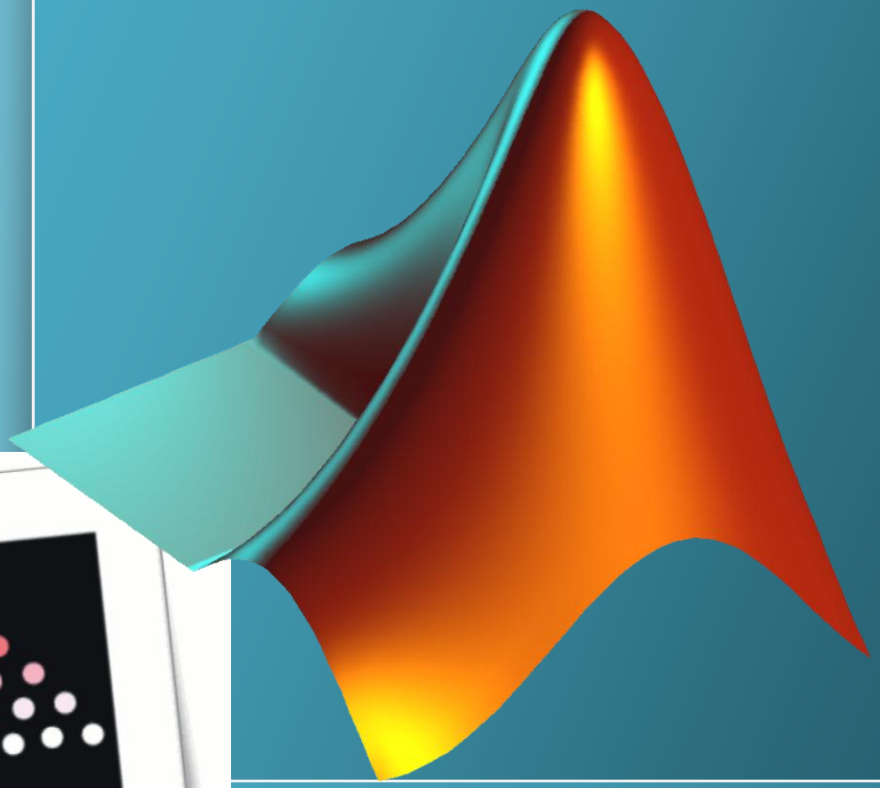
SIMULACIÓN CON MATLAB Y SCILAB DE UN REACTOR TIPO TANQUE AGITADO.



Autores:

Remberto J. Cuadro Alvear

Teddy Cañavera Buelvas

JUNIO 2012

SIMULACIÓN CON MATLAB Y SCILAB DE UN REACTOR TIPO TANQUE AGITADO (CSTR)

REMBERTO JAIME CUADRO ALVEAR

TEDDY CAÑAVERA BUELVAS

UNIVERSIDAD TÉCNOLOGICA DE BÓLIVAR

PROGRAMA DE INGENIERÍA ELECTRÓNICA

Cartagena de Indias, D.T. y C. Junio de 2012

SIMULACIÓN CON MATLAB Y SCILAB DE UN REACTOR TIPO TANQUE AGITADO (CSTR)

REMBERTO JAIME CUADRO ALVEAR

TEDDY CAÑAVERA BUELVAS

MONOGRAFÍA COMO TRABAJO DE GRADO PARA OPTAR AL TITULO DE INGENIERO ELECTRÓNICO

DIRECTOR

MSc. OSCAR SEGUNDO ACUÑA CAMACHO

Universidad Tecnológica de Bolívar

UNIVERSIDAD TÉCNOLOGICA DE BÓLIVAR

PROGRAMA DE INGENIERÍA ELECTRÓNICA

Cartagena de Indias, D.T. y C. Junio de 2012

Cartagena, Junio de 2012

Señores:

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

Comité de evaluación de proyectos.

Ciudad

Estimados señores:

Cordialmente nos permitimos presentar a ustedes la monografía titulada: **"SIMULACIÓN CON MATLAB Y SCILAB DE UN REACTOR TIPO TANQUE AGITADO (CSTR)",** para su estudio, consideración y aprobación, como requisito para obtener el título de Ingeniero Electrónico, además para la aprobación del Minor de Automatización Industrial.

En espera que se cumpla con las normas pertinentes establecidas por la institución.

Sinceramente,

_____ _____

Remberto J. Cuadro Alvear Teddy Cañavera Buelvas

Agradecimientos

Primeramente, quiero dar gracias a Dios y a mi padre por ser mi guía, mi motor, mi mentor y ser quien me dio la claridad para culminar más tarde que te temprano mis estudios universitarios.

Agradezco mas a que a nadie a mi madre, quien es la persona que durante toda la vida me impulsa a salir adelante, a cumplir mis metas y quien mantuvo todo este tiempo incitándome a dominar mi carrera.

A mis abuelos, mis hermanas, mis sobrinos para los que siempre he sido un modelo y con el tiempo han estado a la expectativa de cómo será mi vida a partir del momento en que me convirtiera en profesional.

A mis compañeros, amigos, colegas, profesores, conocidos y todos los que junto a mis familiares siempre estuvieron al tanto de este progreso.

A mi compañero de trabajo Teddy y a mi Director Oscar, ya que gracias a ellos, se pudo culminar esta experiencia invaluable para mi vida.

Por último quiero agradecer a mi futura esposa, por siempre apoyarme, por siempre mantener esas palabras de aliento cuando en momentos no sabía lo que debía hacer, pero que siempre con su empuje me ayudaba a salir adelante.

Remberto J. Cuadro Alvear

Agradecimientos

Agradezco a la universidad Tecnológica de Bolívar por la oportunidad de realizar nuestros estudios.

Nuestros profesores, quien con sus conocimientos hicieron que nuestra vida profesional se enriquecieran, para ser unos excelentes ingenieros.

Oscar segundo Acuña Camacho, director de nuestra monografía de grado, por sus consejos, sentimientos de amistad y recomendaciones a tiempo, que llevar a feliz el termino la misma.

Dios y la Santísima Virgen, luz que guio nuestros senderos y nos han colmado de bienes y sabiduría.

Todas las personas que de una u otra forma nos animaron y colaboraron en la realización de nuestros estudios.

Y por sobre todo quiero dedicar este proyecto a mis padres Manuel Cañavera Sotomayor y Elvira Buelvas Rodríguez

a mis hermanas, Isadora, Farrah y Glenda

a mis sobrinos, Mauren Patricia y Miguel Alejandro

a mi novia, Elena C. Heredia Mercado

por ser la razón de mi vida y soporte para el éxito de este proyecto, porque mis logros son sus logros y realización de mis sueños, son sus sueños.

Teddy Cañavera Buelvas

Nota de Aceptación

_____
_____
_____
_____
_____

_____
Firma del Presidente del Jurado

_____
Firma del Jurado

_____
Firma del Jurado

Pagina dejada en blanco intencionalmente

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

Pagina dejada en blanco intencionalmente

INTRODUCCION

Desde la época de los 60's la idea de emular el comportamiento real de un sistema bajo un ambiente informático se extendió rápidamente entre los investigadores, siendo aceptada como una de las más poderosas herramientas tanto para la comprensión de fenómenos físicos como para la predicción operativa y/o funcional del sistema frente a algún aspecto o condición de operación.

El concepto fundamental del trabajo tiene su base en el enfoque de la simulación, el cual consiste en resolver numéricamente un modelo matemático que rige un sistema físico en el que la solución analítica no se conoce o es de difícil acceso para una aplicación específica. La solución de los modelos matemáticos requiere de la formulación de algoritmos precisos y robustos por lo que la eficacia de este enfoque depende fuertemente de los recursos computacionales.

El modelado y la simulación es un área de constante crecimiento e investigación, que va desde el mismo desarrollo y enfoques de modelado hasta la simulación de sistemas que permitan predecir comportamientos, ajustar diseños, probar condiciones de experimentación, hasta productos como simuladores de entrenamiento para operarios y herramientas de apoyo a la toma de decisiones en diferentes áreas del conocimiento (biología, aeronáutica, industria automotriz, desarrollo de productos, entre otros).

Para el desarrollo de las simulaciones frente a diferentes enfoques de modelado han surgido lenguajes y herramientas comerciales que sirven de apoyo al desarrollo de investigaciones en diferentes ámbitos del conocimiento. De igual manera se ha avanzado con una serie de programas y ambientes Open Source que se convierten en una alternativa significativa frente a los desarrollos comerciales. Los investigadores hacen esfuerzos importantes por evaluar y utilizar estos lenguajes y herramientas a fin de conseguir sus máximas prestaciones.

En este trabajo se analizó el comportamiento de dos herramientas computacionales comúnmente usadas para la simulación de modelos, Matlab (comercial) y Scilab (Open Source). Para hacer el análisis comparativo se hizo una aplicación sobre el benchmark de un CSTR (Continuous System Tank Reactor), bajo unos criterios definidos previamente, los resultados de la investigación se presentan a lo largo de este documento.

Pagina dejada en blanco intencionalmente

1. ESTADO DEL ARTE

Las dificultades que presenta la realización de un proceso antes de su implementación, los peligros que pueda generar y el tiempo invertido al implementarlos, las perturbaciones ocurridas durante su ejecución, entre otras situaciones adversas, ha propiciado el desarrollo de herramientas computacionales como el Matlab (Mathworks) y el Scilab (Open Source) que permitan realizar simulaciones bajo diferentes enfoques de modelado.

Para el caso de estudio de reactores químicos, las herramientas computacionales han sido variadas y su selección ha dependido siempre de la preferencia o bien del diseño metodológico del investigador, según las ventajas que ésta parezca ofrecerle. Sin embargo, la pregunta latente es: ¿hasta dónde los resultados de simulación obtenidos con alguna de las nuevas herramientas de código abierto son comparables computacionalmente con herramientas conocidas como el Matlab.

Han sido innumerables los trabajos desarrollados utilizando Matlab, entre ellos la *simulación borrosa*[1] *de un reactor con reacción exotérmica no lineal* basado en la estructura Takagi – Sugeno – Kang.

[1] Los modelos borrosos o difusos (fuzzy) usan reglas "IF-THEN" para establecer relaciones cualitativas entre las variables. Un modelo borroso consiste de múltiples reglas y cada una de ellas posee un antecedente y un consecuente. El antecedente especifica una cierta parte del espacio de entrada, mientras que le consecuente es un modelo de regresión donde, algunos valores de entrada son utilizados.

En esta simulación, se supuso un modelo excitado, por el flujo de alimentación de reactivo; de modo que se considera al sistema como uno de múltiples entradas y una salida (MISO) y no como un proceso en el que existe un sistema simple entrada - simple salida (SISO).

Según los resultados, el modelo describe al proceso no lineal con un error de prueba máximo de 0.0022, lo cual pone en evidencia el carácter adecuado del mismo al proceso real. Las funciones de pertenencia utilizadas en CSTR_ Fis fueron del tipo de campana de Gauss; pero con otros tipo, tales como la triangular, trapezoidal, etc. Se obtuvieron errores de predicción totalmente satisfactorios.

Al comparar el modelo borroso anteriormente presentado, con alguna variante de modelo lineal determinado mediante técnicas clásicas, como ARX Y ARMAX, se encontró que con ninguno de los dos se obtienen valores promedio del error de predicción menores que el logrado con el sistema de inferencia borroso desarrollado; mientras que el segundo presenta un ajuste del 77%, el primero solo alcanza el 41%.

Por su parte, para el análisis dinámico de un CSTR, presentado por (Guerra & Struck, 2008), enseñó tres estados estacionarios "*obtenidos a través de códigos de MATLAB*", de los cuales, el de menor conversión es estable, mientras que los otros dos fueron inestables. El modelo dinámico les permitió visualizar el comportamiento del CSTR y su sensibilidad respecto a ciertos parámetros, situación relevante, pues es prácticamente imposible operar un proceso sin sufrir variaciones en el sistema.

La linealización del modelo y su ajuste a una función de transferencia de primer orden resultaron muy similares, lo que permite utilizar el modelo lineal para cálculos y análisis posteriores; por lo que con este modelo más sencillo, se espera agilizar este tipo de

procesos. En última instancia, al graficar las respuestas dinámicas de los modelos lineales y no lineales se corroboró la afirmación respecto a la similitud entre los mismos, factor determinante para la aplicación de un mecanismo de control.

Al realizar el estudio de un reactor tipo tanque continuamente agitado (CSTR) diseñaron un modelo matemático en variables de espacio de estado. Posteriormente se utilizó el modelo no lineal para realizar unas pruebas de lazo abierto del sistema y por último se diseñó su sistema de control predictivo[2] por modelo (MPC) que seguidamente, se comparó con una estrategia de control proporcional. Para ambos lazos de control se utilizó el modelo lineal representado a través de variables de estado.

Del análisis del reactor tipo tanque con agitación continua principalmente se dedujo que: ante las dos perturbaciones el proceso es más sensible a los cambios en el flujo de reactante; los máximos valores permisibles para perturbaciones separadas y simultaneas son de ±20% para el flujo de reactante y ±10% en la concentración del reactante. La razón de estos valores es evitar que el algoritmo de control haga que las fracciones de apertura de las válvulas utilizadas tomen valores negativos ó mayores que uno. Las variaciones en el período de muestreo por encima de su valor por defecto originan respuestas sobre-amortiguadas, mientras que valores por debajo del mismo causan respuestas sub-amortiguadas u oscilatorias con la modificación que aumentan el sobre-pico de las salidas y el tiempo de simulación está íntimamente relacionado con la predicción, por tal motivo si éste toma valores muy elevados, se corre el riesgo de sacar al proceso de control.

[2] Para la simulación de la etapa de control se aplicaron dos métodos, los cuales son: Algoritmo de control convencional y Algoritmo de control predictivo.

Para la simulación del modelo se tuvo en cuenta que: la reacción es exotérmica, irreversible y de primer orden, del tipo A => B, donde A es el reactante y B el producto; no se modela el tiempo muerto (retardo); reactante y producto se encuentran en fase líquida; y la densidad y capacidad calórica de la mezcla permanecen constantes, así como otras propiedades termodinámicas de reactante y producto.

Para mostrar la importancia del análisis de estabilidad de un sistema en continuo y cómo éste puede definir las regiones de interés operativo, en este caso para la producción de etanol, (Trejos, Fontalvo, & Gómez, 2009) evaluaron el proceso de obtención del biocombustible en un sistema continuo con células de levadura, a través de modelos matemáticos con dos expresiones cinéticas de crecimiento microbiano Monod y Haldane analizados en un CSTR reactor (Continuous Stirred Tank Reactor). El CSTR descrito por (Oliveira & Moreira, 2005) se muestra en la Figura 1-1.

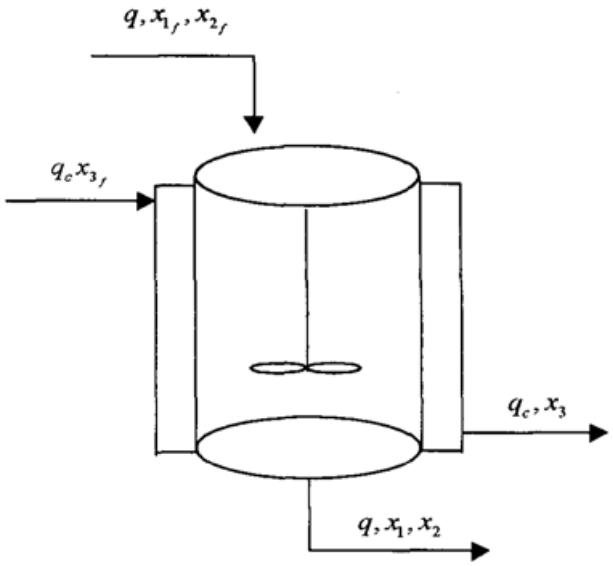


Figura 1-1 - Reactor Tipo Tanque Agitado

El modelo matemático propuesto por (Oliveira & Moreira, 2005) para describir el proceso tiene en cuenta una entrada y una salida en el biorreactor, comportándose como un sistema CSTR, de la siguiente forma:

$$\frac{dx_1}{d\tau} = -Dax_1 exp\left[\frac{x_2}{1+\frac{x_2}{\gamma}}\right] + q\left(x_{1f} - x_1\right) \text{ (1)}$$

$$\frac{dx_2}{d\tau} = \beta Dax_1 exp\left[\frac{x_2}{1+\frac{x_2}{\gamma}}\right] - (q+\delta)x_2 + \delta x_3 + x_{2f} \text{ (2)}$$

$$\frac{dx_3}{d\tau} = \frac{q_c\left(x_{3f} - x_3\right)}{\delta_1} + \frac{\delta(x_2 - x_3)}{\delta_1\delta_2} \text{ (3)}$$

A partir del caso de estudio, la producción de etanol, se demostró que existen diferentes regiones de estabilidad. Cada una de ellas conduce a condiciones de operación con características específicas diferentes. El no considerar las particularidades de dichas regiones, en la selección de las condiciones de operación, puede conducir a alcanzar condiciones de operación de bajo rendimiento y productividad, que no son de interés.

En la simulación del *modelo matemático del sistema series de tanques continuamente agitados isotérmico,* se propuso un modelo matemático de un sistema de tres tanques isotérmicos en serie continuamente agitados, basado en las ecuaciones de balance de masa y de componente. La finalidad fue modelar el sistema e identificar variables que nos permitan realizar acciones correctivas. Para simular el comportamiento de las variables del sistema como la concentración de reactante en cada tanque, se utilizó la aproximación numérica de Euler.

Para implementar el modelo se tuvieron en cuentas varias consideraciones y etapas del proceso, las cuales están regidas cada una por un modelo distinto. Se describen estas etapas de la siguiente forma: modelo del sistema CSTR a volumen constante, modelo del sistema CSTR a volumen variable, método de Euler - sistema CSTR a volumen constante y sistema CSTR a volumen variable.

El modelo matemático proviene del balance de materia y de componente dentro del reactor CSTR del cual resulta el conjunto de ecuaciones diferenciales lineales y no lineales. En el CSTR a volumen constante, este conjunto de ecuaciones nos permite determinar la función de transferencia de la planta en lazo abierto. Los polos reales negativos de la función de transferencia nos permite predecir que el comportamiento de la planta será asintóticamente estable. Un simple método de iteración fue necesario para resolver el estado estable.

El método de Euler[3] permitió determinar el comportamiento del reactante A y del producto B en el tiempo. Se pudo observar a través de la simulación que el reactante A se consume decayendo exponencialmente; mientras que el producto B se produce creciendo exponencialmente. En el volumen CSTR a volumen variable, el comportamiento del volumen en cada tanque creció en el tiempo. Esto permite suponer la necesidad de incorporar un control en las válvulas y de nivel de los tanques por medio de los flujos de entrada y salida.

Las concentraciones del componente A en cada reactor decaen exponencialmente como fue apreciado en el modelo CSTR de volumen constante, mientras que la concentración del producto B se produce con crecimiento exponencial y tiende a la concentración inicial del reactante A. La manipulación de los flujos de entrada y de salida de los tanques utilizando funciones oscilatorias, genera que el volumen crezca con unas pequeñas oscilaciones; sin embargo, la tendencia de crecimiento del volumen permanece invariable debido a la cantidad de volumen.

[3] Este método se aplica para encontrar las soluciones a las ecuaciones diferenciales ordinarias, esto es, cuando la función involucra sólo una variable independiente. El método se basa en forma general en la pendiente estimada de la función para extrapolar desde un valor anterior hasta un nuevo valor.

Mediante una simulación con Scilab, se pudo evaluar de forma objetiva el desempeño de las estrategias de control para las plantas de tratamiento de lodos activados de aguas residuales. El modelo de referencia de simulación resultante nº 1 (BSM1) ha sido la base para un nuevo desarrollo significativo; así en lugar de sólo la evaluación de las estrategias de control a nivel de la unidad de lodos activados (biorreactores y clarificador secundario) de la nueva BSM2 ahora permite a la evaluación de las estrategias de control a nivel de toda la planta, incluyendo el clarificador primario y el tratamiento de lodos con la digestión de lodo anaeróbico (Jeppsson).

La solución numérica del modelo Gray-Scott de auto catálisis cúbico con decaimiento lineal acoplada con la difusión y considerado en un reactor de una dimensión (una celda de difusión del reactor), fue hallada con el uso del software Scilab. Los perfiles de estado de concentración no constante del producto, reactante y auto catálisis en el modelo Gray-Scott se obtuvieron utilizando el método de *pertuburation Homotopía* para los parámetros de valores pequeños (Pandi & Rajendran).

