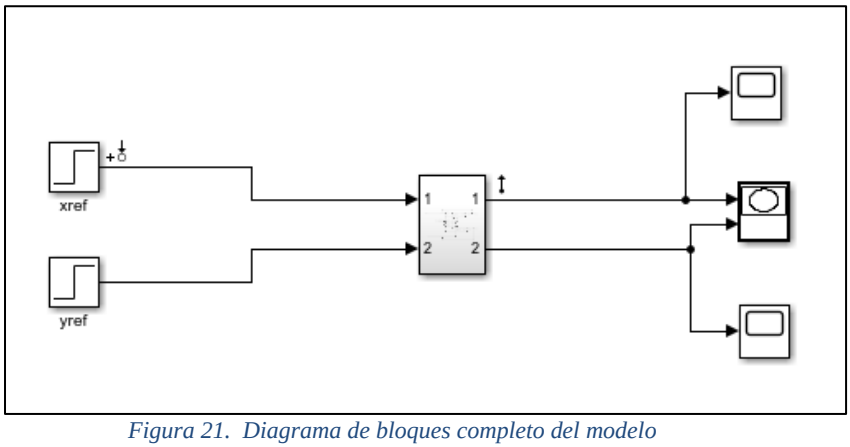


Figura 21. Diagrama de bloques completo del modelo

Una vez definido el modelo se obtienen las siguientes gráficas (Figura 22):

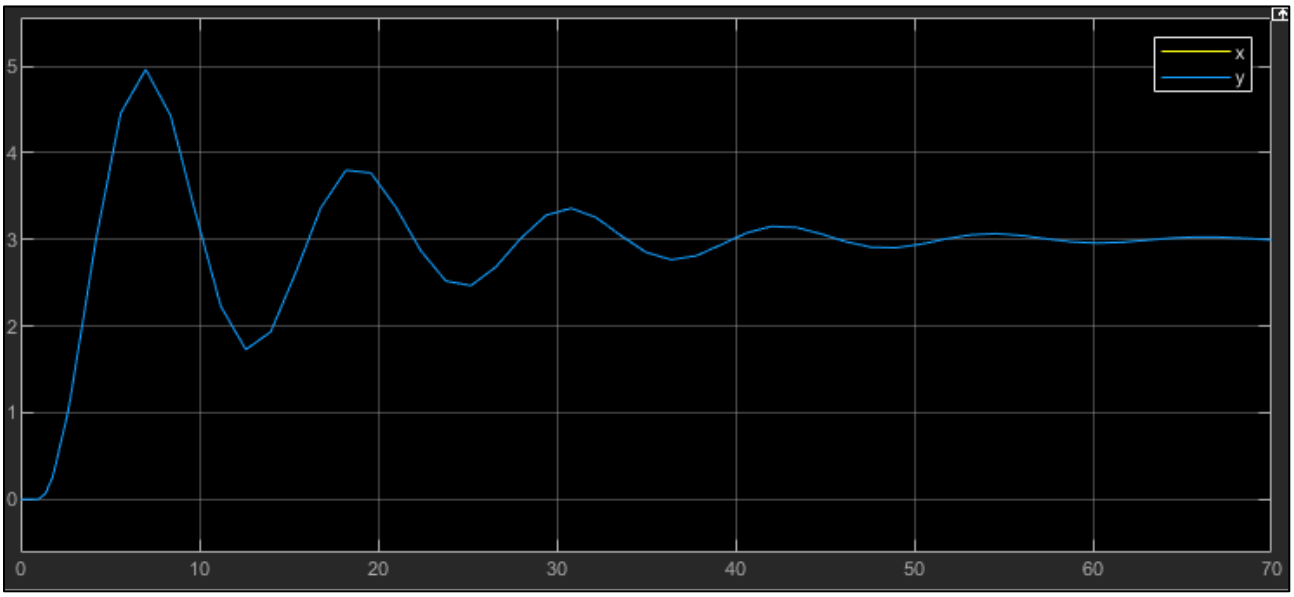


Figura 22. Respuesta de x e y en bucle cerrado

Como se puede apreciar ambas gráficas representadas por la figura 22 son idénticas por lo que se confirma que las ecuaciones de ambas son independientes y por lo tanto no dependen de las de la otra por lo que el sistema puede ser estudiado como de entrada y salida simples.

Respecto a la respuesta del modelo en bucle cerrado se puede decir que es muy oscilatoria debido posiblemente a que con la ganancia inicial los polos se encuentren en el semiplano negativo del lugar de las raíces. Además, el sistema tarda bastante en estabilizarse por completo situándose su tiempo de establecimiento alrededor de los 60 s. Por lo que es recomendable que cuando se desee simular se le dé más tiempo a la simulación. Para ello se debe incrementar el tiempo de parada en el recuadro situado en la ventana "Simulation" del mismo Simulink.

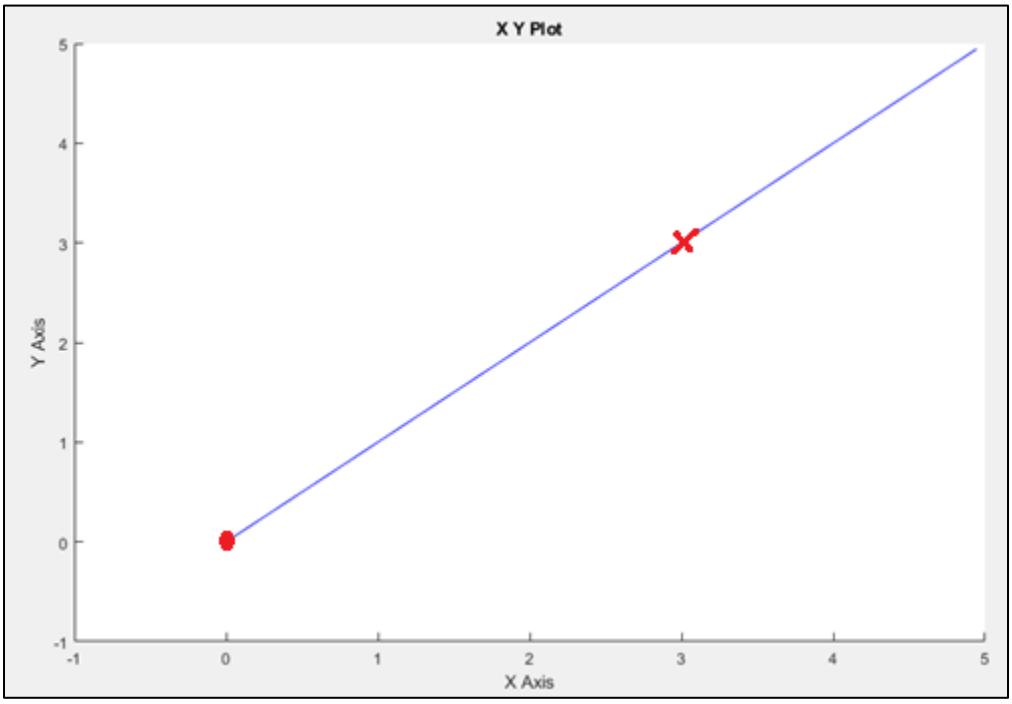


Figura 23. Respuesta de x e y en Graph xy

Finalmente, de este simulink se obtiene esta gráfica mostrada en la figura 23 con los distintos valores alcanzados por las variables, pero sin tener en cuenta el tiempo. La gráfica presenta una línea recta que va desde el 0 al 5 esto se debe a que la "x" y la "y" avanzan al mismo tiempo recorriendo los

mismos valores. Finalmente se llega a estabilizar en el punto (3,3) pero queda registrada la sobreoscilación y de ahí que se alcance el punto (5,5).

A continuación, se pasó a simular la planta directamente en la maqueta diseñada con Easy Java Simulations. En primer lugar, se debe restringir el uso del regulador ya que tan solo se desea simular la acción de la planta introduciendo los siguientes valores en la ecuación en diferencias como marca el código de la figura 24 situada en el submenú Propio:

```
var a1=0, a2=0, a3=0, a4=0;
var b0=1, b1=0, b2=0, b3=0;
```

Figura 24. Valores para ganancia 1 en la ec. En diferencias

De este modo se conseguiría una ganancia de valor 1, de este modo la entrada a la planta solo será el error como representaría el esquema de la figura 25.

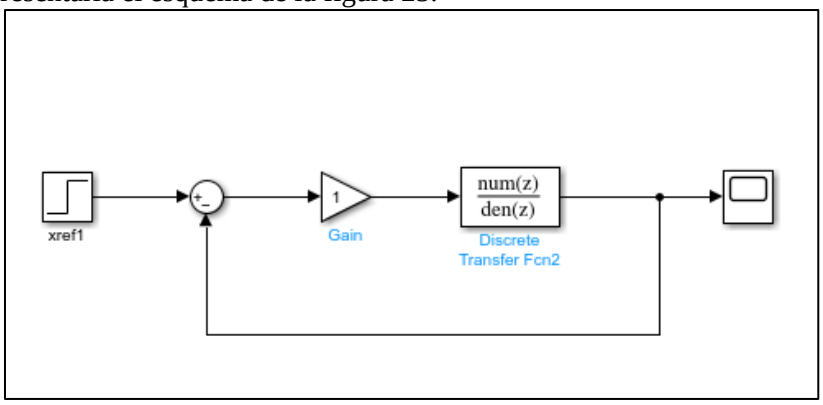


Figura 25. Diagrama de bloques simulación planta

Además, en este caso se ha decido comentar la parte del código que limita error esto es muy recomendable ya que el sistema en la primera sobreoscilación alcanza un valor muy cercano al 5. Por lo que si se realiza el cálculo de la diferencia entre la referencia y la salida en el segundo en el que rebase la referencia que se le haya dado la maqueta estabilizará automáticamente el sistema. Debido a esto el agente irá directo a las coordenadas sin mostrar las oscilaciones y por lo tanto una acción completamente distinta a la que se ha estudiado.

Como se puede observar en la simulación parece que se estabiliza mucho antes, pero no es así, se puede explicar volviendo a la figura 22 y observando que a partir de cierta sobreoscilación el sistema oscila en valores muy cercanos al 3 y de ahí que se cree ese efecto.

5.1.2 Tipos de controladores propuestos.

A fin de lograr que la planta responda como se desea se pueden tomar dos vías distintas para modificar su comportamiento dinámico, una de ellas es modificar físicamente sus características y la otra en la que se buscará añadir un regulador. En particular se empleará el método de análisis ya que la función de transferencia del regulador prácticamente se sabe de antemano y se sabe que no dará demasiados problemas su implementación.

Este proceso coincidirá en ajustar unos parámetros de un regulador para alcanzar unas especificaciones estáticas o dinámicas.

- Estáticas: errores en el seguimiento de una determinada señal: error de posición, error de velocidad y error de aceleración.
- Dinámicas: tiempo de establecimiento, sobresocilación, tiempo de subida. etc...

Una vez estén claras los tipos de especificaciones se ha de hablar de las posibles acciones que se pueden aplicar al sistema para el diseño del regulador, expuestas en sus correspondientes de [7] (cuya

posición en el esquema de bloques se muestra en la figura 26).

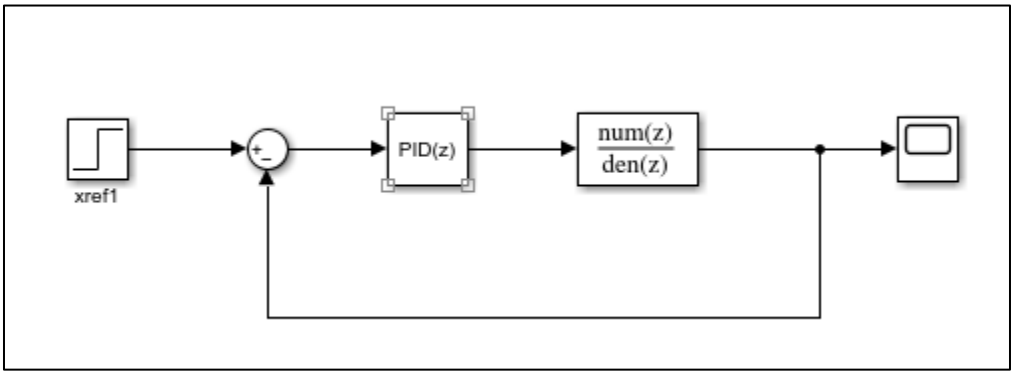


Figura 26. Diagrama de bloques con regulador PID

- Acción proporcional: Se llama acción proporcional porque es proporcional al valor de la señal de error en cada instante de tiempo siendo esta acción esencial en cualquier regulador ya que es el error en cada instante de tiempo el que se pretende anular o incluso reducir. Esta acción tiene como principal característica la ganancia proporcional o *"Kp"*, esta ganancia es la que se multiplicará por la señal de error (entrada del regulador) para obtener la acción de control (salida del regulador). De este modo si se tiene un valor reducido de dicha ganancia la acción será suave y la respuesta será amortiguada y lenta, pasando lo contrario si es una ganancia elevada llegando a obtener una respuesta inestable.

$$u_p = K_p * e(t)$$

- Acción derivada: la acción derivada a diferencia de la proporcional tiene en cuenta la tendencia del error. De esta forma si la derivada del error es pequeña significa que el sistema avanza lentamente por lo que la acción derivada no será demasiado significativa, sin embargo, si la derivada de la señal de error es elevada significa que el sistema irá muy rápido por lo que será necesario actuar en sentido contrario para intentar limitar dicha acción. Por otro lado, existe la posibilidad de que estos cambios bruscos generen fuertes acciones de control que pueden dañar físicamente al sistema.

  Al tener la capacidad de deducir la tendencia al error se evitará la aparición de grandes sobreoscilaciones en el régimen transitorio además de ser capaz de mejorar el resto de las especificaciones dinámicas como el tiempo de establecimiento.

  Para que esta acción pueda ser aplicada será necesario incluir un polo y un cero al lugar de las raíces produciéndose nuevas curvaturas en él. De este modo y si es necesario regulando la ganancia se puede conseguir llevar polos a zonas complejas o consiguiendo que su nueva posición haga que el sistema sea estable.

  Finalmente, la acción es proporcional a la derivada del error siendo el parámetro de la ganancia derivada K_d el parámetro que determina la cantidad de acción que se incluirá en el controlador.

$$u_d(t) = K_d * \frac{de(t)}{dt}$$

- Acción Integral: A diferencia de la acción anterior la acción integral se centra en la duración del error haciendo que su existencia ya sea un problema. De esta manera si el error es pequeño la acción proporcional también lo será y puede que esta no sea lo suficientemente grande para eliminar dicho error de ahí que se empleen acciones integrales en ciertos reguladores.

Hay que entender que esta acción irá almacenando el error haciendo que este aumente con el paso del tiempo esta no dejará de crecer hasta que el error sea eliminado por completo. Con esto y en función del número de acciones integrales que se incluyan en el regulador se puede llegar a eliminar errores de posición, velocidad y aceleración. Ahora bien, se debe tener en cuenta de que si los errores no son eliminados lo suficientemente rápido se acumularán deprisa la respuesta tenderá a ser inestable.

Por lo general este tipo de acción se empleará para mejorar las especificaciones estáticas mejorando a su vez las características en régimen permanente, pero empeorando las de régimen estacionario.

Finalmente, como en casos anteriores tendremos una ganancia, la ganancia integral K_i, cuyo propósito será determinar la cantidad de acción integral que incluye el controlador.

$$u_{i(t)} = K_i * \int_{\tau=0}^{t} e(\tau) * d\tau$$

En cuanto al diseño de los reguladores, los PID son los más utilizados por ser muy sencillos tanto como de diseñar como de implementar además están basados en técnicas analíticas dando soluciones aproximadas al resultado requerido por las especificaciones estáticas y dinámicas. Mientras que los reguladores o controladores algebraicos dan la solución exacta de la función de transferencia global. Sin embargo, estos presentan dos principales inconvenientes: la elevada complejidad de diseñar uno de estos reguladores en control continuo y la posible aparición de oscilaciones ocultas en las respuestas temporales. Se pueden distinguir entre tres tipos distintos y un subtipo:

- Reguladores por asignación de polos.
- Reguladores por cancelación o síntesis directa.
    - Dead-Beat.
- Reguladores de tiempo finito

5.1.2.1 Reguladores proporcionales derivativos.

Tras ver el comportamiento de nuestro sistema ante una entrada en escalón figura 22 se puede determinar que en régimen estacionario posee cierto comportamiento oscilatorio. Además, si se desea implementar en un sistema físico que requiera cierta precisión presentaría un gran problema ya que tarda prácticamente 60 segundos en estabilizarse.

Por estas razones es viable el estudio de un regulador PD, de acción proporcional derivativa, dado que la acción derivativa corrige la aparición de grandes sobreoscilaciones y permite mejorar el tiempo de establecimiento del sistema. Asimismo, al añadir un polo y un cero se curvará el lugar de las raíces permitiendo que los polos que antes estaban fuera del circulo unidad ahora puedan estar dentro haciendo al sistema estable. Por su parte la acción proporcional permitirá mover los polos por el lugar de las raíces para conseguir la configuración deseada para el regulador.

Para el diseño de este regulador se ha empleado el comando “sisotool” en Matlab que permite estudiar controladores del tipo single-input single-output. Una vez abierto se mostrará el lugar de las raíces en la figura 27, unos diagramas de Bode y la respuesta temporal ante un escalón.

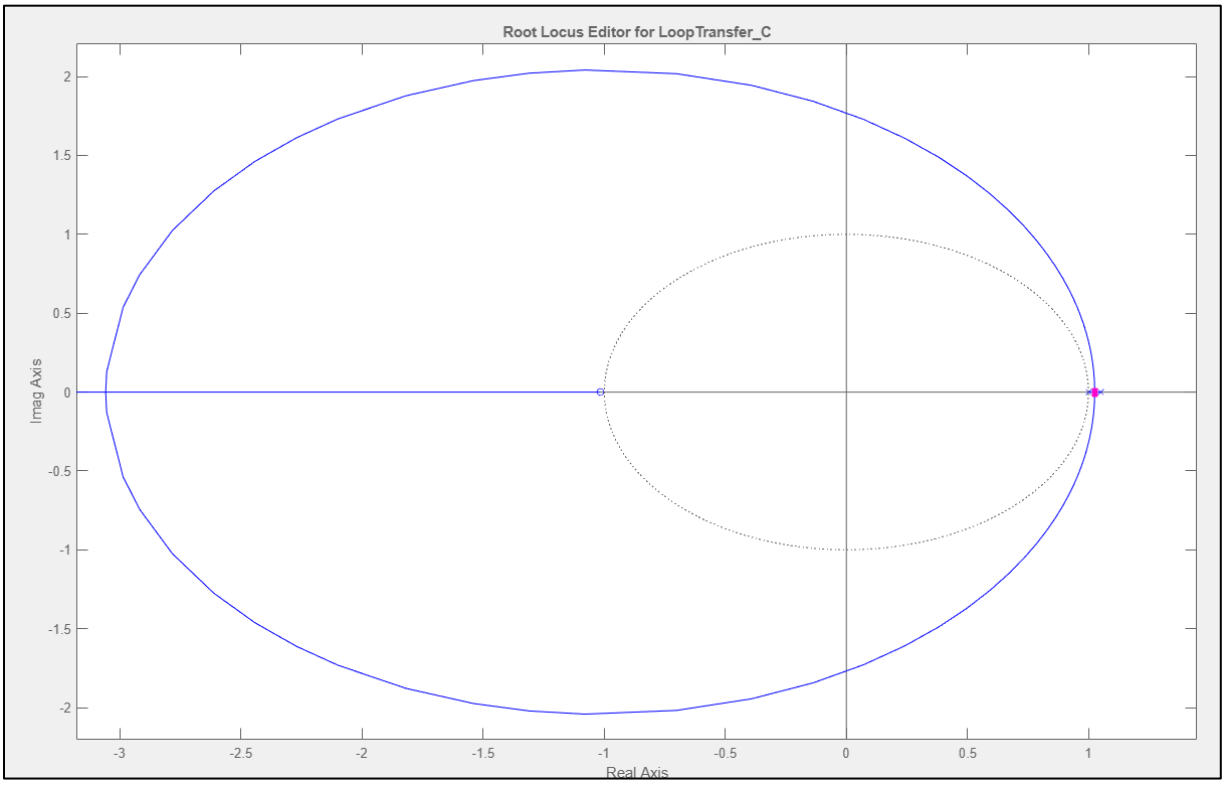


Figura 27. Lugar de las raíces sisotool

Hay que tener en cuenta previamente a la realización de este regulador que el sistema está formado por un doble integrador y un cero fuera del lugar de las raíces. La dinámica de este doble integrador hará que los costes de ganancia empleados para mover los polos sean bastante elevados.

Es cierto que existe la posibilidad de calcular a mano tanto las posiciones del cero y el polo como la ganancia necesaria para cumplir las especificaciones requeridas, pero en este caso se ha preferido diseñar el regulador moviendo a mano los elementos del lugar de las raíces para ver mejor cómo se comporta en cada parte. Por esta misma razón se han obtenido dos reguladores PD distintos que reducen el tiempo de establecimiento anterior y siguen llegando a la referencia.

Para ambos reguladores se ha mantenido la misma posición del polo y el cero añadidos. Siendo la posición del polo un valor muy cercano al centro del circulo unidad, lo que permite asegurar la realizabilidad del sistema y curvar el lugar de las raíces, además crea una rama que lo une al cero situado fuera del círculo unidad, visualizable en la figura 28.

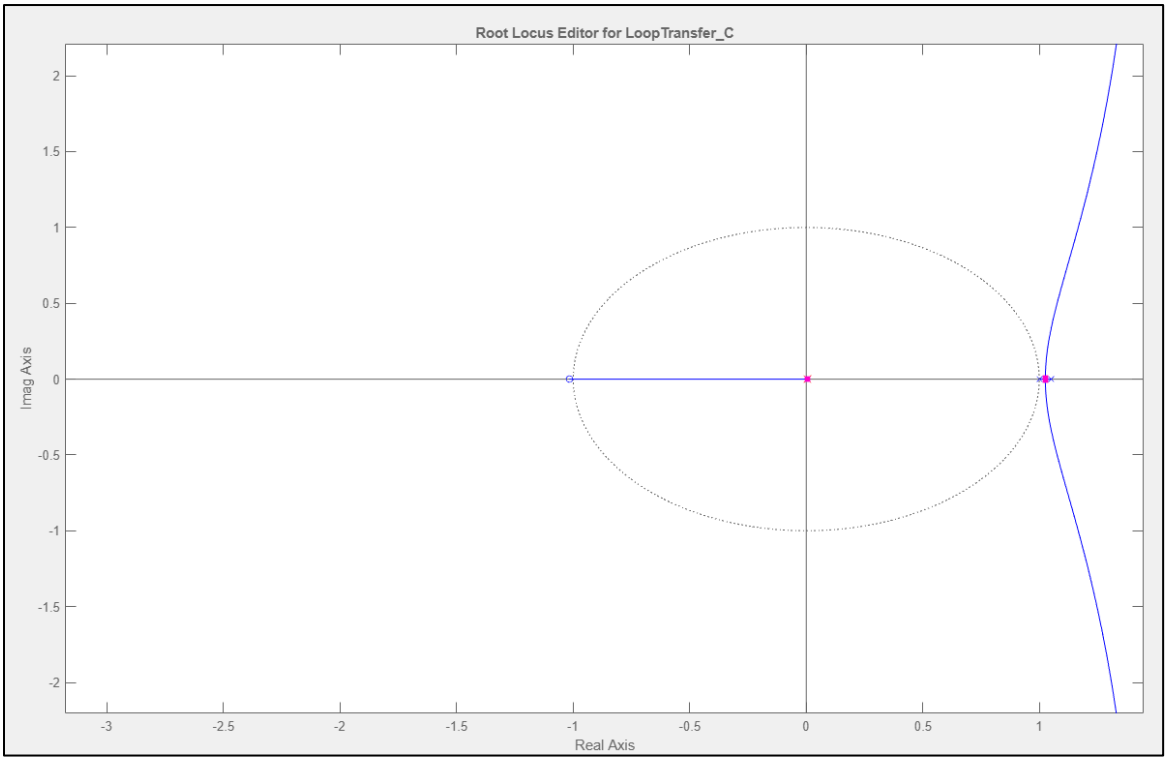


Figura 28. LDR con el polo en 0

Como se puede apreciar en la figura 29, para la posición del cero hay que tener en cuenta que se desea curvar el lugar de las raíces hacia dentro del círculo unidad y que cuanto más cerca esté del módulo 1, cerca del extremo del círculo unidad, el tiempo de establecimiento obtenido será menor.

