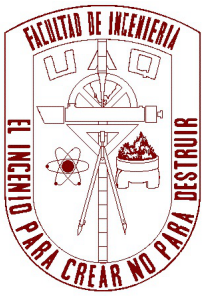


Universidad Autónoma de Querérato

Facultad de Ingeniería

Licenciatura en Matemáticas Aplicadas

"Análisis y realización del método SPPS en la solución de los problemas con valores en la frontera para las ecuaciones ordinarias lineales de segundo orden"

Tesis

Que como parte de los requisitos para obtener el título de:

Licenciado en Matemáticas Aplicadas

Presenta:

Elohim Ortiz Caballero

Dirigido por:

Dra. Kira Khmelnytskaya Gerasimova

Santiago de Querétaro, Qro. Diciembre 2011

Resumen

En 2008 se publicó un método llamado SPPS (Series de Potencias de Parámetro Espectral), el cuál proporciona la solución analítica de la ecuación $(pv')' + qv = \lambda rv$ en términos de series del parámetro espectral λ. En este trabajo se aplica el método SPPS a la solución de problemas con valores en la frontera. Se estudia su precisión, convergencia y se compara con técnicas puramente numéricas disponibles. Se desarrollaron las aplicaciones en Matlab que fueron usados en los ejemplos de prueba que admiten una solución exacta en forma cerrada. Por medio de la comparación de los resultados obtenidos tanto con la solución exacta como también con la solución numérica calculada con el uso de la rutina estandar de Matlab bvp4c se demostró que la implementación numérica de SPPS es sencilla, precisa y robusta, lo cual la convierte en una técnica altamente competitiva con los métodos numéricos existentes.

Índice

1. Descripción del Problema

Los problemas con valores en la frontera de ecuaciones diferenciales ordinarias lineales de segundo orden modelan muchos fenómenos en diferentes áreas del conocimiento tales como Electricidad, Óptica, Dinámica, Biología, Estadística, Economía, etc., de modo que es necesario conocer su solución para poder entender, explicar y predecir el comportamiento del objeto de estudio, bajo ciertas condiciones. Existen métodos para obtener la solución exacta de este tipo de ecuaciones cuando presentan coeficientes constantes, pero cuando los coeficientes son variables no existe un método general para obtener la solución exacta, de modo que se utilizan métodos numéricos para obtener una aproximación de la solución analítica. Es por esto que es importante el estudio de los métodos numéricos para su mejor realización, así como el análisis y comparación de su aplicabilidad y convergencia para obtener resultados óptimos.

2. Antecedentes y Justificación

El área de las ecuaciones diferenciales es un campo de estudio muy amplio, ya que éstas han probado ser una herramienta muy útil para modelar fenómenos en diversas ramas de la ciencia. Gran cantidad de estos fenómenos involucran problemas con valores en la frontera de ecuaciones ordinarias lineales de segundo orden con coeficientes variables. Debido a la falta de un método general para poder resolverlas se tiene que tratar cada ecuación por separado e intentar encontrar su solución, aunque muchas veces se utilizan métodos numéricos para hallar aproximaciones de la solución analítica. Es por eso que los programas de cálculo simbólico como Matlab incorporan dentro de sus herramientas, funciones como la bvp4c para resolver este tipo de problemas.

En el año 2008 el Dr. Vladislav V Kravchenko dio a conocer un método conocido como Spectral Parameter Power Series (SPPS) [10], [12] para encontrar la solución de la ecuación de Sturm-Liouville, como una representación de series de potencias del parámetro espectral. Este representa una nueva y eficiente herramienta para resolver problemas con valores iniciales y en la frontera que involucren la ecuación de Sturm-Liouville. Cabe señalar que éste método es analítico, pero por su naturaleza se puede implementar con ayuda de un software y obtener una solución numérica. A partir de ese año se ha utilizado en varios artículos para resolver problemas relacionados con la ecuación Sturm-Liouville, como por ejemplo: solución numérica de un problema con valores iniciales, que describe la cantidad en que se transmite y refleja una onda electromagnética en una capa no homogénea [5], aplicación al modelo de Jackiw—Pi de grafeno de doble capa [6], problemas inversos [9], solución de ecuaciones obtenidas usando técnicas supersimétricas [8], representación eficiente del discriminante de Hill utilizando el método SPPS [7].

En el presente trabajo se utilizará el método SPPS y se implementará en Matlab para encontrar las soluciones para algunos problemas de valores en la frontera con condiciones de tipo Dirichlet y Neumann para ecuaciones lineales de segundo orden. Cabe señalar que la aplicacion del método SPPS en este trabajo es un caso particular del método general, ya que no se consideran problemas espectrales que representan el área de aplicación del método, sin embargo para el caso $\lambda = 1$, el cuál será el tema de nuestro estudio, el método es aplicable .