En Scilab, Código de Elementos de Ingeniería de las reacciones químicas, se describen una serie de códigos de Scilab, los cuales se pueden utilizar para simular distintos procesos de reacciones de dicha naturaleza, entre ellas las de reactores no ideales. Si bien, Matlab y Scilab, permiten realizar simulaciones de diferentes procesos y solucionar tipos de ecuaciones específicos, existen entre ellos algunas diferencias básicas.

Las funciones en Scilab no son consideradas como archivos separados, como los archivos de Matlab. Una o varias funciones usadas pueden ser definidas en un solo archivo, y el nombre del archivo no está necesariamente relacionado con el nombre de la función. Así mismo, las funciones no están automáticamente cargadas en Scilab, como están en Matlab luego de que su nombre es invocado (Urroz, 2001).

En cuanto a la implementación del sistema de control, en Matlab éste es muy fácil y altamente preciso y el número de funciones disponibles en las herramientas del sistema de control es muy alto; por su parte Scilab viene con RLtool como ayuda en el sistema de control pero las funciones disponibles son insuficientes para completar el estudio. La implementación del procesamiento de imagen en Matlab es totalmente simple pero son escasas; en tanto, las *ready made functions* en Scilab están disponibles.

Muchas de las funciones de Scilab pueden ser ejecutadas de la misma forma que en Matlab. Sin embargo, los comandos tienen nombres ligeramente diferentes, Tabla 1.1

Tabla 1.1 Resumen Comparativa entre Matlab y Scilab

Función ...Scilab	Que hace	Función ...Matlab
abs	Valor Absoluto y Magnitud Compleja	abs
acosh	Inversa del coseno hiperbólico similar: asin, atahn, etc	acosh
mtlb_hold	Mantiene la gráfica en la misma figura	hold
clc	Borra la ventana de comando	clc

En general, con un margen pequeño de diferencia, Matlab ofrece más ventajas frente a Scilab, lo que finalmente pueden llevar a obtener resultados no similares, pero con una diferencia no muy significativa.

Para comparar la eficiencia en cuanto a funciones aplicadas y tiempo de respuesta, se implementa la matriz 2-D Poisson en Scilab y Matlab.

Tabla 1.2 Comparativa Matriz de Poisson

Función	Tamaño	$T_{generado}$	$T_{solución}$	T_{total}
Matlab	$n = 100$	0.11	7.1	7.21
	$n = 300$	1.045	231	232.045
	$n = 500$	2.97	1059	1061.97
Scilab	$n = 100$	0.155	9.52	9.675
	$n = 300$	1.6	408.6	410.2
	$n = 500$	4.57	2425.5	2430.07

Según los resultados expuestos en la Tabla 1.2, Scilab es útil y puede ser fácilmente una alternativa a Matlab, donde T equivale a Tiempo.

2. IMPLEMENTACIÓN DEL MODELO DEL REACTOR QUÍMICO TIPO TANQUE AGITADO CONTINUAMENTE (CSTR) EN MATLAB.

Para desarrollar el algoritmo y realizar la simulación del Modelo del reactor químico tipo tanque agitado CSTR, se escogió Matlab 2009a (versión 7.8.0 – 64 Bits) como herramienta para el desarrollo e implementación de los algoritmos. Matlab es un programa orientado a la implementación de algoritmos numéricos, con rutinas eficientes de manejo de operaciones matemáticas utilizando matrices, considerado un lenguaje de alto nivel en un ambiente interactivo que permite la implementación de desarrollos con alto requerimiento computacional.

Adicionalmente, Matlab dispone de un gran número de funciones repartidas en distintos *toolboxes* para la identificación y optimización de funciones. También ofrece una herramienta gráfica llamada Simulink el cual contiene todos los elementos básicos que se necesitan para la construcción de diagramas de bloques de un modelo. Posee una librería con bloques de operaciones matemáticas básicas, conmutadores, conectores, elementos de simulación y control, entre otros.

Simulink de Matlab utiliza $\boldsymbol{ode45}$[4] para resolver las ecuaciones diferenciales no lineales, $\boldsymbol{ode45}$ está basado en una formula explicita de Runge-Kutta (4,5) y Dormand-Prince **(Klee & Allen, 2011)**. Es un solucionador de un solo paso; en el cálculo $\boldsymbol{y(t_n)}$ solo se necesita la solución en el punto de tiempo precedente inmediatamente anterior, $\boldsymbol{y(t_n - 1)}$.

[4] ode45 de Simulink utiliza el método Dormand-Prince para la solución de las ecuaciones.

En la Tabla 2.1 se hace un breve resumen de las funciones utilizadas en los algoritmos de Matlab para determinar los puntos de equilibrio del sistema.

Tabla 2.1. Funciones de Matlab utilizadas

Nombre de la función	Descripción
$function$	Declara una función, para declarar una función debe ser la primera línea de cualquier función de Matlab
$zeros$	Construye una matriz de ceros
exp	Se utiliza para crear una función exponencial
$global$	Declara variables globales
$input$	Solicita una entrada de usuario
$display$	Muestra un texto
tic	Marca el arranque del reloj
toc	Finaliza el reloj
clc	Limpia la ventana de comando, no borra las variables almacenadas
$clear\ all$	Borra todas las variables almacenadas en el entorno de trabajo
$linspace$	Genera vectores espaciados linealmente
$ode45$	Solucionador de Ecuaciones Diferenciales Ordinarias, comúnmente utilizado. Hay otros solucionadores de ode's
$fsolve$	Es utilizada para resolver sistemas de ecuaciones algebraicas. f*solve* encuentra las raíces (ceros) de un sistema de ecuaciones no lineales
$figure$	Crea una ventana para figuras o gráficos
$subplot$	Divide la figura en paneles rectangulares para almacenar gráficas, comúnmente utilizado cuando se quieren observar varios resultados
$plot$	Utilizado para dibujar en 2-D
$grid$	Muestra cuadriculas en las gráficas

2.1. GENERACIÓN DE DATOS EXPERIMENTALES

Para generar los datos experimentales, se realizaron 4 algoritmos en MATLAB, con el fin de encontrar los resultados simulados de un reactor tipo tanque agitado el cual tiene sus respectivas entradas y salidas, constantes propias, parámetros y algunas consideraciones para poner a prueba sus puntos de equilibrio.

1. $\boldsymbol{cstr.m}$ en este archivo de Matlab, se crea una función donde se definen las condiciones iníciales, los parámetros del reactor y las ecuaciones de estado que especifican el tipo reactor, se utiliza para determinar los puntos de equilibrio del reactor en estado dinámico.

2. $\boldsymbol{cstr\_ode.m}$ con este archivo el usuario ingresa los datos con los que necesita poner a prueba el reactor, aquí se ingresa el vector de tiempo donde se quiere realizar la muestra, las condiciones iniciales del sistema, los datos para las condiciones del reactor, a su vez, este realiza el método con el cual se resuelve el sistema y se visualizan las gráficas inherentes al sistema.

3. $\boldsymbol{cstr\_ss.m}$ al igual que con el archivo $\boldsymbol{cstr.m}$, en este archivo se definen las condiciones iniciales, los parámetros del reactor y las ecuaciones de estado que especifican el reactor, adicional a eso, se crea una matriz de ceros para las condiciones del reactor, ya que para encontrar los puntos de equilibrio de este sistema, es necesario igualar a 0 las ecuaciones. $\boldsymbol{dx(1)/d\tau = dx(2)/d\tau = dx(3)/d\tau = 0}$..

4. $\boldsymbol{cstr\_fsolve.m}$ este archivo es el utilizado para encontrar los puntos de equilibrio del sistema, el usuario ingresa las condiciones iniciales del sistema y el algoritmo arroja los resultados y el tiempo de ejecución.

Los M-Files están divididos en dos parejas, ya que cada uno tiene su utilización específica, hay dos archivos muy similares donde están depositados los parámetros y las ecuaciones de estado del sistema. Estos son $cstr.m$ y $cstr\_ss.m$, los cuales son utilizados para hallar los puntos de equilibrio del sistema en el tiempo y el otro para determinar las raíces de las ecuaciones, respectivamente, esto dependiendo de ciertas condiciones iniciales, las cuales son requeridas en los otros dos m-files llamados $cstr\_ode$ y $cstr\_fsolve$.

En los archivos $cstr.m$ y $cstr\_ss.m$, se encuentran los parámetros y las ecuaciones de estado, (Oliveira & Moreira, 2005) definen las condiciones del sistema, ecuaciones de balance de masa y temperatura, los parámetros del reactor donde se definen los valores de:

- ✓ concentración inicial de la sustancia en el reactor $(x1_f = 1)$
- ✓ temperatura inicial del reactor $(x2_f = 0)$
- ✓ temperatura inicial de la carcasa $(x3_f = -1)$
- ✓ flujo de entrada en el reactor $(q = 1)$
- ✓ flujo de entrada en la carcasa del reactor $(q_c = 1.65102)$
- ✓ energía de activación $(\gamma = 20)$
- ✓ número de Damköhler $(D_a = 0.072)$
- ✓ calor de la reacción $(\beta = 8)$
- ✓ coeficiente de transferencia de calor $(\delta = 0.3)$
- ✓ relación de volumen del reactor $(\delta_1 = 0.1)$
- ✓ la relación de la densidad de la capacidad de calor de la reacción $(\delta_2 = 0.5)$

y las ecuaciones de estado donde están, la ecuación del balance de masa de la sustancia dentro del reactor (1), la ecuación del balance de energía dentro del reactor (2) y la ecuación del balance de energía dentro de la carcasa del reactor (3).

Para determinar los puntos de equilibrio del sistema se utilizan los archivos llamados $cstr\_ss.m$ y $cstr\_fsolve.m$, estos no tienen en cuenta la variación en el tiempo, sino los

puntos donde el sistema adquiere valores estables. El archivo que se necesita ejecutar para determinar los puntos estables del sistema es $cstr\_fsolve$.m.

Para encontrar puntos de equilibrio en estado estacionario de un sistema, se deben igualar a cero (0) las ecuaciones de balance de masa del componente dentro del reactor, el balance de energía dentro del reactor y el balance de energía dentro de la carcasa del reactor, quedando así, un sistema de tres ecuaciones lineales de primer orden con tres incógnitas. El método para resolver el sistema de ecuaciones con la rutina *fsolve* es mostrada a continuación:

$$[x] = fsolve(@fun, x0) \quad (4)$$

Donde:

$[x]$ es el resultado en forma de vector donde se adquieren los puntos de equilibrio del sistema para unas condiciones iniciales asignadas.

fun es la función donde están inmersas los parámetros y las ecuaciones al cual se le pretenden encontrar las raíces. Para este caso $cstr\_ss$.

$x0$ es un vector en el que se consignan las condiciones iniciales el cual el sistema va a buscar las raíces. Se escribe de la forma $[x_1; x_2; x_3]$.

Teniendo en cuenta la sintaxis anterior (4), al realizar las iteraciones para distintas condiciones iniciales se obtuvieron distintos resultados, los cuales están consignados en la Tabla 2.2.

Tabla 2.2 Obtención de las condiciones estables mediante $fsolve$

Condición Inicial	Tiempo de Ejecución [seg]	Matlab (fsolve)		
$[x1 = x2 = x3]$		$x1$	$x2$	$x3$
0	0,2306	0,8933	0,5193	-0,5951
1	0,2101	0,5528	2,7517	0,0000
2	0,2124	0,5528	2,7517	0,0000
3	0,2166	0,5528	2,7517	0,0000
4	0,2128	0,5528	2,7517	0,0000
5	0,2113	0,5528	2,7517	0,0000
6	0,2149	0,5528	2,7517	0,0000
7	0,2199	0,5528	2,7517	0,0000
8	0,2184	0,5528	2,7517	0,0000
9	0,2202	0,5528	2,7517	0,0000
10	0,2240	0,5528	2,7517	0,0000
20	0,2260	0,8933	0,5193	-0,5950
21	0,2290	0,5528	2,7517	0,000
22	0,2320	0,8933	0,5193	-0,5950
23	0,2354	0,5528	2,7517	0,0000
24	0,2324	0,5528	2,7517	0,0000
25	0,2310	0,5528	2,7517	0,0000
30	0,2303	0,8933	0,5193	-0,5950
40	0,2314	0,1890	5,1373	0,6359
50	0,2341	0,5528	2,7517	0,0000
100	0,2364	0,1890	5,1373	0,6359
200	0,2390	0,1890	5,1373	0,6359
500	0,2481	0,1890	5,1373	0,6359
1000	0,2326	0,8933	0,5193	-0,5951

Con la obtención de datos experimentales se puede observar que para los tres estados iniciales se tienen tres distintos puntos de equilibrio, los cuales están consignados en la Tabla 2.3.

Tabla 2.3 Puntos de Equilibrio del CSTR

	x_1^E	x_2^E	x_3^E
Estable (1)	0,8933	0,5193	-0,5951
Inestable (2)	0,5528	2,7517	0,0000
Estable (3)	0,1890	5,1373	0,6369

En la casilla "Tiempo de ejecución", está plasmado el tiempo que tarda Matlab en resolver las ecuaciones para encontrar los puntos de equilibrio; en promedio tardó 0.2326 segundos, para encontrar las raíces del sistema.

Luego de obtener los tres distintos puntos de equilibrio para los diferentes estados, se procede a determinar el comportamiento del sistema en el tiempo, mediante otro algoritmo de Matlab. Para ello se utilizaron los scripts llamados $cstr.m$ y $cstr\_ode.m$ los cuales son utilizados para encontrar los puntos de equilibrio del sistema en el tiempo y a su vez en estado dinámico, es decir con el sistema expuesto a cualquier entrada y/o cambio en las entradas que puedan afectar el sistema haciendo que pueda volverse inestable, observando su comportamiento en el tiempo y las gráficas de su conducta.

Para obtener los resultados, el m-file que se debe ejecutar es $cstr\_ode$, donde el usuario final ingresa el periodo de tiempo en el que pretende observar el comportamiento del sistema ante una entrada, correspondientes a las condiciones iniciales al sistema $[x_1; x_2; x_3]$.

Con la ayuda web de Matlab (Mathwoks), se pudieron encontrar las distintas formas de resolver sistemas de ecuaciones diferenciales ordinarias mostradas en la Tabla 2.4.

Tabla 2.4 Rutinas para resolver ecuaciones diferenciales en Matlab

Rutina	Tipo de problema	Orden de precisión	Cuando utilizarla
ode23	No rígido	Media	La mayoría de las veces, puede ser el primer solucionador que se utilice.
ode45	No rígido	Baja	Para problemas con niveles de error aceptables.
ode113	No rígido	Moderadamente alta	Para problemas con niveles de error rigurosos o para problemas resueltos computacionalmente.
ode15s	Rígido	Moderadamente media	Si ode45 no es el adecuado por problemas de rigidez.
ode23s	Rígido	Baja	Si utiliza tolerancias aceptables para resolver sistemas rígidos y matrices de masa constante.
ode23t	Moderadamente Rígido	Baja	Para casos de problemas moderadamente rígidos y si no se necesita una solución sin amortiguación numérica.
ode23tb	Rígido	Baja	Solo utiliza tolerancias aceptables para resolver sistemas rígidos.

Para la solución de las ecuaciones se utilizó la rutina $ode45$[5], implementando un algoritmo donde el usuario ingresa las condiciones iniciales del sistema, condiciones del estado del reactor y el tiempo donde desea realizar la muestra. Además de eso, se anexó

[5] Se utilizó $ode45$ para este sistema específico debido a que ya se conocían las soluciones de las Ecuaciones Diferenciales del sistema, para cualquier otro caso, se debe realizar una investigación previa con el fin de determinar cuál es el método de integración más adecuado para dar solución al sistema.

un cronometro, con el fin de conocer el tiempo en que Matlab tarda en resolver el sistema de ecuaciones.

La forma como escribe la rutina $ode45$, es expresada en (5):

$$[t, x] = ode45(@fun, t, x0); \text{ (5)}$$

donde, fun es la función donde se ingresaron los parámetros y las ecuaciones de estado, llamado $cstr.m$.

t es el vector de tiempo donde se va a realizar la muestra para este caso se utilizó un tiempo de muestreo que va desde 0 hasta 20seg.

$x0$ es un vector con las condiciones iniciales del sistema, es cual es definido como $[x_1; x_2; x_3]$, donde x_1 es la condición inicial de la entrada de sustancia al reactor, x_2 es la temperatura inicial del reactor y x_3 es la temperatura inicial de la carcasa del reactor. Se tomaron varias muestras, para determinar los diferentes puntos de estabilidad del sistema y el tiempo que tarda Matlab en hallar estos puntos, los cuales son mostrados más adelante.

$[t, x]$ es un vector donde t es el periodo de tiempo donde se realiza la muestra y x muestra los valores asignados de $x_1, x_2\ y\ x_3$ en cada instante de tiempo, es decir, el comportamiento de la entrada hasta que el sistema se encuentre en equilibrio.

Al igual que con la rutina $fsolve$ se realizaron varias iteraciones con $ode45$ con el fin de hallar los distintos puntos de equilibrio del sistema para diferentes entradas y observando su comportamiento en el tiempo, véase Tabla 2.5.

Tabla 2.5 Obtención del comportamiento del sistema con $ode45$ en Matlab

Condición Inicial	Tiempo de Ejecución [seg]	Matlab (ode45)		
[x1 = x2 = x3]		x1	x2	x3
0	0,3520	0,8933	0,5192	-0,5950
1	0,3818	0,8933	0,5193	-0,5950
2	0,3864	0,1890	5,1373	0,6360
3	0,4055	0,1890	5,1373	0,6360
4	0,5825	0,1890	5,1373	0,6359
5	1,0078	0,1890	5,1373	0,6359
6	1,9104	0,1890	5,1373	0,6359
7	3,1407	0,1890	5,1373	0,6359
8	5,0043	0,1890	5,1373	0,6359
9	7,5344	0,1890	5,1373	0,6359
10	10,6117	0,1890	5,1373	0,6359
20	74,5056	0,1890	5,1373	0,6359
30	174,0028	0,1890	5,1373	0,6359
40	280,9546	0,1890	5,1373	0,6359
50	399,2214	0,1890	5,1373	0,6363
100	860,1674	0,1890	5,1373	0,6359
200	-	-	-	-
500	-	-	-	-
1000	-	-	-	-

Del resultado de la simulación para distintas muestras se pueden extraer algunas observaciones:

- Para condiciones de entrada 0 y 1 se obtiene un tipo de salidas estable,
- a partir de condiciones de entrada iguales a 2 o mayores el comportamiento de la salida también es estable y no varía,
- a medida que aumentan las condiciones de entrada, el tiempo de ejecución del algoritmo aumenta sin cambio alguno en la salida del sistema,
- para condiciones iniciales superiores a 100, el tiempo de ejecución aumenta considerablemente, por lo que no se realizaron estas simulaciones.

2.2. ANALISIS DE LOS RESULTADOS EXPERIMENTALES

Se realizaron distintas pruebas con el fin de observar el comportamiento del sistema y conocer visualmente cual es el tiempo aproximado en que el sistema logra estabilizarse para ciertas entradas.

En la Figura 2-1 se observa el tiempo de estabilización cuando la entrada es igual a 0.