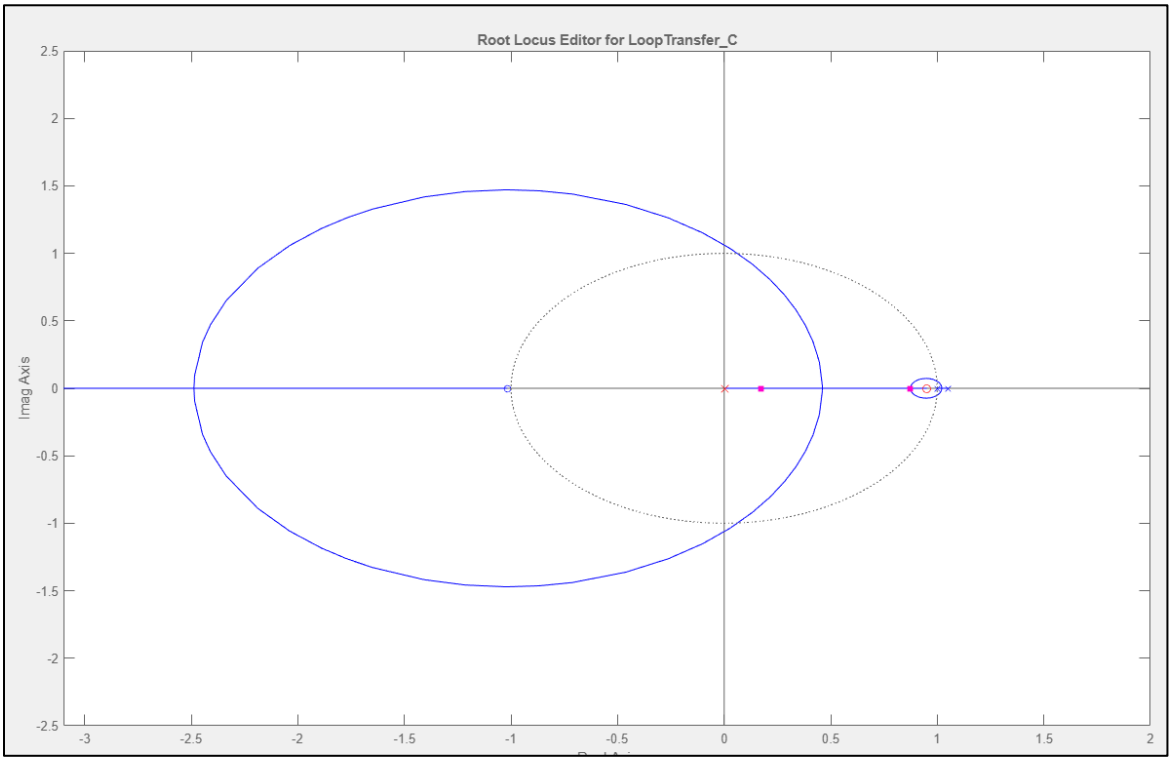


Figura 29. Lugar de las raíces con regulador sisotool.

Una vez incluidos el cero y el polo se procede a regular la ganancia. Con el primer regulador lo que se buscaba era reducir el tiempo de establecimiento y conseguir sobreoscilaciones menores.

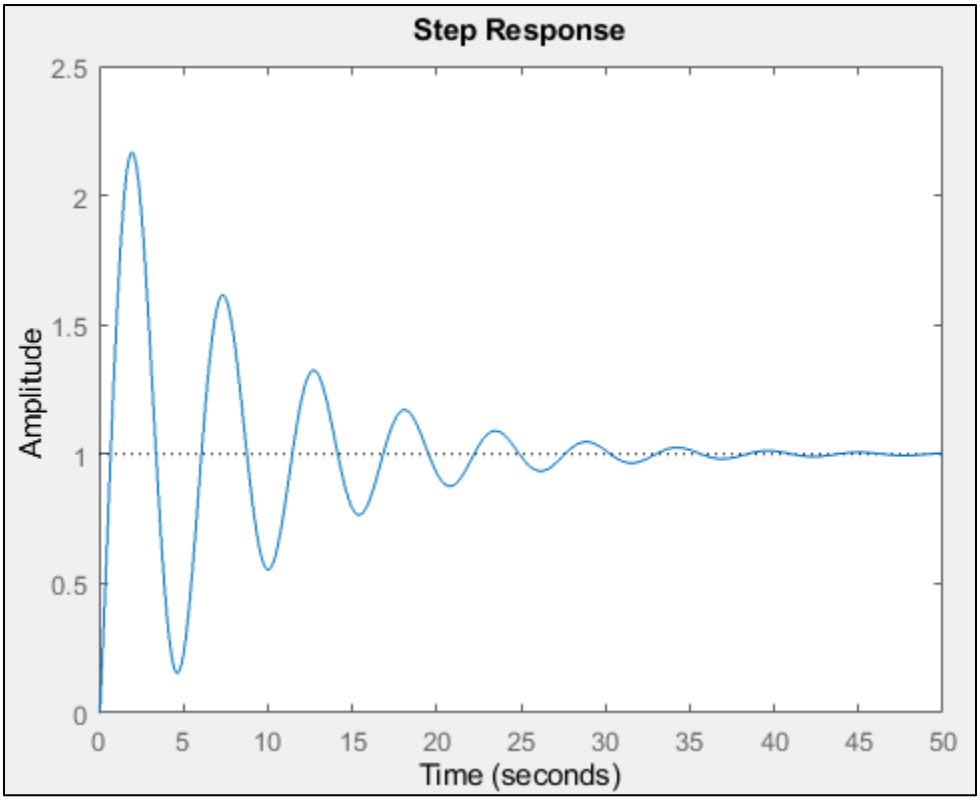


Figura 30. Respuesta temporal primer regulador

Y realmente se consiguió como se puede observar en la figura 30, sin embargo, los polos con esta ganancia estaban situados en la parte compleja del lugar de las raíces. Por lo que el objetivo del segundo regulador sería conseguir reducir el comportamiento oscilatorio e incluso reducir el tiempo de establecimiento. Para ello se aumentó la ganancia hasta conseguir que ambos polos coincidieran en el eje real.

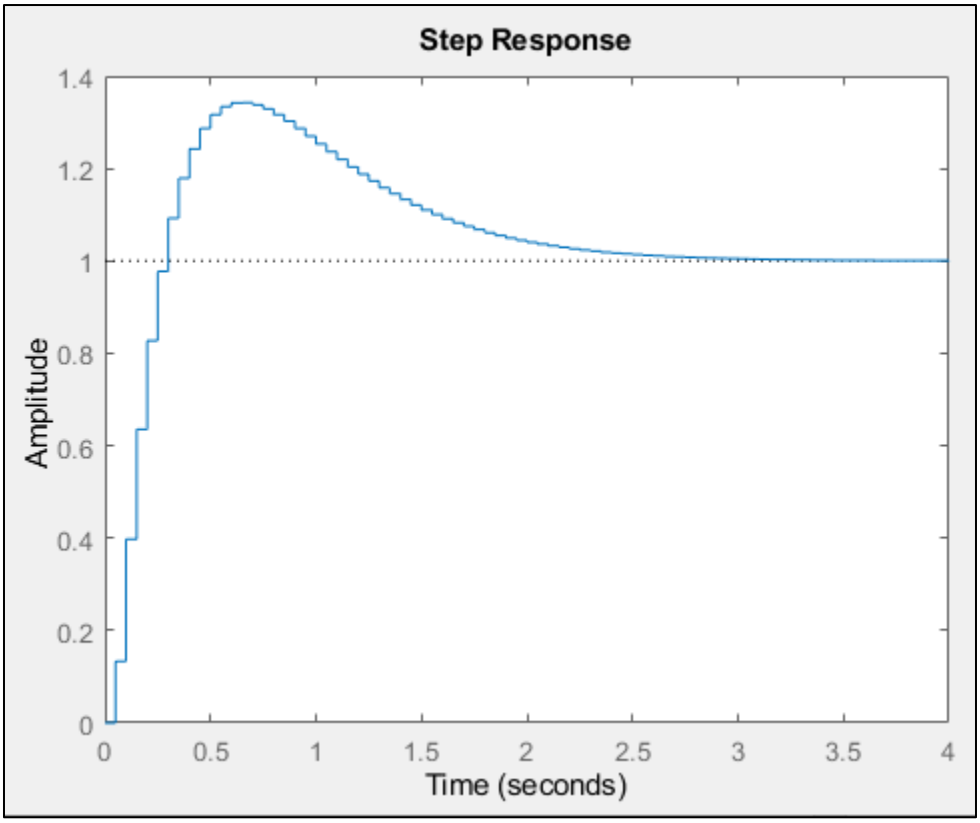


Figura 31. Respuesta temporal segundo regulador

Finalmente se obtuvo que el mayor pico de sobreoscilación era del 34%, apreciable en la figura 31, esto se puede reducir aplicando acción integral, pero haría que el regulador a diseñar fuera más complejo incluso la acción de control requerida podría dañar el sistema físico. Hay que destacar que se consigue un tiempo de establecimiento mucho menor a los anteriores casos.

Las funciones de transferencia de los reguladores obtenidos serían las siguientes mostradas en la figura 32:

$$\frac{92.249\ (z-0.9512)}{(z-0.003597)} \qquad \frac{366.6\ (z-0.9512)}{(z-0.003597)}$$

Figura 32. FDT regulador 1 y 2.

Lo más destacable de estas funciones es su elevada ganancia lo que se puede traducir en grandes variaciones de la acción de control. Para poder visualizarla se emplea el comando "step" que permite introducir una entrada del tipo escalón seguido del comando "feedback" con el que se puede crear un bucle cerrado. De forma que si introducimos `step(feedback(reg2,Gz))` en el command window de Matlab obtendremos las siguientes variaciones de la acción de control de los reguladores , correspondiendo la figura 33 con el primero y la figura 34 con el segundo.

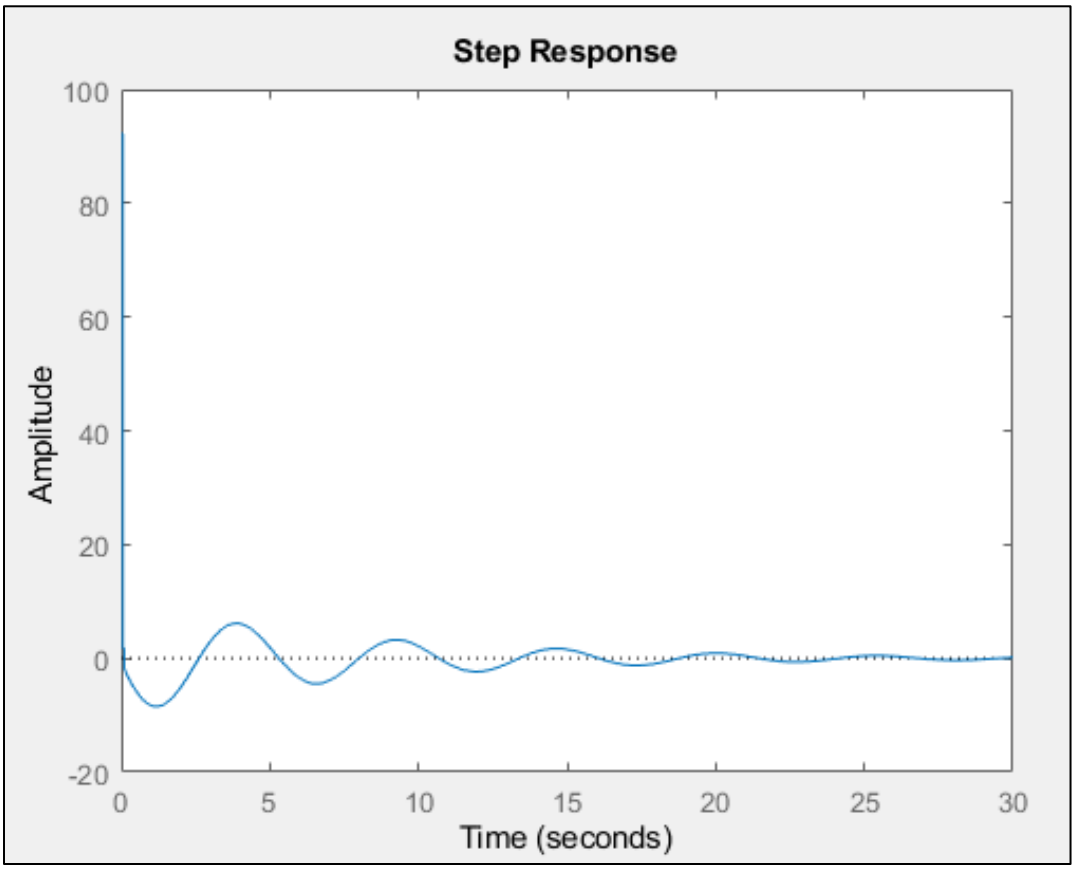


Figura 33. Acción de control regulador 1

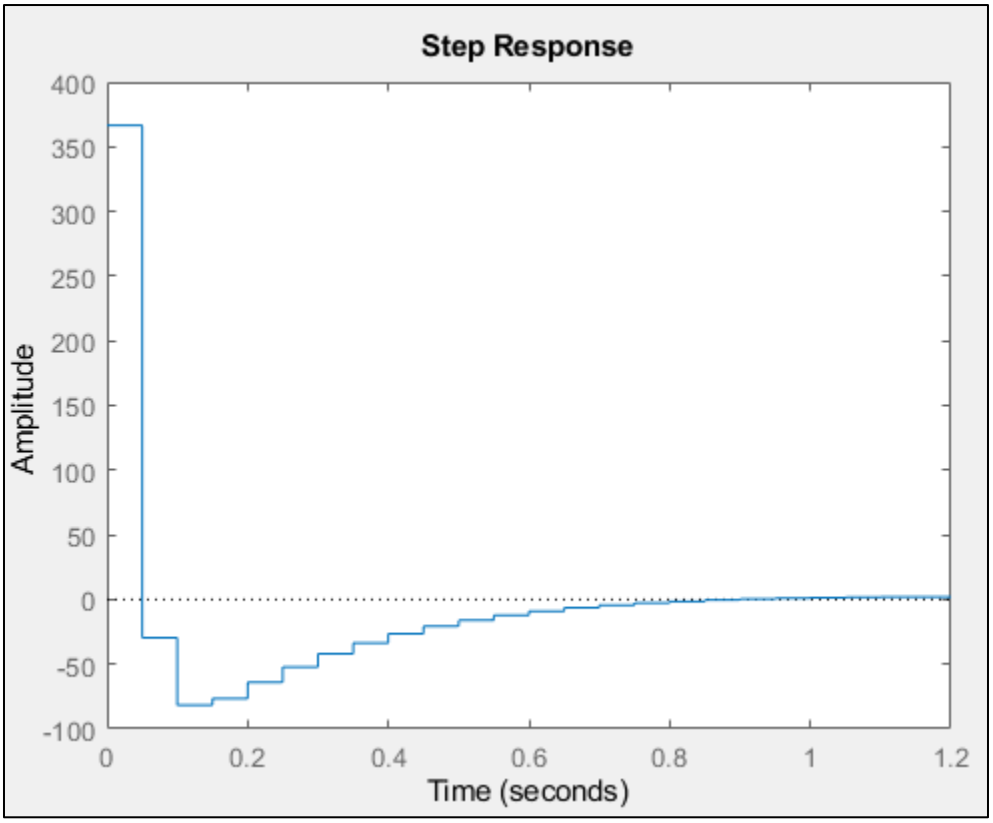


Figura 34. Acción de control regulador 2

De estas dos figuras se puede extraer que en ambos casos se produce una variación bastante importante de la acción de control en un tiempo muy reducido. Esto aplicado a un sistema físico podría llegar a ser perjudicial para sus actuadores. También se puede observar que sus acciones de control tienen formas similares a sus respuestas temporales, solo que están invertidas. Esto sucede gracias al efecto de la acción derivativa, cuando esta percibe el error que se va a producir actúa en sentido contrario para intentar alcanzar la referencia y hacer el sistema más estable.

Al igual que con el modelo de la planta se procede a implementar estos dos reguladores en la maqueta de Easy Java simulations. Para ello se variarán los valores del regulador en la ecuación en diferencias. Es importante recordar que al ser tratado como un sistema "SISO" el regulador es el mismo para las "x" y para las "y". Al igual que en el caso de la simulación de la planta se abstendrá de emplear los limitadores de error en cada uno de los experimentos, de esta forma se podrá visualizar con claridad el efecto del regulador en el sistema. Seguidamente se incluirán en las ecuaciones en diferencias los valores de las raíces del numerador y denominador tal y como se muestran en el código de las figuras 35 y 36.

```
//PD1
 var a1=0.003597, a2=0, a3=0, a4=0;
 var b0=92.25, b1=-87.75, b2=0, b3=0;
```

Figura 35. Valores Regulador 1 en la ecuación en diferencias.

El resultado es que el agente llega a rebasar en una primera pasada las coordenadas, pero la acción derivada se opone a la acción de control reduciendo de manera importante la sobreoscilación que presentaba el sistema de la planta. Antes de llegar a estabilizarse oscila un par de veces como era de esperar después ver su acción de control en Matlab. Finalmente llega a una estabilización aparente en muchísimo menos tiempo del esperado, alrededor de los 5 segundos, el motivo viene dado por lo explicado anteriormente, al cabo de unos segundos las oscilaciones producidas por la acción de control se vuelven casi imperceptibles.

```
//PD2
var a1=0.003597, a2=0, a3=0, a4=0;
var b0=366.6, b1=-348.7, b2=0, b3=0;
```

Figura 36. Valores Regulador2 en la ecuación en diferencias.

De igual manera que con el regulador anterior este sobrepasa ligeramente el destino en su primera pasada sin embargo al llegar al pico de la sobreoscilación este se dirige de una forma más directa y suave y sin oscilar directamente al punto. Consiguiendo un tiempo de establecimiento reducido respecto al otro regulador siendo este ligeramente superior a los 2 segundos.

5.1.2.2 Asignación de polos.

El objetivo de estos reguladores es conseguir que la ecuación característica de un sistema global en bucle abierto (que incluya una planta y un regulador) tenga sus raíces en una situación preestablecida obteniéndose así un comportamiento deseado.

Para el diseño se deberán tener en cuenta:

$$G_p(z) = \frac{B(z)}{A(z)} = \frac{b_m z^m + b_{m-1} z^{m-1} + \ldots + b_1 z + b_0}{a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0}$$

Donde el grado de las raíces del numerador de la planta tiene que ser menor que el denominador m<n y los polinomios B y A deben ser primos entre ellos.

$$G_r(z) = \frac{Q(z)}{P(z)} = \frac{q_\mu z^\mu + q_{\mu-1} z^{\mu-1} + \ldots + q_1 z + q_0}{p_\eta z^\eta + p_{\eta-1} z^{\eta-1} + \ldots + p_1 z + p_0}$$

Aquí se debe cumplir que $\eta \geq \mu$ para consolidar la causalidad del regulador.

Donde la ecuación característica será:

$$1 + G_p(z) + G_r(z) = 0$$

Si se sustituye por los polinomios:

$$A(z)P(z) + B(z)Q(z) = \prod_i (z - p_i)$$

Siendo $p_i$los polos que se desean asignar al bucle cerrado y el valor de i es la cantidad de estos polos que deberán incluirse. En el caso en el que el grado del numerador de la planta sea igual al del regulador se incluirán 2n-1 polos siendo n la cantidad de polos de la planta. Sin embargo, solo existirá solución única si:

$$n + \eta = \mu + \eta + 1$$

De esta forma habría el mismo número de incógnitas que de ecuaciones. Además, este controlador posee ciertas propiedades mostradas en [9]:

- La dinámica deseada en bucle cerrado dependerá de los polos en bucle cerrado y, por lo tanto, la estructura y los parámetros del controlador.
- Existe la posibilidad de cancelar los ceros del proceso inicial. Por lo que cualquier cero que esté situado en el círculo unidad podrá ser cancelado si en el controlador se ponen polos en su misma posición. Si se intentara cancelar un cero fuera del circulo unidad la respuesta del sistema en bucle cerrado podría tener un comportamiento inestable.

- El denominador del controlador contiene todos los ceros del modelo del proceso que se pueden cancelar. Por lo que hay que tener cuidado con los ceros situados dentro del circulo unidad porque pueden dar lugar a polos negativos que producirán oscilaciones ocultas.

Resumiendo lo planteado lo que se busca es la obtención de una función de transferencia global como la siguiente:

$$G_g(z) = \frac{B(z)Q(z)}{A(z)P(z) + B(z)Q(z)} = \frac{B(z)Q(z)}{\prod_{i=1}^{n+\eta}(z - p_i)}$$

La implementación del regulador según el método de asignación de polos se ha obtenido empleando un script de Matlab. Primeramente, se definen los polos, los ceros y la ganancia de la planta en control discreto tal y como se muestra en la figura 37. Estos valores son necesarios para calcular la ecuación característica mostrada anteriormente.

```
K = 0.00036317 ;
p1 = 1;
p2 = 1.051;
c1 = -1.017;
```

Figura 37. Valores de Gz en discreto.

Seguidamente se probará el valor de 0,85 como polo que se desea asignar en bucle cerrado. Además, se emplearán un polo y un cero en el regulador de forma que se cumpla la condición para obtener un solo resultado.

A continuación, se define la ecuación característica con los polinomios ya sustituidos y en función del polo, el cero y la ganancia que se han creado. Hay que destacar que el grado del polinomio del polo añadido es de 3, pero se le ha añadido una z porque solo se pueden poner como máxmo los polos en pares. Asimismo se emplea en comando de Matlab “coeffs” el cual devolverá por un lado los coeficientes “v” con los correspondientes grados de “t” (z^2,z,1) así como se muestra en el código de la Figura 38.

```
T_eq =(z-p1)*(z-p2)*(z-g1) + K*b*(z-c1)*(z-q1) - z*(z-pd)^2;
[v t] = coeffs(T_eq,z);
```

Figura 38. Ecuación característica asignación de polos.

Seguidamente, se emplea el comando “solve” para resolver los coeficientes obtenidos anteriormente, esto será guardado en una variable llamada “sol” la cual emplearemos después para resolver el sistema de ecuaciones en función de la ganancia b, el polo q1 y el cero g1 con el código de la figura 39. Obteniendo así para cada uno de ellos una solución única.

```
sol = solve(v);

b = double(sol.b(1));
g1 = double(sol.g1(1));
q1 = double(sol.q1(1));
```

Figura 39. Sistema de ecuaciones

Finalmente se obtiene la función de transferencia del regulador del controlador con los valores de b, q1 y g1. No sin antes dejar de utilizar la z como incógnita y volver a usarla como parte de la función de transferencia. Para ello se empleará el código de la figura 40 y se mostrará el resultado de la fdt del regulador en la figura 41.

```
clear z;
z = tf('z',T);
Gr = (b*(z-ql))/((z-gl))
```

Figura 40. Obtención Gr

```
506.3 (z-0.9393)
----------------
   (z+0.1671)
```

Figura 41. Gr asignación de polos

Previamente a su implementación en la maqueta sería interesante estudiar su comportamiento con la planta y su acción de control. Empleando la herramienta "sisotool" se puede observar que el lugar de las raíces, mostrado en la figura 42, con este regulador es muy similar a cuando se incluía el PD ya que tiene una distribución de polos y ceros muy parecida.