Análogamente se utilizará la función bvp4c de Matlab para calcular la solución de este tipo de problemas. bvp4c combina varios métodos numéricos incorporados con el fin de analizar qué método obtiene una mejor aproximación a la solución exacta y poder obtener resultados óptimos.

Al no contar con un método que nos aporte una solución analítica para este tipo de problemas, es importante saber qué método numérico podemos usar con la seguridad de que nos dará una mejor aproximación que cualquier otro. Pues de esta manera podremos garantizar una mayor exactitud en los resultados y con esto una mejor modelación del fenómeno en cuestión.

3. Objetivos

- Entender en qué consiste el Método SPPS.
- Aprender a usar Matlab y en particular los comandos y las funciones para calcular soluciones numéricas de problemas con valores en la frontera de ecuaciones diferenciales lineales de segundo orden.

- Aprender a programar el método SPPS en Matlab.
- Analizar cómo se comportan las soluciones numéricas de cada método con respecto a la solución exacta en algunos ejemplos que se propongan.
- Sacar conclusiones acerca de la aplicabilidad del método.

4. Metodología

- Repasar teoría de ecuaciones lineales de segundo orden y conceptos relacionados con problema con valores en la frontera.
- Estudiar el método SPPS.
- Estudiar las funciones necesarias para programar el método SPPS en Matlab.
- Estudiar la función bvp4c y algunas otras que se utilizan en conjunto para calcular las soluciones numéricas de un problema con valores en la frontera de una ecuación diferencial lineal de segundo orden.
- Proponer ejemplos de problemas con valores en la frontera cuya solución exacta se conozca y darles solución numérica con los dos programas anteriores.
- Calcular los errores absoluto y relativo entre la solución exacta y la solución SPPS, analizar los resultados obtenidos en cuanto a la convergencia, eficiencia y estabilidad de la implementación numérica SPPS. De manera análoga, realizar el procedimiento anterior con bvp4c.
- Redactar tesis.

5. Preliminares

La ecuación diferencial lineal de segundo orden tiene la forma

$$\frac{d^2y}{dx^2} + P(x)\frac{dy}{dx} + Q(x)y = R(x) \tag{1}$$

donde, como la notación lo sugiere, $P(x), Q(x)\, R(x)$ son funciones de x solamente. No hay pérdida de generalidad en suponer que el coeficiente de la segunda derivada es 1 pues siempre se puede obtener (1) aplicando una división.

Contrario a lo que sucede con las ecuaciones diferenciales lineales de primer orden, en general (1) no se puede resolver explícitamente en términos de funciones elementales ni incluso en términos de antiderivadas. Existen algunos métodos para resolver algunos casos muy especiales de esta ecuación y muchas veces es necesario utilizar series. Cuando $R(x) = 0$ en (1) la ecuación resultante se le llama homogénea. En caso contrario la ecuación se llama no homogénea.

Debido a que en general no es posible obtener una solución por simple inspección de la ecuación (1) es importante asegurar la existencia y unicidad de la solución de dicha ecuación.

Teorema 1 (de Existencia y Unicidad) *Sean $P(x), Q(x)$ y $R(x)$ funciones continuas en un intervalo cerrado $[a, b]$. Si x_0 es un punto cualquiera en $[a, b]$ y si y_0 y y_0' son números cualesquiera, entonces la ecuación (1) tiene una y sólo una solución $y(x)$ en el intervalo $[a, b]$ tal que $y(x_0) = y_0$ y $y'(x_0) = y_0'$.*

5.1. Solución general de la ecuación homogénea

Definición 1 *Si dos funciones $f(x)$ y $g(x)$ definidas en el intervalo $[a, b]$ cumplen que una es múltiplo de la otra, se dice que son linealmente dependientes en $[a, b]$. Si esto no se cumple se dice que son linealmente independientes.*

Teorema 2 ([15]) *Sean $y_1(x)$ y $y_2(x)$ soluciones linealmente independientes de la ecuación homogénea*

$$y'' + P(x)y' + Q(x)y = 0 \tag{2}$$

en el intervalo $[a, b]$. Entonces

$$c_1 y_1(x) + c_2 y_2(x) \tag{3}$$

es la solución general de (2) en $[a, b]$, en el sentido de que cada solución de (2) en el intervalo se puede obtener de (3) para ciertos valores de c_1 y c_2.

Primero se demuestran dos lemas, el primero nos dice en que puntos la función de x , $W(y_1, y_2) = y_1y_2' - y_2y_1'$ llamada Wronskiano, se hace cero, y el segundo da una relación entre el valor del Wronskiano y la dependencia lineal de dos soluciones de la ecuación (2).