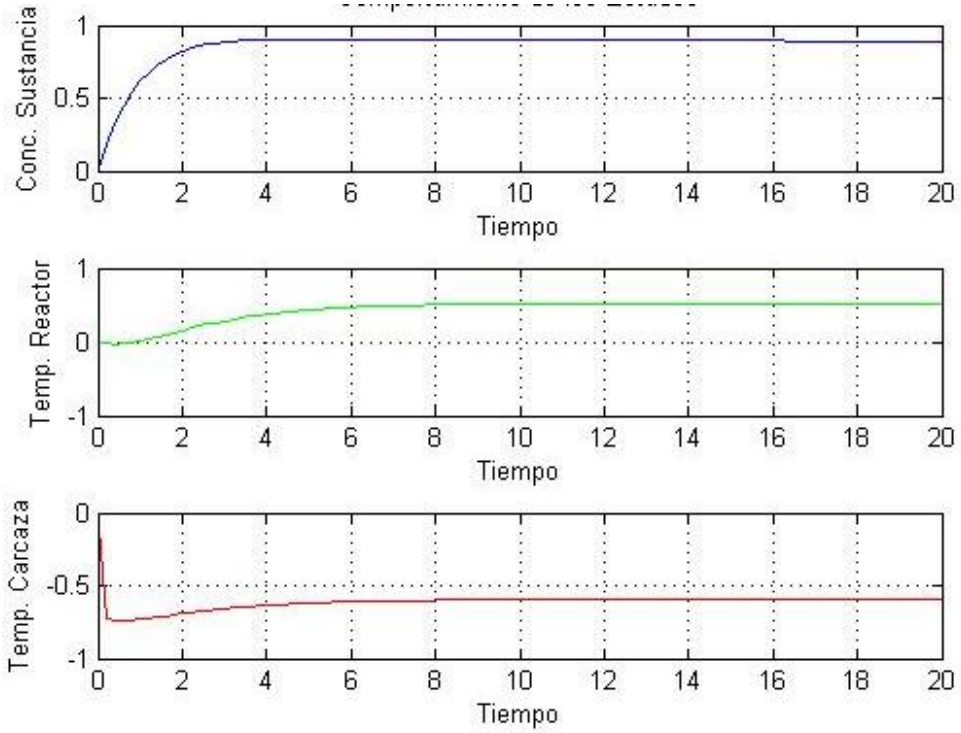


Figura 2-1 - Comportamiento del sistema para condiciones iniciales iguales a 0

De la Figura se pueden obtener ciertas observaciones:

1. En la gráfica de Concentración de sustancia, se puede observar que el sistema tardó aproximadamente unos **3.5 segundos** en buscar su punto de equilibrio, equivalente a 0.8933.
2. En las gráficas de las temperaturas del Reactor y la carcasa, se puede evidenciar que el sistema tarda aproximadamente **8 segundos** en llegar a su punto de equilibrio, equivalentes a 0.5193 y -0.5950, respectivamente.

Ahora se procede a analizar la Figura 2-2, con condiciones de entrada iguales a 1

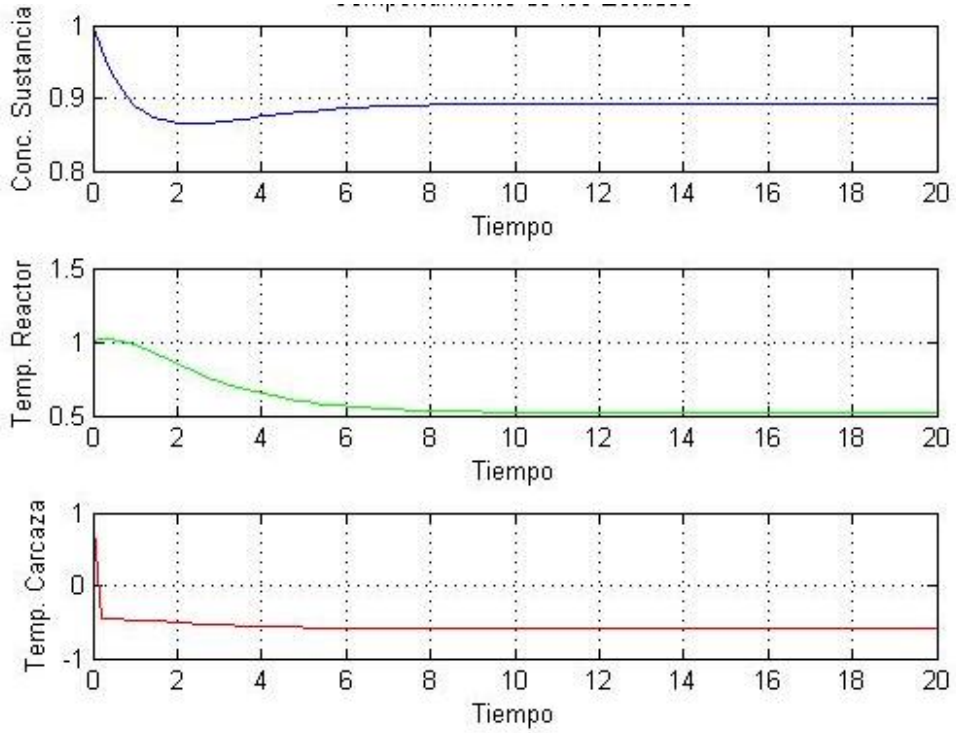


Figura 2-2 - Comportamiento del sistema para condiciones iniciales iguales a 1

Analizando las gráficas se puede concluir que:

En la gráfica para condiciones de entrada iguales a 1, el sistema tarda aproximadamente **8 segundos** en estabilizarse. Su punto es equivalente a 0.8933

1. En el caso de la temperatura del reactor, el sistema tarda aproximadamente **9 segundos** en llegar al punto de equilibrio equivalente a 0.5193
2. Para la gráfica de la temperatura de la carcasa, el sistema tarda en llegar a su punto de estabilización **5 segundos**, equivalente a -0.5950. Véase Figura 2-2.

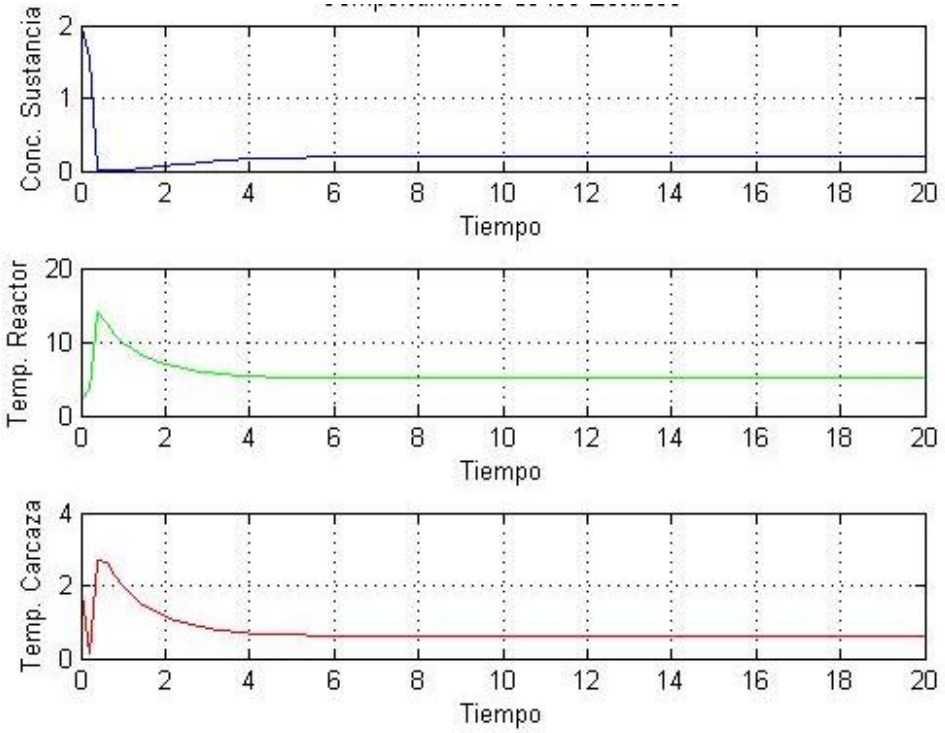


Figura 2-3 - Comportamiento del sistema para condiciones iniciales iguales a 2

Para el caso, donde la entrada al sistema tiene amplitud equivalente a 2, analizando la Figura 2-3 se puede analizar qué:

1. En la gráfica de Concentración de sustancia, el sistema tiene una caída que tiende a 0 y luego busca su punto de estabilización equivalente a 0.1890 en aproximadamente **5 segundos**.
2. Para la gráfica de la temperatura del reactor, el sistema tiene un pico de 14.11 en menos de 1 segundo y luego tarda casi **5 segundos** en llegar a su punto estable para esta entrada, en este caso su punto de estabilización es equivalente a 5.1370.
3. Mientras que en la gráfica de la temperatura de la carcasa, se puede observar que hay una caída de temperatura equivalente a 0.1131 en aprox. 0.2 segundos y

luego un pico de 2.6595 en aprox. 0.5 segundos antes de llegar a su punto estable equivalente a 0.6360 en aproximadamente **5 segundos.** A partir de este punto el sistema empieza a tener un comportamiento similar.

Para distintas condiciones de entrada, véase Figura 2.4 a Figura 2.7.

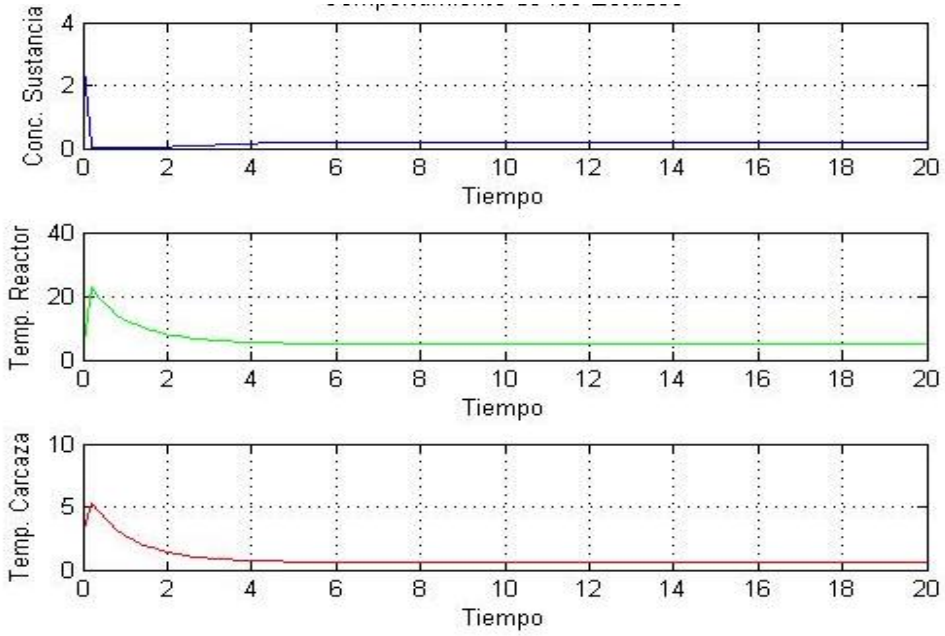


Figura 2-4 - Comportamiento del sistema para condiciones iniciales iguales a 3

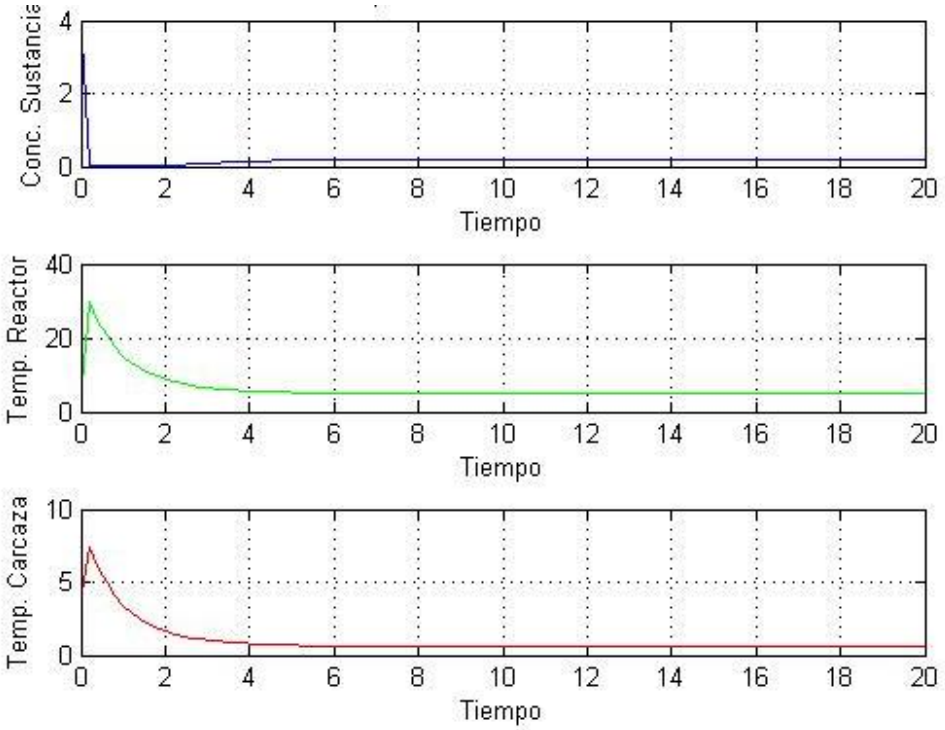


Figura 2-5 - Comportamiento del sistema para condiciones iniciales iguales a 4

En la curva de la Temperatura del Reactor se puede observar que el sistema tiene un pico muy alto en este estado y que luego regresa a su punto estable

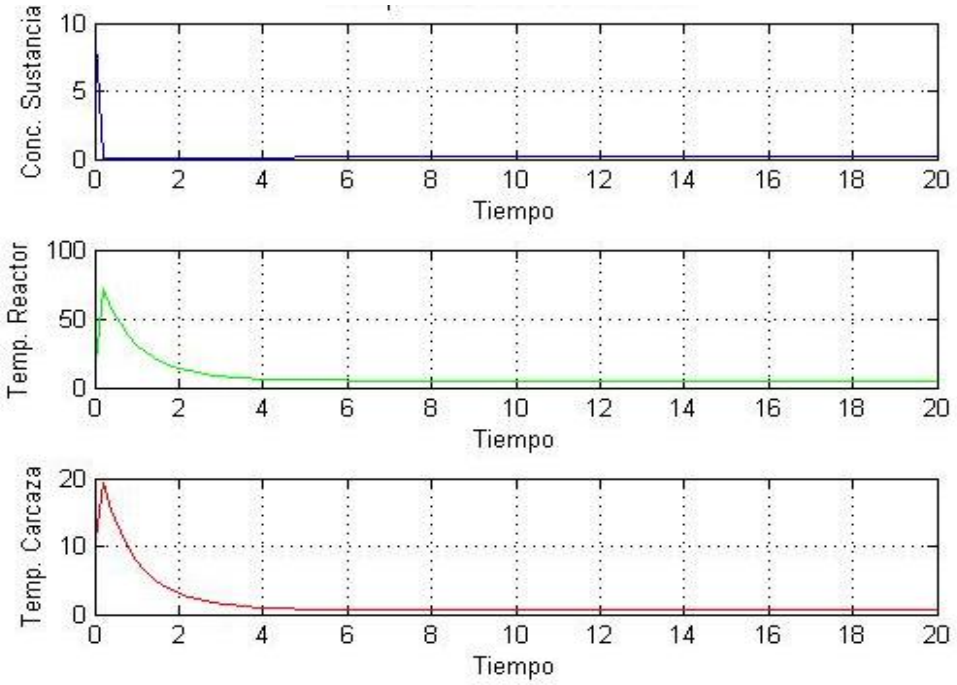


Figura 2-6 - Comportamiento del sistema para condiciones iniciales iguales a 5

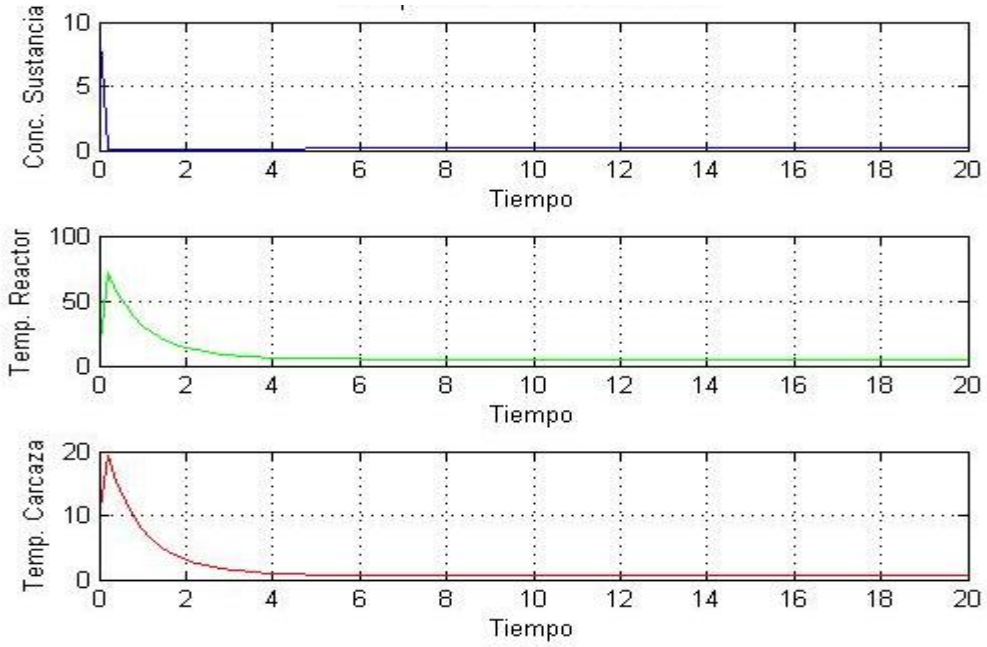


Figura 2-7 - Comportamiento del sistema para condiciones iniciales iguales a 10

Observando las Figura 2-4 hasta la Figura 2-7, se puede concluir que el sistema a partir de una entrada equivalente a 3 o mayor a ese valor, comienza a comportarse de manera similar, teniendo picos en las entradas y luego llega a su punto de estabilización.

2.3. REPRESENTACIÓN DEL SISTEMA DE UN CSTR MEDIANTE SIMULINK

En la Figura 2-8 se describe el diagrama de bloques de Simulink en el cual se relacionan los modelos del reactor seleccionado, el cual está descrito por las ecuaciones (1), (2), (3). Los bloques identificados como Scope, se utilizan para observar el comportamiento del sistema en el tiempo mediante una gráfica, se adecuó un bus de datos para verificar los tres comportamientos en una sola figura.