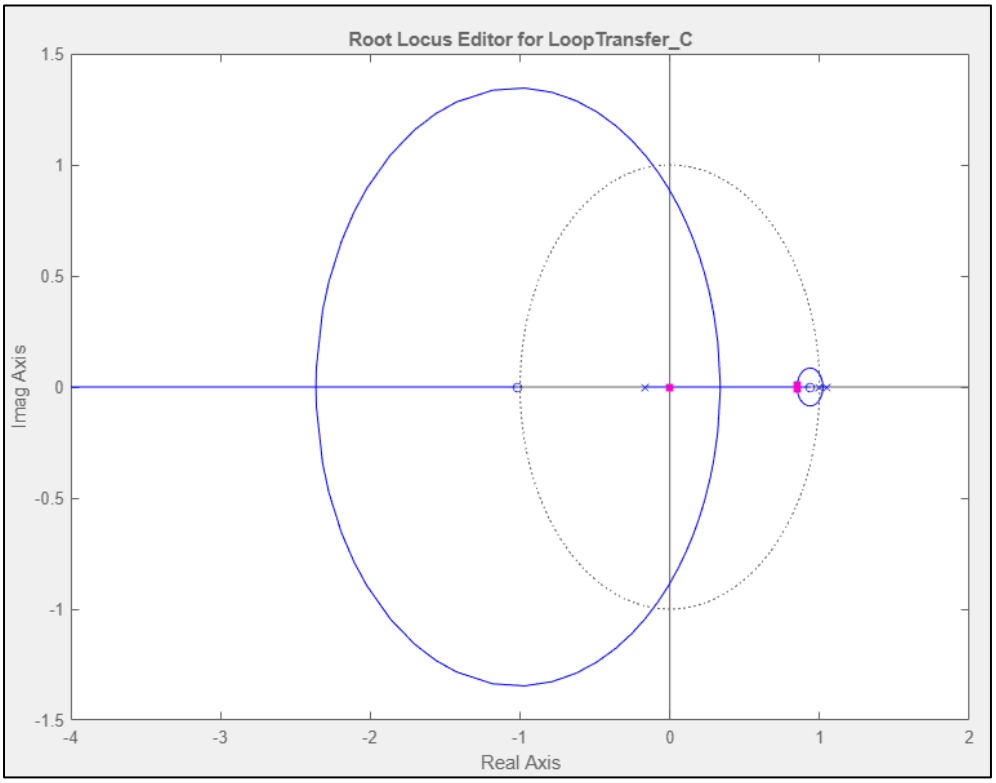


Figura 42. LDR asignación de polos.

Al igual que con los reguladores PD existe una leve sobreoscilación en el régimen transitorio de la figura 43. Pero a diferencia de ellos el regulador obtenido mediante asignación de polos tiene un tiempo de establecimiento mucho menor al del primer PD y muy similar al del segundo siendo este ligeramente más rápido.

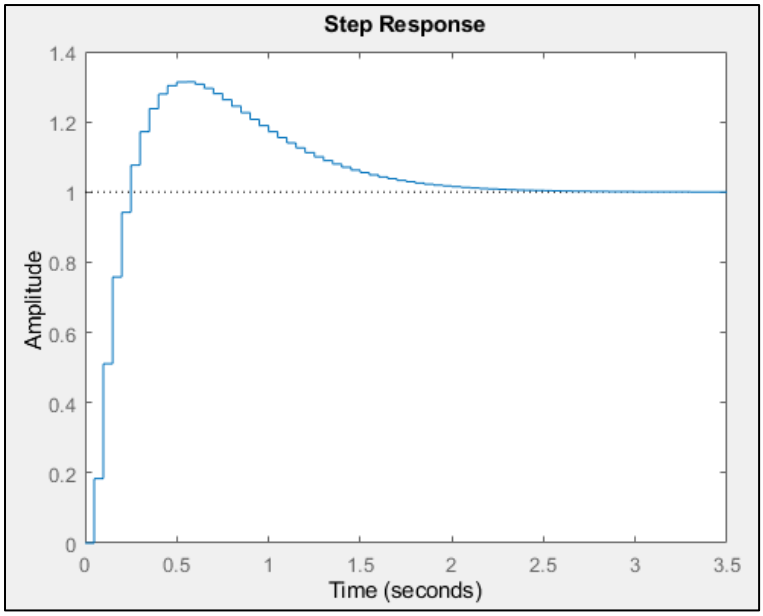


Figura 43. Respuesta temporal asignación de polos.

Para finalizar, en la gráfica de la acción de control, de la figura 44, al igual que en el resto de los reguladores existe una variación muy grande en muy poco tiempo de la misma acción de control. Además, esta variación es mayor que la del segundo regulador por lo que a pesar de ser ligeramente más rápido, este, lo consigue con una acción de control bastante mayor.

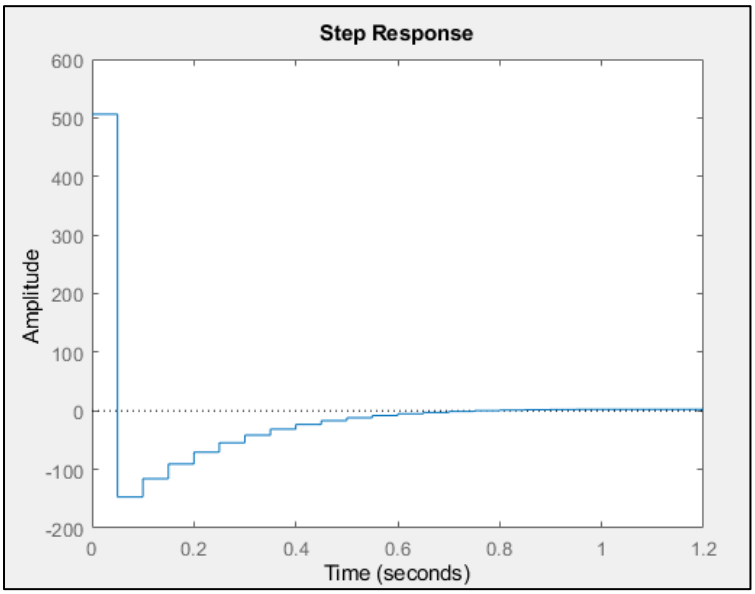


Figura 44. Acción de control asignación de polos

Viendo que la dinámica obtenida es muy similar a la de los reguladores anteriores es muy posible que responda de la misma manera en la maqueta. Pero antes de ello es interesante saber qué pasaría si el polo colocado en bucle cerrado tuviera una dinámica distinta. Esto se puede atisbar en la figura 45:

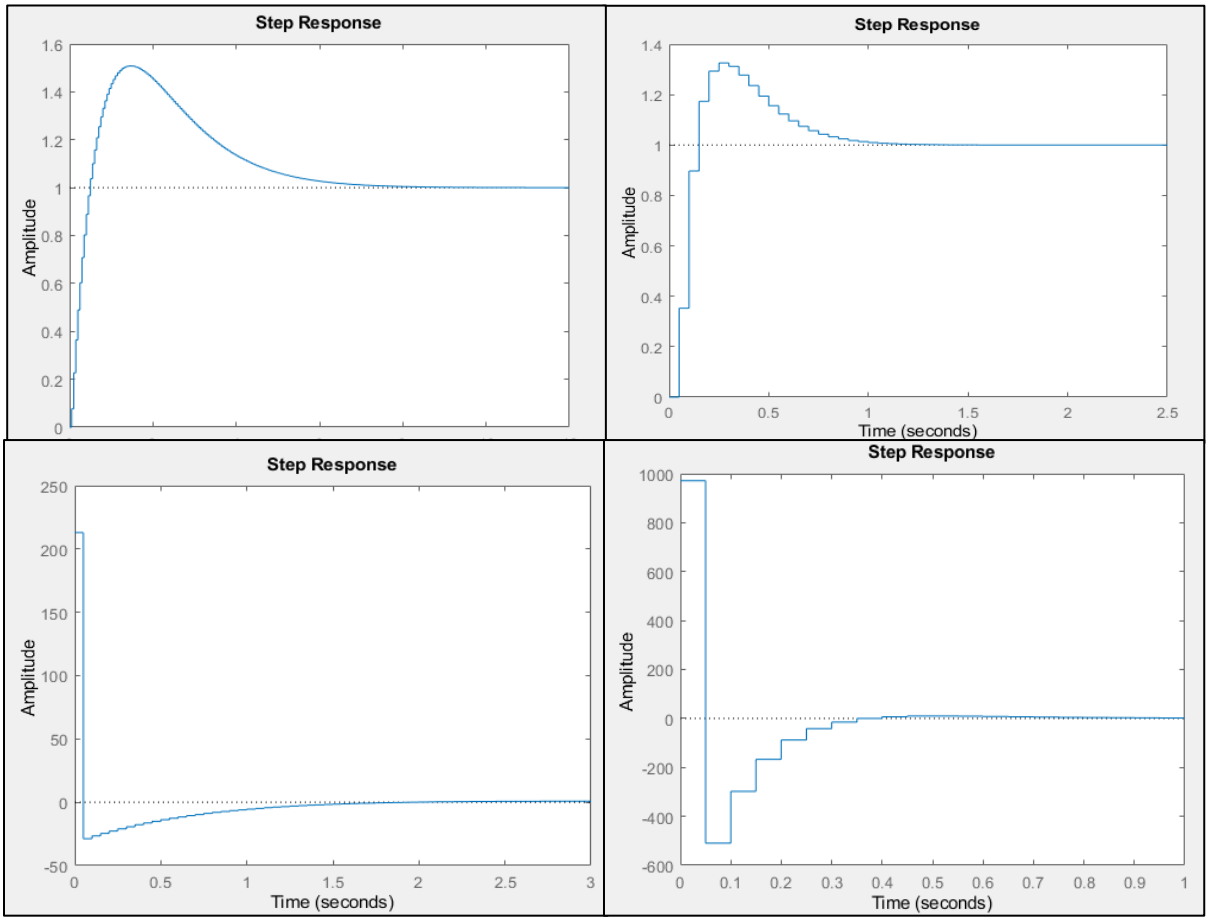


Figura 45. Polo en 0.95 vs Polo en 0.7

Como era de esperar al acercarse al círculo unidad el sistema tendrá una dinámica más lenta y requerirá de una menor acción de control, por otro lado, pasará lo contrario contra más se aleje.

Al igual que con los reguladores proporcionales no será necesaria ni la limitación de la acción de control ni la del error ya que la sobreoscilación no superará los límites del escenario. Al igual que en los anteriores casos se sustituirán los valores de la función de transferencia en la ecuación en diferencias mediante el código de la figura 46 y así obtener el efecto del regulador en la maqueta.

```
//Asignación de polos
  var a1=0.1671, a2=0, a3=0, a4=0;
  var b0=506.3, b1=-475.6, b2=0, b3=0;
```

Figura 46. Ecuación en diferencias Asignación de polos.

Respecto a su funcionamiento en la maqueta como era de esperar supera ligeramente el punto al que debe llegar el agente en la primera pasada, sin embargo, la acción del regulador hace que sobre oscile menos que con el segundo regulador proporcional y una vez llega al pico de esta oscilación se dirige rápidamente a las coordenadas consiguiendo así el mejor tiempo de establecimiento.

5.1.2.3 Síntesis directa o cancelación.

Al igual que en asignación de polos se asumirá que:

$$G_p(z) = \frac{B(z)}{A(z)} = \frac{b_m z^m + b_{m-1} z^{m-1} + \ldots + b_1 z + b_0}{a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0}$$

$$G_r(z) = \frac{Q(z)}{P(z)} = \frac{q_\mu z^\mu + q_{\mu-1} z^{\mu-1} + \ldots + q_1 z + q_0}{p_\eta z^\eta + p_{\eta-1} z^{\eta-1} + \ldots + p_1 z + p_0}$$

Teniendo que la función en de transferencia en bucle cerrado sería:

$$G_g(z) = \frac{G_p(z) G_r(z)}{1 + G_p(z) G_r(z)}$$

$$G_g(z) = \frac{B(z) Q(z)}{A(z) P(z) + B(z) Q(z)}$$

Por lo que si se despeja esta ecuación se obtiene la función de transferencia del controlador:

$$G_r(z) = \frac{1}{G_p(z)} \frac{G_g(z)}{1 - G_g(z)}$$

Hasta aquí todo parece ser igual, sin embargo, en este caso la elección del bucle cerrado no es el paso principal, sino que se debe realizar un estudio sobre si el sistema global junto a este controlador es realizable y estabilizable justo como se expone en [4].

- Realizabilidad: Una vez se sustituyen los polinomios de la planta y el regulador en la función de transferencia en bucle cerrado se obtiene:

$$G_g(z) = \frac{B(z) Q(z)}{A(z) P(z) + B(z) Q(z)}$$

Denominando "o" como grado del polinomio del numerado y "e" como grado del polinomio del denominador. Para que el sistema fuese realizable "e" debe ser igual o mayor a "o" de esta manera se tendría un mayor número de polos que de ceros y el sistema sería realizable. De igual manera:

$$o = m + \mu$$

$$e = n + \eta$$

Siendo la excedencia de polos sobre ceros en $G_g(z)$:

$$(n + \eta) - (m + \mu) = (n - m) + (\eta - \mu)$$

Por lo que si se implementa un regulador con el mismo número de ceros que de polos se tendría la misma diferencia entre polos y ceros en la función de transferencia en bucle cerrado y en la planta. Por lo tanto, se intuye la importancia de que $G_g(z)$ mantenga como mínimo la misma diferencia entre polos y ceros que el proceso.

- Estabilidad: Como se puede intuir viendo las ecuaciones anteriormente mostradas el regulador cancela polos y ceros del proceso, haciendo que los polos del proceso pasen a ser sus ceros y los ceros del proceso pasen a ser sus polos. Por ese motivo si hay algún polo o cero fuera del circulo unidad pueden darse comportamientos inestables.

A continuación, se mostrará el desarrollo suponiendo la existencia de un polo y un cero inestable

$$G_r(z) = \frac{B(z)}{A(z)} = \frac{(z-\beta) * B'(z)}{(z-\alpha) * A'(z)}$$

Sustituyendo en la ecuación de la fdt del regulador:

$$G_r(z) = \frac{(z-\alpha) * A'(z)}{(z-\beta) * B'(z)} \frac{G_g(z)}{1 - G_g(z)}$$

$$G_r(z) = \frac{(z-\alpha) * Q'(z)}{(z-\beta) * P'(z)}$$

Dando lugar a la siguiente ecuación característica:

$$(z-\beta) * (z-\alpha) * [\, A'(z)P'(z) + B'(z)Q'(z)] = 0$$

Cuyo resultado en bucle cerrado será inestable ya que al sacar factor común los polos y los ceros no pueden ser cancelados. Debido a esta cuestión se plantean dos restricciones.

- Si existen polos inestables el factor 1-$G_g(z)$ debe contenerlos como ceros.
- Si existen ceros inestables el factor $G_g(z)$ debe contenerlos como polos.

Sabiendo esto se procede a crear la función de transferencia en bucle cerrado del sistema en tiempo discreto donde se le han añadido los polos en 0.9 por su cercanía al círculo unidad del lugar de las raíces. Consiguiéndose así una acción de control más reducida y un tiempo de establecimiento mayor que si fuera más cercano a 0.

$$G_{bc} = \frac{K_{bc} * (z + 1.017)}{(z - 0.9)^2}$$

$$1 - G_{bc} = \frac{(z - 0.9)^2 - K_{bc} * (z + 1.017)}{(z - 0.9)^2}$$

Como $1 - G_{bc}$ debe tener una diferencia entre polos y ceros de 1 se le debe añadir un grado de libertad en forma de cero.

$$1 - G_{bc} = \frac{(z - 1.051) * (z - \alpha_2)}{(z - 0.9)^2}$$

Solo se tendría como incógnitas la ganancia en bucle cerrado y α_2 tendiendo así dos ecuaciones y dos incógnitas. Sin embargo, si se desea añadir la ecuación del error de posición que hace que sea 0 no podríamos cumplir todas las especificaciones obligando a añadir más grados de libertad.

$$G_{bc} = \frac{K_{bc} * (z + 1.017) * (z - \alpha_1)}{z * (z - 0.9)^2}$$

$$1 - G_{bc} = \frac{(z - 0.9)^2 - K_{bc} * (z + 1.017) * (z - \alpha_1)}{z * (z - 0.9)^2}$$

$$1 - G_{bc} = \frac{(z - 1.051) * (z - \alpha_2) * (z - \alpha_3)}{z * (z - 0.9)^2}$$

Al añadir otra incógnita en $1 - G_{bc}$ es necesario aumentar el grado del denominador en 1, de ahí que

se haya añadido la *"z"*. Además, al tener grado 3 en denominador de $1 - G_{bc}$ se deberá aumentar de la misma manera en G_{bc}, a su vez, creando la necesidad de añadir otra incógnita en el numerador. Resolviendo los polinomios del numerador se obtienen las siguientes ecuaciones:

$$z^3 - z^2 * (\alpha_2 + \alpha_3) + z * \alpha_2 * \alpha_3 - 1.051 * z^2 + 1.051 * (\alpha_2 + \alpha_3) - 1.051 * \alpha_2 * \alpha_3$$

$$z^3 - z^2 + 0.81 * z - K_{bc} * (z^2 + z * (1.017 - \alpha_1) - 1.017 * \alpha_1)$$

Para crear el sistema de ecuaciones se deberán despejar en función del grado del polinomio dando asi un sistema con cuatro ecuaciones y cuatro incógnitas, modeladoe en el código de la figura 47. Donde se incluye una cuarta ecuación para forzar un error de posición 0 a pesar de que la planta ya incluya un integrador.

Debido a su complejidad el sistema fue resuelto en matlab. Para ello se creó un nuevo script donde se definieron las ecuaciones ya despejadas en función del grado de "z".

```
function F = root2d(x)

F(1) = -x(2) - x(3) - 1.0551 + 1 + x(4);
F(2) = x(2)*x(3) + 1.051*(x(2) + x(3)) - 0.81 + x(4)*(1.017 - x(1));
F(3) = -1.051*x(2)*x(3) - 1.017*x(4)*x(1);
F(4) = x(4)*(1 + 1.017)*(1 - x(1)) - 0.81;
```

Figura 47. Sistema de ecuaciones Matlab síntesis directa.

Con las siguientes líneas del código de la figura 48 se llama al script y se resuelve el sistema mediante el comando *"solve"*.

```
fun = @root2d
x0 = [0,0,0,0];
x = fsolve(fun,x0)
```

Figura 48. Resolver sistema de ecuaciones síntesis directa.

Es importante tener en cuenta que uno de los valores de α_2 o α_3 debe ser 1 para que se cumpla que el error de posición sea nulo. Una vez resuelto se definirán las funciones de transferencia, en la Figura 49, del controlador y esta misma restada a uno para cumplir con la ecuación de formación del regulador.

```
DBC = (x(4)*(z+1.017)*(z-x(1)))/(z*(z-0.9)^2);
ADBC = ((z-1.051)*(z-x(2))*(z-x(3)))/(z*(z-0.9)^2);
```

Figura 49. Definición de las fdt síntesis directa.

Donde se obtendría una función de transferencia como se muestra en la figura 50, cuya diferencia entre polos y ceros es la misma que la del diseño del proceso por lo que se cumpliría el criterio de realizabilidad. Solo queda sustituir en la ecuación de la función del regulador las fdt obtenidas anteriormente:

```
Gr= (1/Gz)*(DBC/ADBC);
Gr = minreal(Gr,0.001)
```

Figura 50. Obtención Gr síntesis directa.

Finalmente, la función de transferencia para el diseño del regulador corresponderá con la figura 51:

$$\frac{2132.8\ (z-1)\ (z-0.4815)}{(z-1.06)\ (z+0.3405)}$$

Figura 51. FDT del regulador síntesis directa.

A continuación, se mostrará el lugar de las raíces correspondiente con el producto de la acción del regulador con la planta en bucle cerrado. En la figura 52 se puede apreciar como los polos y los ceros no han sido cancelados directamente si no que los que han sido añadidos según las restricciones son capaces de curvar el lugar de las raíces. Para mitigar la acción de los que estuvieran situados fuera del círculo unidad, de esta forma se puede comprobar la estabilidad del sistema de una forma visual.

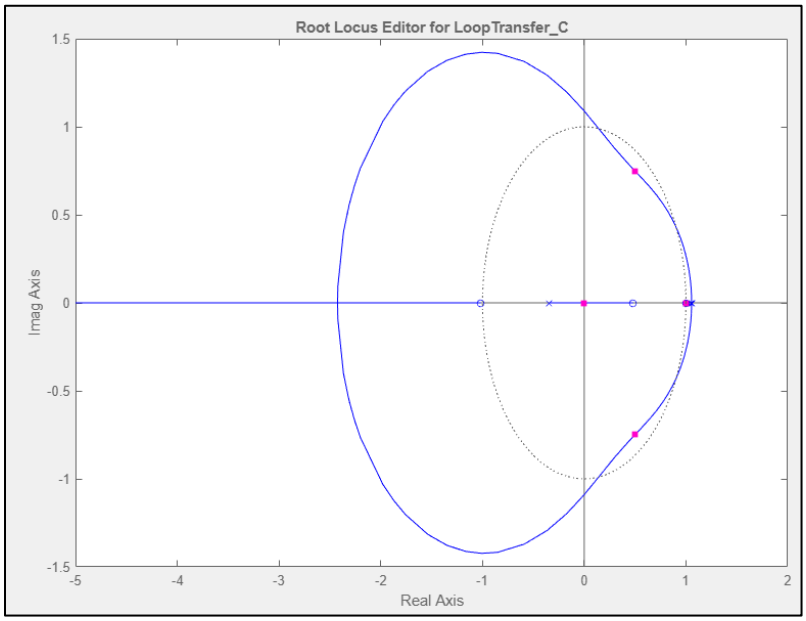


Figura 52. LDR de DBC.

Cuya respuesta temporal ante una entrada en escalón será la siguiente mostrada en la figura 53:

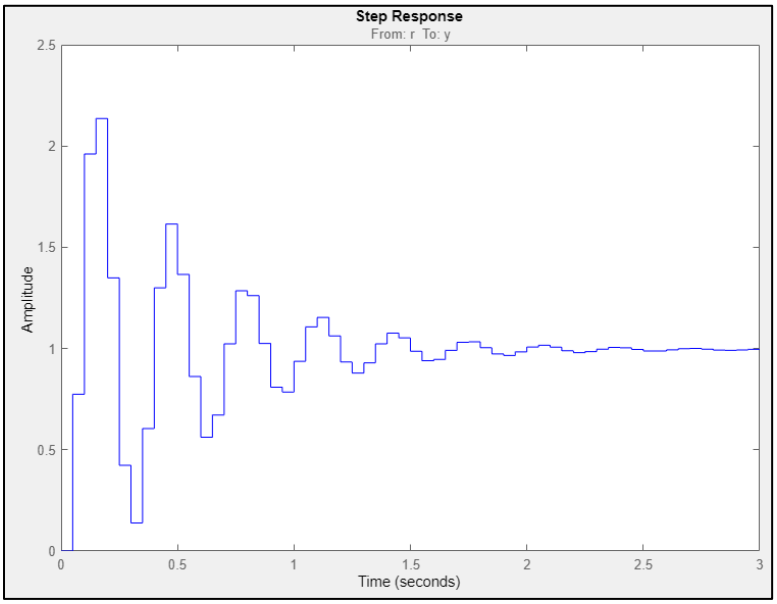


Figura 53. Respuesta temporal síntesis directa.

Esta respuesta presenta un elevado número de oscilaciones en un tiempo muy reducido lo que significa que en ese espacio de tiempo sufrirá grandes variaciones de la acción de control. Por otra parte, el sistema se estabiliza rápidamente.

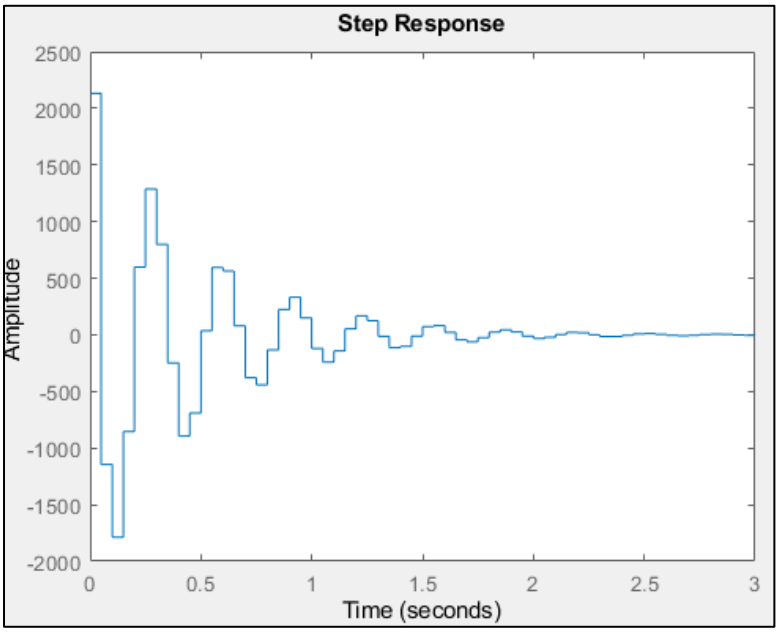


Figura 54. Acción de control síntesis directa.