Lema 1 ([15]) *Sean $y_1(x)$ y $y_2(x)$ soluciones de (2) en $[a, b]$, entonces el Wronskiano $W(y_1, y_2)$ es cero o distinto de cero en todo el intervalo.*

Demostración. Observe que

$$W' = y_1y_2'' + y_1'y_2' - y_2y_1'' - y_2'y_1' = y_1y_2'' - y_2y_1''$$

Así, como y_1 y y_2 son soluciones de (2), se tiene

$$y_1'' + Py_1' + Qy_1 = 0$$

y

$$y_2'' + Py_2' + Qy_2 = 0$$

Multiplicando la primer ecuación por y_2 y la segunda por y_1, y restando la primera de la segunda se obtiene

$$(y_1y_2'' - y_2y_1'') + P(y_1y_2' - y_2y_1') = 0$$

esto es

$$\frac{dW}{dx} + PW = 0$$

Pero la solución de esta ecuación lineal de primer orden es

$$W = ce^{-\int P dx}$$

Usando el hecho de que la exponencial no se hace cero, se concluye que el Wronskiano es cero si $c = 0$ ó distinto de cero si $c \neq 0$. ■

Lema 2 ([15]) *Sean $y_1(x)$ y $y_2(x)$ dos soluciones de la ecuación (2) en $[a, b]$, entonces son linealmente dependientes en este intervalo si y sólo si su Wronskiano es cero.*

Demostración. Supongamos que y_1 y y_2 son linealmente dependientes, se quiere demostrar que $y_1y_2' - y_2y_1' = 0$. Si una de estas funciones es cero, se cumple claramente el resultado. Por lo que supongamos que ninguna es cero, entonces por su dependencia lineal se sigue que una es múltiplo constante de la otra. Esto es $y_1 = cy_2$ para algún c, entonces $y_1' = cy_2'$. Así, se tiene

$$y_1y_2' - y_2y_1' = y_1(cy_1') - (cy_1)y_1' = 0$$

Ahora supongamos que $W = 0$ y probaremos la dependencia lineal. Si $y_1 = 0$ en $[a, b]$ entonces las funciones son linealmente dependientes. Si y_1 no es totalmente cero en $[a, b]$, por continuidad se tiene que existe un intervalo $[d, e]$ de $[a, b]$ en el cual y_1 no se hace cero. Por lo tanto se cumple

$$\frac{y_1y_2' - y_2y_1'}{y_1^2} = 0$$

en $[d, e]$. La ecuación anterior se puede reescribir de la forma $(y_2/y_1)' = 0$, integrando se obtiene $y_2 = ky_1$ para alguna constante k y todo $x \in [d, e]$. Así, como y_2 y ky_1 tienen los mismos valores en $[d, e]$, sus derivadas son iguales y por el teorema 1 se deduce que

$$y_2(x) = ky_1(x)$$

para todo $x \in [a, b]$, es decir y_1 y y_2 son linealmente dependientes. ■

Del Lema 1 y el Lema 2 se concluye que el Wronskiano de dos soluciones de (2) linealmente independientes es distinto de cero.

Demostración del Teorema 2. Supongamos que $y(x)$ es una solución de (2) en $[a, b]$. Debemos demostrar que existen constantes c_1 y c_2 tales que

$$y(x) = c_1y_1(x) + c_2y_2(x)$$

para todo $x \in [a, b]$. Por el teorema 1, una solución de (2) en todo $[a, b]$ está determinada por su valor y el valor de su derivada en un sólo punto. Puesto que $c_1y_1(x) + c_2y_2(x)$ y $y(x)$ son soluciones de (2) en $[a, b]$, basta sólo con demostrar que para algún $x_0 \in [a, b]$ existen c_1 y c_2 tales que

$$\begin{aligned} c_1y_1(x_0) + c_2y_2(x_0) &= y(x_0) \\ c_1y_1'(x_0) + c_2y_2'(x_0) &= y(x_0) \end{aligned} \tag{4}$$

Pero para que el sistema anterior tenga solución se debe cumplir que el determinante

$$\begin{vmatrix} y_1(x_0) & y_2(x_0) \\ y_1'(x_0) & y_2'(x_0) \end{vmatrix} = y_1(x_0)y_2'(x_0) - y_2(x_0)y_1'(x_0)$$

sea distinto de cero. Pero como y_1 y y_2 son linealmente independientes por el lema 2 $W(y_1, y_2) = y_1y_2' - y_2y_1' \neq 0$, de modo que el sistema (4) tiene solución única, por lo tanto existen c_1 y c_2 tales que $y(x) = c_1y_1(x) + c_2y_2(x)$ para todo $x \in [a, b]$. ■