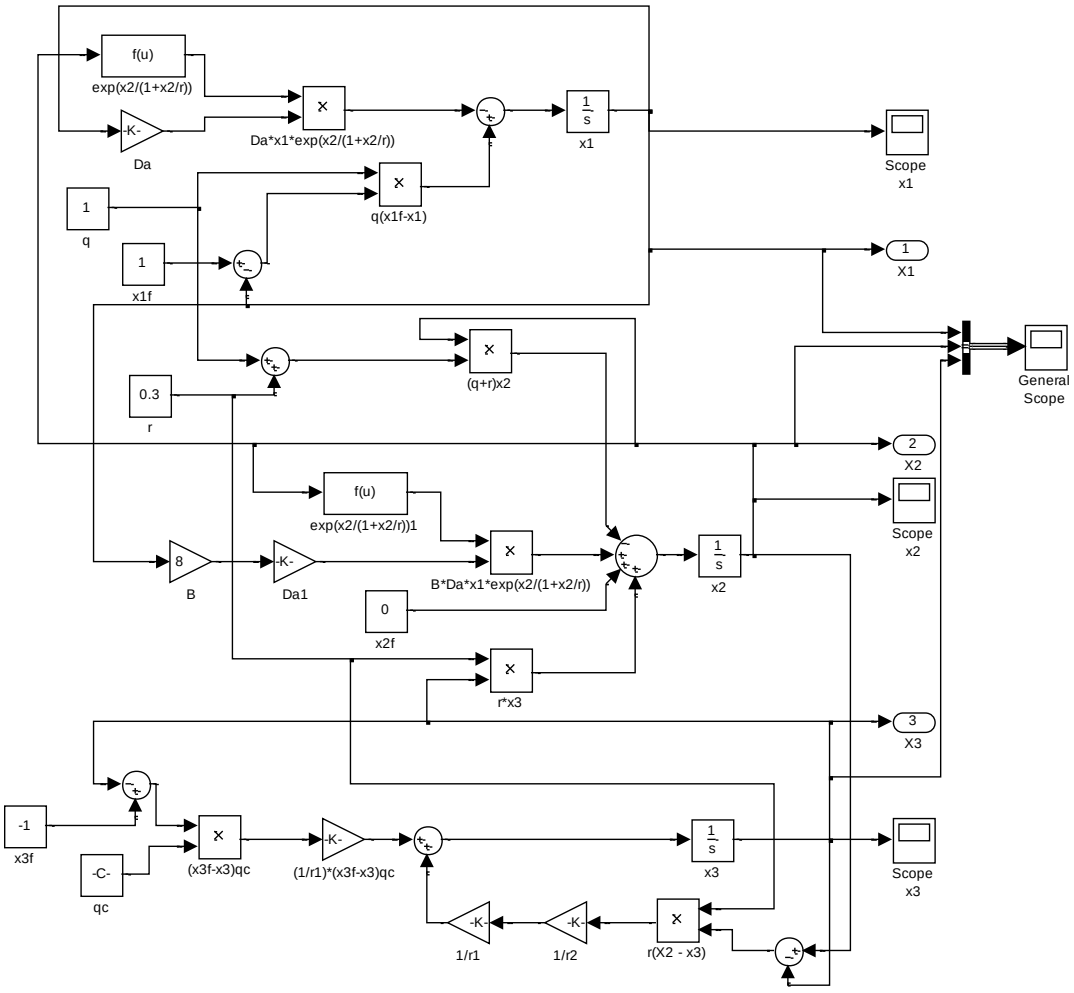


Figura 2-8 – Diagrama de bloques del sistema en Simulink

Al realizar las simulaciones en Simulink para distintas condiciones de entrada al sistema, se obtienen los siguientes resultados gráficos:

Para una condición de entrada igual a 0 se obtiene, véase Figura 2-9,

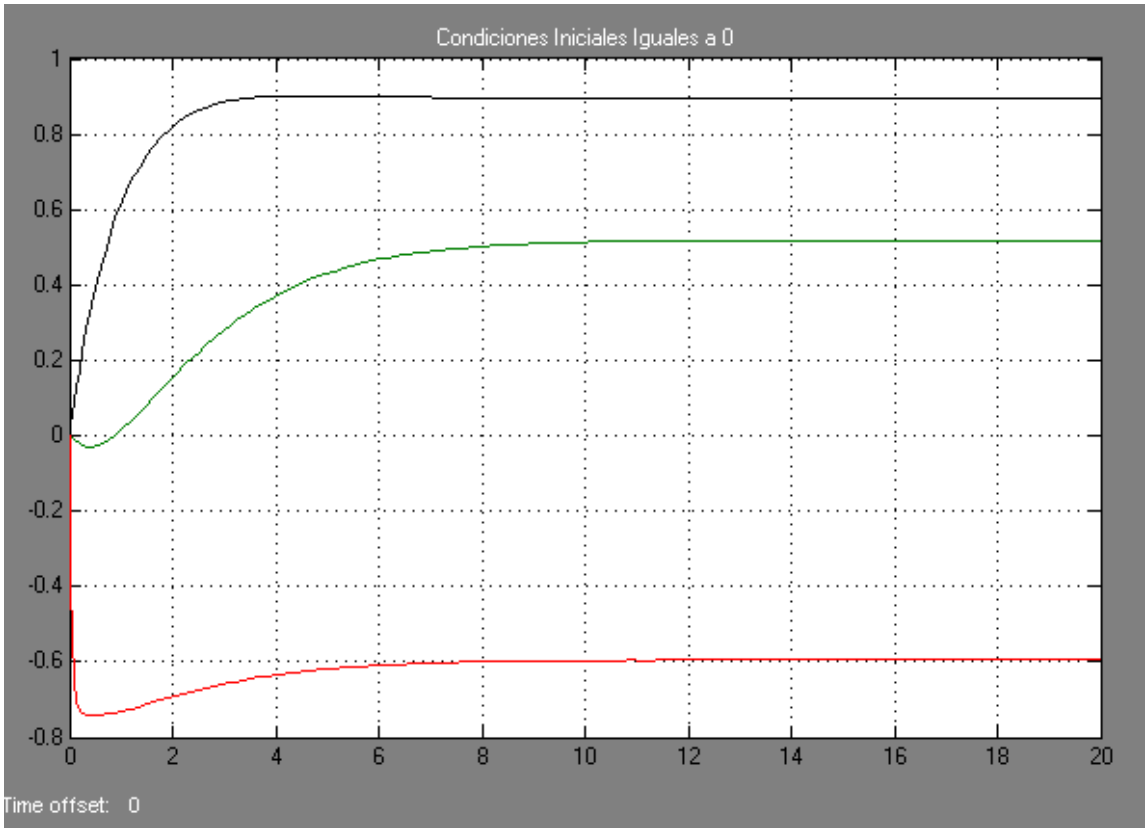


Figura 2-9 – Condiciones iniciales iguales a 0 en Simulink

El sistema adquiere su valor estable en los tres estados aproximadamente a los 10 segundos, en la grafica se puede apreciar que la estabilidad se alcanza de forma natural.

Para identificar los datos de las gráficas debemos tener en cuenta,

- La curva de color negro es la equivalente a la Concentración de la Sustancia.
- La curva de color verde es la equivalente a la Temperatura del Reactor.
- La curva de color rojo es la equivalente a la Temperatura de la carcasa.

Para una condicion de entrada igual a 1, el sistema tiene el comportamiento mostrado en la Figura 2-10

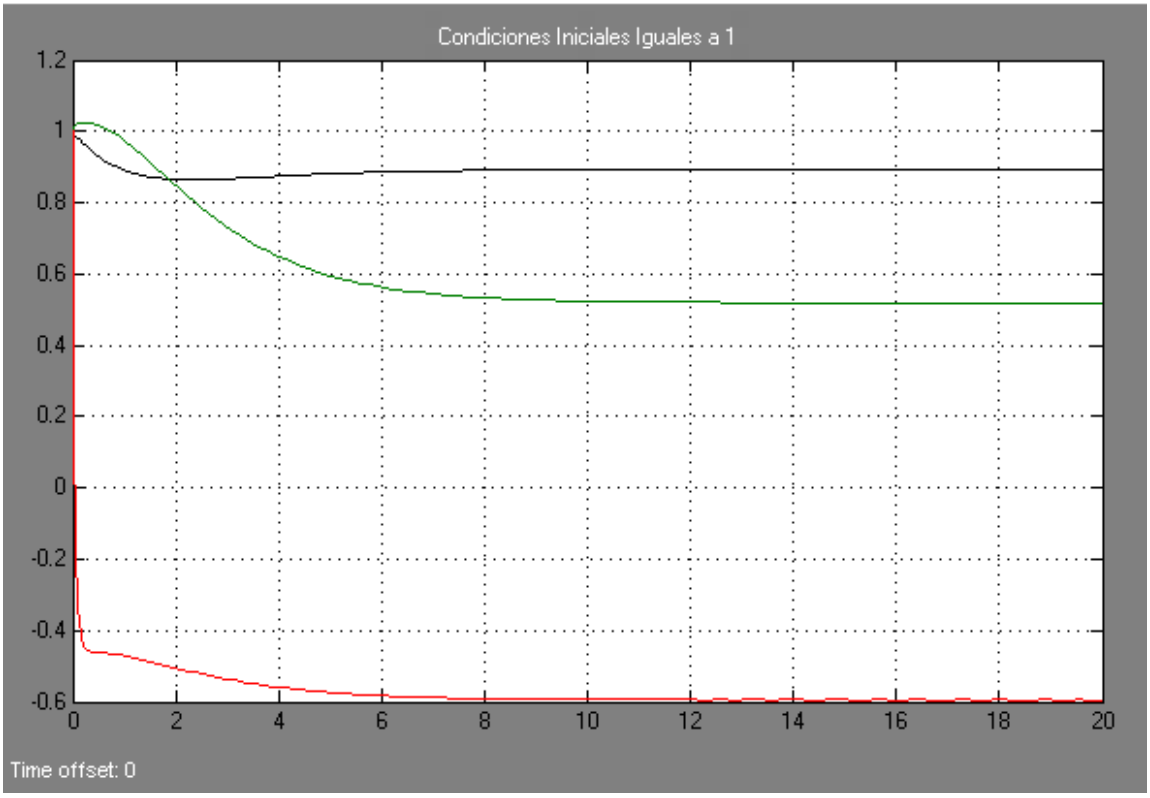


Figura 2-10 – Condiciones iniciales iguales a 1 en Simulink

En la gráfica se puede observar que el sistema para una condición de entrada equivalente a 1, el tiempo aproximado de estabilización es igual a 8 segundos.

Tanto para condiciones iniciales iguales a 0 e iguales a 1, recordando que los puntos de estabilización de concentración de la sustancia = 0.8933, la temperatura del reactor = 0.5193 y el de la temperatura de la carcasa = -0.5950, en la Figura 2-2 y Figura 2-3 se pueden observar dichos puntos.

Para condiciones de entrada equivalentes a 2 y mayores el sistema tiene un comportamiento similar. La Figura 2-11 y Figura 2-12, muestran graficas para condiciones iniciales de amplitud 2 y 5, respectivamente.

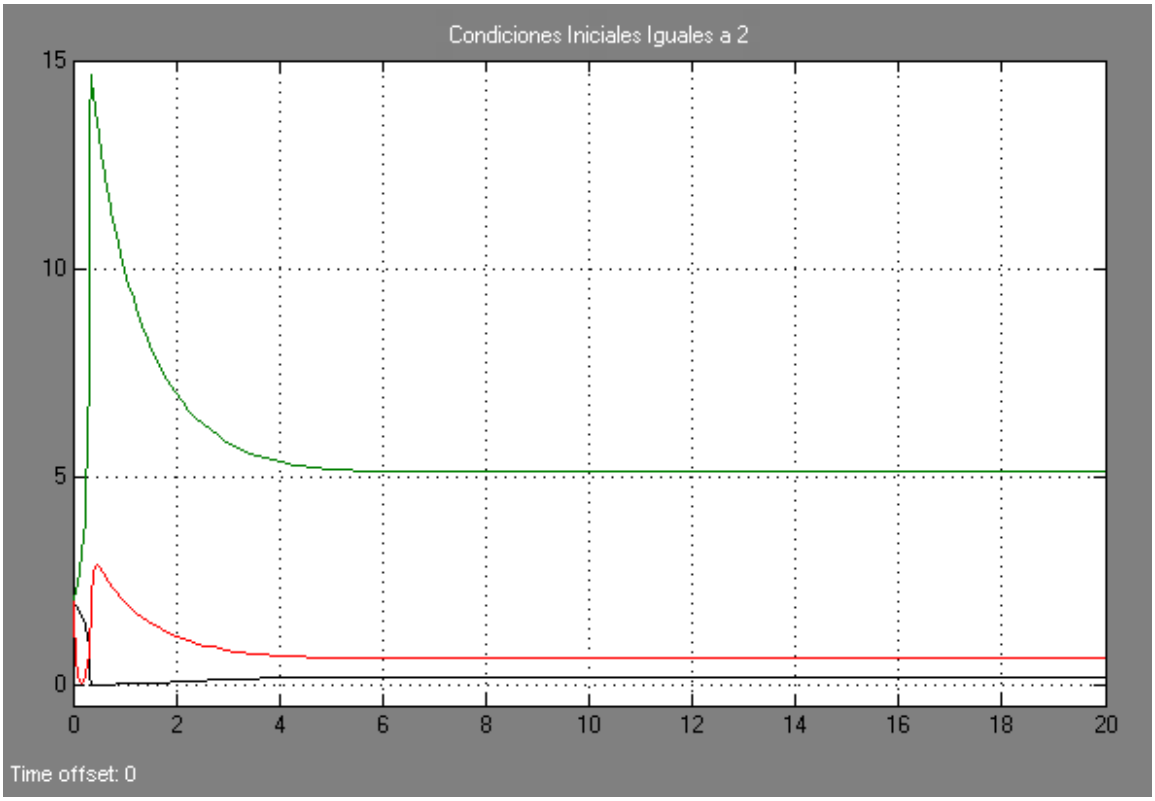


Figura 2-11 – Condiciones iniciales iguales a 2 en Simulink

Ante una entrada equivalente a 2, las temperaturas del reactor y de la carcasa adquieren unos picos bastante elevados y luego llegan a su punto de estabilidad, la concentración de la sustancia reacciona lentamente y en poco tiempo llega a su punto de estabilidad.

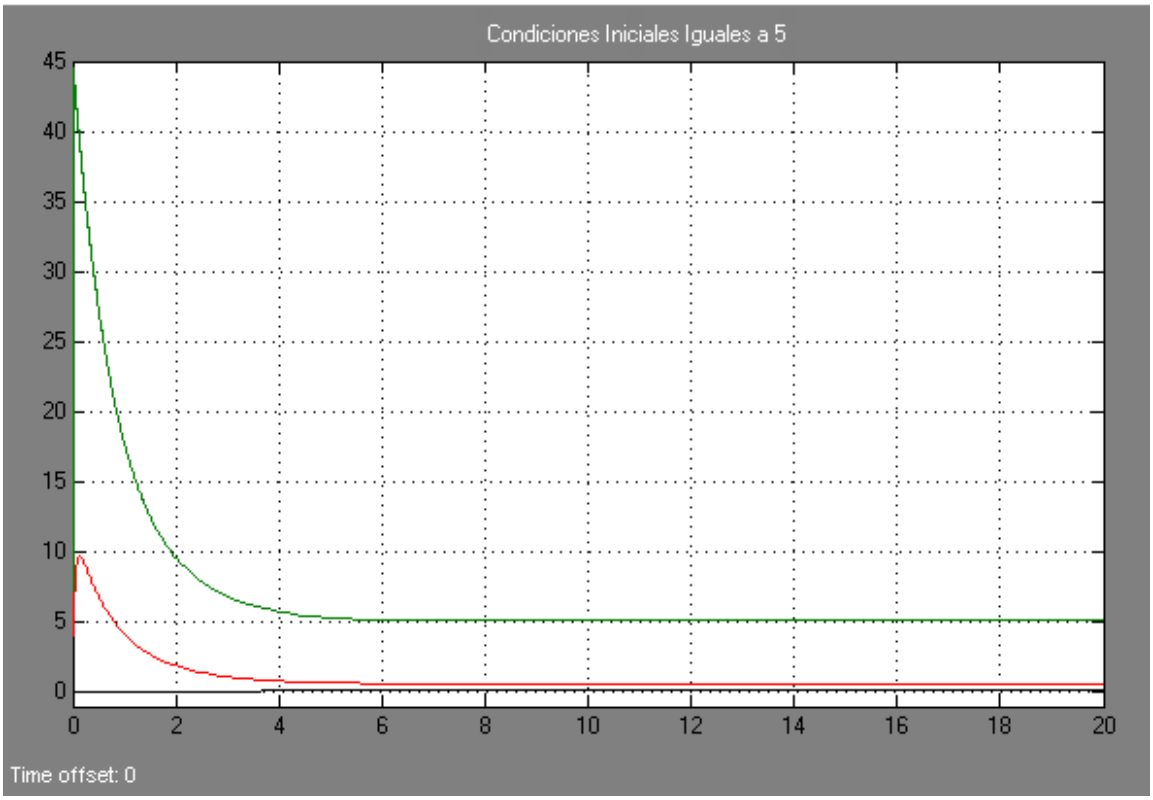


Figura 2-12 – Condiciones iniciales iguales a 5 en Simulink

Si se observa detalladamente ambas la Figura 2-11 y la Figura 2-12, se concluye que el comportamiento del sistema es similar ante una entrada mayor a 2, donde a medida que aumentan las condiciones de entradas la amplitud de la salida aumenta notablemente y luego alcanza su punto estable en un punto determinado.

3. IMPLEMENTACIÓN DEL MODELO DEL REACTOR QUÍMICO TIPO TANQUE AGITADO CONTINUAMENTE (CSTR) EN SCILAB.

Nuevamente al realizar la simulación del Modelo del reactor químico tipo tanque agitado CSTR se escogió Scilab (versión 5.3.3 – 32 Bits) como herramienta para el desarrollo e implementación de los algoritmos. Scilab es un software matemático, con un lenguaje de programación de alto nivel, interactivo, de uso libre y disponible para distintos sistemas operativos, fue creado para realizar cálculos numéricos aunque también ofrece la posibilidad de realizar cálculos simbólicos como derivadas, funciones polinomiales y racionales.

Adicionalmente, Scilab dispone de un entorno similar a Simulink de Matlab para la simulación de sistemas dinámicos y resolución de sistemas de ecuaciones diferenciales llamado XCOS. El cual dispone de varios paquetes que incluyen algunas herramientas para la simulación sencilla de circuitos eléctricos y termodinámica. Para lanzarlo solo basta escribir en la consola de Scilab $xcos$ o bien abriéndolo desde el menú Aplicaciones.

XCOS es un editor grafico para la construcción de sistemas híbridos dinámicos. Los modelos pueden ser ensamblados, cargados, grabados, compilados y simulados. Provee una interfaz de varios compiladores de diagramas de bloques y el simulador hibrido Scicos. XCOS utiliza $cvode$ para resolver sistemas de ecuaciones diferenciales ordinarias e ida para resolver ecuaciones diferenciales algebraicas. Por defecto se utiliza $cvode$.

En la Tabla 3.1 se hace un breve resumen de las funciones utilizadas en los algoritmos de Scilab para determinar los puntos de equilibrio del sistema.

Tabla 3.1 Funciones de Scilab utilizadas

Nombre de la función	Descripción
$function$	Declara una función, para declarar una función debe ser la primera línea de cualquier función de Scilab
$zeros$	Construye una matriz de ceros
exp	Se utiliza para crear una función exponencial
$global$	Define variables globales
$input$	Solicita una entrada de usuario
$disp$	Muestra un texto
$tic()$	Marca el arranque del reloj
$toc()$	Finaliza el reloj
clc	Limpia la ventana de comando, no borra las variables almacenadas
$clear\ all$	Borra todas las variables almacenadas en el entorno de trabajo
$linspace$	Genera vectores espaciados linealmente
ode	Soluciona de forma explícita sistemas de ecuaciones diferenciales ordinarias
$fsolve$	Es utilizada para resolver sistemas de ecuaciones algebraicas. f*solve* encuentra las raíces (ceros) de un sistema de ecuaciones no lineales
$figure$	Crea una ventana para figuras o gráficos
$subplot$	Divide la figura en paneles rectangulares para almacenar gráficas, comúnmente utilizado cuando se quieren observar varios resultados
$plot$	Utilizado para dibujar en 2-D
gca	Es una rutina utilizada para el manejo de ejes en una figura
$grid$	Muestra cuadriculas en las gráficas

3.1. GENERACIÓN DE DATOS EXPERIMENTALES

De forma similar a lo realizado en Matlab, en Scilab se crearon 4 algoritmos con el fin de encontrar los resultados simulados de un reactor tipo tanque agitado, el cual cuenta con sus respectivas entradas y salidas, constantes propias, parámetros y algunas consideraciones para poner a prueba sus puntos de equilibrio.

1. $\boldsymbol{cstr.sce}$ en este archivo de Scilab, se crea una función donde se definen las condiciones iniciales, los parámetros del reactor y las ecuaciones de estado que especifican el tipo reactor, se utiliza para determinar los puntos de equilibrio del reactor en el tiempo.

2. $\boldsymbol{cstr\_ode.sce}$ con este archivo el usuario ingresa los datos con los que necesita poner a prueba el reactor. Aquí se ingresa el vector de tiempo donde se quiere realizar la muestra, las condiciones iniciales del sistema, los datos para las condiciones del reactor, el cual realiza el método de integración con el cual se resuelve el sistema y se visualizan las gráficas inherentes al sistema.

3. $\boldsymbol{cstr\_ss.sce}$ al igual que con el archivo $\boldsymbol{cstr.sce}$, en este archivo se definen las condiciones iniciales, los parámetros del reactor y las ecuaciones de estado que especifican el reactor, adicional a eso, se crea una matriz de ceros para las condiciones del reactor, ya que para encontrar los puntos de equilibrio de este sistema, es necesario igualar a 0 las ecuaciones. $\boldsymbol{dx(1)/d\tau = dx(2)/d\tau = dx(3)/d\tau = 0}$.

4. $\boldsymbol{cstr\_fsolve.sce}$ este archivo es el utilizado para encontrar los puntos de equilibrio del sistema, el usuario ingresa las condiciones iniciales del sistema y el algoritmo arroja los resultados y el tiempo de ejecución.

Los cuatro algoritmos fueron divididos en dos parejas, una pareja para encontrar los puntos de equilibrio en estado estacionario y la otra pareja para encontrar los puntos de equilibrio en estado dinámico.

En los archivos $cstr.sce$ y $cstr\_ss.sce$, se encuentran los parámetros y las ecuaciones de estado donde, nuevamente, están definidas las condiciones del sistema, las ecuaciones de balance de masa y temperatura, los valores son iguales a los que fueron implementados en el Capitulo 2, en Matlab.