En última instancia se encontrará la figura 54 cuya representación es la gráfica de la acción de control cuyo comportamiento es exactamente como se esperaba. Produciéndose un movimiento oscilatorio muy rápido en el régimen estacionario que a su vez produce una gran variación en la acción de control.

A continuación, se incluirán los valores del polinomio del numerador y el denominador dentro del cálculo de la ecuación en diferencias como se representa En el código de la figura 55.

```
//Sintesis directa
 var a1=-0.7195, a2=0.3609, a3=0, a4=0;
  var b0=2133, b1=-3160, b2=1027, b3=0;
```

Figura 55. Valores de la síntesis directa en la ecuación en diferencias.

En un principio es cierto que posee elevados cambios en la acción de control además es un sistema que presenta un gran número de sobreoscilaciones en régimen transitorio, pero inicialmente se prescindirán de los limitadores de la acción de control y de error para visualizar una simulación lo más fiel posible al modelo obtenido con Matlab.

Una vez iniciada la simulación el agente se dirige rápidamente al punto rebasándolo ligeramente y reproduciendo las oscilaciones que anteriormente se visualizaron en la Figura X.X. No obstante, cuando parece que se dispone a estabilizarse comienza a realizar sobreoscilaciones mayores hasta que finalmente satura.

Este resultado se debe al efecto Windup, este aparece cuando se posee acción integral y hay cierto nivel de saturación, de modo que el error de saturación se integra generando un deterioro en la respuesta haciendo que esta sea inestable aun cuando el sistema se encuentre prácticamente estabilizado. Este efecto no ha podido ser detectado durante el estudio de la señal porque Matlab suele presentar aproximaciones de la salida que no tienen por qué ser reales. Sin embargo, se puede visualizar su efecto con un sencillo esquema de Simulink que estará dispuesto como sería el sistema en bucle cerrado con el regulador y la planta en la parte superior y con una realimentación negativa de la salida tal y como representa la figura 56.

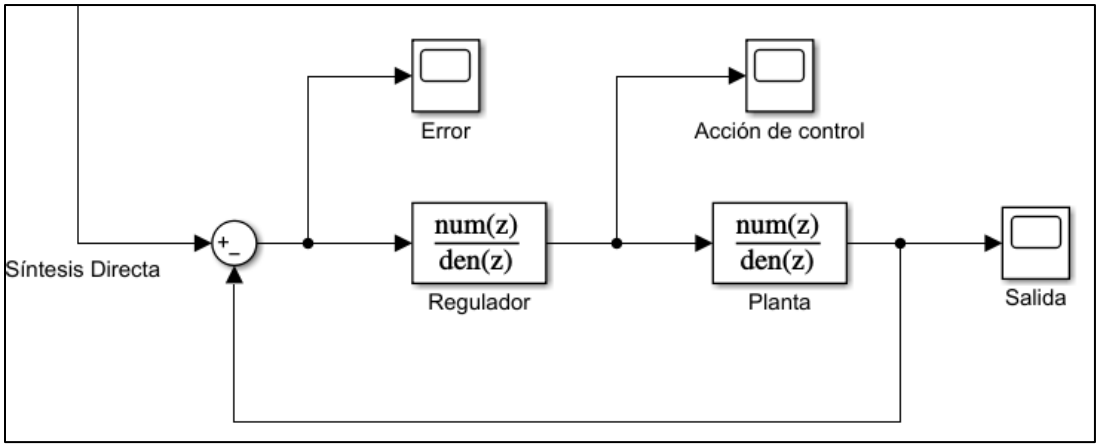


Figura 56. Esquema Windup Simulink.

Una vez se ejecute la simulación se obtiene la siguiente respuesta del error en la figura 57, donde se puede apreciar como este se va acumulando. Haciendo a su vez que la acción de control sature en la figura 58.

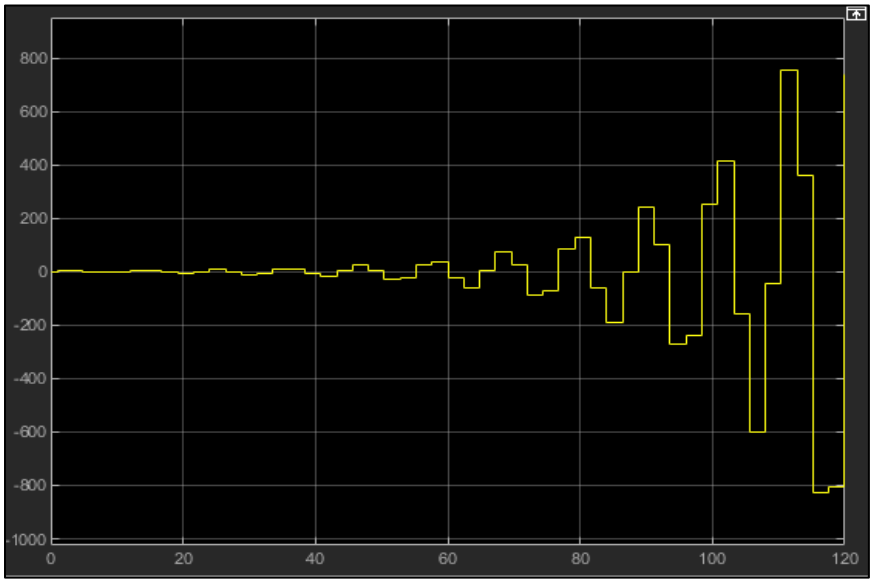


Figura 57. Error Windup Simulink.

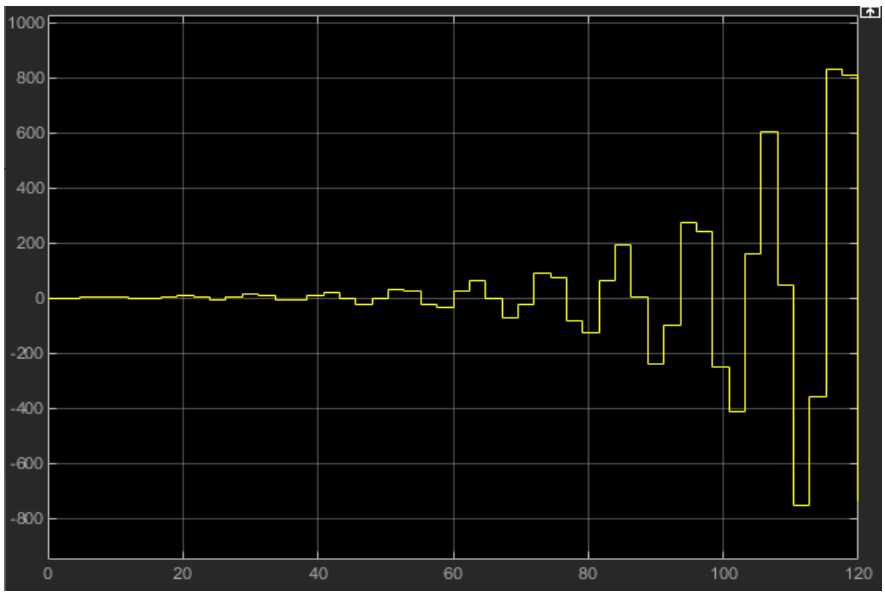


Figura 58. Acción de control Windup Simulink.

El efecto windup puede ser contrarrestado de dos formas diferentes expuestas en [10], siendo la primera una ya propuesta en el mismo Figura como es la limitación de la acción de control, pero esta presenta un ligero problema y es que ralentizará demasiado el sistema. Por otro lado, existe la posibilidad de implementar un sistema de descarga de la acción integral cuyo efecto es evitar que se acumule el error. La implementación de este último sería tan sencilla como incluir un bloque de saturación o de limitación de la señal realimentando este al integrador junto a una ganancia a diseñar figura 59.

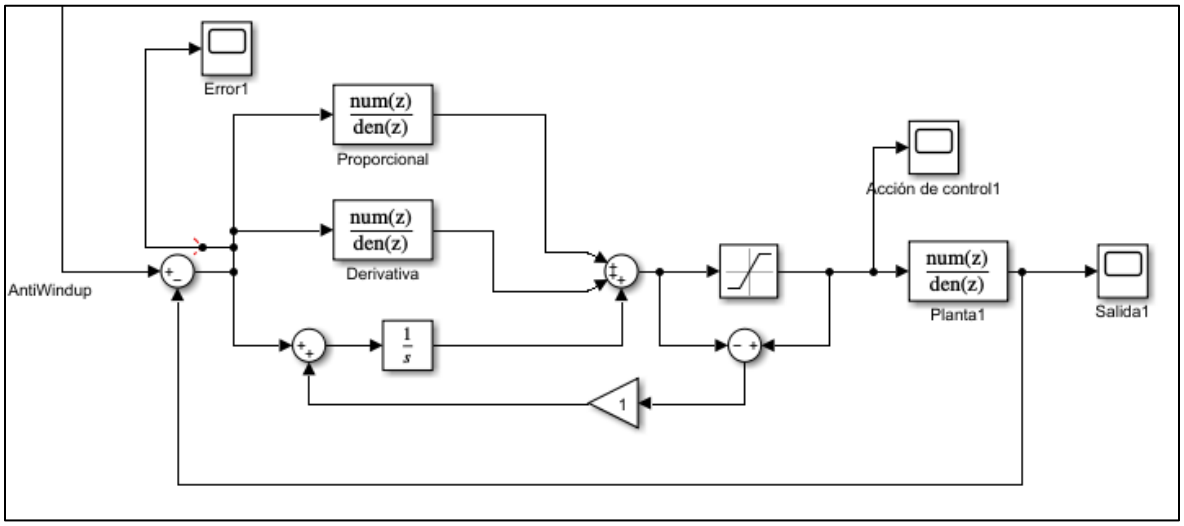


Figura 59. Anti-Windup Simulink.

5.1.2.4 Dead-Beat.

El controlador Dead-Beat o controlador de tiempo mínimo permite alcanzar la referencia en el mínimo número de periodos de muestreo posibles. Cuyo mínimo tiempo de establecimiento viene dado por la expresión:

$$T_e = n * T$$

Donde "T" será el valor del periodo de muestreo y "n" el orden del sistema en lazo cerrado. De igual manera que con el método de síntesis directa se deberán tener en cuenta los mismos criterios de estabilidad y realizabilidad incluso se deberán de respetar las mismas restricciones en caso de existir polos o ceros alternantes o que estén situados fuera del círculo unidad. Para alcanzar dichas características se deben de poner todos los polos del sistema en la posición z = 0.

$$G_g(z) = \frac{K * (z + a) * (z + b) * \dots}{z^n}$$

Debido a estas especificaciones este tipo de controladores presenta una serie de limitaciones y desventajas:

- Es usual que el sistema sature debido a las elevadas acciones de control necesarias para alcanzar la referencia en tan poco tiempo.
- Un mal modelado de la planta puede causar respuestas muy pobres.

El planteamiento será muy parecido al de síntesis directa solo que esta vez los polos son definidos directamente como z=0. De la misma manera se empieza partiendo de que la diferencia entre numerador y denominador en la fdt del sistema en bucle cerrado es de 1, por lo que se incluirá tan solo un grado de libertad en $1 - G_{bc}$.

$$G_{bc} = \frac{K_{bc} * (z + 1.017)}{z^2}$$

$$1 - G_{bc} = \frac{z^2 - K_{bc} * (z + 1.017)}{z^2}$$

$$1 - G_{bc} = \frac{(z - 1.051) * (z - \alpha_2)}{z^2}$$

Al igual que en apartado anterior, solo se tendría como incógnitas la ganancia en bucle cerrado y α_2 tendiendo así dos ecuaciones y dos incógnitas. No obstante, si se desea añadir la ecuación del error de posición que hace que sea 0 no podríamos cumplir todas las especificaciones obligando a añadir más grados de libertad.

$$G_{bc} = \frac{K_{bc} * (z + 1.017) * (z - \alpha_1)}{z^3}$$

$$1 - G_{bc} = \frac{z^3 - K_{bc} * (z + 1.017) * (z - \alpha_1)}{z^3}$$

$$1 - G_{bc} = \frac{(z - 1.051) * (z - \alpha_2) * (z - \alpha_3)}{z^3}$$

Una vez obtenido un sistema con cuatro incógnitas y cuatro ecuaciones se procede con la resolución del polinomio:

$$z^3 - z^2 * (\alpha_2 + \alpha_3) + z * \alpha_2 * \alpha_3 - 1.051 * z^2 + 1.051 * (\alpha_2 + \alpha_3) - 1.051 * \alpha_2 * \alpha_3$$

$$z^3 - K_{bc} * (z^2 + z * (1.017 - \alpha_1) - 1.017 * \alpha_1)$$

A continuación, se creará otro script en el que se incluirá el código de la figura 60 con las ecuaciones ya despejadas e igualadas en función del valor del grado de la incógnita "z" que les esté multiplicando.

```
function F = root3d(x)

F(1) = -x(2) - x(3) - 1.0551  + x(4);
F(2) = x(2)*x(3) + 1.051*(x(2) + x(3)) + x(4)*(1.017 - x(1));
F(3) = -1.051*x(2)*x(3) + x(4)*(-1.017 * x(1));
F(4) = x(4)*(1 + 1.017)*(1 - x(1)) - 1;
```

Figura 60. Sistema de ecuaciones Matlab Dead-Beat.

Una vez definidas las ecuaciones se procede con la obtención valor de cada una de las incógnitas añadidas al sistema mediante el comando "fsolve", mediante el código de la figura 61.

```
fun = @root3d
x0 = [0,0,0,0];
x = fsolve(fun,x0)
```

Figura 61. Resolver sistema de ecuaciones Dead-Beat.

Es importante tener en cuenta que uno de los valores de α_2 o α_3 debe ser 1 para que se cumpla que el error de posición sea nulo. Una vez obtenidos todos los valores, entre los que, si se encuentra un valor unitario para las incógnitas añadidas, ya se pueden definir las funciones de transferencia del sistema en bucle cerrado empleando las líneas de código de la figura 62.

```
DBC = (x(4)*(z+1.017)*(z-x(1)))/(z^3);
ADBC = ((z-1.051)*(z-x(2))*(z-x(3)))/(z^3);

Gr= (1/Gz)*(DBC/ADBC);
Gr = minreal(Gr,0.001)
```

Figura 62. Definición de las fdt Dead-Beat.

Siendo la función de transferencia del regulador la mostrada en la figura 63:

$$\frac{3659.8\ (z-1)\ (z-0.627)}{(z-1.045)\ (z+0.7714)}$$

Figura 63 FDT del regulador Dead-Beat.

Como se apreciar en la Figura anterior (Figura 63) la diferencia entre polos y ceros en esta fdt es de 0 por lo que al ser multiplicada por la de la planta se obtendría la misma diferencia que en la función del sistema en bucle cerrado por lo que seguramente el sistema sea realizable. Por otro lado, si se desea ver de una forma visual la realizabilidad y estabilidad del sistema se mostrará a continuación el lugar de las raíces del sistema en bucle cerrado de la figura 64:

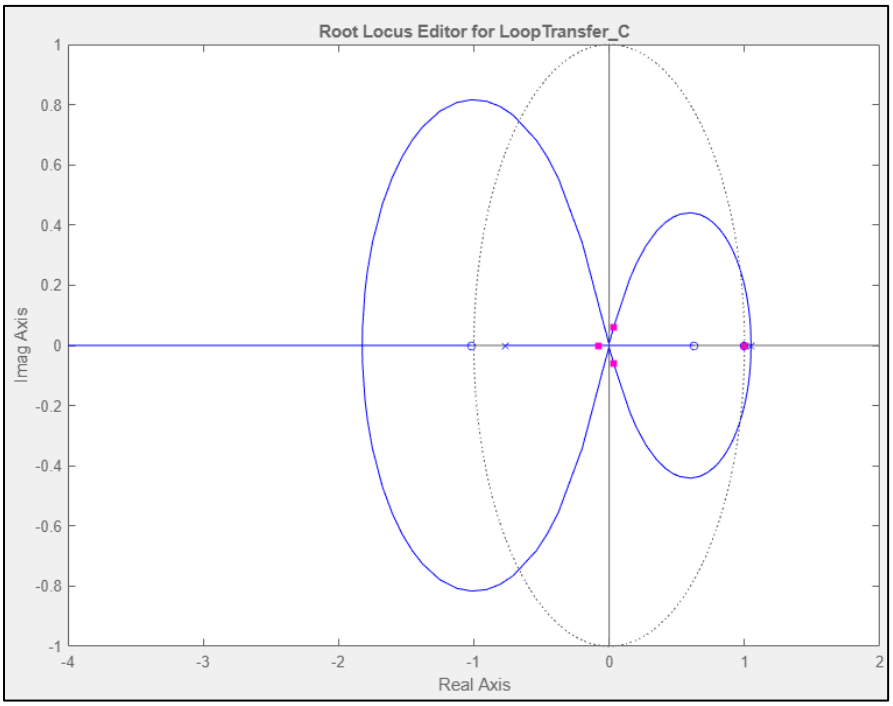


Figura 64. LDR del regulador con la planta Dead-Beat.

Donde al igual que en síntesis directa los polos no han sido cancelados directamente si no que el LDR ha sido curvado de tal forma que los polos estén dentro del circulo unidad. Por otro lado, con el valor de la ganancia anteriormente obtenido hay un polo situado en el eje negativo del lugar de las raíces lo que significa que existe la posibilidad de que se produzca oscilaciones ocultas.

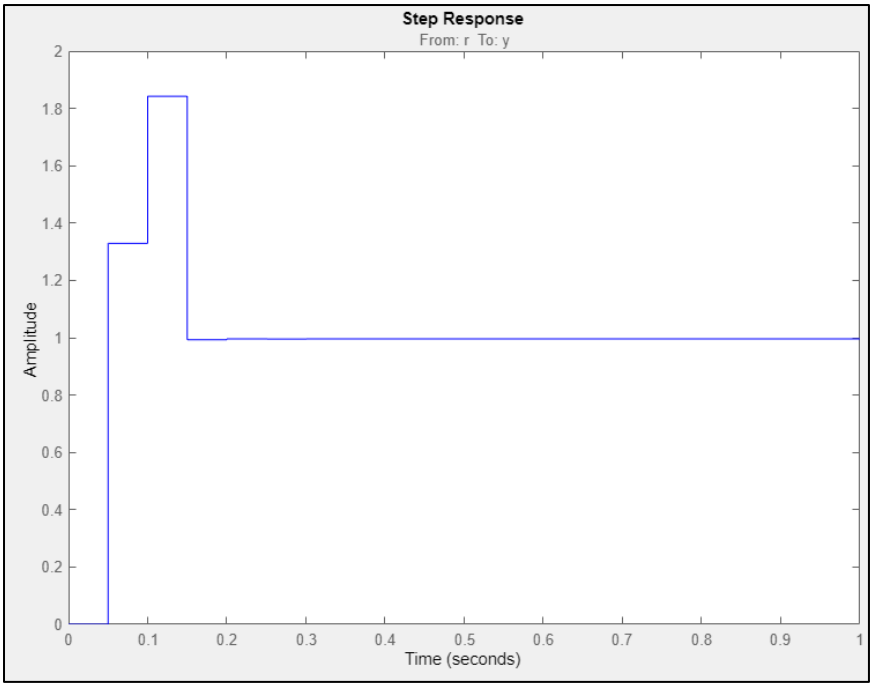


Figura 65. Respuesta temporal Dead-Beat.

La respuesta del sistema ante un escalón unitario, tal y como muestra la figura 65 es extremadamente rápida. Cabe destacar que cumple con el tiempo para el que estaba diseñado, dado por el producto entre el periodo de muestreo y el orden del sistema en lazo cerrado.

$$T_e = n * T = 3 * 0.05 = 0.15\ s$$

Además, en su respuesta encontramos una sobreoscilación de un valor ciertamente considerable teniendo en cuenta que es un escalón unitario la entrada. Finalmente, hay que recalcar que seguramente se requiera de una acción de control bastante elevada para que el sistema sea tan rápido como se podrá comprobar a continuación en la figura 66, esto puede llevar a que el sistema sature o que el sistema físico sufra alguna clase de problema.

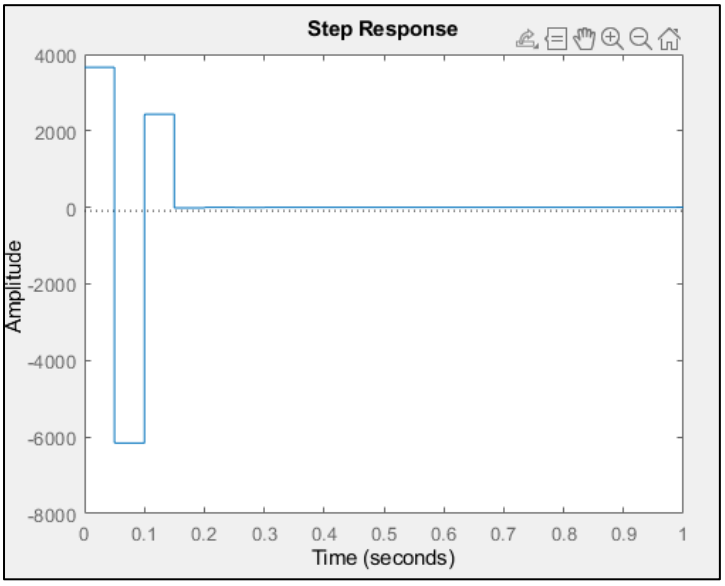


Figura 66. Acción de control Dead-Beat.

A continuación, se ha verificado su comportamiento en la maqueta de la misma forma que el resto de los reguladores. En primer lugar, prescindiendo de los limitadores de acción de control y de error y añadiendo el valor de los componentes de los polinomios del numerador y denominador en la ecuación en diferencias como se muestran en la figura 67.

```
//Dead-Beat
var a1=0.274, a2=0.8064, a3=0, a4=0;
var b0=3660, b1=-5954, b2=2295, b3=0;
```

Figura 67. Valores del Dead-Beat en la ecuación en diferencias.

Como era de esperar su respuesta es bastante rápida de hecho a penas se puede percibir el movimiento, pero se puede llegar a distinguir la sobreoscilación que se produce en el régimen transitorio. No obstante, el sistema no llega a saturar y el agente es capaz de volver a las coordenadas escogidas sin realizar ninguna otra oscilación.

Como prueba de que el sistema es excesivamente rápido a pesar de aplicar el limitador de error seguirá haciendo el mismo recorrido ya que al programa a penas le da tiempo a hacer que pare el agente.

5.2. Agentes con restricciones no holonómicas.

Las restricciones holonómicas son aquellas en las que las ecuaciones de movimiento son integrables y se pueden expresar en como una ecuación entre coordenadas.

Por otra parte, los modelos con restricciones no holonómicas son sistemas con restricciones en las velocidades no integrables, es decir no es posible expresarlas, exclusivamente, en término de las posiciones. Estas restricciones asumen que no existe deslizamiento en las ruedas del robot móvil y las mismas se encuentran relacionadas con las velocidades del vehículo. De este modo para que un robot con dichas condiciones alcance unas coordenadas necesita previamente orientarse a ese punto o si se desea finalizar con cierta orientación y posición se deberá maniobrar.

Un ejemplo claro sería el movimiento de un coche, este no se puede desplazar lateralmente porque sus ruedas apuntan en dos direcciones y solo dos de ellas pueden elegir orientación. Es por ello que para alcanzar determinada posición debe orientarse y maniobrar previamente a ponerse en movimiento hacia las coordenadas.