5.2. Uso de una solución para encontrar otra

Como se ha visto en la sección 5.1 una vez que se tienen dos soluciones linealmente independientes y_1 y y_2 de la ecuación homogénea (2), la solución general esta dada por la expresión $c_1y_1(x) + c_2y_2(x)$, donde c_1 y c_2 son constantes arbitrarias. Supongamos se conoce una solución $y_1(x)$, entonces para obtener una segunda solución, defínase $y_2 = y_1v$ donde

$$v = \int \frac{1}{y_1^2} e^{-\int P dx} dx,$$

claramente y_1 y y_2 son linealmente independientes por la definición 1, ya que una no es múltiplo constante de la otra. Entonces

$$y_2' = vy_1' + v'y_1 \qquad\qquad y_2'' = vy_1'' + 2v'y_1' + v''y_1. \tag{5}$$

Por otro lado, derivando v se tiene

$$v' = \frac{1}{y_1^2} e^{\int -P dx},$$

tomando logaritmos en ambos miembros

$$\log v' = -2 \log y_1 - \int P dx,$$

derivando

$$\frac{v''}{v'} = -2\frac{y_1'}{y_1} - P,$$

esto es

$$v''y_1 + v'(2y_1' + Py_1) = 0,$$

como y_1 es solución de (2)

$$v(y_1'' + Py_1' + Qy_1) + v''y_1 + v'(2y_1' + Py_1) = 0,$$

reescribiendo

$$vy_1'' + 2v'y_1' + v''y_1 + P(vy_1' + v'y_1) + Q(vy_1) = 0,$$

sustituyendo (5) y $y_2 = vy_1$ en la igualdad anterior se tiene

$$y_2'' + Py_2' + Qy_2 = 0,$$

es decir, y_2 es solución de (2).

5.3. Reducción de una ecuación diferencial lineal homogénea de segundo orden

Teorema 3 *Dada la ecuación lineal de segundo orden*

$$f(x)y'' + g(x)y' + h(x)y = 0 \tag{6}$$

con $x \in [a,b], f(x) \neq 0$ *en* $[a,b], a, b, \in \mathbb{R}$, *siempre es posible reducirla en una ecuación de la forma:*

$$u'' + q(x)u = 0 \tag{7}$$

donde $q = \dfrac{h}{f} - \dfrac{1}{4}\left(\dfrac{g}{f}\right)^2 - \dfrac{1}{2}\left(\dfrac{g}{f}\right)'$.

Demostración. Sea $u = y \cdot e^{\frac{1}{2}\int \frac{g}{f}dx}$ .Entonces

$$y = u \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx} \tag{8}$$

$$y' = -\frac{1}{2}\frac{g}{f}u \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx} + u' \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx} \tag{9}$$

$$y'' = -\frac{1}{2}\left(\frac{g}{f}\right)' u \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx} + \frac{1}{4}\left(\frac{g}{f}\right)^2 u \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx} - \frac{g}{f}u' \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx} + u'' \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx} \tag{10}$$

Sustituyendo (8),(9) y (10) en (6)

$$f\left[-\frac{1}{2}\left(\frac{g}{f}\right)' u \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx} + \frac{1}{4}\left(\frac{g}{f}\right)^2 u \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx} - \frac{g}{f}u' \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx} + u'' \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx}\right]$$

$$+g\left[-\frac{1}{2}\frac{g}{f}u \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx} + u' \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx}\right] + h \cdot u \cdot e^{-\frac{1}{2}\int \frac{g}{f}dx} = 0$$

De donde se tiene

$$\left[-\frac{f}{2}\left(\frac{g}{f}\right)' + \frac{f}{4}\left(\frac{g}{f}\right)^2 - \frac{1}{2}\left(\frac{g^2}{f}\right) + h\right] u + f \cdot u'' = 0 \tag{11}$$

Nótese que $f \neq 0$, pues (6) es de segundo orden. Así, multiplicando a (11) por $\frac{1}{f}$

$$\left[-\frac{1}{2}\left(\frac{g}{f}\right)' + \frac{1}{4}\left(\frac{g}{f}\right)^2 - \frac{1}{2}\left(\frac{g}{f}\right)^2 + \frac{h}{f}\right] u + u'' = 0$$

$$\left[-\frac{1}{2}\left(\frac{g}{f}\right)' - \frac{1}{4}\left(\frac{g}{f}\right)^2 + \frac{h}{f}\right] u + u'' = 0$$