Para determinar los puntos de equilibrio del sistema se utilizan los archivos llamados $cstr\_ss.sce$ y $cstr\_fsolve.sce$. Estos no tienen en cuenta la variación en el tiempo, sino los puntos donde el sistema adquiere valores estables en estado estacionario. El archivo que se necesita ejecutar para determinar los puntos estables del sistema es $cstr\_ss.sce$.

Para que Scilab pueda ejecutar los algoritmos se deben inicialmente "*cargar*" en la consola de Scilab, para hacerlo solo basta con hacer click en FILE, luego en EJECUTAR, aparecerá una ventana y desde allí buscará el archivo a ejecutar en la ruta o carpeta donde se tenga almacenado, para este caso $cstr\_ss.sce$.Luego en la misma Consola de Scilab se escribe el comando a continuación descrito, $\boldsymbol{exec\ cstr\_fsolve.sce}$ para ejecutar el algoritmo deseado. Scilab tiene múltiples rutinas para resolver sistemas de ecuaciones, en este caso la más adecuada para la solución de estas es "$fsolve$[6]".

[6] La función de Scilab $fsolve$ es utilizada para resolver sistemas de ecuaciones no lineales. Esta función no soporta dispersión jacobiana. f*solve* está basado en el método de Powell, aunque quizá no es el mejor método. En Scilab, todavía no es posible afinar los parámetros del método fsolve – Tomado de http://wiki.scilab.org/Contributor%20-%20FSolve.

El método para resolver el sistema de ecuaciones mediante la rutina *fsolve* es mostrada en (6):

$$[x] = fsolve(x0, fun); \text{ (6)}$$

Donde $[x]$ es el resultado en forma de vector donde se adquieren los puntos de equilibrio del sistema para unas condiciones iniciales asignadas,

fun es la función donde están inmersas los parámetros y las ecuaciones al cual se le pretenden encontrar las raíces. Para este caso, $cstr\_ss$,

$x0$ es un vector en el que se consignan las condiciones iniciales que el sistema usará para buscar las raíces. Se escribe de la forma $[x_1; x_2; x_3]$

Se realizaron iteraciones en Scilab con el algoritmo $cstr\_fsolve.sce$, con el fin de encontrar los puntos de equilibrio y el tiempo que Scilab tardaba en encontrar estas raíces. De forma similar que en Matlab, Scilab establece un cronometro para determinar el tiempo en que este resuelve la ecuación, mediante ***tic()*** para iniciar el conteo y ***toc()*** para finalizarlo.

Teniendo en cuenta la sintaxis anterior, al realizar las iteraciones para distintas condiciones iniciales se obtuvieron resultados diferentes, los cuales están consignados en la Tabla 3.2.

Tabla 3.2 Obtención de las condiciones estables en Scilab mediante $fsolve$

Condición Inicial	Tiempo de Ejecución [seg]	Scilab (fsolve)		
[x1 = x2 = x3]		x1	x2	x3
0	0,002	0,8933	0,5193	-0,5950
1	0,003	0,8933	0,5193	-0,5950
2	0,004	0,5528	2,7517	0,0000
3	0,002	0,5528	2,7517	0,0000
4	0,002	0,5528	2,7517	0,0000
5	0,002	0,1890	5,1373	0,6359
6	0,002	0,1890	5,1373	0,6359
7	0,003	0,1890	5,1373	0,6359
8	0,002	0,1890	5,1373	0,6359
9	0,002	0,1890	5,1373	0,6359
10	0,003	0,1890	5,1373	0,6359
20	0,003	0,1890	5,1373	0,6359
21	0,003	0,1890	5,1373	0,6359
22	0,003	0,1890	5,1373	0,6359
23	0,003	0,1890	5,1373	0,6359
24	0,003	0,1890	5,1373	0,6359
25	0,003	0,1890	5,1373	0,6359
30	0,003	0,1890	5,1373	0,6359
40	0,003	0,1890	5,1373	0,6359
50	0,003	0,1890	5,1373	0,6359
100	0,003	0,1890	5,1373	0,6359
200	0,003	0,1890	5,1373	0,6359
500	0,003	0,1890	5,1373	0,6359
1000	0,003	0,1890	5,1373	0,6359

En la casilla “Tiempo de ejecución” se plasmó el tiempo que tarda Scilab en resolver las ecuaciones para encontrar los puntos de equilibrio. En promedio demoró 0.00275 segundos, para encontrar las raíces del sistema.

Con la ayuda de Scilab, se pudo encontrar los diferentes tipos de métodos para la solución de las ecuaciones diferenciales, los cuales se pueden observar en la Tabla 3.3.

Tabla 3.3 Tipos de métodos para resolver ecuaciones diferenciales en Scilab

Tipo	**Descripción**
Ninguno	Automáticamente selecciona el método más adecuado. Los métodos utilizados en este tipo son, el Método de Adams y el Método BDF (Formula de Derivación Inversa).
Adams	Utiliza el método de Adams. Utilizado en problemas no rígidos
Stiff	Utiliza el método BDF. Utilizado en problemas rígidos
rk	Utiliza el método Bogacki-Shampine de 4to Orden
rkf	Utiliza el método Dormand-Prince de 4to y 5to Orden
fix	Igual al tipo rkf. Es el método más simple para probar
root	Solucionador con capacidad de encontrar raíces. Utiliza una variante de lsoda para encontrar las raíces de un vector dado
discrete	Para simulación discreta en el tiempo

Para la solución de las ecuaciones se utilizó la rutina ode de tipo rfk, implementando un algoritmo donde el usuario ingresa las condiciones iniciales del sistema, condiciones del estado del reactor y el tiempo en el que se desea realizar la muestra, además de eso, se anexó un cronometro, con el fin de conocer el tiempo en que Matlab tarda en resolver el sistema de ecuaciones.

Para ello se utilizaron los scripts llamados $cstr.sce$ y $cstr\_ode.sce$ los cuales son utilizados para solucionar las ecuaciones del modelo utilizado, con el fin de encontrar los puntos de equilibrio del sistema en el tiempo, lo puede realizar también para un sistema dinámico, es decir con el sistema expuesto a cualquier entrada y/o cambio en las entradas que puedan afectar el sistema haciendo que este pueda volverse inestable, observando su comportamiento en el tiempo y las gráficas de la conducta de este.

Para obtener los resultados, se debe ejecutar desde la consola de Scilab el archivo $cstr.sce$, luego ingresar el comando ***exec cstr_ode.sce;*** para que se ejecute el

algoritmo que resolverá las ecuaciones en un rango de tiempo determinado. El usuario final ingresa el periodo de tiempo en el que pretende observar el comportamiento del sistema ante una entrada, correspondientes a las condiciones iniciales al sistema $[x_1; x_2; x_3]$.

Para la solución de las ecuaciones, en este caso, se utilizó la rutina ode, la cual es la función estándar en Scilab para resolver ecuaciones diferenciales ordinarias.

Si se ejecuta el algoritmo $cstr\_ode.sce$, habiendo cargado el $cstr.sce$, se pueden determinar las raíces del sistema y su comportamiento en el tiempo.

Para ejecutar desde la consola de Scilab la rutina ode sin abrir $cstr\_ode.sce$, la forma como escribe correctamente, es expresada en (7):

$$[x] = ode(x0, t0, t, fun); \text{ (7)}$$

donde, fun es la función donde se ingresaron los parámetros y las ecuaciones de estado, llamada $cstr.sce$

$t0$ es el tiempo inicial para comenzar la muestra.

t es el vector de tiempo donde se va a realizar la muestra. Para este caso se utilizó un tiempo de muestreo que va desde 0 hasta 20 segundos.

$x0$ es un vector con las condiciones iniciales del sistema, es cual es definido como $[x_1; x_2; x_3]$, donde x_1 es la condición inicial de la entrada de sustancia al reactor, x_2 es la temperatura inicial del reactor y x_3 es la temperatura inicial de la carcasa del reactor. Se tomaron varias muestras, para determinar los diferentes puntos de estabilidad del sistema y el tiempo que tarda Scilab en hallar estos puntos.

$[x]$ es un vector donde muestra los valores asignados de $x_1, x_2\ y\ x_3$ en cada instante de tiempo.

Al igual que con la rutina $fsolve$ se realizaron varias iteraciones con ode con el fin de hallar los distintos puntos de equilibrio del sistema para diferentes entradas y observando su comportamiento en el tiempo. Ver Tabla 3.4.

Tabla 3.4 Obtención del comportamiento del sistema con ode en Scilab.

Condición Inicial	**Tiempo de Ejecución [seg]**	**Scilab (ode)**		
[x1 = x2 = x3]		x1	x2	x3
0	0,0430	0,8933	0,5192	-0,5951
1	0,0410	0,8933	0,5193	-0,5950
2	0,0720	0,1890	5,1373	0,6360
3	0,0530	0,1890	5,1373	0,6360
4	0,0600	0,1890	5,1373	0,6359
5	0,0640	0,1890	5,1373	0,6359
6	0,0680	0,1890	5,1373	0,6359
7	0,0650	0,1890	5,1373	0,6359
8	0,0670	0,1890	5,1373	0,6359
9	0,0620	0,1890	5,1373	0,6359
10	0,0700	0,1890	5,1373	0,6359
20	0,0690	0,1890	5,1373	0,6359
30	0,0730	0,1890	5,1373	0,6359
40	0,0750	0,1890	5,1373	0,6359
50	0,0720	0,1890	5,1373	0,6359
100	0,0700	0,1890	5,1373	0,6359
200	0,0770	0,1890	5,1373	0,6359
500	0,0770	0,1890	5,1373	0,6359
1000	0,0790	0,1890	5,1373	0,6359

Del resultado de la simulación para distintas muestras se pueden extraer algunas observaciones:

- Para condiciones de entrada 0 y 1 se obtiene un tipo de salidas estable,

- a partir de condiciones de entrada iguales a 2 o mayores el comportamiento de la salida también es estable y no varía,
- a medida que aumentan las condiciones de entrada, el tiempo de ejecución del algoritmo aumenta.

3.2. ANALISIS DE RESULTADOS EXPERIMENTALES

En la Figura 3-1 se puede observar el tiempo que tarda el sistema en estabilizarse, cuando las condiciones de entrada son equivalentes a 0.

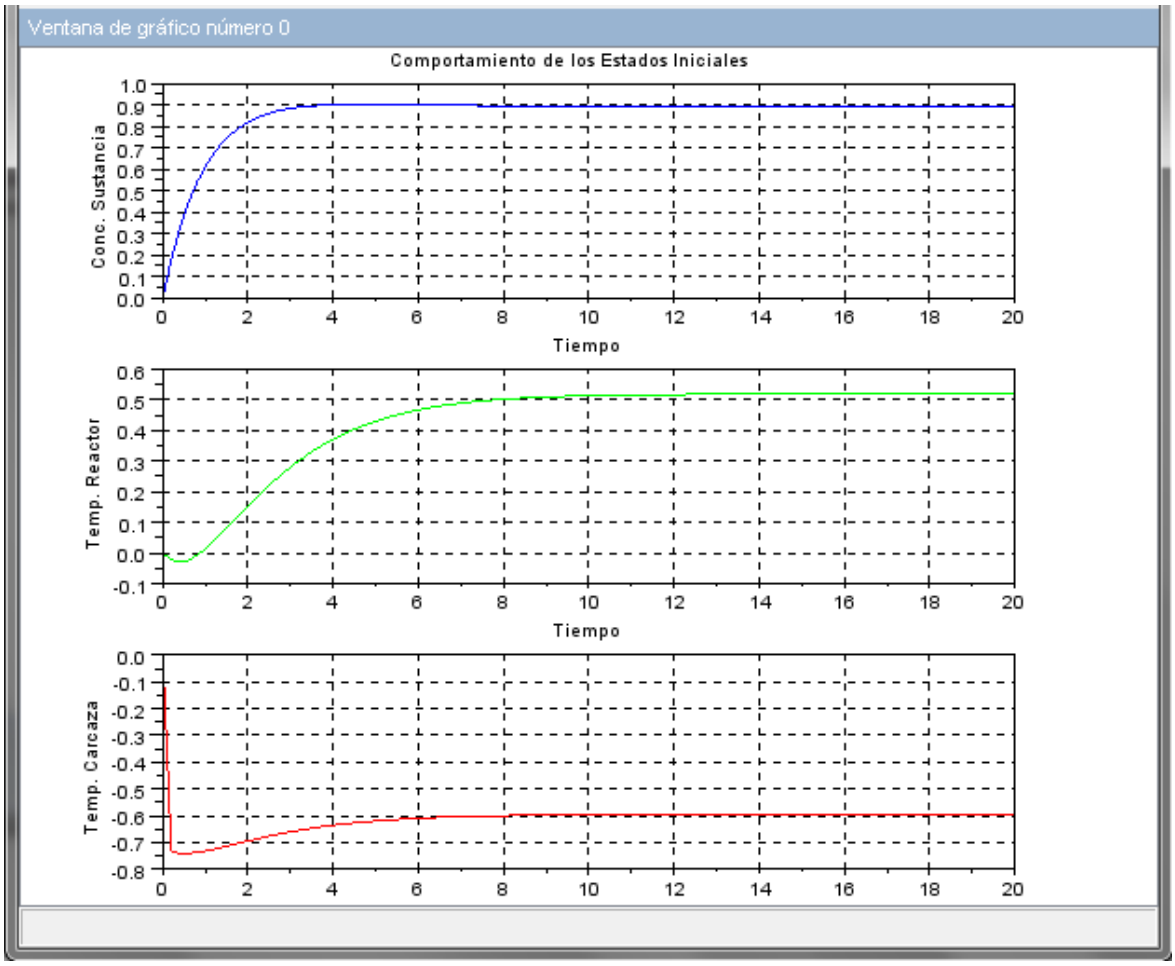


Figura 3-1 - Comportamiento del sistema para condiciones iniciales iguales a 0

De la gráfica se pueden adquirir ciertas observaciones:

1. En la gráfica de Concentración de sustancia, se puede observar que el sistema tardó aproximadamente unos **3.5 segundos** en buscar su punto de equilibrio, equivalente a 0.8933,
2. en las gráficas de las temperaturas del Reactor y la carcasa, se puede evidenciar que el sistema tarda aproximadamente **8 segundos** en llegar a su punto de equilibrio, equivalentes a 0.5193 y -0.5950, respectivamente.

Continuando con el análisis de la Figura 3-2, para condiciones de entrada iguales a 1

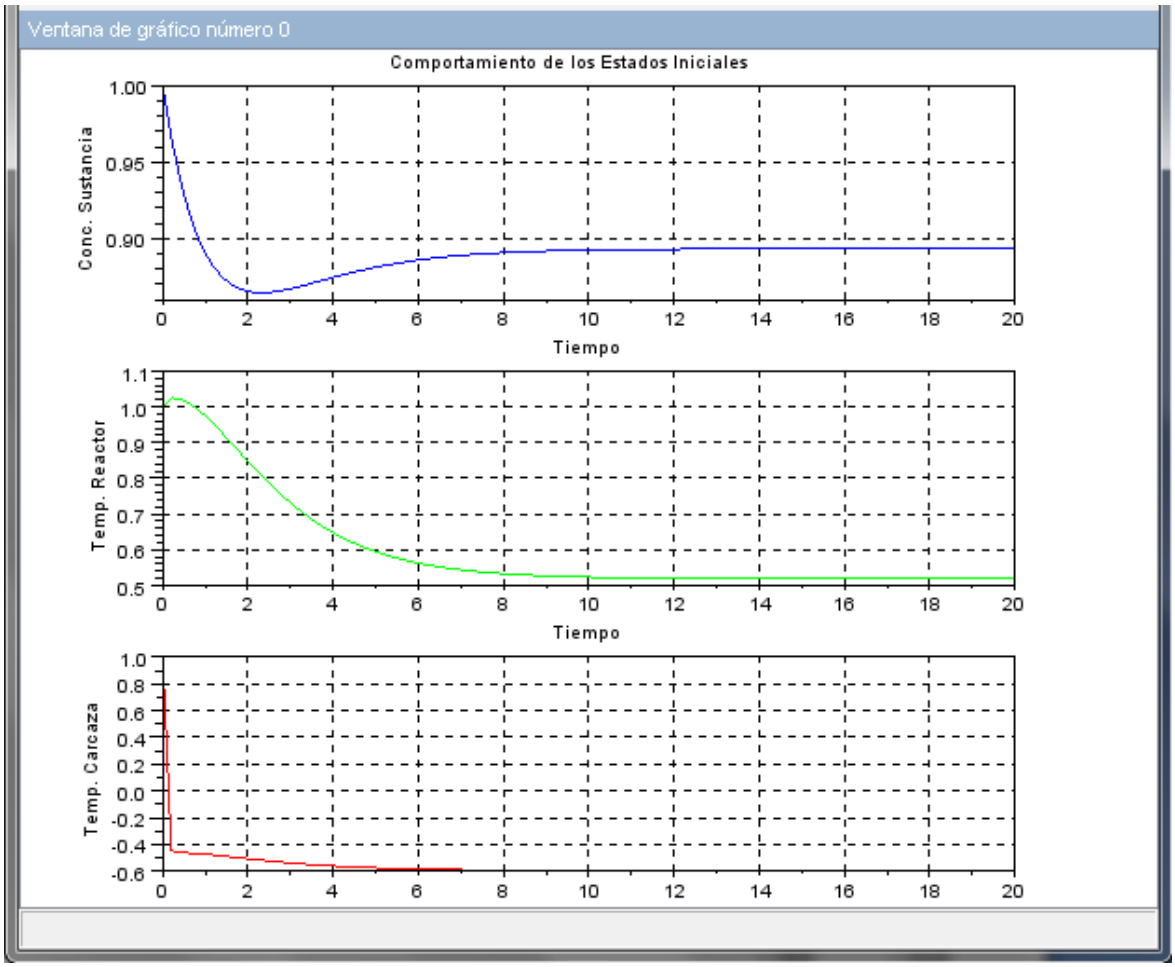


Figura 3-2 - Comportamiento del sistema para condiciones iniciales iguales a 1

Analizando las gráficas se puede concluir que:

1. En la gráfica para condiciones de entrada iguales a 1, la concentración de sustancia tarda aproximadamente **8 segundos** en estabilizarse. Su punto es equivalente a 0.8933,
2. En el caso de la temperatura del reactor, el sistema tarda aproximadamente ***9*** **segundos** en llegar al punto de equilibrio equivalente a 0.5193,
3. Para la gráfica de la temperatura de la carcasa, el sistema tarda en llegar a su punto de estabilización **5 segundos**, equivalente a -0.5950.