5.2.1 Sistema dinámico no lineal de los agentes con restricciones no holonómicas.

El modelo dinámico propuesto para el robot con restricciones será el siguiente:

$$\frac{dxpos}{dt} = xvel$$

$$\frac{dypos}{dt} = yvel$$

$$\frac{dthw}{dt} = \omega$$

Cuyas leyes de movimiento serán muy similares al del modelo no lineal, solo que ahora la velocidad se obtendrá de una forma algo distinta, pero se mantendrá el control de la posición. Previamente a la obtención de las velocidades lineales es necesario calcular la distancia existente entre el punto a alcanzar y el agente para ello se empleará la acción de control o salida del regulador que en este caso corresponderá con el resultado de la ecuación en diferencias.

$$\rho = \sqrt{ux^2 + uy^2}$$

Seguidamente será necesario conocer el ángulo en el que se encuentran las coordenadas en cada uno de los instantes de la simulación para obtener este ángulo se emplea la arcotangente de la acción de control en y entre la de las x.

$$\phi = artg(\frac{uy}{ux})$$

A continuación, se obtienen las velocidades lineal y angular con las funciones trigonométricas coseno y seno, respectivamente, de la diferencia del ángulo entre el agente y el punto y el ángulo obtenido integrando la velocidad angular.

$$v = -kv * \rho * \cos(\phi - thw)$$

$$\omega = -k\omega * \sin(\phi - thw)$$

Finalmente, los valores de las velocidades lineales para abscisas y ordenadas se calcularán de la siguiente forma:

$$xvel = v * \cos(thw)$$
$$yvel = v * \sin(thw)$$

Como se puede observar hay ciertas ecuaciones que poseen operadores no lineales del tipo trigonométrico como son los senos, cosenos y arcotangentes. Estos hacen que el sistema sea no lineal por lo que no se puede obtener la ecuación en diferencias, sería necesario calcular el punto de equilibrio y linealizar alrededor de este. Por otra parte, se decidió realizar un estudio empleando la herramienta Simulink de Matlab para poder comprobar el comportamiento del sistema ante entradas del tipo escalón.

En primera instancia se deberán definir en un script de Matlab los valores para las ganancias de la velocidad lineal y la velocidad angular. A continuación, mediante los bloques ya empleados en los anteriores diseños que brinda Simulink se irá definiendo el sistema de la figura 68:

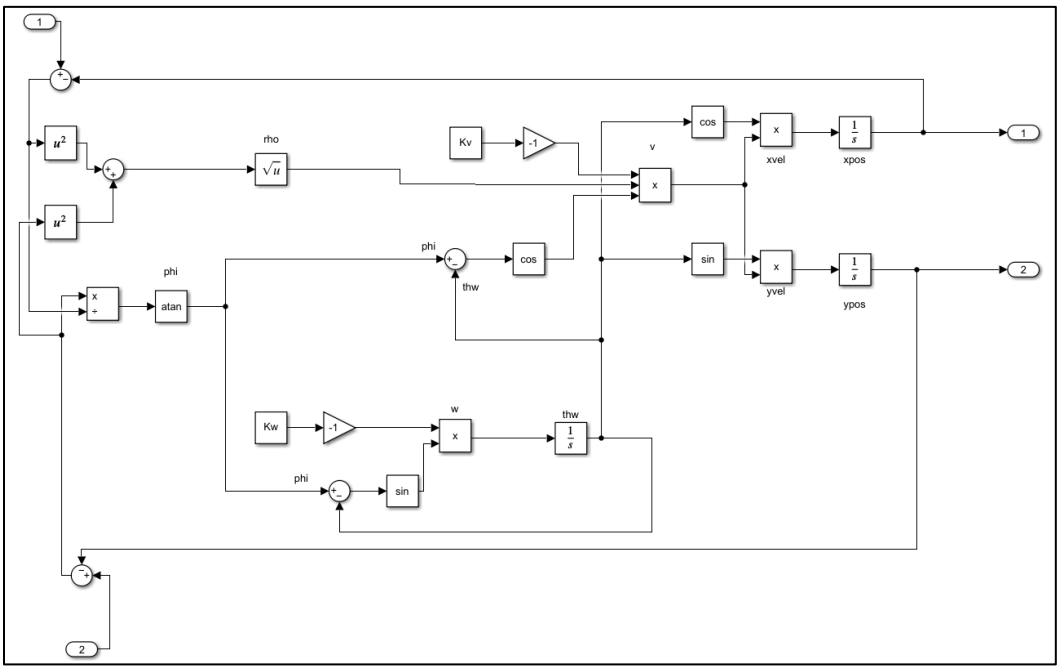


Figura 68. Simulink sistema no lineal con restricciones no holonómicas.

Debido a que para definir ciertas variables del sistema como son "rho" y "phi" se requiere del error de la realimentación, lo óptimo seria comenzar a formar el Simulink desde el final. En particular con

la realización de las ecuaciones de la posición cuyo valor se obtiene integrando la velocidad con el esquema de la figura 69.

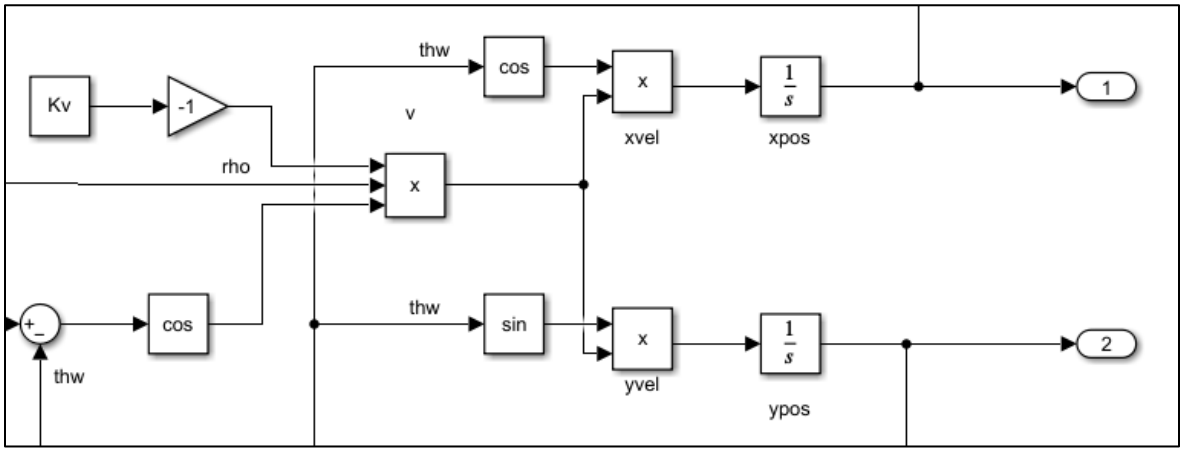


Figura 69. Simulink Posición y velocidad sistema no lineal.

Asimismo, para obtener tanto los valores de la velocidad lineal como los de las velocidades de "x" e "y" es necesario realizar los cálculos de la orientación y los cálculos de "rho" y "phi" con el esquema de la figura 70.

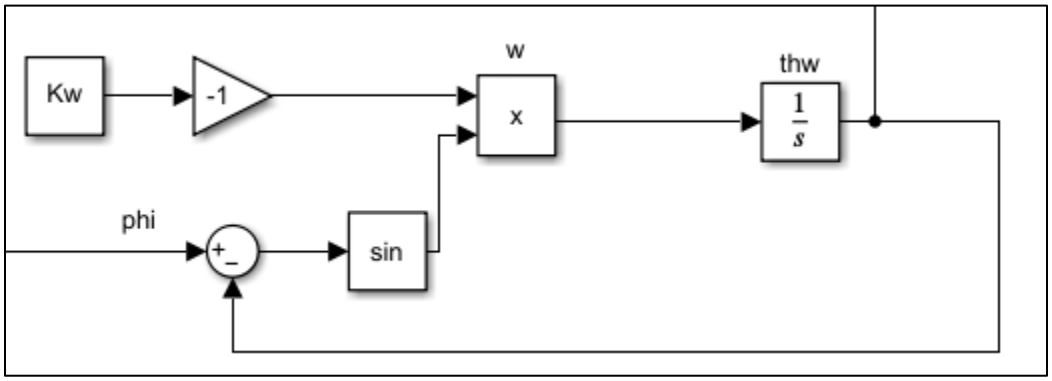


Figura 70. Simulink Orientación no lineal.

Finalmente se emplaza la realimentación que restará el valor de referencia y el de la posición actual en "x" e "y". De esta forma se obtendría el valor de entrada de la planta y se podrán realizar los cálculos de "rho" y "phi" tal y como se puede observar en la figura 71.

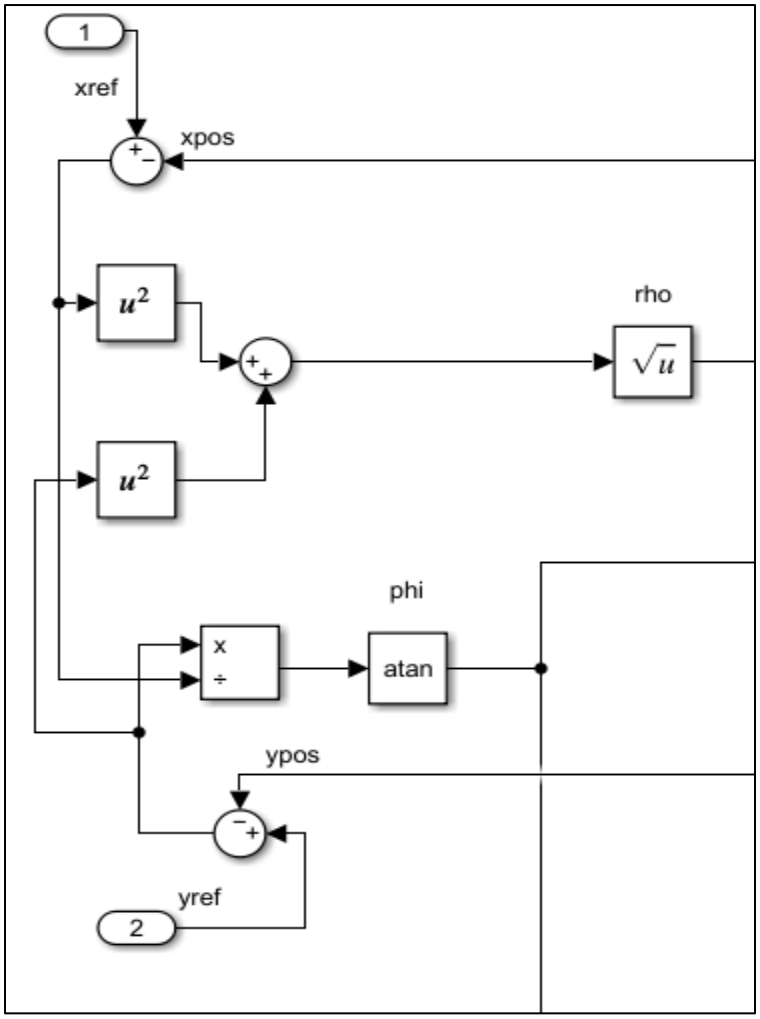


Figura 71. Simulink rho y phi no lineal.

Al igual que en el caso no lineal se emplearon bloques tipo step para dar la entrada en escalón mientras que para visualizar las señales de salida se utilizan scopes y el bloque XYGraph. Se impuso como valor a alcanzar el 3 positivo en ambas posiciones. Una vez iniciada la simulación se puede observar en el grafico XY, de la figura 72, que el sistema evoluciona hacia el lado contrario y sin un rumbo claro.

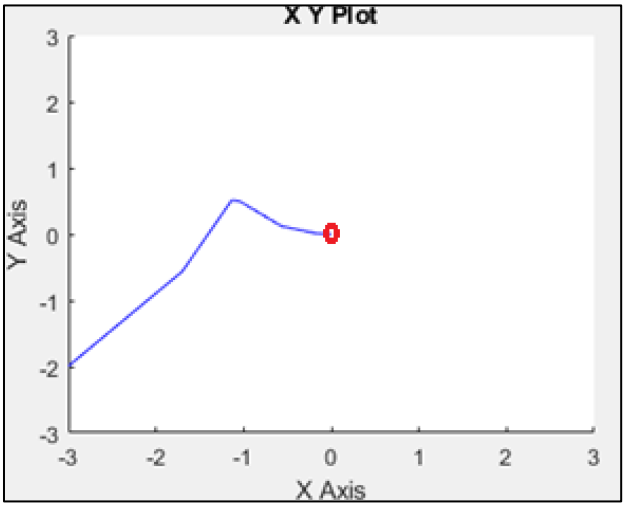


Figura 72. XYGraph sistema no lineal.

Este comportamiento se puede traducir como que el sistema es inestable y es posible que requiera de acción integral. La lectura de las gráficas de abscisas y ordenadas mostradas ambas en la figura 73 confirman que la acción de este sistema en bucle cerrado es inestable.

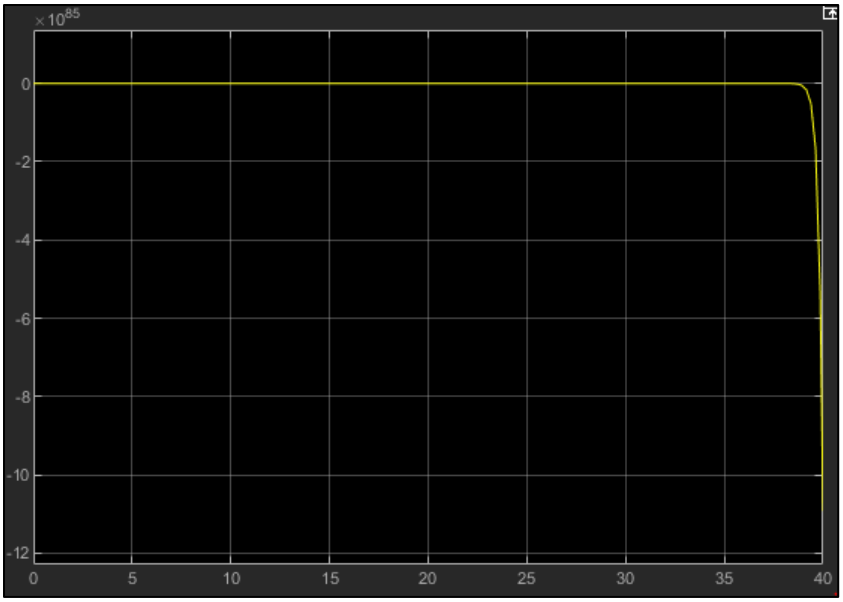


Figura 73. Gráfica X e Y no lineal.

Su implementación en la maqueta es muy similar al caso del sistema lineal sin restricciones no holonómicas. Empezando por la definición de las leyes de control en el submenú de evolución, tal y como se muestra en la figura 74, hay que destacar que la orientación definida como "thw" es distinta a la de "th", esta última se emplea para orientar cualquier agente independientemente del resto de las ecuaciones que conformen su modelo. Simplemente se ha mantenido porque su cálculo es necesario en una de las estructuras dentro del submenú de relaciones fijas, donde se recuerda que se reproducirá el código como si este se encontrara en un "script".

Ahora bien, para que se visualice correctamente el movimiento del agente con restricciones no holonómicas se debe sustituir en el menú HTMLView la variable de orientación de los agentes por la orientación con restricciones. De esta forma variará la orientación durante el desplazamiento y no solo al principio de la simulación.

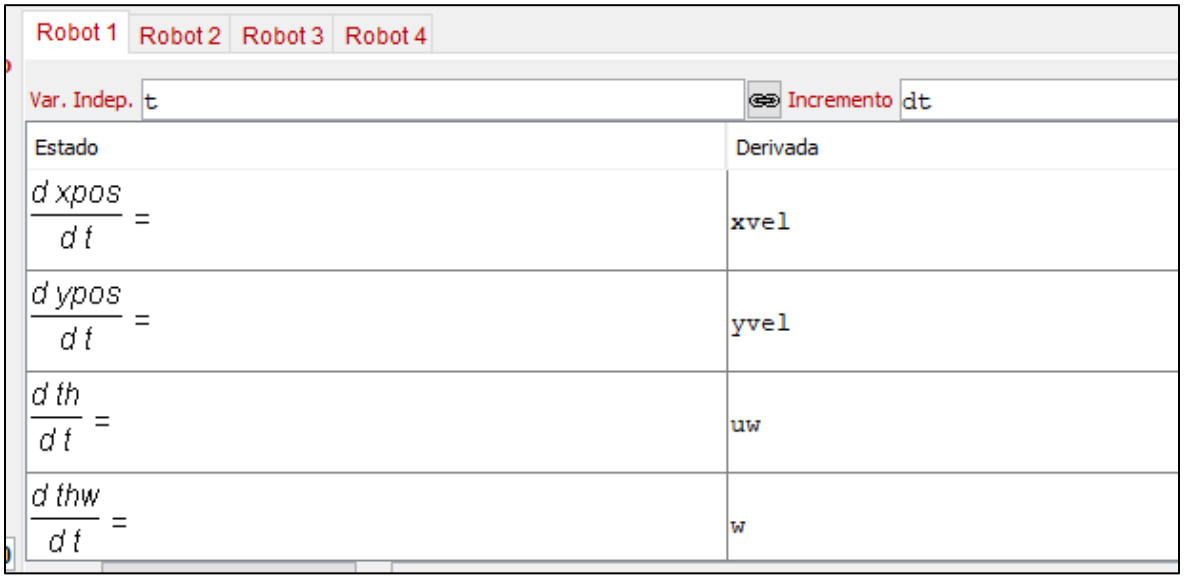


Figura 74. Evolución sistema no lineal

Al ser un sistema más complejo que lineal es necesario definir las ecuaciones del modelo en las

relaciones fijas empleando el código mostrado en la figura 75. Más concretamente en el segundo modo del switch porque se ejecutará en bucle esto será necesario porque al obtenerse la salida del regulador se podrán realizar los cálculos del resto de variables para cada uno de los intervalos de tiempo.

```
ux = controlSignalGenericxR1();
uy = controlSignalGenericyR1();

rho = Math.sqrt( Math.pow(ux,2) + Math.pow(uy,2));
phi = Math.atan(uy/ux);

v = -kv*rho* Math.cos(phi -thw);
w = -kw* Math.sin(phi -thw);

xvel = v* Math.cos(thw);
yvel = v* Math.sin(thw);
```

Figura 75. Modelo no lineal en relaciones fijas.

Para este caso se ha empleado un regulador cuyo único efecto es incluir una ganancia de valor 1 de esta forma el resultado del cálculo de la ecuación en diferencias será el error de realimentación siendo este la entrada de la planta. Para ello en la figura 76 simplemente se le da el valor de 1 tanto en a0 como b0, siendo el valor de a0 siempre unitario en cualquier regulador.

```
var a0=1, a1=0, a2=0, a3=0, a4=0;
var b0=1, b1=0, b2=0, b3=0;
```

Figura 76. Valores ec.en dif. no lineal.

Una vez ejecutada la simulación el agente mantendrá su comportamiento inestable y seguirá una trayectoria muy similar a la mostrada en el gráfico XY.

Debido a la disposición del sistema y sabiendo que este es no lineal sería complejo un estudio para obtener un controlador que fuerce una respuesta estable y es ahí donde la validación de una segunda ley de control mediante el cual se pretenderá estabilizar un sistema muy similar.

5.2.2 Control no lineal por realimentación de estado.

Se ha planteado el siguiente modelo con restricciones no holonómicas que como se puede apreciar sigue una dinámica muy similar al modelo anterior. Donde se pueden obtener dos puntos de equilibrio: uno al inicio donde el valor de las velocidades sea 0 y un segundo que será escogido, el valor de referencia, donde se desea que el sistema se estabilice y se detenga.

$$\frac{dxpos}{dt} = xvel$$

$$\frac{dypos}{dt} = yvel$$

$$\frac{dthw}{dt} = \omega$$

De igual manera que en el apartado anterior es necesario calcular la distancia que existe entre el agente y las coordenadas a alcanzar por lo que se seguirá manteniendo el control por posición. No obstante, en contraposición al sistema al sistema anterior no se empleará el error de posición

proveniente de la acción de control como tal, sino que con el nuevo control por realimentación de estado no lineal se cambiará la anterior entrada a planta por la diferencia entre posición actual y la referencia.

$$\rho = \sqrt{(xpos - xref)^2 + (ypos - yref)^2}$$

De la misma manera se definirá el ángulo en el que se encontrará el punto a alcanzar en cada instante de tiempo.

$$\phi = artg(\frac{ypos - yref}{xpos - xref})$$

Finalmente, las ecuaciones de las velocidades lineales y la velocidad angular se definirán igual que las del primer sistema no lineal con restricciones no holonómicas.

$$v = -kv * \rho * \cos(\phi - thw)$$

$$\omega = -k\omega * \sin(\phi - thw)$$

$$xvel = v * \cos(thw)$$

$$yvel = v * \sin(thw)$$

Estas mismas ecuaciones hacen que el sistema sea no lineal debido a los operadores trigonométricos contenidos por todas ellas. Además, el valor de cada una de las salidas no dependerá de una única entrada por lo que el sistema sería acoplado y MIMO (multiple-input multiple-output). Debido a ser un sistema no lineal no se podrá obtener su función de transferencia directamente asimismo al ser un sistema acoplado no podemos estudiar la entrada y la salida de las" x "o las "y"por separado, por lo que se decidió estudiar su estabilidad mediante un esquema de bloques de Simulink como el de la figura 77.

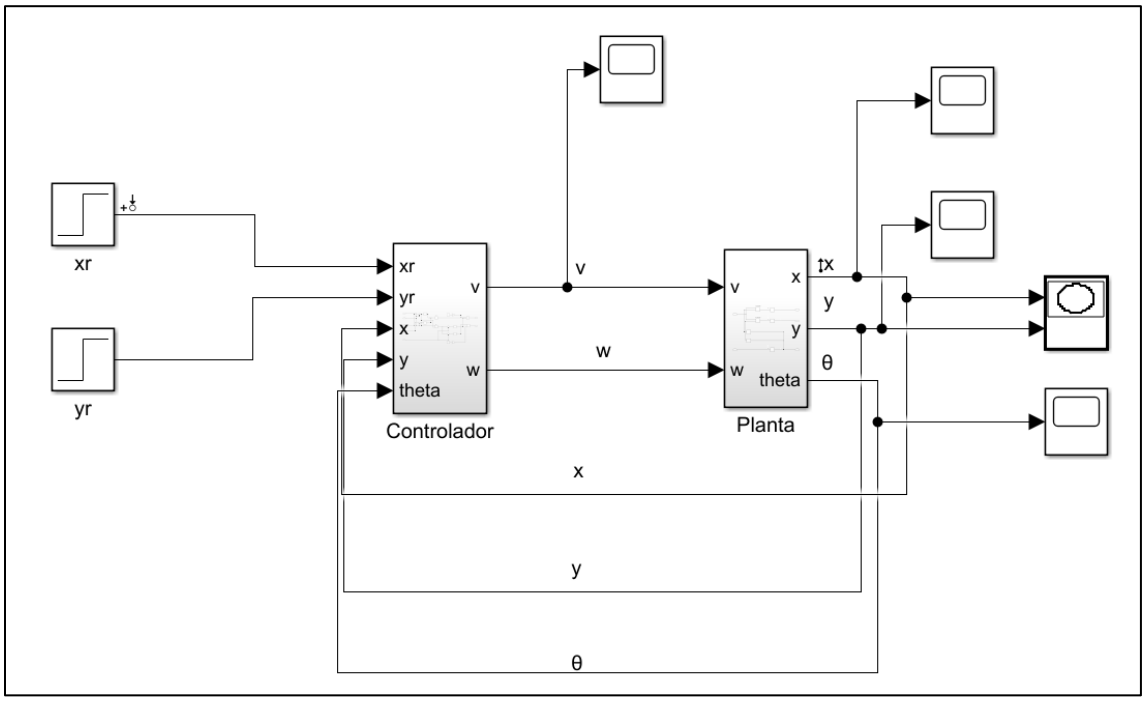


Figura 77. Simulink Segunda ley de control y realimentación de estado no lineall.