Obteniendo la ecuación (7). ■

5.4. Problemas lineales con valores en la Frontera

Considérese el problema

$$y'' + P(x)y' + Q(x)y = R(x) \tag{12}$$

$$\begin{aligned} l_1[y] &= k_0 y(a) + k_1 y'(a) + h_0 y(b) + h_1 y'(b) = \alpha \\ l_2[y] &= m_0 y(a) + m_1 y'(a) + w_0 y(b) + w_1 y'(b) = \beta \end{aligned} \tag{13}$$

donde P, Q y R son funciones continuas en el intervalo $[a, b]$, $k_i, h_i, m_i, w_i \;\; i = 0, 1$ y α, β son constantes dadas, tales que una condición no es múltiplo de la otra. La ecuación (12) y las condiciones (13) se llaman problema no homogéneo con valores en la frontera. La ecuación (12) sujeta a las condiciones (13) con $R(x) \equiv 0$ y $\alpha = \beta = 0$ se llama problema homogéneo con valores en la frontera.
Las condiciones (13) son generales y en particular incluyen las

- Condiciones de Dirichlet

Este tipo de condición indica los valores que debe tomar la solución $y(x)$ en los extremos del intervalo a considerar. Para $k_1, h_0, h_1, m_0, m_1, w_1 = 0$ y $k_0, w_0 = 1$ en (13) se obtiene

$$y(a) = \alpha, \;\; y(b) = \beta.$$

- Condiciones de Neumann

Esta condición indica los valores que debe tomar la derivada de la solución $y'(x)$ en los extremos del intervalo considerado. Para $k_0, h_0, h_1, m_0, m_1, w_0 = 0$ y $k_1, w_1 = 1$ en (13) se obtiene

$$y'(a) = \alpha, \quad y'(b) = \beta.$$

- Condiciones de Robin

Esta condición es una combinación lineal de valores de la solución $y(x)$ y de los de su derivada $y'(x)$ en los extremos del intervalo usado. Para $h_0, h_1, m_0, m_1 = 0$ y $k_1, w_1 = 1$ en (13) se obtiene

$$y'(a) + k_0 y(a) = \alpha, \;\; y'(b) + w_0 y(b) = \beta.$$

La existencia y unicidad de la solución para estos problemas no siempre ocurre, pues pueden tener solución única, infinitas soluciones o ninguna. Por esta razón, es importante saber cuándo existe solución. Es claro que un problema homogéneo siempre tiene al menos la solución trivial. El siguiente teorema nos dice cuando un problema homogéneo tiene solamente la solución trivial.

Teorema 4 (vea p.ej. [1]) *Sean $y_1(x)$ y $y_2(x)$ soluciones linealmente independientes de la ecuación homogénea asociada a la ecuación (12)*

$$y'' + P(x)y' + Q(x)y = 0. \tag{14}$$

Entonces el problema homogéneo con valores en la frontera tiene como única solución a la trivial si y sólo si

$$\Delta = \begin{vmatrix} l_1[y_1] & l_1[y_2] \\ l_2[y_1] & l_2[y_2] \end{vmatrix} \neq 0$$

Demostración. Cualquier solución de la ecuación homogénea (14) se puede escribir de la forma

$$y(x) = c_1 y_1(x) + c_2 y_2(x).$$

Esta es solución del problema homogéneo si y sólo si

$$\begin{aligned} l_1[c_1 y_1 + c_2 y_2] &= c_1 l_1[y_1] + c_2 l_1[y_2] = 0 \\ l_2[c_1 y_1 + c_2 y_2] &= c_1 l_2[y_1] + c_2 l_2[y_2] = 0. \end{aligned} \tag{15}$$

Donde (15) tiene como solución única a la trivial si y sólo si $\Delta \neq 0$. ■

De este teorema se concluye que el problema homogéneo con valores en la frontera tiene infinitas soluciones si y sólo si $\Delta = 0$.

Teorema 5 ([1]) *El problema no homogéneo (12),(13) tiene solución única si y sólo si el problema homogéneo tiene como solución única la trivial.*

Demostración. Sean $y_1(x)$ y $y_2(x)$ soluciones linealmente independientes de la ecuación homogénea (14) y $z(x)$ una solución particular de (12). Entonces la solución general de la ecuación no homogénea (12) es de la forma

$$y(x) = c_1 y_1(x) + c_2 y_2(x) + z(x).$$

La cual es solución del problema (12), (13) si y sólo si

$$\begin{aligned} l_1[c_1 y_1 + c_2 y_2 + z] &= c_1 l_1[y_1] + c_2 l_1[y_2] + l_1[z] = \alpha \\ l_2[c_1 y_1 + c_2 y_2 + z] &= c_1 l_2[y_1] + c_2 l_2[y_2] + l_2[z] = \beta. \end{aligned} \tag{16}$$

Pero el sistema homogéneo en (16) tiene solución única si y sólo si $\Delta \neq 0$, si y solo si (por el teorema anterior) el problema homogéneo tiene como única solución a la trivial. ■

5.5. Definiciones de Error Absoluto y Relativo

Sea X un valor de una medida que se toma como exacto y x una aproximación de X, entonces se define el error absoluto como

$$\varepsilon_{abs} = |X - x|$$

es decir, es la magnitud de la diferencia entre la aproximación y el valor exacto. Y se define el error relativo como

$$\varepsilon_{rel} = \frac{|X - x|}{|X|}$$

esto es, es el cociente del error absoluto y la magnitud del valor exacto.