En el caso donde las condiciones de entrada al sistema son equivalentes a 2, Figura 3-3, se puede analizar qué:

1. En la gráfica de Concentración de sustancia, el sistema tiene una caída casi que llegando a 0 y luego busca su punto de estabilización equivalente a 0.1890 en aproximadamente **5 segundos**,

2. Para la gráfica de la temperatura del reactor, el sistema tiene un pico de 14.11 en menos de 1 segundo y luego tarda casi **5 segundos** en llegar a su punto estable para esta entrada, en este caso su punto de estabilización es equivalente a 5.1370,

3. Mientras que en la gráfica de la temperatura de la carcasa, se puede observar que hay una caída de temperatura equivalente a 0.1131 en aprox. 0.2 segundos y luego un pico de 2.6595 en aprox. 0.5 segundos antes de llegar a su punto estable equivalente a 0.6360 en aproximadamente **5 segundos.**

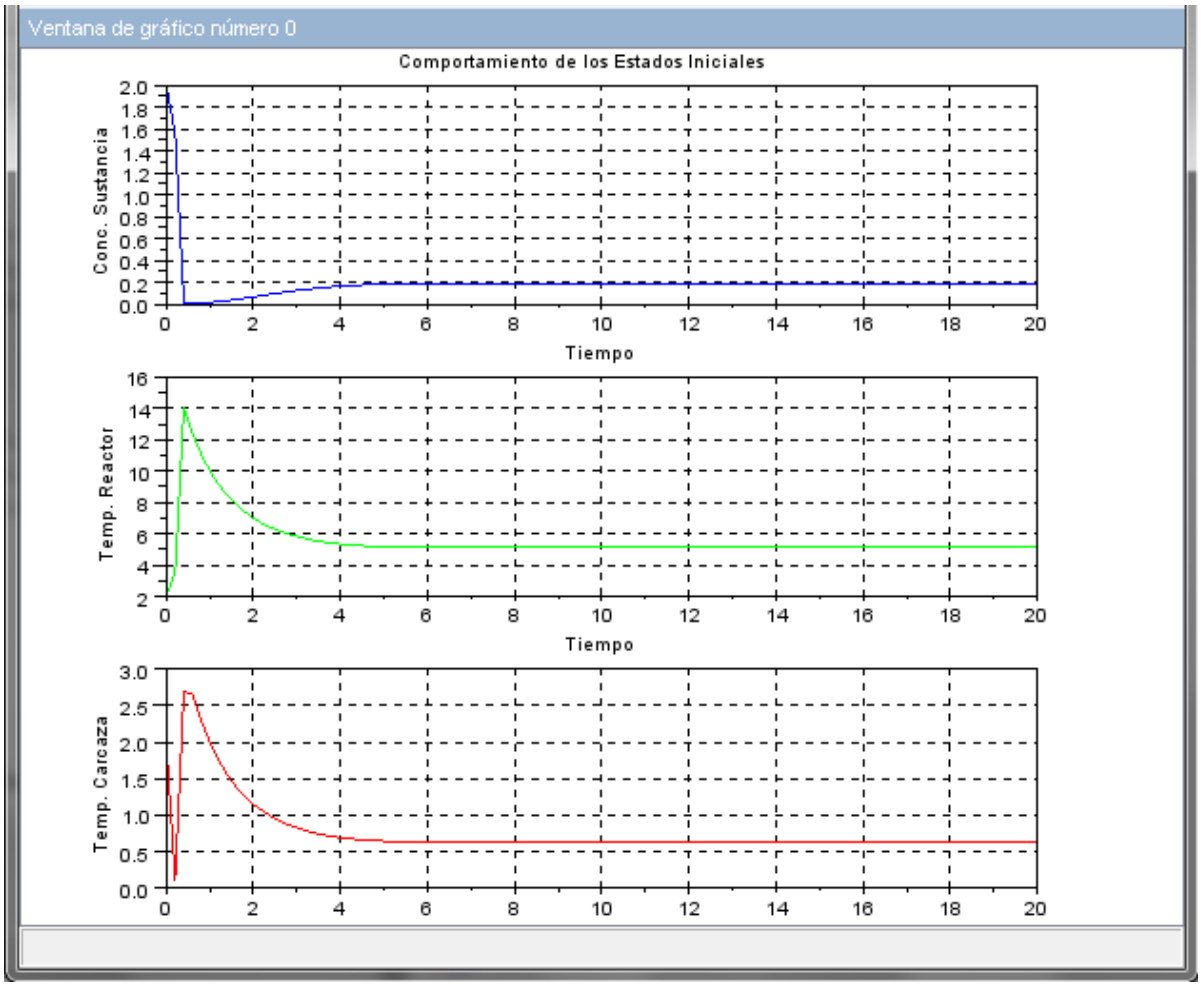


Figura 3-3 - Comportamiento del sistema para condiciones iniciales iguales a 2

A partir de este punto el sistema empieza a tener un comportamiento similar, para distintas condiciones de entrada, véase Figura 3-4 a la Figura 3-6.

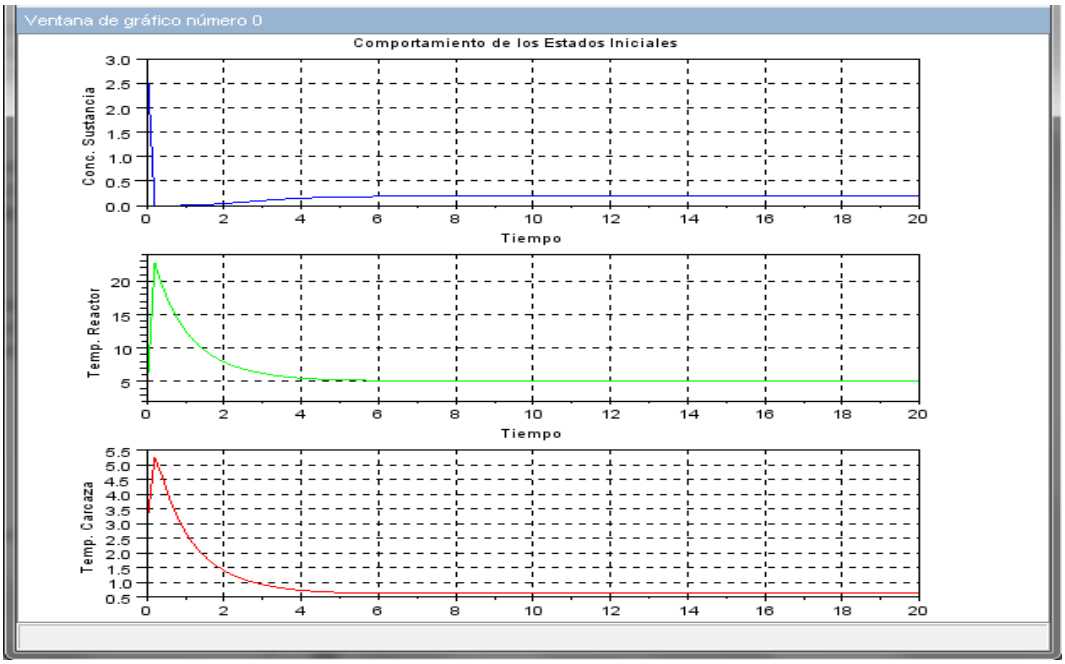


Figura 3-4 - Comportamiento del sistema para condiciones iniciales iguales a 3.

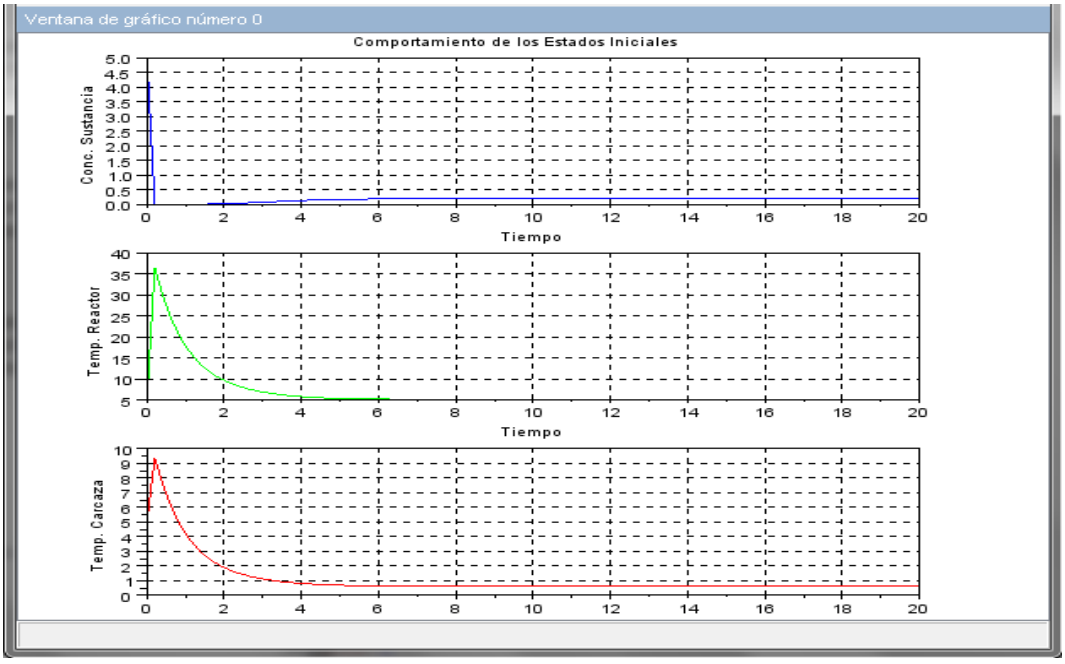


Figura 3-5 - Comportamiento del sistema para condiciones iniciales iguales a 5.

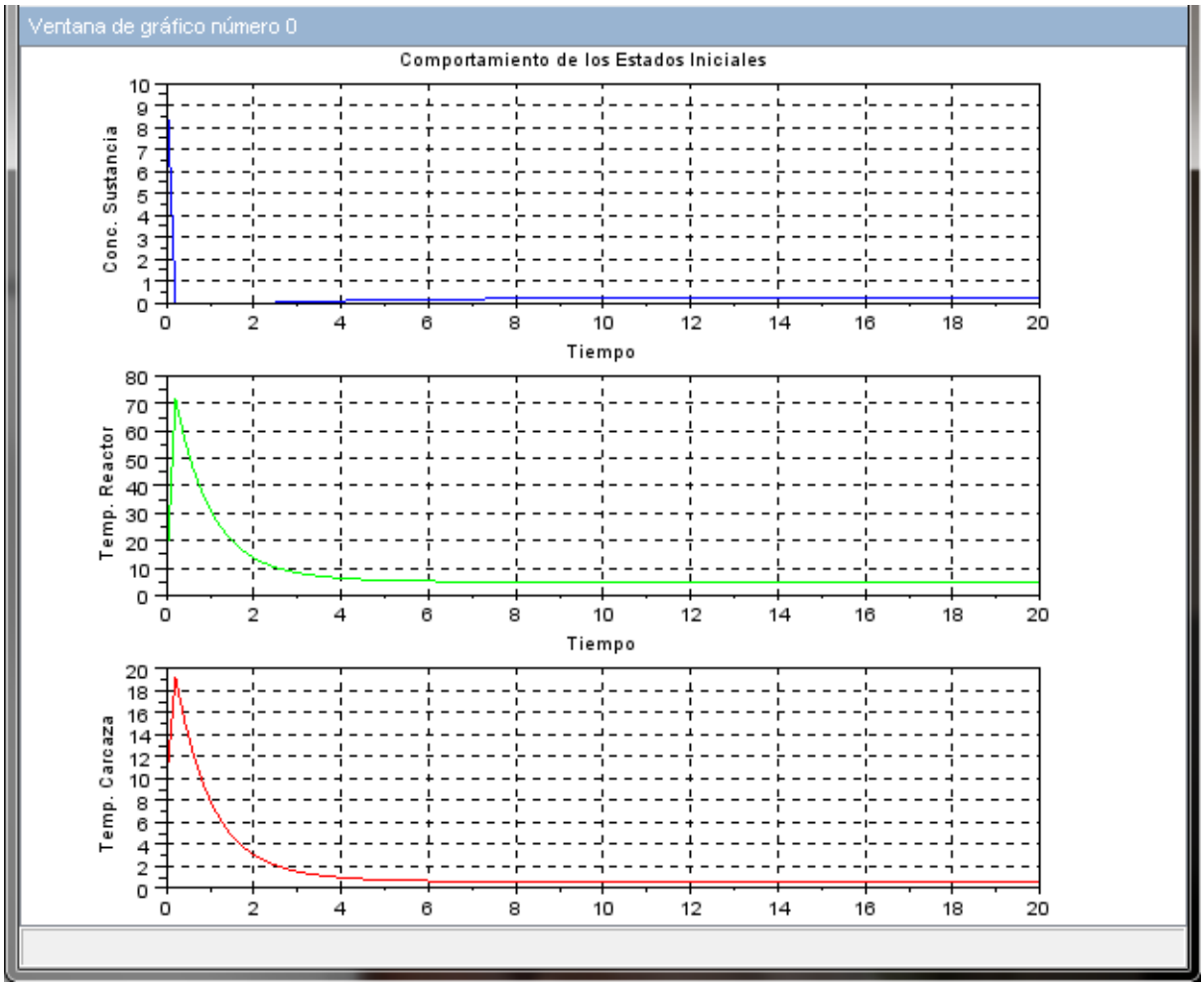


Figura 3-6 - Comportamiento del sistema para condiciones iniciales iguales a 10

Observando las Figuras 3-4, 3-5 y 3-6 se puede concluir que el sistema a partir de una entrada equivalente a 3 o mayor, repite su comportamiento, teniendo picos en las entradas y luego estabilizándose en el tiempo.

3.3. REPRESENTACIÓN DEL SISTEMA DE UN CSTR MEDIANTE XCOS

En la Figura 3-7 se describe el diagrama de bloques de XCOS en el cual se relacionan los modelos del reactor seleccionado descrito por las ecuaciones (1), (2) y (3) del capítulo 2. Los bloques identificados como Scope, se utilizan para observar el comportamiento del sistema en el tiempo mediante una gráfica, similar a un osciloscopio, se adecuó un bus de datos para verificar los tres comportamientos en una sola figura.

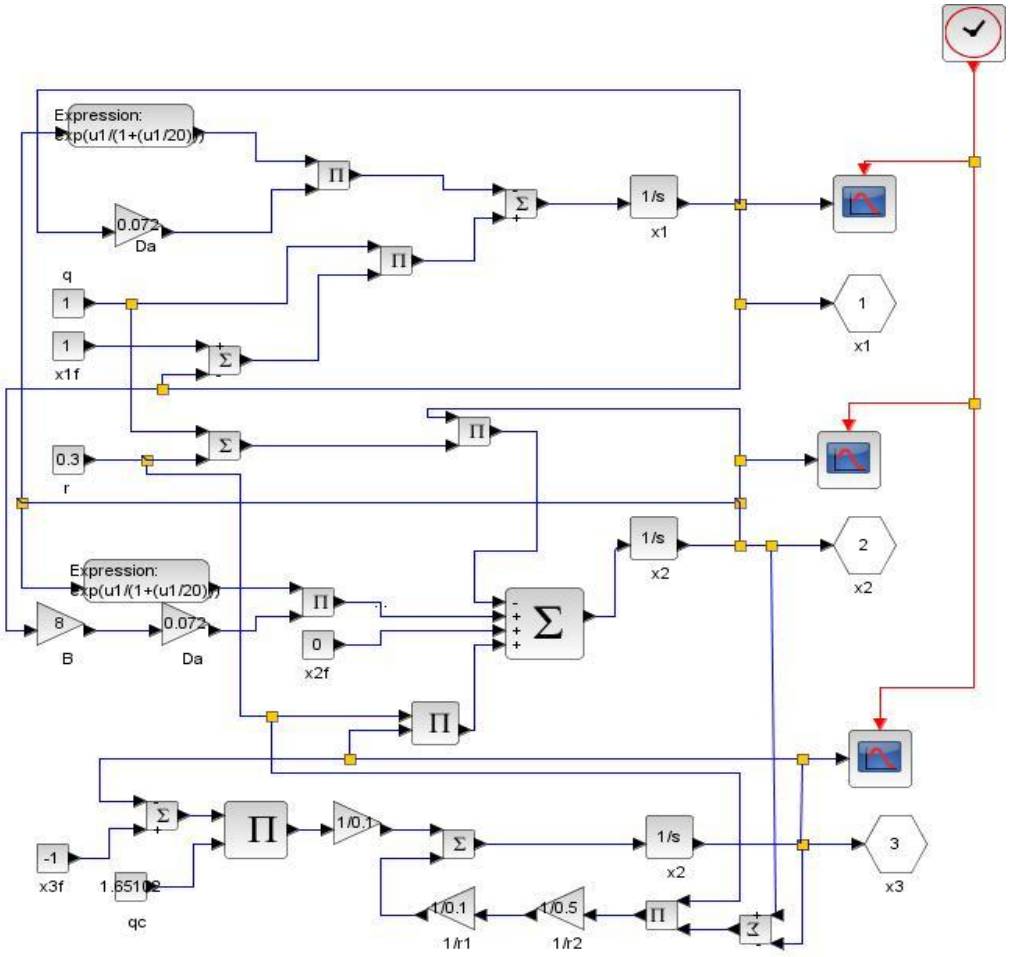


Figura 3-7 – Diagrama de bloques del sistema en XCOS

Al realizar las simulaciones con los diagramas de bloques de Scilab para distintas condiciones de entrada al sistema se obtienen los siguientes resultados gráficos.

Para condiciones de entrada equivalente a 0 se obtiene, véase Figura 3-8.

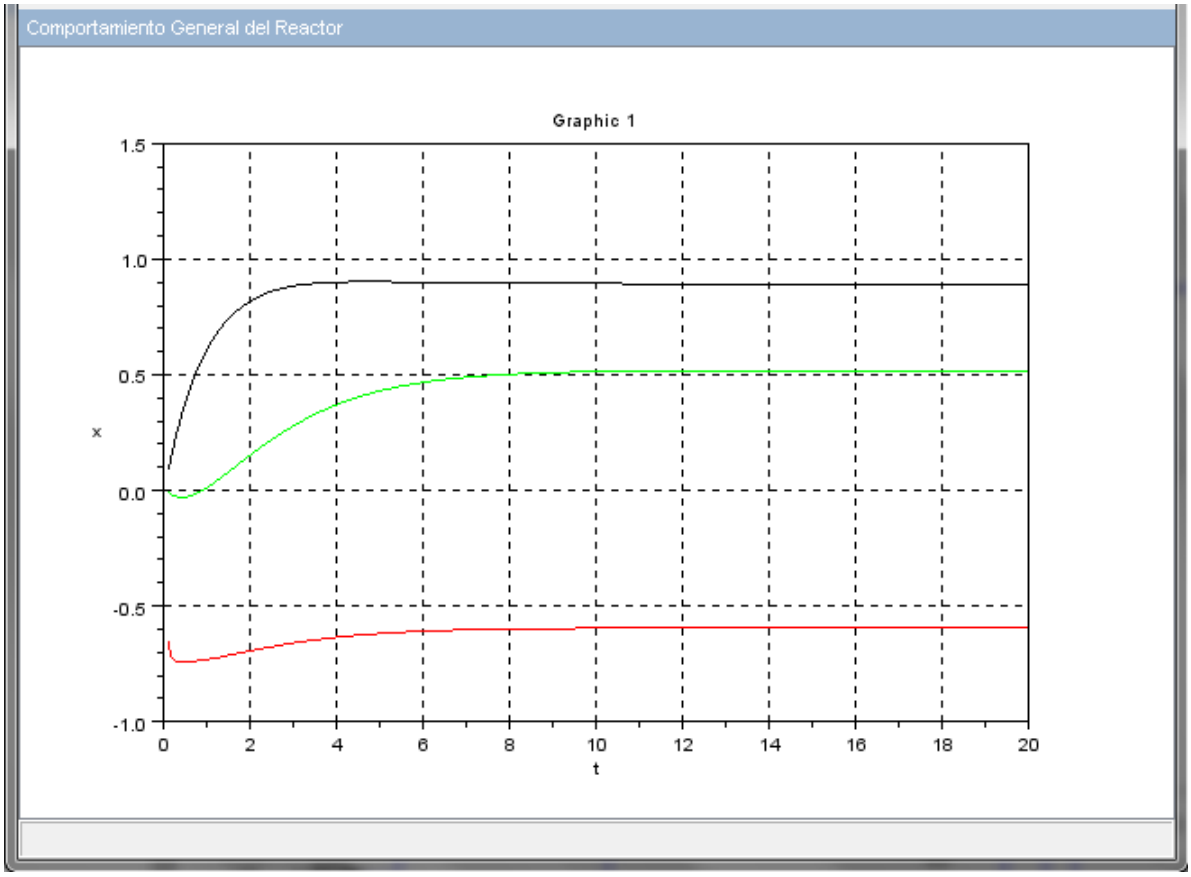


Figura 3-8 - Comportamiento del sistema para entrada igual a 0

En el grafico anterior se observa que el sistema llega a sus puntos de equilibrio aproximadamente 8 segundos después de haber empezado. El sistema encuentra el equilibrio en puntos equivalentes a 0.893 para la concentración de la sustancia en el reactor, 0.5192 para la temperatura del reactor y -0.5956 para la temperatura de la carcasa.

Los colores de las curvas que representan los distintos estados están descritos a continuación:

1. La curva de color negro representa el estado de la Composición de la sustancia.
2. La curva de color verde representa la Temperatura del Reactor.
3. La curva de color rojo representa la Temperatura de la Carcasa del reactor.

Si mediante el diagrama de bloques de XCOS se ingresa una condición de entrada equivalente a 1, los puntos de equilibrio pueden ser observados en la Figura 3-9.