A diferencia del Simulink del apartado 5.2.1 este ha sido diseñado separando las ecuaciones entre una planta y un regulador. La planta estará conformada por los integradores y las cuentas necesarias para obtener la posición, la orientación del robot y las velocidades en abscisas y ordenadas. Mientras

que dentro del bloque “controlador” estarán definidas las ecuaciones que dan valor a las variables necesarias para la obtención de la velocidad lineal y la velocidad lineal. Esta disposición se debe al deseo de visualizar la evolución de la velocidad a lo largo del recorrido. Para la visualización de cada una de las variables se han vuelto a emplear los bloques tipo scope y el XY Graph.

Previamente a iniciar la simulación hay que definir ciertos parámetros en un script de Matlab como las ganancias de las velocidades lineales y angular, la posición y orientación iniciales y el periodo de muestreo.

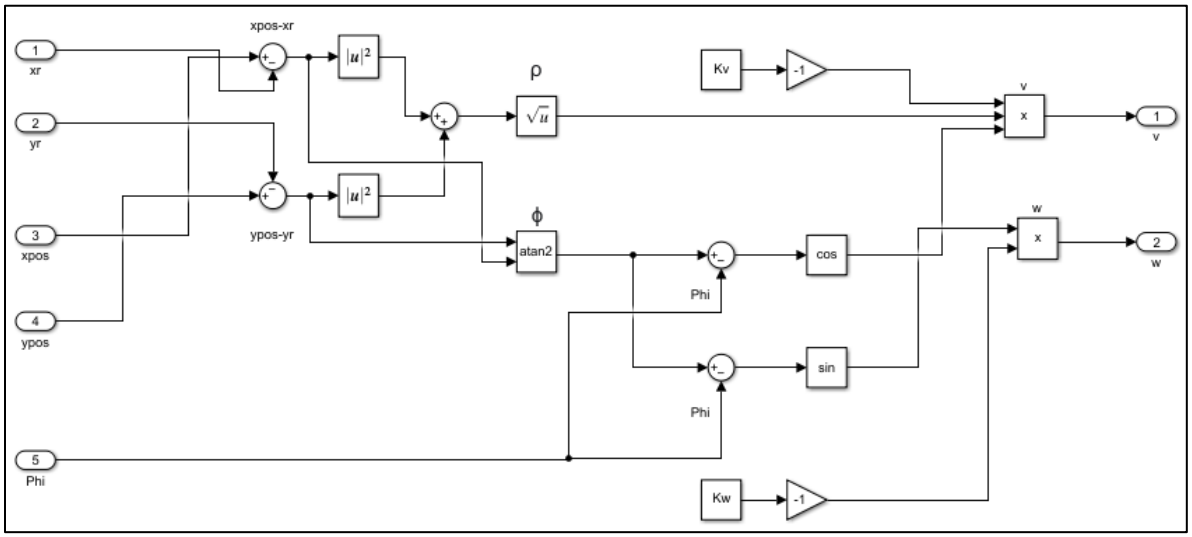


Figura 78. Simulink-Controlador Segunda ley de control.

Como se aprecia en la figura 78 se ha definido el sistema de ecuaciones de una forma muy similar al del modelo del apartado 5.2.2 solo que esta vez los valores de las variables que conforman rho y phi vienen directamente de la realimentación del sistema. Seguidamente se realizarán las operaciones que se mostraron anteriormente. Hay que tener en cuenta que al ser un sistema distinto los valores de las ganancias kv y kw son muy distintos para su obtención simplemente se fueron probando valores inferiores para que no saturase el sistema.

En última estancia la figura 79 representa el esquema de la planta. Dentro de este y como se ha comentado anteriormente se realizan los últimos cálculos de la ley de control donde se obtendría la salida deseada, en este caso la posición en x e y. Destacar que se experimentó con incluir una situación inicial donde el agente está situado en el punto (1,1) y posee una orientación de 0 radianes, es importante que estas condiciones iniciales se incluyan dentro de los integradores y no en el bloque step para que Matlab las reconozca como tal.

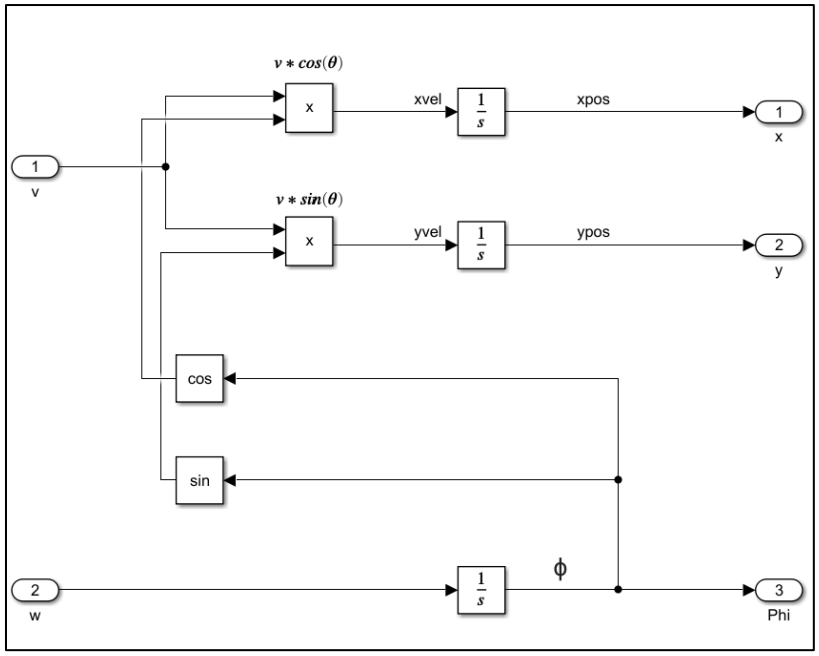


Figura 79. Simulink-Planta Segunda ley de control.

Una vez se obtuvo el esquema se propuso el punto (3,2) para ser alcanzado. Para este punto, la posición inicial anteriormente comentada y junta a un valor de 0.5 en las ganancias kv y kw el sistema respondió de la siguiente manera:

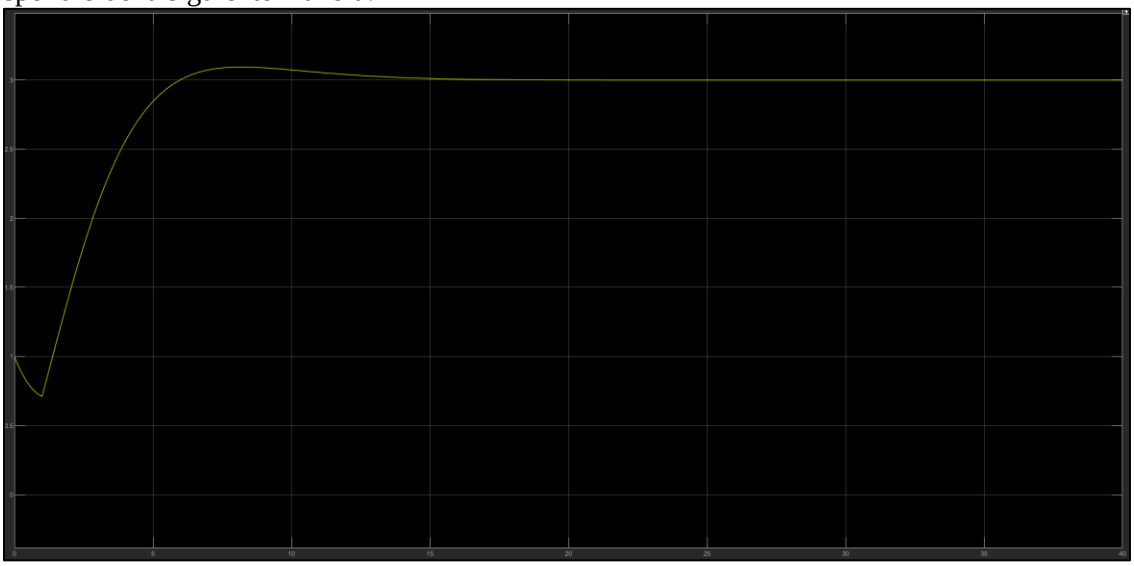


Figura 80. Respuesta temporal xpos Segunda ley de control.

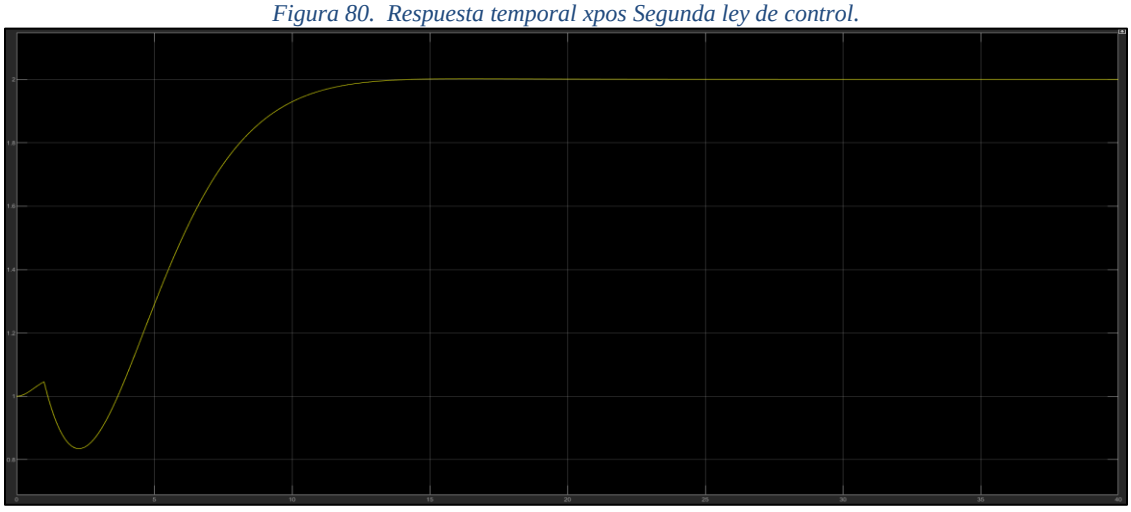


Figura 81. Respuesta temporal ypos Segunda ley de control.

Para empezar tanto en la figura 80como en la figura 81 el sistema inicialmente posee unas pequeñas oscilaciones negativas que en el caso de iniciar en el punto (0,0) no aparecen. En un principio no debería ocurrir ya que las posiciones no se encuentran en planos negativos por lo que posiblemente se deba a que el sistema tiene polos situados en el plano negativo del lugar de las raíces, esta situación suele producir oscilaciones no deseadas en el régimen transitorio.

Además, como se esperaba se llega a cumplir los valores de referencia con un tiempo de establecimiento de aproximadamente 15 segundos. Es cierto que para el caso de las x siempre se producirá una ligera sobreoscilación en el instante previo a la estabilización del sistema.

Los valores equivalentes a la acción de control vendrían representados por la velocidad lineal y la velocidad angular. Que como se pueden observar en las siguientes figuras (82 y 83) no poseen valores elevadísimos como en casos anteriores.

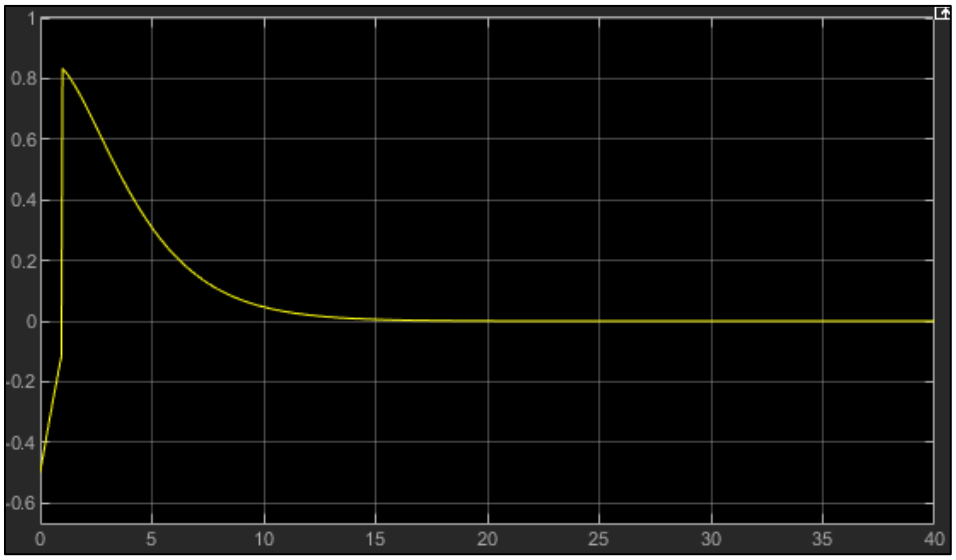


Figura 82. Respuesta temporal v Segunda ley de control.

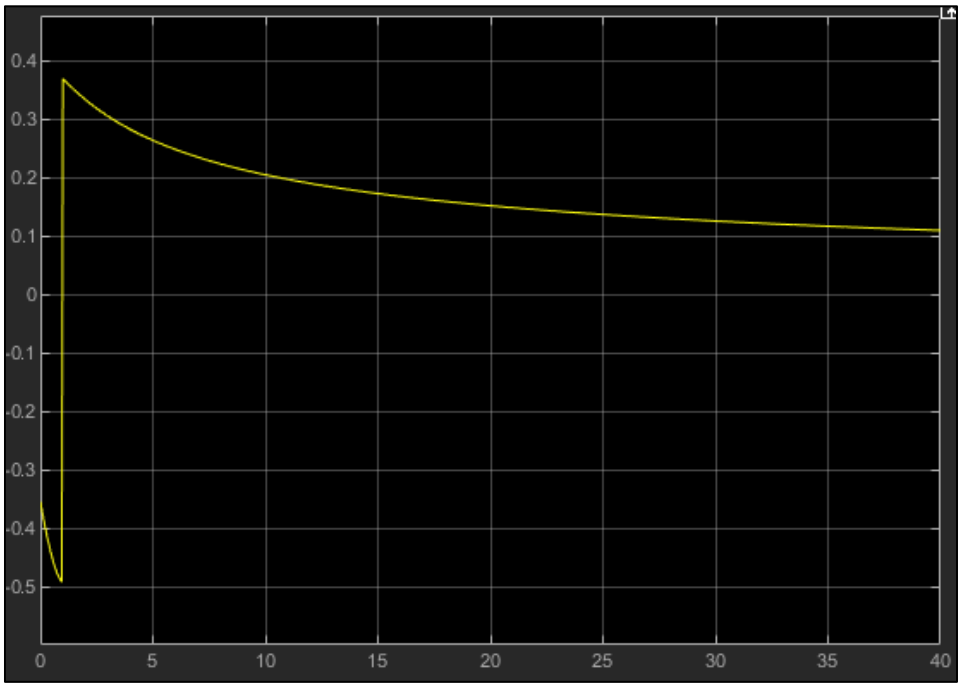


Figura 83. Respuesta temporal w Segunda ley de control.

En última instancia hay que comentar que con el bloque Graph XY, mostrado en la figura 84, se

puede apreciar a la perfección la trayectoria que debería seguir el sistema. Este inicialmente sufre la sobreoscilación inicial pero después este no se dirige realizando una trayectoria en línea recta al punto si no que debe maniobrar para poder llegar a este.

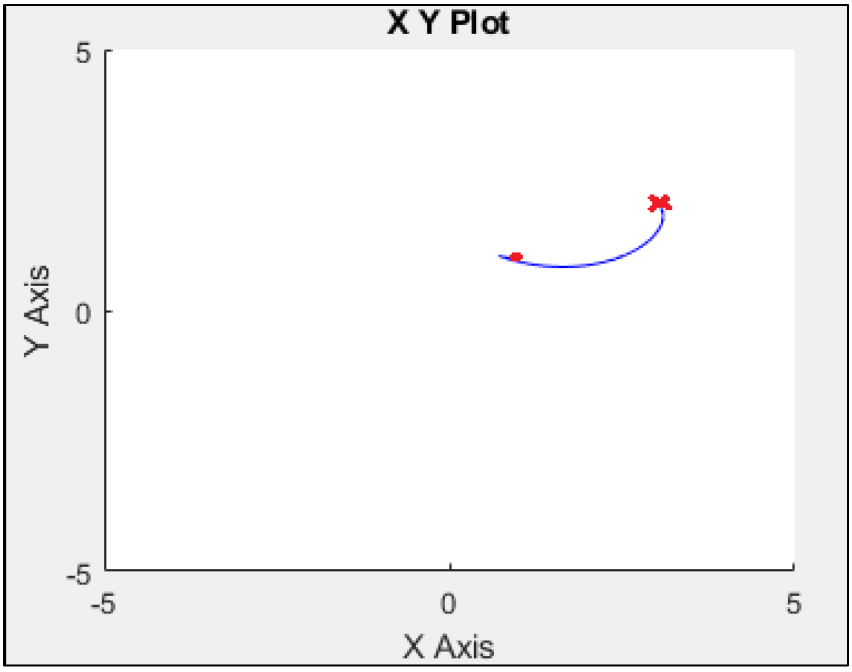


Figura 84. Respuesta temporal XY Segunda ley de control.

Respecto a la simulación y al igual que con el primer modelo no lineal se incluyen la mayoría de las ecuaciones en el case 1 del switch como se puede comprobar en la figura 85. Mientras que las leyes de control de posición y orientación se han definido idénticamente en el submenú de evolución.

```
 ux_R2 = controlSignalGenericxR2();
 uy_R2 = controlSignalGenericyR2();

 rho2 = Math.sqrt( Math.pow(xpos_R2-xref_R2,2) + Math.pow(ypos_R2-yref_R2,2));
theta2 = Math.atan2(ypos_R2-yref_R2, xpos_R2-xref_R2);

 v2 = -kv *rho2* Math.cos(theta2 -thw_R2);
 w2 = -kw *Math.sin(theta2 -thw_R2);

 xvel_R2 = v2* Math.cos(thw_R2);
 yvel_R2 = v2* Math.sin(thw_R2);
```

Figura 85. Case 1 Segunda ley de control.

Nótese que esta vez no se empleará la acción de control obtenida mediante la ecuación en diferencias por lo que no será necesario emplear la limitación de la acción de control. Por otra parte, se mantendrá la limitación para que cuando llegue al punto el agente no de vueltas sobre si mismo, para ello esta limitación ha sido ligeramente modificada. Simplemente se igualó a cero la variable propia del sistema que indica la orientación del robot y se ha reducido el error para que sea lo más fiel posible a los resultados obtenidos vía Simulink.

El sistema simulado en la maqueta sigue una trayectoria muy similar a la obtenida anteriormente solo que esta vez no presenta ninguna sobreoscilación porque parte del punto (0,0). Lo más importante de esta simulación es que se aprecia a la perfección como el agente va variando su orientación respecto a las coordenadas a alcanzar y como realiza la maniobra necesaria para alcanzar estas.

Como curiosidad, dentro de las posibilidades que existen para estabilizar esta clase de sistemas existe

el Criterio de estabilidad de Lyapunov donde se pueden distinguir dos métodos destinados a estabilizar el sistema como se prueba en [11]:

- Primer método: busca la estabilidad local de un sistema no lineal a partir de su aproximación lineal. Partiendo de un sistema:

$$\dot{x} = f\big(x(t)\big)$$

  Donde se definirá el punto de equilibrio como:

$$x(0) = x_0$$

  Se diría que este punto de equilibrio cumpliría con la estabilidad de Lyapunov si para cada $\epsilon > 0$ existe $\delta > 0$ de forma que todas las ecuaciones cumplen || x(0) - x_0 ||$\leq \delta$ y a su vez se verifica que || x(t) - x_0 ||$\leq \epsilon$. Esto significa que el sistema será estable si se cumple la distancia δ entre el punto inicial y el de equilibrio y si el sistema se mantiene lo suficientemente cerca al punto de equilibrio, siendo la distancia ϵ, respecto a su posición actual.

  A su vez se puede tener un punto de equilibrio asintomáticamente estable si todas sus soluciones que son iniciadas cerca de dicho punto tienden a este cuando el valor del tiempo es infinito. Esto se traduce como que el sistema inicia muy cerca del punto de equilibrio pero que convergerá en algún momento del tiempo.

- Segundo método: este método está pensado para sistemas físicos considerando que estos requieren de energía para su funcionamiento. Si el valor de la energía decrece el sistema tenderá a estabilizarse sobre un punto final donde el valor de dicha energía sea nulo. Como ejemplo se puede presentar el caso de una masa vibrante unida a un muelle, esta vibrará alrededor de un punto hasta que se estabilice el sistema alcanzado un valor máximo de longitud teniendo en cuenta la resistencia que ofrece la masa siendo este el punto donde la energía es 0. Como curiosidad la mayoría de estos sistemas serán asintomáticamente estables.

6. Conclusiones.

Durante la realización de este trabajo de fin de grado se ha diseñado y programado una maqueta, que puede ser ejecutada en casi cualquier buscador, con el programa Easy Java Simulations en la que se encontrarán cuatro agentes capaces de alcanzar distintas coordenadas dadas por el usuario además de la ganancia para los casos donde el sistema es no lineal. A su vez se emplearon tres dinámicas distintas para el movimiento de los agentes llegándose a desarrollar cuatro reguladores discretos distintos para el modelo lineal.

En primer lugar, se estudió el comportamiento de un sistema lineal cuya dinámica permitía que fuera estudiado como un sistema de una única salida y entrada. En bucle cerrado la planta ya presentaba una respuesta estable pero lenta y con oscilaciones en el régimen transitorio. Por lo que se propuso el diseño de dos reguladores del tipo PD que permitían mejorar la señal en el transitorio obteniendo una respuesta más rápida y estable, pero con unos valores de acción de control elevados. Seguidamente se diseñó un regulador algebraico mediante la asignación de polos con el que se obtuvieron resultados ligeramente mejores que con los reguladores anteriores, pero con una acción de control mucho mayor. También se modeló un regulador con el método de síntesis directa cuya respuesta al ser simulada presentaba el efecto Windup que la hacía inestable. Finalmente, para este mismo modelo se probó a implementar un Dead-beat cuyo objetivo es llevar a la respuesta a la estabilidad lo más rápido posible, y así fue a pesar de que el sistema presenta una altísima acción de control y sufre una gran sobreoscilación.

En segundo lugar, se inició el estudio de un sistema no lineal con restricciones no holonómicas cuya respuesta era inestable y de ahí que se decidiera implementar y validar una segunda ley de control mediante la realimentación de estados no lineal, lo que permitiría estabilizar el sistema. El diseño del regulador en base a este criterio fue un completo éxito obteniendo una trayectoria en la simulación en la que se podía observar a la perfección como debe maniobrar un robot con restricciones no holonómicas y como este iba variando su orientación en cada instante de tiempo.

Finalmente hay que destacar que este trabajo ha resultado muy interesante con respecto a la ampliación de materia impartida durante este grado, además se han fortalecido y afianzado conocimientos sobre control informática e incluso robótica. De igual manera ha sido altamente beneficioso aprender a utilizar la herramienta Easy Java Simulations abriendo un mundo de posibilidades para realizar futuros experimentos y permitiendo crear una aplicación que puede ser empleada para impartir docencia.

7. Referencias bibliograficas.

[1] *Oh, Kwang-Kyo, Myoung-Chul Park, and Hyo-Sung Ahn. "A survey of multi-agent formation control." Automatica 53 (2015): 424-440.*
Obtenido en: https://www.sciencedirect.com/science/article/abs/pii/S0005109814004038

[2] *Información sobre EJS: https://www.um.es/fem/EjsWiki/Main/WhatIsEJS?*

[3] Jensen, Simon Holm, Anders Møller, and Peter Thiemann. "Type analysis for JavaScript." *International Static Analysis Symposium*. Springer, Berlin, Heidelberg, 2009.

[4] Crockford, Douglas. *JavaScript: The Good Parts: The Good Parts*. " O'Reilly Media, Inc.", 2008.