6. Descripción de la función bvp4c de Matlab

bvp4c

Sintaxis

```
sol = bvp4c(odefun,bcfun,solinit,options)
solinit = bvpinit(x, yinit, params)
```

Argumentos

`odefun` Manipulador de la función que evalúa la ecuación diferencial $f(x, y)$. Puede tener la forma

```
dydx = odefun(x,y,parameters)
```

donde `x` es un escalar correspondiente a x e `y` es un vector columna correspondiente a y. `parameters` es un vector de parámetros desconocidos y `odefun` es un vector columna.

`bcfun` Manipulador de la función que calcula los residuos en las condiciones de frontera. Para condiciones con valores en la frontera en dos puntos de la forma $bc(y(a), y(b))$, `bcfun` puede ser de la forma

```
res = bcfun(ya,yb,parameters)
```

donde `ya` y `yb` son vectores columna que corresponden a $y(a)$ y $y(b)$.
`parameters` es un vector de parámetros desconocidos.
La función retorna el vector columna `res`.

`solinit` Estructura que contiene las estimaciones iniciales. Se crea usando la función `bvpinit`. `solinit` contiene los siguientes datos:

`x`	Nodos ordenados de la malla inicial. Las condiciones de frontera se imponen en a =`solinit.x(1)` y b =`solinit.x(end)`
`y`	Estimación inicial de la solución de modo que `solinit.y(:,i)` sea una estimación de la solución en el nodo `solinit.x(i)`
`parameters`	Opcional. Es un vector que provee estimaciones para los parámetros desconocidos.

`options` Argumento de integración opcional. Estructura que se crea con `bvpset`

Descripción

`sol = bvp4c(odefun,bcfun,solinit)` integra un sistema de ecuaciones diferenciales de la forma

$$y' = f(x, y)$$

en el intervalo $[a, b]$ sujeto a las condiciones de frontera

$$bc(y(a), y(b)) = 0$$

`bvp4c` también puede encontrar parámetros p desconocidos para problemas de la forma

$$\begin{aligned} y' &= f(x, y, p) \\ 0 &= bc(y(a), y(b), p) \end{aligned}$$

donde p corresponde a `parameters`. `bvp4c` retorna los valores finales de los parámetros desconocidos en `sol.parameters`. `bvp4c` calcula una solución continua en $[a, b]$ y que tiene primera derivada continua.

La función `deval` evalúa la solución en los puntos `xint` del intervalo [a,b]:

```
sxint = deval(sol,xint)
```

La estructura `sol` que `bvp4c` retorna tiene los siguientes datos:

`sol.x`	Malla que `bvp4c` selecciona
`sol.y`	Aproximación a $y(x)$ en los puntos malla de `sol.x`
`sol.yp`	Aproximación a $y'(x)$ en los puntos malla de `sol.x`
`sol.parameters`	Valores que `bvp4c` retorna para los parámetros desconocidos, sólo si los hay
`sol.solver`	cadena 'bvp4c'
`sol.stats`	Estadísticas de costo computacional

`sol` puede reemplazarse por otro nombre. Los datos `x,y,yp,parameters` y `solver` son creados por `bvp4c`.

bvpinit

Sintaxis

```
solinit = bvpinit(x,yinit,parameters)
```

Descripción

`solinit = bvpinit(x,yinit,parameters)` forma la aproximación inicial que `bvp4c` utiliza para resolver el problema con valores en la frontera (BVP) dado.

`x` es un vector que especifica la malla inicial. Si se quiere resolver un BVP en $[a, b]$, se debe proporcionar a y b que corresponden a `x(1)` y `x(end)` respectivamente.

La función `bvp4c` adapta esta malla a la solución, de modo que una aproximación de la forma `xb = linspace(a,b,10)` a menudo es suficiente. En ocasiones cuando la solución cambia bruscamente se debe proporcionar los puntos malla donde ocurre este cambio. Las entradas de `x` deben ir en orden creciente ($a < b$) o en orden decreciente ($a > b$). Para problemas con valores en la frontera en dos puntos los valores de `x` deben ser distintos. Esto es, si $a < b$, las entradas deben satisfacer `x(1)` $<$ `x(2)` $<$ `...`$<$ `x(end)`. Análogamente si $a > b$.

`yinit` es una aproximación para la solución. Puede ser un vector o una función.