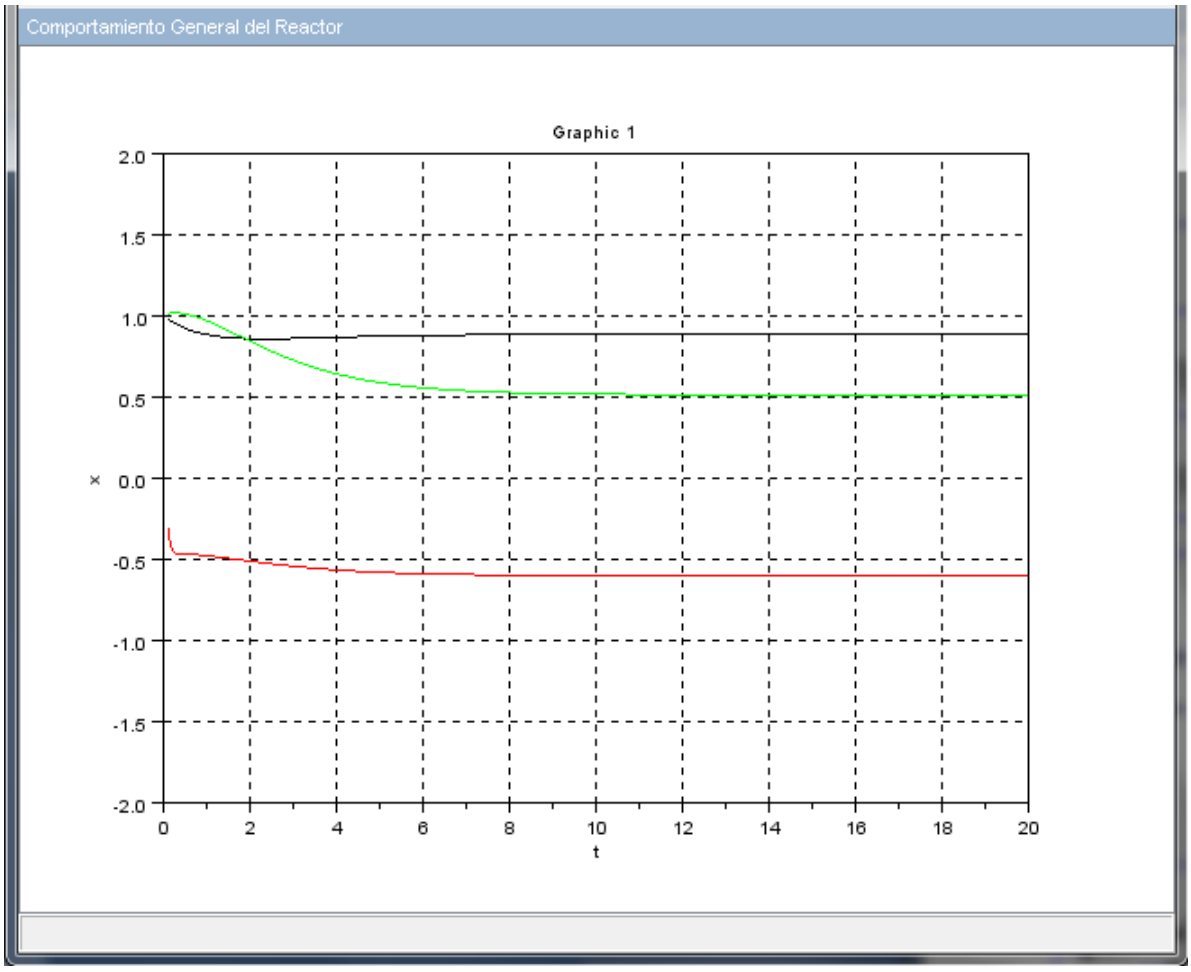


Figura 3-9 - Comportamiento del sistema para entrada igual a 1

Para los datos de entrada mayores a uno, se pudo determinar con anticipación que el sistema se comporta de igual forma, teniendo picos en instantes posteriores al arranque y luego estabilizándose en un lapso no mayor a 10 segundos

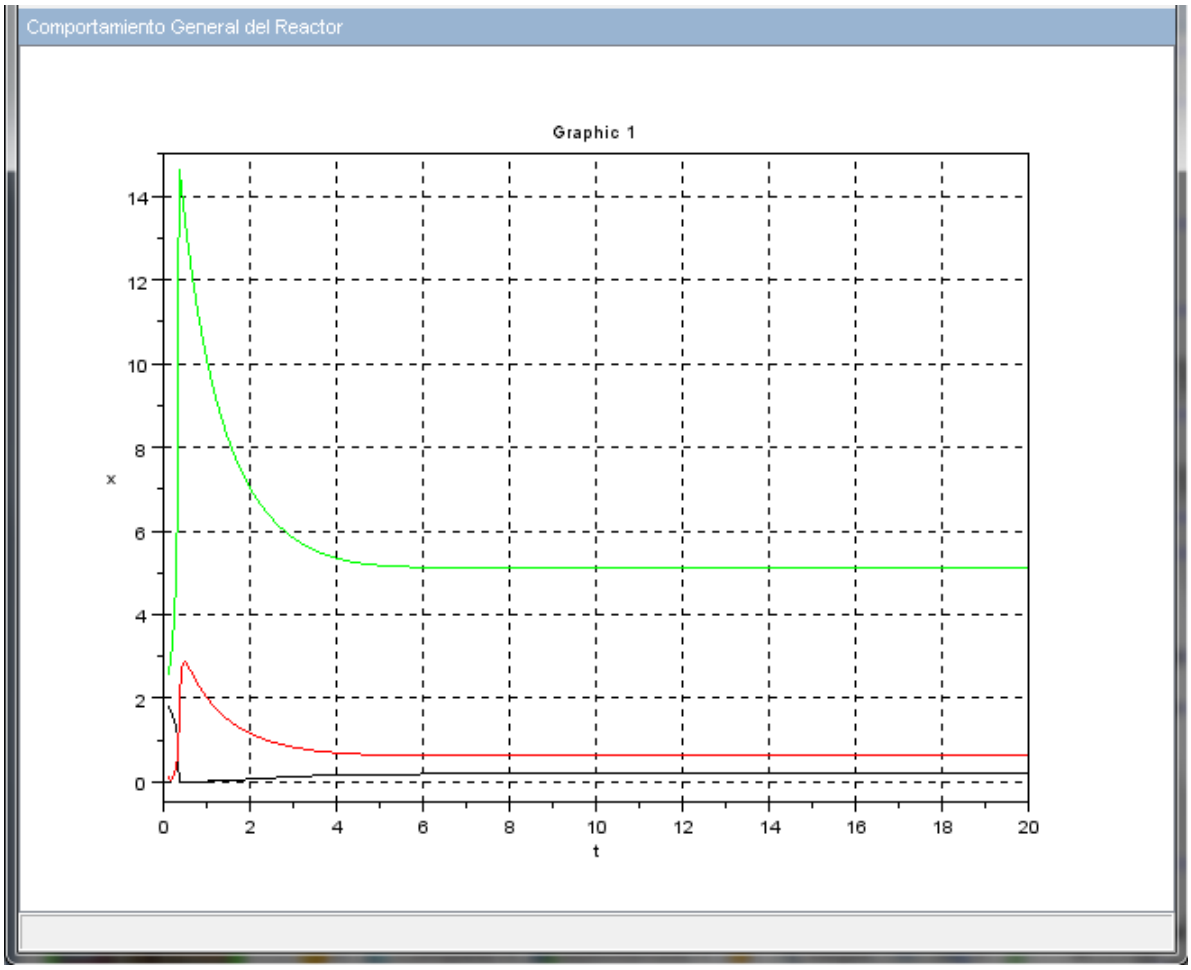


Figura 3-10 - Comportamiento del sistema para entrada igual a 5

Si se observa la Figura 3-10, se puede concluir que para condiciones iniciales equivalentes a 5, el sistema adquiere un comportamiento distinto a como lo hace para condiciones iniciales de amplitud 0 y 1, ya que al empezar la reacción, la Temperatura del reactor tiene un aumenta indiscriminadamente y luego se disminuye hasta su punto de equilibrio.

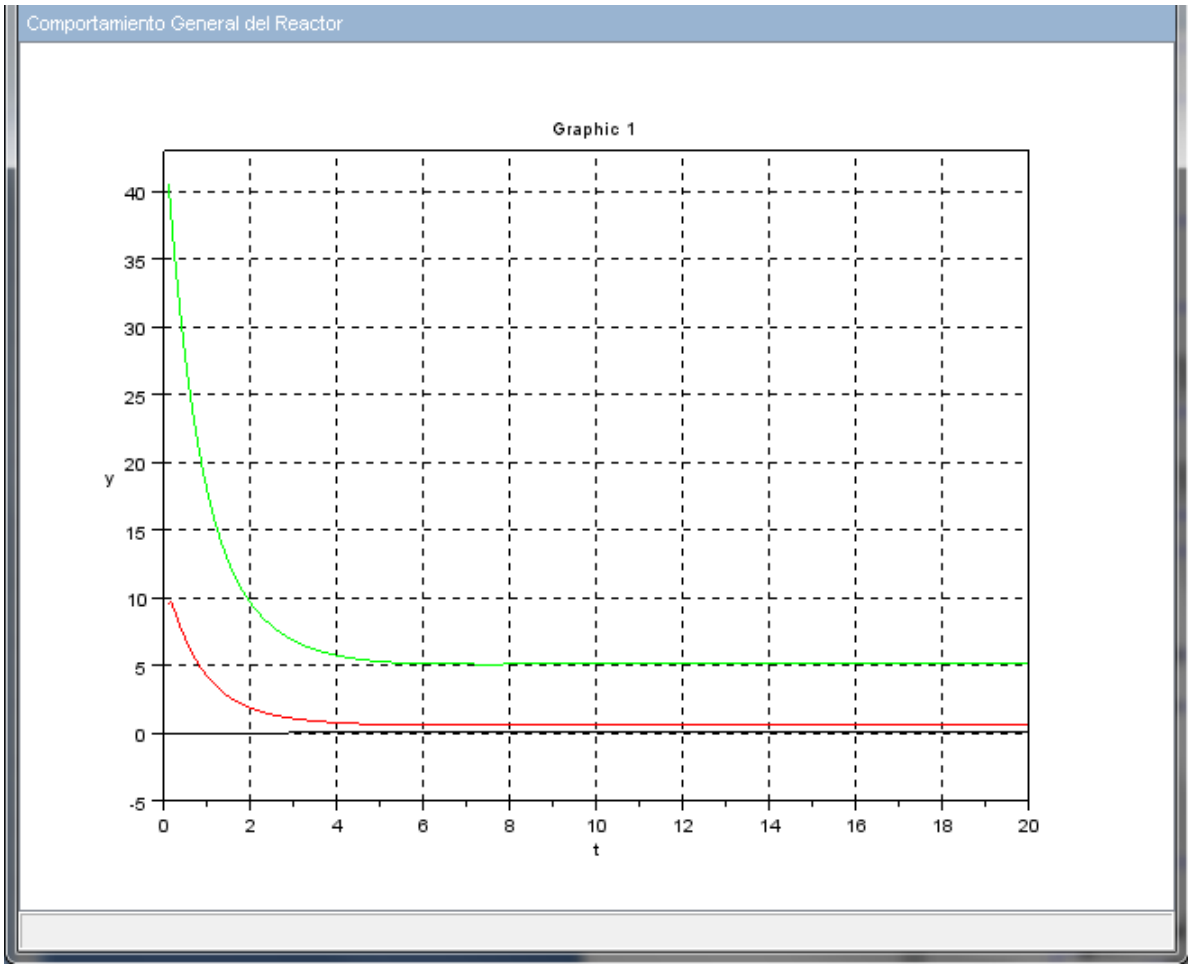


Figura 3-11 - Comportamiento del sistema para entrada igual a 10

De esta forma también se puede observar que en las figuras inmediatamente anteriores, el sistema, para unas condiciones iniciales de amplitud 2, 5 y 10, las temperaturas tienen unos picos bastante pronunciados y luego alcanzan su punto estable, concluyendo así, como se dijo en el capitulo anterior, que el sistema debe adecuarse para tener unas condiciones de entrada equivalentes a 0 y 1 para no tener picos en las entradas sino un sistema que se estabilice de forma más sutil.

4. ANALISIS DE RESULTADOS

En este capítulo se presenta el análisis de resultados con las dos aplicaciones. Para hacer este análisis, se establecieron los siguientes criterios:

- Tamaño de los archivos generados.
- Requerimientos mínimos para la instalación.
- Tiempos de ejecución según método de integración.
- Error en los resultados.
- Comparativa en métodos de integración utilizado por ambas herramientas.

4.1. TAMAÑO DE LOS ARCHIVOS

Empezando desde el nivel más básico, se estudió un esquema comparativo entre los tamaños de los archivos de las herramientas utilizadas y los tamaños de los algoritmos desarrollados.

Tabla 4.1 Comparativa de tamaños de archivos

	Matlab 2009a (versión 7.8.0 – 64Bits)	**Scilab (versión 5.3.3 – 32 Bits)**
Archivo Instalación	680MB	696MB
Algoritmo cstr	4.43KB	4.44KB
Algoritmo cstr_ss	4.45KB	4.53KB
Algoritmo cstr_fsolve	704 bytes	704 bytes
Algoritmo cstr_ode	1.80KB	1.17KB
Diagrama de Bloques	44.7KB	190KB
Imagen	34.8KB	51.3KB

De la Tabla 4.1 se puede destacar que:

1. Los archivos para la instalación, las imágenes y los algoritmos desarrollados en ambas herramientas tienen tamaños de archivos muy similares.
2. Se nota una diferencia en el tamaño de los Diagramas de bloques implementados, aunque el tamaño en Scilab sea mayor al de Matlab, se puede tomar como irrelevante ya que no está en el orden de los MB.

4.2. REQUERIMIENTOS MÍNIMOS DE INSTALACIÓN

Adicional, se tuvo en cuenta los requerimientos mínimos de hardware Y OS[7] que tienen ambas herramientas computacionales, con las que también, dependiendo del tipo de aplicación en cuanto a toolboxes, el usuario puede decidir cuál de las dos es la adecuada.

Requerimientos Scilab:

- Procesador: Intel Pentium IV o mejor.
- Espacio en Disco: 640MB.
- Memoria (RAM): 1GB o mayor.
- Video: Compatible con OpenGL 1.2.
- Resolución de pantalla: 1024 x 768 al menos.
- Sistema Operativo: Compatible con WINXP o WIN7, GNU/Linux, Mac OS X.

[7] Operative System= Sistema Operativo

Requerimientos Matlab:

- Procesador: Intel Pentium IV o mejor.
- Espacio en Disco: 640MB.
- Memoria (RAM): 512MB, 1GB recomendado.
- Video: Compatible con OpenGL 1.2.
- Resolución de pantalla: 1024 x 768 al menos.
- Sistema Operativo: Compatible con WINXP o WIN7, GNU/Linux, Mac OS X, Solaris.

Si se realiza una observación a los datos anteriores se puede comprobar, que tanto los tamaños de los archivos como los requerimientos mínimos para su funcionamiento son equivalentes.

4.3. TIEMPOS DE EJECUCIÓN

En este ítem, se presentan los tiempos de ejecución de cada una de las aplicaciones en la simulación para diferentes métodos de integración numérica. De acuerdo con esto, los resultados obtenidos se reseñan a continuación.

Para determinar los tiempos mostrados en la Tabla 4.2, se tomó como muestra un estado inicial de amplitud 1, para la Concentración de la Sustancia, para la Temperatura del Reactor y para la Temperatura de la Carcasa.

En los resultados arrojados al utilizar los distintos métodos de integración, se observa que con el método de Bogacki-Shampine el tiempo de ejecución de Matlab es ligeramente superior a Scilab, mientras que en los métodos de integración de Adams, Dormand-Prince y BDF continúa la tendencia de Scilab al arrojar los resultados en menos tiempo que Matlab.

Tabla 4.2 Métodos de integración y tiempos de ejecución de las aplicaciones**.**

Métodos de integración	Tiempos de ejecución	
	Matlab	Scilab
Adams	0.537	0.025
Bogacki-Shampine	0.370	0.776
Dormand-Prince	0.412	0.062
BDF[8]	0.440	0.025

4.4. ERROR EN LOS RESULTADOS

Al observar los datos arrojados por el sistema cuando se utilizó el comando $fsolve$, se nota que hay datos que son distintos para Matlab y para Scilab, véase Tabla 2.2 y Tabla 3.2, esto se debe a que ambos utilizan distintos métodos para hallar las raíces o en este caso los puntos de estabilidad del sistema.

Scilab utiliza el método DFP (Davidon-Fletcher-Powell), el cual consiste en generar aproximaciones sucesivas de la inversa del Hessiano de la función f. Matlab utiliza el método de cuasi-newton o método de Broyden el cual consiste en aproximar la matriz jacobiana en cada iteración a partir de la matriz tangente utilizada en la iteración anterior. La primera iteración del método se realiza como en el método de Newton pero a partir de la segunda iteración la matriz jacobiana es sustituida por otra matriz.

[8] Backward Differentiation Formula = Formula de Derivación hacia Atrás.

4.5. COMPARATIVA EN MÉTODOS DE INTEGRACIÓN UTILIZADO POR AMBAS HERRAMIENTAS.

Teniendo en cuenta los métodos de integración aplicados por las técnicas utilizadas en los comandos $ode45$ de Matlab y ode de Scilab, se estableció un criterio de comparación con el fin de conocer el margen de error que hay en los métodos utilizados por ambas herramientas.

En la Tabla 4.3, se muestran la comparación de los métodos utilizados, arrojando los siguientes resultados:

Tabla 4.3 Comparativa de datos vs tiempo para ambas herramientas

Datos para rango de tiempo entre 1 segundo y 15 Segundos							
Tiempo	Matlab (ode45)			Scilab (ode)			**Tiempo**
Segundos	x_1	x_2	x_3	x_1	x_2	x_3	**Segundos**
1,0101	0,8897	0,9737	-0,4726	0,8896	0,9737	-0,4726	**1,0101**
2,0202	0,8653	0,8451	-0,5069	0,8653	0,8451	-0,5066	**2,0202**
3,0303	0,8674	0,7282	-0,5382	0,8674	0,7281	-0,5381	**3,0303**
4,0404	0,8749	0,6454	-0,5610	0,8748	0,6453	-0,5606	**4,0404**
5,0505	0,8815	0,5928	0,5749	0,8815	0,5928	-0,5749	**5,0505**
6,0606	0,8862	0,5613	-0,5836	0,8861	0,5613	-0,5835	**6,0606**
7,0707	0,8891	0,5430	-0,5882	0,8891	0,5429	-0,5885	**7,0707**
8,0808	0,8909	0,5321	-0,5913	0,8909	0,5325	-0,5914	**8,0808**
9,0909	0,8919	0,5267	-0,5930	0,8919	0,5266	-0,5930	**9,0909**
10,1010	0,8925	0,5234	-0,5941	0,8925	0,5233	-0,5939	**10,1010**
11,1111	0,8929	0,5215	-0,5945	0,8928	0,5215	-0,5944	**11,1111**
12,1212	0,8931	0,5205	-0,5949	0,8930	0,5205	-0,5947	**12,1212**
13,1313	0,8932	0,5199	-0,5944	0,8931	0,5199	-0,5948	**13,1313**
14,1414	0,8932	0,5196	-0,595	0,8932	0,5196	-0,5949	**14,1414**
15,1515	0,8933	0,5195	-0,5947	0,8932	0,5194	-0,5949	**15,1515**

De la tabla anterior se puede destacar que:

- Se tomaron condiciones iniciales del sistema con amplitud 1 para realizar la experimentación.

- El margen de error de los métodos de integración utilizados por las herramientas son nulos, teniendo en cuenta que para la comparación se utilizó el método Runge-Kutta (Dormand-Prince) en los dos programas y se obtuvo un resultado el cual muestra que para este tipo de modelos la respuesta de Scilab comparada con la de Matlab son idénticas.

- Se escogieron puntos aleatorios para realizar la comparación, corriendo los algoritmos diseñados se puede tener una mayor certeza de lo previamente mostrado, ya que se muestran todos los puntos del sistema en función del tiempo.

5. CONCLUSIONES

- ✓ Se desarrolló un estudio comparativo entre dos aplicaciones, una aplicación comercial (Matlab) y una aplicación de Open Source (Scilab), aplicado a un reactor tipo tanque agitado, más conocido como CSTR (Continuous System Tank Reactor) por sus siglas en inglés.

- ✓ Las dos aplicaciones consumen recursos computacionales similares en la instalación de sus componentes y librerías, por lo que se puede afirmar que bajo este criterio, son comparables y no existe una marcada diferencia entre uno y otro.

- ✓ En cuanto al tamaño de los archivos generados en la aplicación de los algoritmos, a diferencia de los diagramas de bloques, donde Matlab genera archivos con una capacidad de almacenamiento en kBytes de aproximadamente una tercera parte de los archivos generados por Scilab, los demás archivos tienen un tamaño equivalente.