[5] *Applets, Java. "A. Java Applets." (1998).*

[6] *Sorribes, A.G." Laboratorios virtuales web como herramienta de apoyo para prácticas de ingeniería no presenciales". Editorial Universitat Politècnica de València. ISBN: 978-84-9048-833-1.*

[7] *Llobregat, J. J., Lacruz, Á. C., Calvo, V. C., & Salvador, A. C. (2016). Control automático: Tiempo continuo y tiempo discreto. REVERTÉ.*

[8] *Roig, J. V. (2020). Diapositivas de la asignatura Ingeniería de Control.*

[9] *Sorribes, A. G. (2020). Diapositivas de la asignatura Control Avanzado por Computador.*

[10] *Castaño Giraldo, S. A.Anti Windup en un control PID. Obtenido en: https://controlautomaticoeducacion.com/control-realimentado/anti-windup-en-un-control-pid/*

[11] *Carlos Mario Vélez S. Universidad EAFIT. Diapositivas estabilidad de Lyapunov.* Obtenido en:
http://prototipando.es/phocadownload/pfc/hermes/backup/PROGRAMACION_C/Curso%20Sistemas%20No%20Lineales/lyapunov.pdf

PLIEGO DE CONDICIONES

JAIME SANCHA MIGUEL

Durante la ejecución de este trabajo se ha diseñado una maqueta mediante Easy Java/Javascript Simulations mediante la cual se han validado distintos reguladores para un modelo dinámico lineal. Además de la validación de un modelo con restricciones no holonómicas estabilizado mediante una segunda ley de control obtenida mediante una realimentación de estados no lineal. Tanto los archivos generados para la creación de la maqueta como los códigos del desarrollo de los reguladores discretos y sus esquemas de Simulink serán accesibles a cualquiera que lo desee para uso particular y posterior ampliación del estudio sobre el tema para futuros trabajos o incluso para impartir docencia.

El contenido generado durante la realización del trabajo se corresponde con:

- Un archivo EJSS donde se podrán encontrar todas las variables y constantes junto a sus leyes de control y las distintas funciones y estructuras internas que permiten integrar todos estos elementos en la simulación. Así como un escenario grafico correspondiente con la parte visual de la maqueta.

- Códigos de Matlab donde se definen las constantes y variables de los modelos para más tarde obtener sus funciones de transferencia, lugares de las raíces, respuestas temporales y acciones de control.

- Esquemas de Simulink diseñados principalmente para simular respuestas de los diferentes modelos ante referencias del tipo escalón. Y realizar comprobaciones sobre sus respuestas temporales y sus acciones de control.

Por otro parte es necesario comentar las condiciones de los softwares empleados en la generación de dichos materiales:

- Matlab: La licencia ha sido cedida por la UPV con un periodo de caducidad de un año y puede ser usada hasta en cuatro dispositivos. Hay que destacar que es muy recomendable que el equipo donde se vaya a emplear este programa posea unas buenas especificaciones para evitar colapsos y tiempos de carga excesivos.

- Easy Java/Javascript Simulations: Es un programa cuya licencia es completamente gratuita cuyos principales fines son la enseñanza y el aprendizaje. El principal requisito es poseer un buscador de internet siendo el más recomendado el Google Chrome.

PRESUPUESTO

JAIME SANCHA MIGUEL

ÍNDICE DE LA PRESUPUESTOS:

1. Coste de los materiales.

Dado que este trabajo no requiere de ninguna clase de soporte físico la principal herramienta física empleada para la realización de este trabajo de fin de grado fue un portátil *MSI Leopard GP73 8RD*, con un procesador Intel Core 8750-H, 16GB de memoria RAM ampliable a 32GB, con almacenamiento de 1TB de HDD y 250GB de SSD y con una gráfica Nvidia 1050 Ti. Valorado en unos 1049,99 €.

Materiales	**Precio (€)**
MSI Leopard GP73 8RD	1049.99

2. Coste del software.

Como bien se comentó en el pliego de condiciones para el desarrollo del trabajo se emplearon dos softwares distintos, siendo uno de ellos completamente. Hay que destacar que estos son los que se emplearon directamente en la realización, pero también es necesario poseer un sistema operativo y un buscador a pesar de que este sea completamente gratuito.

Software	**Coste (€)**	**Coste Proporcional (€)**
Matlab & Simulink	1200 (Anual)	600
Microsoft Windows 10 Pro	199 (Perpetuo)	49.75
Microsoft Office365	69 (Anual)	17.25
Google Chrome	0 (Perpetuo)	0
Easy Java/Javascript Simulations	0 (Perpetuo)	0
Total		667

3. Coste de la mano de obra.

Para la ejecución del trabajo se han requerido de un total de 133h de trabajo por parte del autor y 16h por parte del tutor. El computo de horas del autor se puede dividir en cinco partes:

- El diseño y la programación de la maqueta.
- Desarrollo y estudio de los modelos dinámicos.
- Creación de los distintos esquemas.
- Búsqueda de información.
- Redacción.

Se ha calculado que aproximadamente un coste aproximado de 18 €/h para el autor y 40 €/h para el tutor por sus labores de revisión y asesoramiento.

Personal	Tareas	Coste temporal (h)	Coste (€)
Autor	Diseño y programación	20	360
	Desarrollo y estudio de los modelos dinámicos	30	540
	Creación de los distintos esquemas	3	54
	Búsqueda de información	5	90
	Redacción	75	1350
Tutor	Revisión y asesoramiento	16	640
Total		133	3034

4. Total.

Finalmente se agruparán todos los costes necesarios en la siguiente tabla:

Tipo de coste	Precio (€)
Materiales	1049.99
Software	667
Mano de obra	3034
Total	4750.99

Siendo el coste del presente trabajo de fin de grado de cuatro mil setecientos cincuenta euros con noventa y nueve céntimos (4750.99€).

ANEXOS

JAIME SANCHA MIGUEL

ÍNDICE DE LOS ANEXOS:

1. Códigos de Matlab.

1.1 Obtención de la fdt del modelo lineal.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%Modelo Lineal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

T=0.05;
M=3.5;
C=0.5
F=1
f=1

T=0.05;
z =tf('z',T)
s = tf('s');
num = [0,0,-(1/M)];
den = [-1,(1/C)*0.5,0];
Gs = tf(num,den)

Gz = c2d(Gs,T,'zoh');
Gz = minreal(Gz,0.001)
zpk(Gz)
sisotool(Gz)
```

1.2 Obtención de los reguladores PD.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%PID-PD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
reg1 = (92.249*(z-0.9512))/(z-0.003597);
reg2 = (366.6*(z-0.9512))/(z-0.003597);

% Proceso
step(feedback(reg1*Gz,1))

% Acción de control
step(feedback(reg1,Gz))

% Proceso
step(feedback(reg2*Gz,1))

% Acción de control
step(feedback(reg2,Gz))
```

1.3 Obtención del controlador por asignación de polos.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%Asignación de Polos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

K = 0.00036317 ;
p1 = 1;
p2 = 1.051;
c1 = -1.017;

pd = 0.85;
syms g1 q1 b z ;

T_eq =(z-p1)*(z-p2)*(z-g1) + K*b*(z-c1)*(z-q1) - z*(z-pd)^2;
[v t] = coeffs(T_eq,z);

% Resolvemos el sistema
sol = solve(v);

b = double(sol.b(1));
g1 = double(sol.g1(1));
q1 = double(sol.q1(1));

clear z;
z = tf('z',T);
Gr = (b*(z-q1))/((z-g1))

% Proceso
step(feedback(Gr*Gz,1))

% Acción de control
step(feedback(Gr,Gz))
```

1.4 Obtención del controlador por síntesis directa.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%Síntesis Directa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fun = @root2d
x0 = [0,0,0,0];
x = fsolve(fun,x0)

DBC = (x(4)*(z+1.017)*(z-x(1)))/(z*(z-0.9)^2);
ADBC = ((z-1.051)*(z-x(2))*(z-x(3)))/(z*(z-0.9)^2);

Gr= (1/Gz)*(DBC/ADBC);
Gr = minreal(Gr,0.001)

% Proceso
step(feedback(Gr*Gz,1))
% Acción de control
step(feedback(Gr,Gz))
```

```
function F = root2d(x)

F(1) = -x(2) - x(3) - 1.0551 + 1 + x(4);
F(2) = x(2)*x(3) + 1.051*(x(2) + x(3)) - 0.81 + x(4)*(1.017 - x(1));
F(3) = -1.051*x(2)*x(3) - 1.017*x(4)*x(1);
F(4) = x(4)*(1 + 1.017)*(1 - x(1)) - 0.81;
```

1.5 Obtención del controlador por Dead-Beat.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%Dead Beat
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


fun = @root3d
x0 = [0,0,0,0];
x = fsolve(fun,x0)

DBC = (x(4)*(z+1.017)*(z-x(1)))/(z^3);
ADBC = ((z-1.051)*(z-x(2))*(z-x(3)))/(z^3);

Gr= (1/Gz)*(DBC/ADBC);
Gr = minreal(Gr,0.001)

% Proceso
step(feedback(Gr*Gz,1))

% Acción de control
step(feedback(Gr,Gz))
```

```
function F = root3d(x)

F(1) = -x(2) - x(3) - 1.0551  + x(4);
F(2) = x(2)*x(3) + 1.051*(x(2) + x(3)) + x(4)*(1.017 - x(1));
F(3) = -1.051*x(2)*x(3) + x(4)*(-1.017 * x(1));
F(4) = x(4)*(1 + 1.017)*(1 - x(1)) - 1;
```

1.6 Definición de las constantes para el modelo no lineal y la segunda ley de control.

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %% Modelo no Lienal// Segunda ley de control
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


T=0.05;
z=tf('z',T);
v=0;
xvel=0;
thw=0;
Kv=0.5;
```

```
Kw=0.5;
w=0;
xvel=0;
yvel=0;
Kp=0;
Kvel=0;
x0=1;
y0=1;
theta0=0;
```

2. Códigos de EJS.

2.1 Relaciones fijas.

```
if(xref_R1>=5) xref_R1=5;
if(xref_R1<=-5) xref_R1=-5;
if(yref_R1>=5) yref_R1=5;
if(yref_R1<=-5) yref_R1=-5;


switch (modo_R1){
 //Etapa Orientación
 case 0:
   //Calculos orientación
   th_ref_R1 = Math.atan2(yref_R1-ypos, xref_R1-xpos);
   error_orientacion_R1 = th_ref_R1 - th;
   if(error_orientacion_R1 > Math.PI){  //Giro a derechas negativo
        error_orientacion_R1 = -(2*Math.PI-error_orientacion_R1);
   }
   if(error_orientacion_R1 < -Math.PI){  //Giro a izquierdas positivo
        error_orientacion_R1 = (2*Math.PI+error_orientacion_R1);
   }
   error_or_abs_R1 = Math.abs(error_orientacion_R1);


   //Calculo acción de control
   uw = controlOrientacion(error_orientacion_R1);

   //Condición paso siguiente etapa
   if(error_or_abs_R1 < 0.05 && error_or_abs_R1 != 0){
     uw = 0;
     xant_R1 = xref_R1;
     yant_R1 = yref_R1;
     modo_R1 = 1;
   }
   break;

 //Etapa Desplazamiento
 case 1:
    ux = controlSignalGenericxR1();
    uy = controlSignalGenericyR1();

    rho = Math.sqrt( Math.pow(ux,2) + Math.pow(uy,2));
    phi = Math.atan(uy/ux);
```

```
    v = -kv*rho* Math.cos(phi -thw);
    w = -kw* Math.sin(phi -thw);

    xvel = v* Math.cos(thw);
    yvel = v* Math.sin(thw);

    //Limitación acción de control
    if(ux>=12) ux=12;
    if(ux<=-12) ux=-12;
    if(uy>=12) uy=12;
    if(uy<=-12) uy=-12;

   //Limitación del error
    if(Math.abs(xref_R1 - xpos) < 0.1  && Math.abs(yref_R1 - ypos) < 0.1 ){
        ux = 0;
        uy = 0;
        xvel = 0;
        yvel = 0;
        modo_R1 = 2;
    }

    //Cambio de referencia dinamico
    if(xref_R1!=xant_R1 || yref_R1!=yant_R1){
       ux = 0;
       uy = 0;
       xvel = 0;
       yvel = 0;
       ekx1_R1=0;
       ukx1_R1=0;
       eky1_R1=0;
       uky1_R1=0;
       modo_R1 = 0;
    }

    break;

  // Estapa de Espera a cambio de referencia
  case 2:
    if(xref_R1!=xant_R1 || yref_R1!=yant_R1){
       modo_R1 = 0;
    }
    break;
}

// State
paused = _model.isPaused();
```

2.2 Funciones para obtener la acción de control en x e y.

```
function controlSignalGenericxR1(){
  var uux;
  if(Tsample>0){
    if(t-n1_zoh_R1*Tsample>=Tsample){
      uux = controlDiscretoxR1();
      n1_zoh_R1=n1_zoh_R1+1;
```

```
    }
  }
  return uux;
}

function controlSignalGenericyR1(){
  var uuy;
  if(Tsample>0){
    if(t-n2_zoh_R1*Tsample>=Tsample){
      uuy = controlDiscretoyR1();
      n2_zoh_R1=n2_zoh_R1+1;
    }
  }
  return uuy;

}

function controlDiscretoxR1(){
  //d-b
  /*var a1=0.5042, a2=0, a3=0, a4=0;
  var b0=1365, b1=-1435, b2=0, b3=0;*/

    //PD1
  /*   var a1=-0.003597, a2=0, a3=0, a4=0;
     var b0=92.25, b1=-87.75, b2=0, b3=0; */

    //PD2
   /*  var a1=0.003597, a2=0, a3=0, a4=0;
     var b0=366.6, b1=-348.7, b2=0, b3=0;*/

 /* var a1=0, a2=0, a3=0, a4=0;
  var b0=1, b1=0, b2=0, b3=0;*/

   //Asignación de polos
  /*   var a1=0.1671, a2=0, a3=0, a4=0;
     var b0=506.3, b1=-475.6, b2=0, b3=0; */


   //Sintesis directa
  /*  var a1=-0.7195, a2=0.3609, a3=0, a4=0;
     var b0=2133, b1=-3160, b2=1027, b3=0;*/

     //Dead-Beat
    var a1=0.274, a2=0.8064, a3=0, a4=0;
     var b0=3660, b1=-5954, b2=2295, b3=0;

  //Error de realimentacion
  var ekx = xref_R1 - xpos;

  //Calculo accion de control
  var ukx = b0*ekx + b1*ekx1_R1 + b2*ekx2_R1 + b3*ekx3_R1 + a1*ukx1_R1 + a2*ukx2_R1
+ a3*ukx3_R1 + a4*ukx4_R1;//0.9231*ukx1 + 2.116*ekx -2.101*ekx1;

  //Actualización
  ekx3_R1=ekx2_R1;
  ekx2_R1=ekx1_R1;
  ekx1_R1=ekx;
```

```
  ukx4_R1=ukx3_R1;
  ukx3_R1=ukx2_R1;
  ukx2_R1=ukx1_R1;
  ukx1_R1=ukx;

  return ukx;
}

function controlDiscretoyR1(){
  //d-b
  /*var a1=0.5042, a2=0, a3=0, a4=0;
  var b0=1365, b1=-1435, b2=0, b3=0;*/

    //PD1
  /*   var a1=-0.003597, a2=0, a3=0, a4=0;
     var b0=92.25, b1=-87.75, b2=0, b3=0; */

    //PD2
   /*  var a1=0.003597, a2=0, a3=0, a4=0;
     var b0=366.6, b1=-348.7, b2=0, b3=0;*/

 /* var a1=0, a2=0, a3=0, a4=0;
  var b0=1, b1=0, b2=0, b3=0;*/

   //Asignación de polos
  /*   var a1=0.1671, a2=0, a3=0, a4=0;
     var b0=506.3, b1=-475.6, b2=0, b3=0; */

   //Sintesis directa
  /*  var a1=-0.7195, a2=0.3609, a3=0, a4=0;
     var b0=2133, b1=-3160, b2=1027, b3=0;*/

     //Dead-Beat
    var a1=0.274, a2=0.8064, a3=0, a4=0;
     var b0=3660, b1=-5954, b2=2295, b3=0;
  //Error de realimentacion
  var eky = yref_R1 - ypos;

  //Calculo accion de control
  var uky =  b0*eky + b1*eky1_R1 + b2*eky2_R1 + b3*eky3_R1 + a1*uky1_R1 + a2*uky2_R1
+ a3*uky3_R1 + a4*uky4_R1; //0.9231*uky1 + 2.116*eky -2.101*eky1;

  //Actualización
  eky3_R1=eky2_R1;
  eky2_R1=eky1_R1;
  eky1_R1=eky;

  uky4_R1=uky3_R1;
  uky3_R1=uky2_R1;
  uky2_R1=uky1_R1;
  uky1_R1=uky;

  return uky;
}
```

2.3 Función para ajustar el control de orientación.

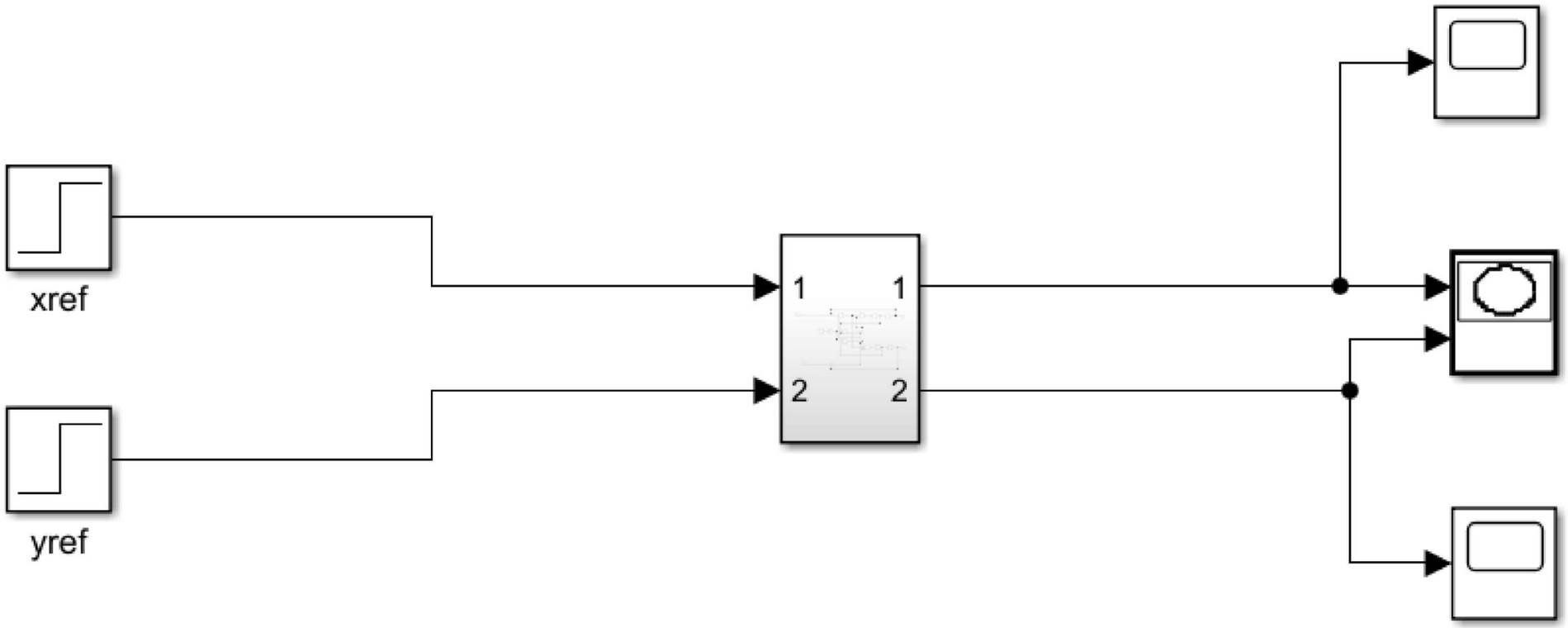
```
function controlOrientacion(error_orientacion){
  var Ko=1.5;
  var uuw;
  uuw = Ko*error_orientacion;

return uuw;
}
```

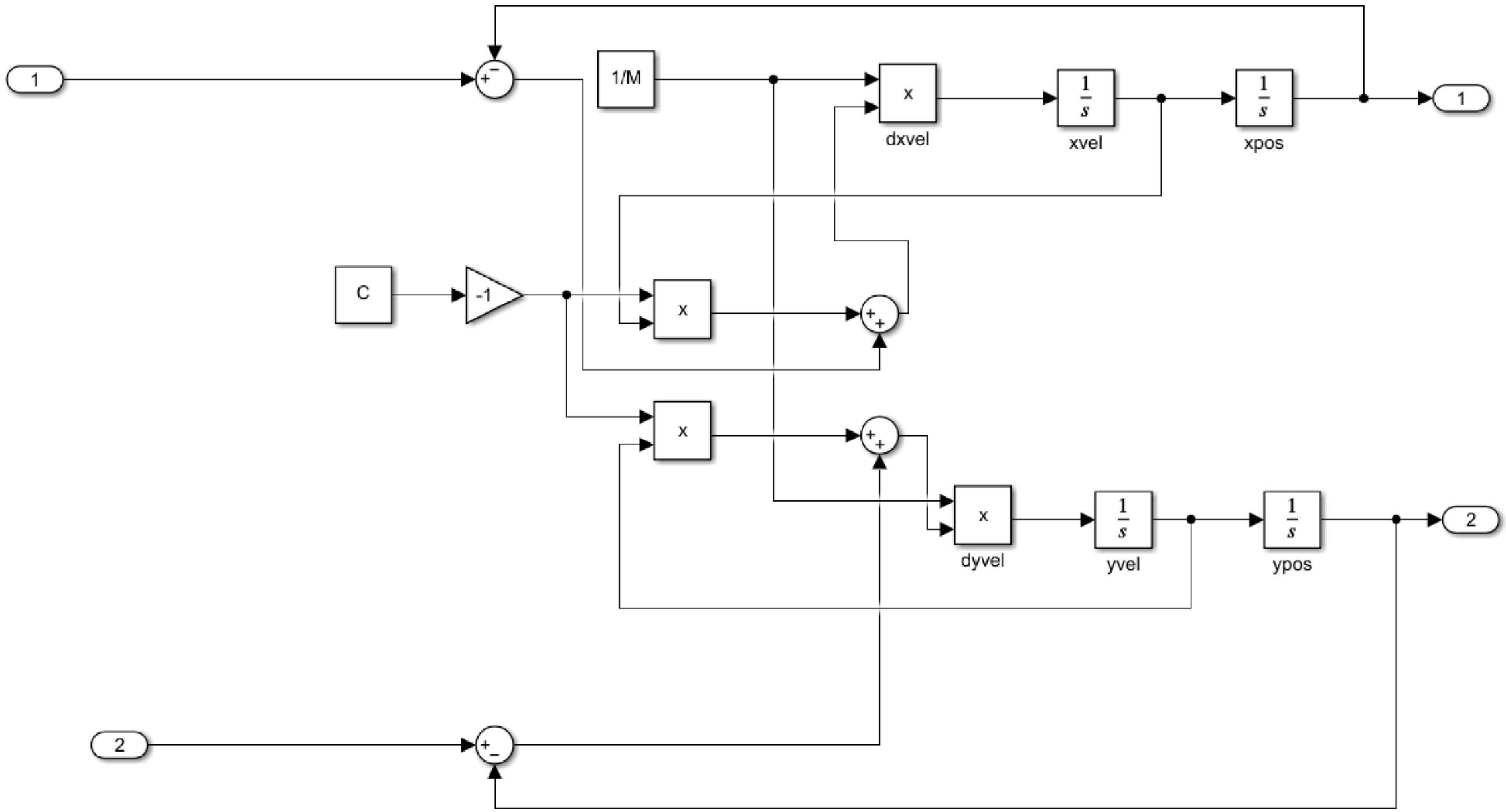
PLANOS

JAIME SANCHA MIGUEL

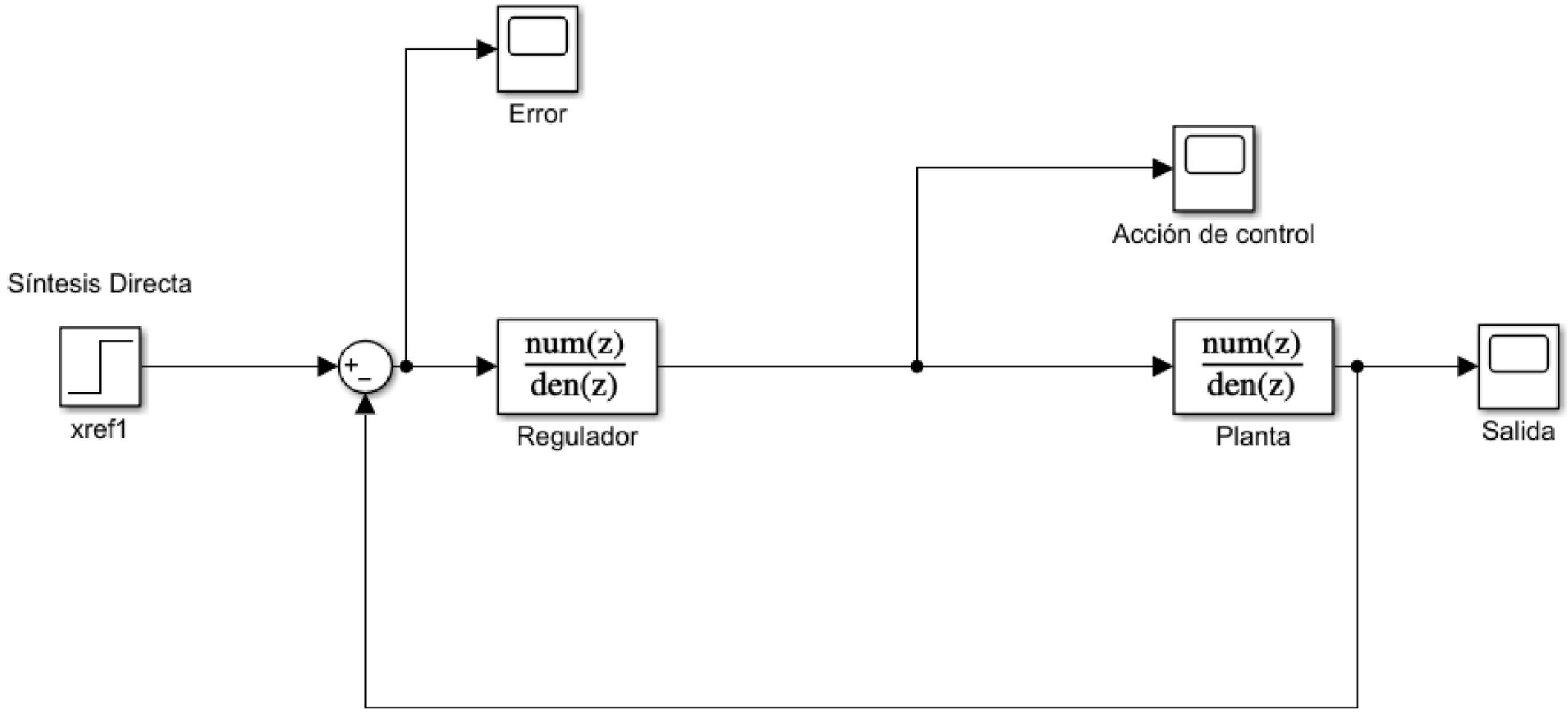
ÍNDICE DE PLANOS:



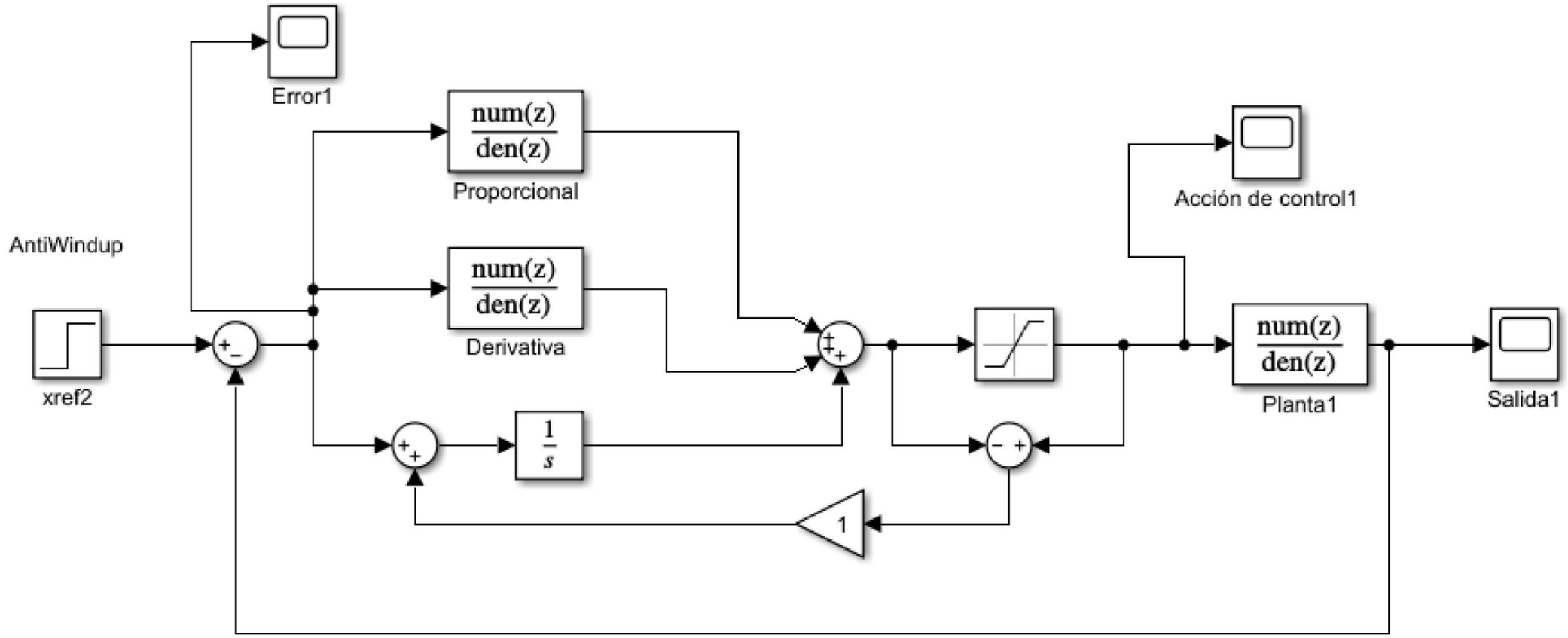
PROYECTO: DISEÑO Y VALIDACIÓN DE ESTRATEGIAS DE CONTROL DE FORMACIONES EN SISTEMAS MULTIRROBOT EN UNA PLATAFORMA VIRTUAL BASADA EN EASY JAVA SIMULATIONS.		Fecha: 25/06/2021 Escala S/E
Autor:Jaime Sancha Miguel	Plano: Diagrama completo modelo lineal	Plano Nº 01



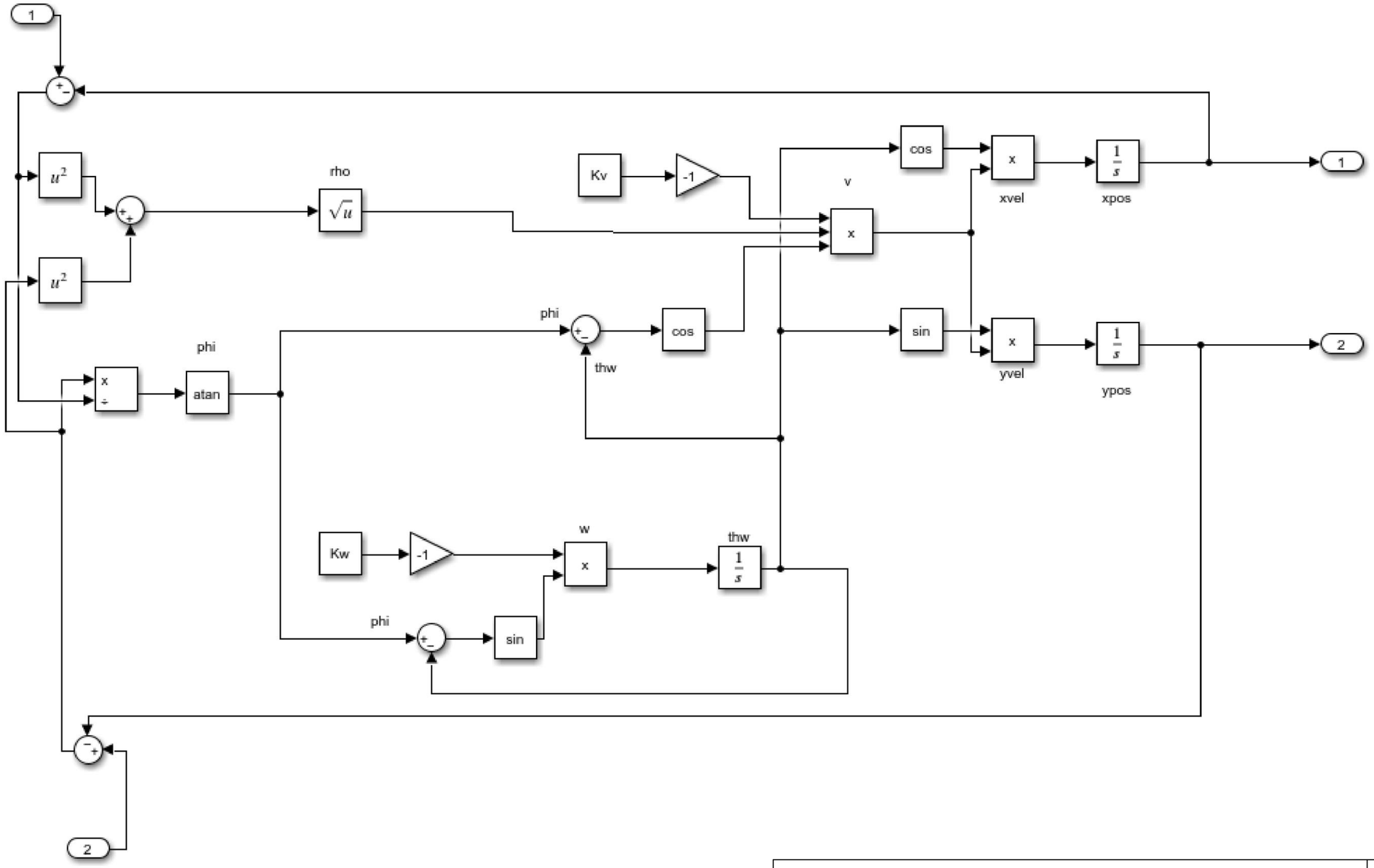
PROYECTO: DISEÑO Y VALIDACIÓN DE ESTRATEGIAS DE CONTROL DE FORMACIONES EN SISTEMAS MULTIRROBOT EN UNA PLATAFORMA VIRTUAL BASADA EN EASY JAVA SIMULATIONS.		Fecha: 25/06/2021 Escala S/E
Autor:Jaime Sancha Miguel	Plano: Ecuaciones modelo lineal	Plano Nº 02



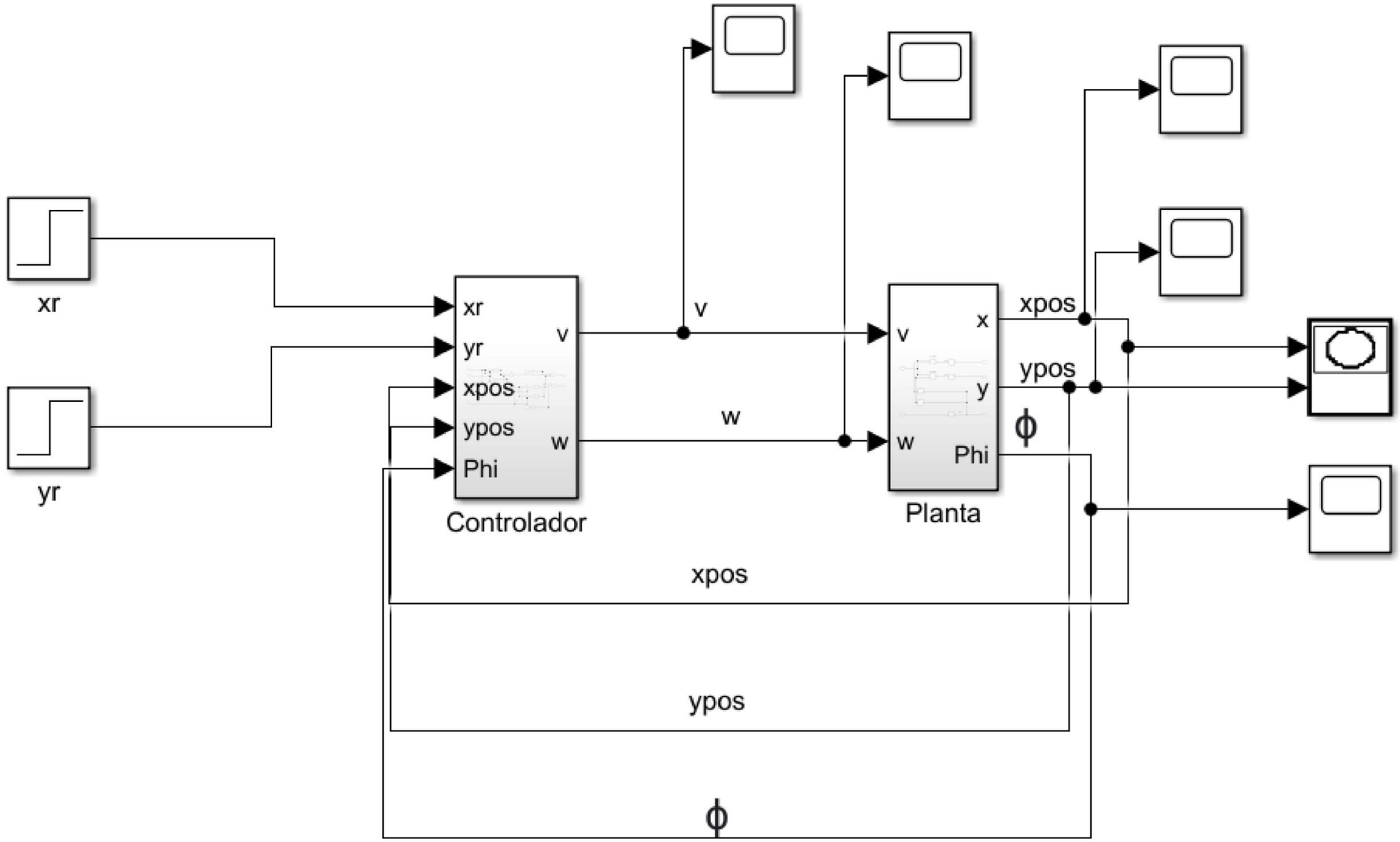
PROYECTO: DISEÑO Y VALIDACIÓN DE ESTRATEGIAS DE CONTROL DE FORMACIONES EN SISTEMAS MULTIRROBOT EN UNA PLATAFORMA VIRTUAL BASADA EN EASY JAVA SIMULATIONS.		Fecha: 25/06/2021 Escala S/E
Autor:Jaime Sancha Miguel	Plano: Esquema para comprobar la respuesta con síntesis directa	Plano Nº 03



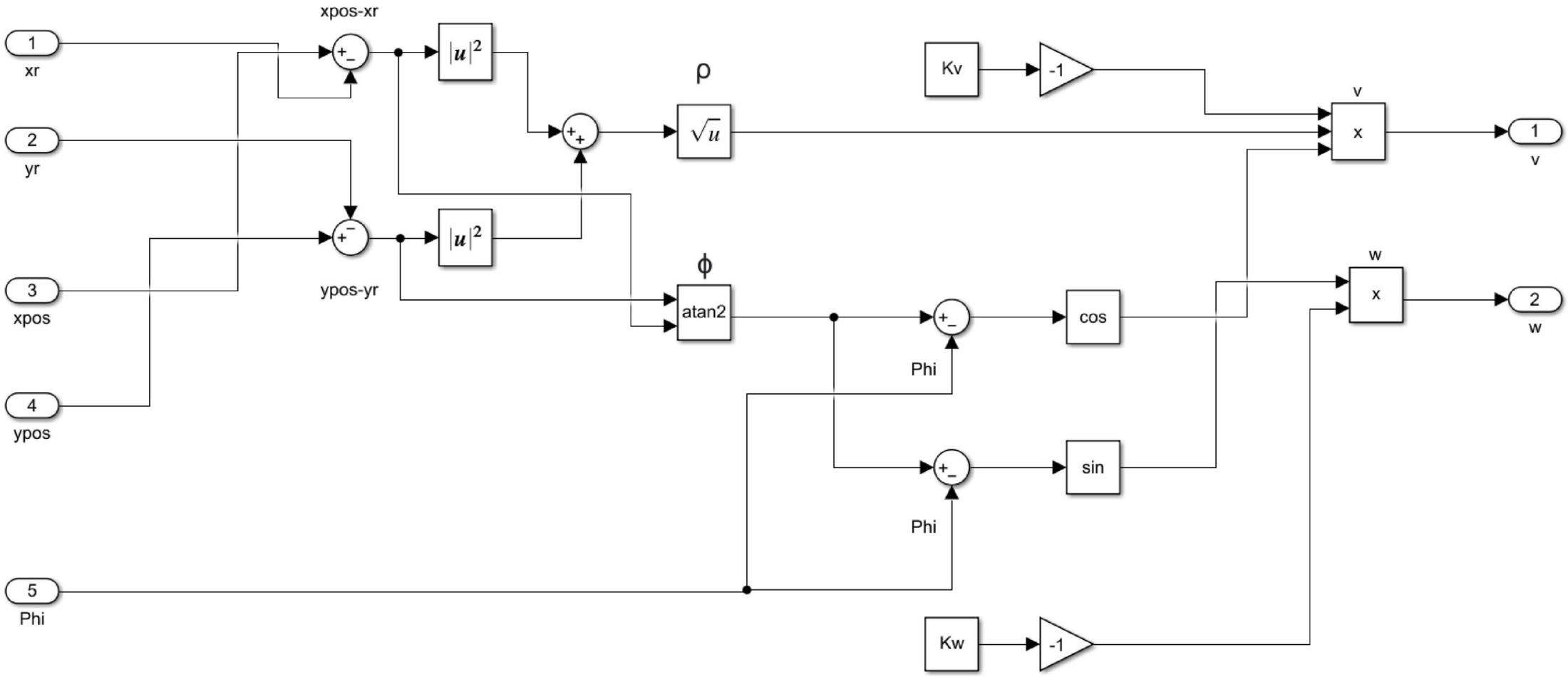
PROYECTO: DISEÑO Y VALIDACIÓN DE ESTRATEGIAS DE CONTROL DE FORMACIONES EN SISTEMAS MULTIRROBOT EN UNA PLATAFORMA VIRTUAL BASADA EN EASY JAVA SIMULATIONS.		Fecha: 25/06/2021 Escala S/E
Autor:Jaime Sancha Miguel	Plano: Ejemplo Anti-Windup	Plano Nº 04



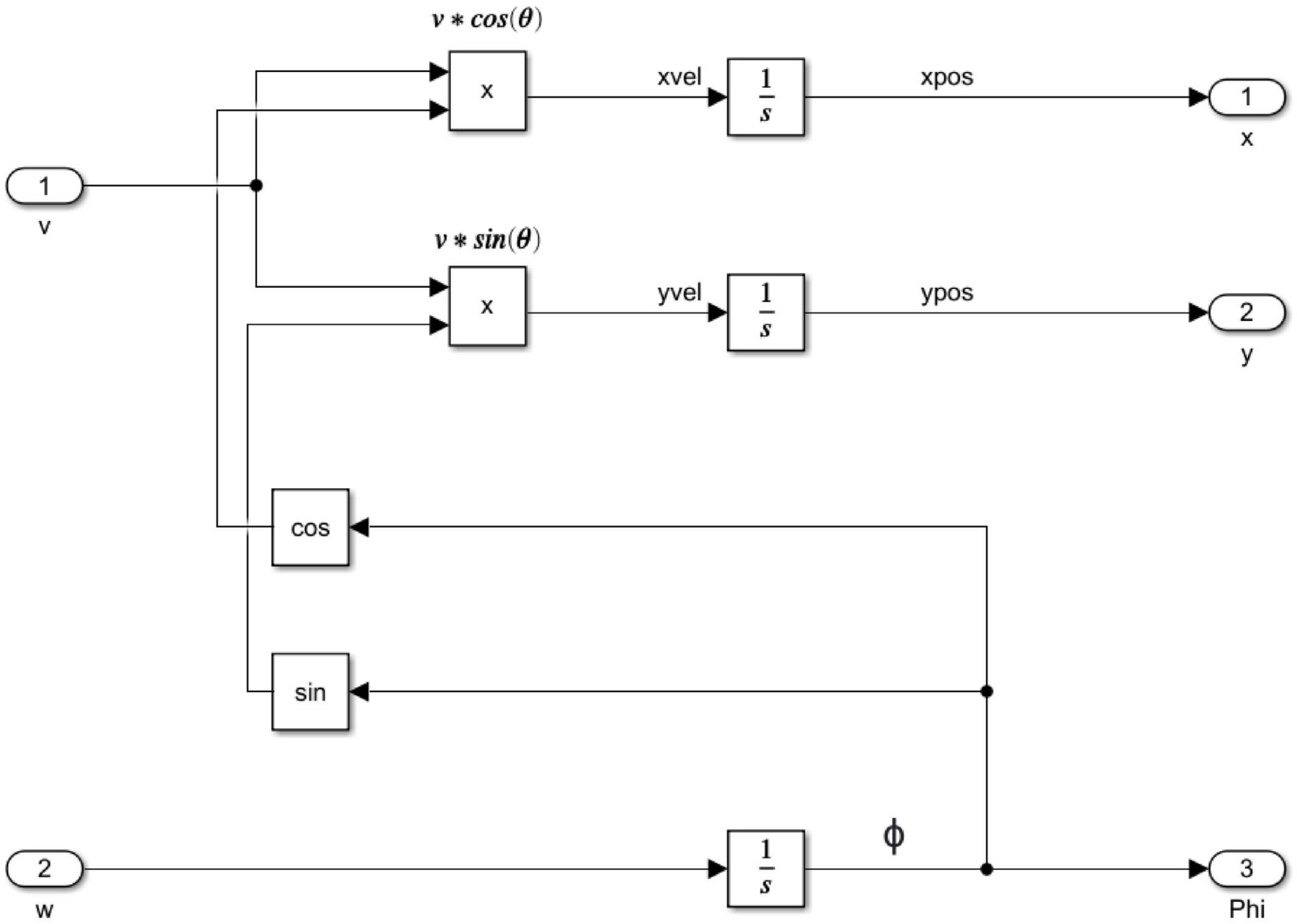
PROYECTO: DISEÑO Y VALIDACIÓN DE ESTRATEGIAS DE CONTROL DE FORMACIONES EN SISTEMAS MULTIRROBOT EN UNA PLATAFORMA VIRTUAL BASADA EN EASY JAVA SIMULATIONS.		Fecha: 25/06/2021 Escala S/E
Autor: Jaime Sancha Miguel	Plano: Esquema con las ecuaciones del modelo no lineal con restricciones no holonómicas	Plano Nº 05 ~~06~~



PROYECTO: DISEÑO Y VALIDACIÓN DE ESTRATEGIAS DE CONTROL DE FORMACIONES EN SISTEMAS MULTIRROBOT EN UNA PLATAFORMA VIRTUAL BASADA EN EASY JAVA SIMULATIONS.		Fecha: 25/06/2021 Escala S/E
Autor:Jaime Sancha Miguel	Plano: Esquema estabilidad segunda ley de control.	Plano Nº 06



PROYECTO: DISEÑO Y VALIDACIÓN DE ESTRATEGIAS DE CONTROL DE FORMACIONES EN SISTEMAS MULTIRROBOT EN UNA PLATAFORMA VIRTUAL BASADA EN EASY JAVA SIMULATIONS.		Fecha: 25/06/2021 Escala S/E
Autor:Jaime Sancha Miguel	Plano: Controlador segunda ley de control.	Plano Nº ~~00~~ 07



PROYECTO: DISEÑO Y VALIDACIÓN DE ESTRATEGIAS DE CONTROL DE FORMACIONES EN SISTEMAS MULTIRROBOT EN UNA PLATAFORMA VIRTUAL BASADA EN EASY JAVA SIMULATIONS.		Fecha: 25/06/2021 Escala S/E
Autor:Jaime Sancha Miguel	Plano: Planta segunda ley de control	Plano Nº 08