- Vector: `bvpinit` repite el correspondiente elemento del vector como una constante en todos los puntos malla. El vector debe ser de la forma $[y_0\ y_0']$ donde y_0 es una constante y $y_0' = 0$ es su derivada.
- Función: Para un punto malla dado `x(j)`, la función aproximación retorna un valor `y(j)` que es la aproximación al valor correspondiente de la solución.

Si se cuenta con la presencia de parámetros en la ecuación, se puede utilizar el vector `parameters` para dar aproximaciones a éstos.

`solinit` es una estructura que tiene los siguientes campos:

`x`	Nodos ordenados de la malla inicial
`y`	Aproximación inicial de la solución, donde `solinit.y(:,i)` es una aproximaxión para el nodo `solinit.x(i)`.
`parameters`	Opcional. Vector que provee de aproximaciones iniciales a los parámetros.

bvpset

Sintaxis

```
options = bvpset('propiedad1','valor1','propiedad2','valor2',...)
```

Descripción

Crea una estructura `options` que es empleada por `bvp4c`, en la cuál las propiedades tienen los valores especificados.

Propiedades de Tolerancia de Error

Debido a que `bvp4c` utiliza una fórmula de colocación, la solución numérica está basada en una malla de puntos en la cual las ecuaciones de colocación se satisfacen. La selección de la malla y el control del error se basan en los residuos de esta solución, de modo que la solución calculada $S(x)$ es la solución exacta de un problema perturbado $S'(x) = f(x, S(x)) + res(x)$. En cada subintervalo de la malla, una norma del residuo en la i-ésima componente de la solución, `res(i)`, es estimada y es menor o igual que una

cierta tolerancia. Esta tolerancia es una función de las tolerancias absolutas y relativas, `AbsTol`, `RelTol` las cuales son definidas por el usuario.

$$\left\| \frac{\texttt{res(i)}}{\max\left(\texttt{abs(f(i))}, \frac{\texttt{AbsTol(i)}}{\texttt{RelTol}}\right)} \right\| \leq \texttt{RelTol} \tag{17}$$

A continuación se describen las propiedades de tolerancia de Error.

Propiedad	**Valor por defecto**	**Descripción**
`RelTol`	1e-3	Tolerancia de error relativo que aplica a todas las componentes del vector de residuos. La solución calculada $S(x)$ es la solución exacta de $S'(x) = f(x, S(x)) + res(x)$. En cada subintervalo de la malla, el residuo $res(x)$ satisface $\left\| \frac{\texttt{res(i)}}{\max\left(\texttt{abs(f(i))}, \frac{\texttt{AbsTol(i)}}{\texttt{RelTol}}\right)} \right\| \leq \texttt{RelTol}$ El valor máximo para esta propiedad es de 1e-14. Si se proporciona un valor mayor, Matlab lo omite y toma 1e-14.
`AbsTol`	1e-6	Tolerancias de error absoluto que aplican a las componentes correspondientes del vector de residuos. `AbsTol(i)` es un límite por debajo del cual los valores de las componentes correspondientes no son importantes. Si un valor escalar es especificado, éste aplica a todas las componentes.

Propiedad del Tamaño de Malla

bvp4c resuelve un sistema de ecuaciones algebraicas para determinar la solución numérica de un problema con valores en la frontera en cada punto de la malla. El tamaño del sistema depende del número de ecuaciones diferenciales (n) y el número de puntos de la malla considerada (N). Cuando la cantidad máxima de puntos malla es alcanzada, bvp4c se detiene y retorna la solución obtenida hasta ese momento. Esta solución no satisface la tolerancia indicada, pero provee una excelente estimación inicial para volver a calcular la solución con a) tolerancias mayores o b) un incremento en el valor NMax (cantidad de puntos malla).

Propiedad	**Valor**	**Descripción**
`NMax`	Entero positivo $\{\lfloor 10000/n \rfloor\}$	Cantidad máxima de puntos malla permitida al resolver un problema con valores en la frontera, donde `n` es el número de ecuaciones diferenciales del problema. El valor por defecto `NMax` permite que el sistema algebraico sea de unas 10000 ecuaciones. Para sistemas de pocas ecuaciones diferenciales, el valor por defecto de `NMax` es suficiente para obtener una solución precisa.

Así, si se desea utilizar esta opción basta con definir el comando `options = bvpset('NMax',k),`donde k es el número de puntos malla que se requiera y añadir `options` como argumento de `bvp4c`.

7. Método SPPS

Recordemos la definición de función continua por partes y el Teorema fundamental del Cálculo.

Definición 2 *[13]. Una función $f(x)$, $a \leq x \leq b$ es continua por partes, si existe un conjunto finito de puntos $a = x_0 < x_1 < ... < x_k < x_{k+1} = b$ tales que*
1). f es continua en $x \neq x_i$, $\quad i = 0, ..., k+1$
2). $\lim_{\varepsilon \to 0} f(x_i + \varepsilon)$ existe $\quad i = 0, ..., k$
3). $\lim_{\varepsilon \to 0} f(x_i - \varepsilon)$ existe $\quad i = 1, ..., k+1$.