- ✓ La sintaxis de Matlab y Scilab es aproximadamente igual en un 80%, al revisar los comandos de funciones y demás aplicaciones básicas de la herramienta.

- ✓ Al comparar los tiempos de ejecución de las dos aplicaciones de acuerdo con la selección de mismo método de integración numérica, con excepción del método de Bogacki-Shampine, Scilab reduce en valores cercanos a una décima parte del tiempo de ejecución que utiliza la misma aplicación en Matlab en los experimentos de simulación desarrollados.

- ✓ Los valores calculados a partir de los experimentos de simulación en cada una de las dos aplicaciones (Matlab y Scilab), son aproximadamente iguales, lo cual sugiere un cierto grado de confiabilidad para el uso de Scilab como herramienta de apoyo en desarrollos y procesos de investigación.

Al realizar la comparación entre los datos expresados versus instantes de tiempos específicos para las herramientas Matlab y Scilab, se nota que no existe una variación significativa entre los métodos de integración utilizados en las dos aplicaciones.

6. BIBLIOGRAFÍA

Acuña, O., & Gutierrez, D. (2007). Aspectos computacionales de algunos métodos de ajuste paramétrico de modelos aplicados a ciertos procesos de polimerización. *VII Congreso de la Asociación Colombiana de Automática.* ACA.

Campbell, S., Chancelier, J.-P., & Nikouhkan, R. (2008). *Modeling and Simulation in Scilab/Scicos.* Springer.

Caro, A., & Sepúlveda, C. *Fundamentos de Scilab y Aplicaciones.* 2004.

Fritzon, P. (s.f.). Introduction to Modeling and Simulation od Technical and Physical systems with Modelica. *Wiley and IEEE Press* .

Guerra, F., & Struck, A. (2008). *Análisis Dinámico de un CSTR.* Mexico D.F.

Jeppsson, U. e. *Benchmark simulation model No. 2: General protocol and exploratory case studies.*

Klee, H., & Allen, R. (2011). *Simulation of Dynamic Systems with Matlab and Simulink.* CRC Press.

Mathews, J., & Fink, K. (1999). *Numerical Methods using Matlab.* Prentice Hall.

Mathwoks. (s.f.). Recuperado el 28 de Mayo de 2012, de http://www.mathworks.com/support/product/product.html?product=ML

Oliveira, D., & Moreira, E. (2005). A Chemical Reactor Benchmark for Parallel Adaptive Control Using Feedfoward Neural Networks. *IEEE* .

Pandi, T., & Rajendran, L. *Analytical solution of Non-Linear differential equation in Oscilatory chemical reactions.*

Russo, L., & Bequette, W. (1992). *CSTR Performance Limitations Due to Cooling Jacket Dynamics - Open and Closed-Loop Considerations.* Miami Beach.

Scilab. (s.f.). Recuperado el 27 de Mayo de 2012, de http://www.scilab.org/products/xcos

Trejos, V., Fontalvo, J., & Gómez, M. (2009). Descripción matemática y análisis de estabilidad de procesos fermentativos. *Dyna , 76* (158), 111-121.

Urroz, G. (2001). *Ordinary Differential Equations with SCILAB.*

LISTA DE ANEXOS

ANEXO 1 – Algoritmo cstr.m

```
%***************************** Algoritmo cstr.m *****************************
% Este archivo contiene el código cstr.m para la obtención de los Condiciones de
estabilidad de un % CSTR en el tiempo. El modelo y los parámetros fueron tomados de "A
Chemical Reactor % Benchmark for Parallel Adaptive Control Using Feedfoward
Neural Networks", Oliveira et al.
% Elaborado por Remberto J. Cuadro Alvear. Mayo de 2012
% Universidad Tecnológica de Bolívar - Cartagena
%**********************************************************************

function dx = cstr(t,x)

%----------------------------------------------------------------
% Definición de las condiciones iniciales.
%----------------------------------------------------------------

global dx                   % Se toman datos para las diferentes condiciones del reactor.
x1 = x(1);                  % x1 Concentración de sustancia en el reactor
x2 = x(2);                  % x2 Temperatura del reactor
x3 = x(3);                  % x3 Temperatura de la carcasa del reactor

%----------------------------------------------------------------
% Definición de los parámetros del reactor.
%----------------------------------------------------------------

x1f = 1;                    % Concentración inicial de la sustancia en el reactor
x2f = 0;                    % Temperatura inicial del reactor.
x3f = -1;                   % Temperatura inicial de la carcasa del reactor.
q   = 1;                    % Flujo de entrada en el reactor.
qc  = 1.65102;              % Flujo de entrada en la carcasa del reactor.
Da  = 0.072;                % Factor pre-exponencial no térmico. Número de Damköhler.
r   = 20;                   % Energía de activación.
B   = 8;                    % Calor de la reacción.
s   = 0.3;                  % Coeficiente de transferencia de calor.
s1  = 0.1;                  % Relación de volumen del reactor.
s2  = 0.5;                  % Relación de la capacidad de la densidad del calor de la
                            reacción.
```

```
%-----------------------------------------------------------------------
% Definición de las ecuaciones de estado.
%-----------------------------------------------------------------------

dx(1)  = q.*(x1f - x1) - x1*Da*exp(x2/(1+(x2/r)));                    % Ecuacion del balance
de masa de la sustancia dentro del reactor.
dx(2)  = B.*Da*x1*exp(x2/(1+(x2/r))) - (q + s)*x2 + s*x3 + x2f;       % Ecuación del balance
de energía dentro del reactor.
dx(3)  = (qc*(x3f - x3)/s1) + s*(x2 - x3)/(s1*s2);                    % Ecuación del balance
de energía dentro de la chaqueta del reactor.

end
```

ANEXO 2 – Algoritmo cstr_ode.m

```
%************************** Algoritmo cstr_ode.m  ************************
% Este archivo llamado cstr_ode.m para la obtención de los Condiciones de estabilidad de un CSTR % en el tiempo. El usuario ingresa los datos en un ambiente interactivo y recibe la solución del      % sistema.
% Elaborado por Remberto J. Cuadro Alvear. Mayo de 2012
% Universidad Tecnológica de Bolívar - Cartagena
%************************************************************************

clear all
clc
global dx

% ------------------------------------------------
%  Entrada de Datos
% ------------------------------------------------

display('Tiempo donde se va a realizar la muestra')
t0    = input('Tiempo inicial para la muestra.          t0: ');
tf    = input('Tiempo para terminar la muestra.       tf: ');
[time]= linspace(t0,tf);
clc
display('Condiciones iniciales para solución del sistema')
x1    = input('Condición inicial de Entrada de la sustancia.          x1: ');
x2    = input('Condición inicial de Temperatura del reactor.          x2: ');
x3    = input('Condición inicial de Temperatura de la carcasa.        x3: ');
x     = [x1;x2;x3];
clc
display('Ingrese los datos de las condiciones del reactor')
dx(1) = input('Ingrese la condición de la Concentración de Sustancia:      dx1: ');
dx(2) = input('Ingrese la condición de la Temperatura del Reactor          dx2: ');
dx(3) = input('Ingrese la condición de la Temperatura de la Carcasa        dx3: ');
dx    = [dx(1);dx(2);dx(3)];
clc
```

```
% ------------------------------------------------
% Solución del Sistema de Ecuaciones Diferenciales
% ------------------------------------------------

display('Calculando datos, por favor espere...')
tic
[t,x]     =ode45(@cstr,time,x);
T         =toc;
clc
display('Los puntos de equilibrio del CSTR son:');
disp('    Tiempo      x1       x2       x3')
disp('    _____________________________________')
disp([t,x])
x
display('El tiempo para la solución de la ecuación fue: ');
T
display('Presione una tecla para continuar');
pause

% ------------------------------------------------
% Gráficas de las variables
% ------------------------------------------------

figure
subplot(3,1,1),plot(t,x(:,1),'b'),
ylabel('Conc. Sustancia')
xlabel('Tiempo')
title('Comportamiento del sistema')
grid on
subplot(3,1,2),plot(t,x(:,2),'g'),
ylabel('Temp. Reactor')
xlabel('Tiempo')
grid on
subplot(3,1,3),plot(t,x(:,3),'r'),
ylabel('Temp. Carcaza')
xlabel('Tiempo')
grid on
```

ANEXO 3– Algoritmo cstr_ss.m

```
%************************** Algoritmo cstr_ss.m  **************************
% Este archivo contiene el código cstr_ss.m para la obtención de los Puntos de equilibrio
% en Estado Estacionario de un CSTR.
% El modelo y los parámetros fueron tomados de "A Chemical Reactor Benchmark for
% Parallel Adaptive Control Using Feedfoward Neural Networks", Oliveira et al.
% Elaborado por Remberto J. Cuadro Alvear. Mayo de 2012
% Universidad Tecnológica de Bolívar - Cartagena
%************************************************************************

function dx = cstr_ss(x)

%----------------------------------------------------------------
% Definición de las condiciones iniciales.
%----------------------------------------------------------------

dx = zeros(3,1);                % Se hace un vector de 0.
x1 = x(1);                      % x1 Concentración de sustancia en el reactor
x2 = x(2);                      % x2 Temperatura del reactor
x3 = x(3);                      % x3 Temperatura de la carcasa del reactor

%----------------------------------------------------------------
% Definición de los parámetros del reactor.
%----------------------------------------------------------------

x1f = 1;                        % Concentración inicial de la sustancia en el reactor
x2f = 0;                        % Temperatura inicial del reactor.
x3f = -1;                       % Temperatura inicial de la carcasa del reactor.
q   = 1;                        % Flujo de entrada en el reactor.
qc  = 1.65102;                  % Flujo de entrada en la carcasa del reactor.
Da  = 0.072                     % Número de Damköhler.
r   = 20;                       % Energía de activación.
B   = 8;                        % Calor de la reacción.
s   = 0.3;                      % Coeficiente de transferencia de calor.
s1  = 0.1;                      % Relación de volumen del reactor.
s2   = 0.5                      % Relación de la capacidad de la densidad del calor de la
reacción.
```

```
%-----------------------------------------------------------------
% Definición de las ecuaciones de estado.
%-----------------------------------------------------------------

dx(1)  = q.*(x1f - x1) - x1*Da*exp(x2/(1+(x2/r)));                    % Ecuación del
balance de masa de la sustancia dentro del reactor.
dx(2)  = B.*Da*x1*exp(x2/(1+(x2/r))) - (q + s)*x2 + s*x3 + x2f;       % Ecuación del
balance de energía dentro del reactor.
dx(3)  = (qc*(x3f - x3)/s1) + s*(x2 - x3)/(s1*s2);                    % Ecuación del
balance de energía dentro de la chaqueta del reactor.

end
```

ANEXO 4 – Algoritmo cstr_fsolve.m

```
%************************ Algoritmo cstr_fsolve.m  ***************************
% Este archivo llamado cstr_fsolve.m para la obtención de los Condiciones de
% estabilidad de un CSTR en el tiempo.
% El usuario ingresa los datos en un ambiente interactivo y recibe la solución del sistema.
% Elaborado por Remberto J. Cuadro Alvear. Mayo de 2012
% Universidad Tecnológica de Bolívar - Cartagena
%*******************************************************************************

clear all
clc

% ------------------------------------------------
%  Entrada de Datos
% ------------------------------------------------

x1      = input('Condición inicial de entrada de la sustancia.        x1: ');
x2      = input('Condición inicial de Temperatura del reactor.        x2: ');
x3      = input('Condición inicial de Temperatura de la carcasa.      x3: ');
x       =[x1;x2;x3];

% ------------------------------------------------
% Solución de la ecuación Diferencial
% ------------------------------------------------

display('Calculando datos, por favor espere...')
tic
[x]     =fsolve(@cstr_ss,x);
T       =toc;
display('Los puntos de equilibrio son')
x
display('Tiempo para solución de la ecuación')
T
```

ANEXO 5– Algoritmo cstr.cse

```
//*************************** Algoritmo cstr.cse  ***************************
// Este archivo contiene el código cstr.sce para la obtención de los Condiciones de
// estabilidad de un CSTR en el tiempo. El modelo y los parámetros fueron tomados de "A
// Chemical Reactor Benchmark for Parallel Adaptive Control Using Feedfoward Neural
// Networks", Oliveira et al.
// Elaborado por Remberto J. Cuadro Alvear. Mayo de 2012
// Universidad Tecnológica de Bolívar - Cartagena
//************************************************************************

function dx = cstr(t,x)
global dx

//---------------------------------------------------------------
// Definición de las condiciones iniciales.
//---------------------------------------------------------------

x1 = x(1);                  // x1 Concentración de sustancia en el reactor
x2 = x(2);                  // x2 Temperatura del reactor
x3 = x(3);                  // x3 Temperatura de la carcasa del reactor

//---------------------------------------------------------------
// Definición de los parámetros del reactor.
//---------------------------------------------------------------

x1f = 1;                    // Concentración inicial de la sustancia en el reactor
x2f = 0;                    // Temperatura inicial del reactor.
x3f = -1;                   // Temperatura inicial de la carcasa del reactor.
q   = 1;                       // Flujo de entrada en el reactor.
qc  = 1.65102;              // Flujo de entrada en la carcasa del reactor.
Da  = 0.072                    // Número de Damköhler.
r   = 20;                   // Energía de activación.
B   = 8;                    // Calor de la reacción.
s   = 0.3;                  // Coeficiente de transferencia de calor.
s1  = 0.1;                  // Relación de volumen del reactor.
s2  = 0.5;                     // Relación de la capacidad de la densidad del calor de la
                               reacción.
```

```
//---------------------------------------------------------------------
// Definición de las ecuaciones de estado.
//---------------------------------------------------------------------

dx(1)  = q.*(x1f - x1) - x1*Da*exp(x2/(1+(x2/r)));                    // Ecuación del
balance de masa de la sustancia dentro del reactor.
dx(2)  = B.*Da*x1*exp(x2/(1+(x2/r))) - (q + s)*x2 + s*x3 + x2f;       // Ecuación del
balance de energía dentro del reactor.
dx(3)  = (qc*(x3f - x3)/s1) + s*(x2 - x3)/(s1*s2);                    // Ecuación del
balance de energía dentro de la chaqueta del reactor.

endfunction
```

ANEXO 6– Algoritmo cstr_ode.sce

```
//************************ Algoritmo cstr_ode.sce  ****************************
// Este archivo llamado cstr_ode.sce para la obtención de los Condiciones de estabilidad de // un CSTR en el tiempo. El usuario ingresa los datos en un ambiente interactivo y recibe la // solución del sistema.
// Elaborado por Remberto J. Cuadro Alvear. Mayo de 2012
// Universidad Tecnológica de Bolívar - Cartagena
//*****************************************************************************

clear variables
clc

// ------------------------------------------------
// Entrada de Datos
// ------------------------------------------------

disp('Intervalo de tiempo para la muestra')
t0    = input('Tiempo inicial para iniciar la muestra.          t0: ');
tf    = input('Tiempo final para terminar la muestra.           tf: ');
[t]   = linspace(t0,tf);
clc
disp('Condiciones iniciales del sistema')
x1    = input('Condición inicial de Entrada de la sustancia.          x1: ');
x2    = input('Condición inicial de Temperatura del reactor.          x2: ');
x3    = input('Condición inicial de Temperatura de la carcasa.        x3: ');
x0    =[x1;x2;x3];
clc
disp('Comportamiento del sistema')
dx1    = input('Condición inicial de Entrada de la sustancia.         dx1: ');
dx2    = input('Condición inicial de Temperatura del reactor.         dx2: ');
dx3    = input('Condición inicial de Temperatura de la carcasa.       dx3: ');
dx     =[dx1;dx2;dx3];
clc
```

```
// ---------------------------------------------------------------------
// Solución del sistema de ecuaciones Diferenciales
// ---------------------------------------------------------------------
tic();
[x]       =ode(x0,t0,t,cstr)
T         = toc();
disp('Los puntos de equilibrio del CSTR son:')
disp('    Tiempo        x1          x2          x3')
disp('    _____________________________________________________')
disp([t',x'])
disp('El tiempo para la solución de la ecuación fue:')
disp(T)
// ---------------------------------------------------
// Gráficas de las variables
// ---------------------------------------------------
subplot(311)
plot(t,x(1,:),'b')
xlabel('Tiempo')
ylabel('Conc. Sustancia')
set(gca(),"grid",[1 1])
title('Comportamiento de los Estados Iniciales')
subplot(312)
plot(t,x(2,:),'g')
xlabel('Tiempo')
ylabel('Temp. Reactor')
set(gca(),"grid",[1 1])
subplot(313)
plot(t,x(3,:),'r')
xlabel('Tiempo')
ylabel('Temp. Carcaza')
set(gca(),"grid",[1 1])
```

ANEXO 7– Algoritmo cstr_ss.sce

```
//************************ Algoritmo cstr_ss.sce  ****************************
// Este archivo contiene el código cstr_ss.sce para la obtención de los Puntos de equilibrio
// en estado Estacionario de un CSTR. El modelo y los parámetros fueron tomados de "A
// Chemical Reactor Benchmark for Parallel Adaptive Control Using Feedfoward Neural
// Networks", Oliveira et al.
// Elaborado por Remberto J. Cuadro Alvear. Mayo de 2012
// Universidad Tecnológica de Bolívar - Cartagena
//*****************************************************************************

function dx = cstr_ss(x)

//-----------------------------------------------------------------
// Definición de las condiciones iniciales.
//-----------------------------------------------------------------

dx = zeros(3,1);          // Se hace un vector de 0.
x1 = x(1);               // x1 Concentración de sustancia en el reactor
x2 = x(2);               // x2 Temperatura del reactor
x3 = x(3);               // x3 Temperatura de la carcasa del reactor

//-----------------------------------------------------------------
// Definición de los parámetros del reactor.
//-----------------------------------------------------------------

x1f = 1;                  // Concentración inicial de la sustancia en el reactor
x2f = 0;                 // Temperatura inicial del reactor.
x3f = -1;                // Temperatura inicial de la carcasa del reactor.
q   = 1;                 // Flujo de entrada en el reactor.
qc  = 1.65102;           // Flujo de entrada en la carcasa del reactor.
Da  = 0.072;             // Número de Damköhler.
r   = 20;                // Energía de activación.
B   = 8;                 // Calor de la reacción.
s   = 0.3;               // Coeficiente de transferencia de calor.
s1  = 0.1;               // Relación de volumen del reactor.
s2  = 0.5;               // Relación de la capacidad de la densidad del calor de la reacción.
```

```
//---------------------------------------------------------------------
// Definición de las ecuaciones de estado.
//---------------------------------------------------------------------

dx(1)  = q.*(x1f - x1) - x1*Da*exp(x2/(1+(x2/r)));                    // Ecuación del
balance de masa de la sustancia dentro del reactor.
dx(2)  = B.*Da*x1*exp(x2/(1+(x2/r))) - (q + s)*x2 + s*x3 + x2f;       // Ecuación del
balance de energía dentro del reactor.
dx(3)  = (qc*(x3f - x3)/s1) + s*(x2 - x3)/(s1*s2);                    // Ecuación del
balance de energía dentro de la chaqueta del reactor.

endfunction
```

ANEXO 8 – Algoritmo cstr_fsolve.sce

```
//************************ Algoritmo cstr_fsolve.sce  ************************
// Este archivo llamado cstr_fsolve.sce para la obtención de los Condiciones de estabilidad
// de un CSTR en el tiempo. El usuario ingresa los datos en un ambiente interactivo y
//recibe la solución del sistema.
// Elaborado por Remberto J. Cuadro Alvear. Mayo de 2012
// Universidad Tecnológica de Bolívar - Cartagena
//*****************************************************************************

clear variables
clc

// ------------------------------------------------
// Entrada de Datos
// ------------------------------------------------

x1    = input('Condición inicial de Entrada de la sustancia.          x1: ');
x2    = input('Condición inicial de Temperatura del reactor.          x2: ');
x3    = input('Condición inicial de Temperatura de la carcasa.        x3: ');
x0    =[x1;x2;x3];

// ------------------------------------------------
// Solución de la ecuación Diferencial
// ------------------------------------------------

tic();
[x]       = fsolve(x0,cstr_ss);
T         = toc();
disp('Los puntos de equilibrio del CSTR son:')
disp(x)
disp('El tiempo para la solución de la ecuación fue:')
disp(T)
```