El límite en 2 se denota como $f(x_i + 0)$ y se le llama límite por la derecha. El límite en 3 se denota como $f(x_i - 0)$ y se le llama límite por la izquierda. Nótese que esta definición implica que f es acotada en $[a, b]$ y por tanto existe el supremo de f en $[a, b]$. Así, una función continua por partes en $[a, b]$, por el criterio de Lebesgue [2], es Riemann integrable en $[a, b]$.

Teorema 6 *(Teorema Fundamental del Cálculo [16]) Sea $f : [a, b] \to \mathbb{R}$ una función continua y defínase a $F(x)$ como sigue,*

$$F(x) = \int_a^x f(s)ds,$$

entonces $F(x)$ es derivable en $[a, b]$.

El método SPPS fue dado a conocer por el Dr. Kravchenko en 2008, el cual se utiliza para obtener la solución general de la ecuación de Sturm-Liouville

$$(pv')' + qv = \lambda rv$$

en forma de series de potencias del parámetro espectral λ. Un caso particular y muy utilizado es cuando $p = 1, q = 0, \lambda = 1$, el cuál se describe en el siguiente teorema donde $r(x) = q(x)$ es el potencial de la ecuación.

Teorema 7 *Supóngase que $q(x)$ es una función $C[0, d]$. La solución general de la ecuación*

$$-\frac{dv(x)}{dx^2} + q(x)v(x) = 0 \tag{18}$$

en $(0, d)$ es de la forma

$$v = c_1 v_1 + c_2 v_2$$

donde c_1 y c_2 son constantes arbitrarias y v_1, v_2 vienen dadas por las igualdades

$$v_1 = \sum_{k=0}^{\infty} \widetilde{X}_0^{(2k)}(x) \qquad\qquad v_2 = \sum_{k=0}^{\infty} X_0^{(2k+1)}(x), \tag{19}$$

donde $\widetilde{X}_0^{(n)}$ y $X_0^{(n)}$ se definen como sigue

$$\widetilde{X}_0^{(0)} \equiv 1 \qquad\qquad X_0^{(0)} \equiv 1$$

y para $n \in \mathbb{N}$

$$\widetilde{X}_0^{(n)}(x) = \begin{cases} \int_0^x \widetilde{X}_0^{(n-1)}(s)q(s)ds & \text{si } n \text{ es impar} \\ \int_0^x \widetilde{X}_0^{(n-1)}(s)ds & \text{si } n \text{ es par} \end{cases}$$

$$X_0^{(n)}(x) = \begin{cases} \int_0^x X_0^{(n-1)}(s)ds & \text{si } n \text{ es impar} \\ \int_0^x X_0^{(n-1)}(s)q(s)ds & \text{si } n \text{ es par} \end{cases}$$

Demostración. Primero probaremos que las series $\sum_{k=0}^{\infty} \widetilde{X}_0^{(2k)}(x)$, $\sum_{k=0}^{\infty} \widetilde{X}_0^{(2k+1)}(x)$, $\sum_{k=0}^{\infty} X_0^{(2k+1)}(x)$ y $\sum_{k=0}^{\infty} X_0^{(2k)}(x)$ convergen uniformemente. Puesto que $q(x)$ es integrable $[0,d]$, para n par y $x \in [0,d]$ se tiene

$$
\begin{aligned}
\left|\widetilde{X}_0^{(n)}(x)\right| &= \left|\int_0^x \widetilde{X}_0^{(n-1)}(\xi_1)d\xi_1\right| \leq \int_0^x \left|\widetilde{X}_0^{(n-1)}(\xi_1)\right| d\xi_1 = \int_0^x \left|\int_0^{\xi_1} \widetilde{X}_0^{(n-2)}(\xi_2)d\xi_2\right| d\xi_1 \\
&\leq \int_0^x\int_0^{\xi_1} |q(\xi_2)| \int_0^{\xi_2} \left|\widetilde{X}_0^{(n-3)}(\xi_3)\right| d\xi_3 d\xi_2 d\xi_1 \\
&\leq \int_0^x\int_0^{\xi_1} |q(\xi_2)| \int_0^{\xi_2}\int_0^{\xi_3} \left|q(\xi_4)\widetilde{X}_0^{(n-4)}(\xi_4)\right| d\xi_4 d\xi_3 d\xi_2 d\xi_1 \leq ... \\
&\leq \int_0^x\int_0^{\xi_1} |q(\xi_2)| \int_0^{\xi_2}\int_0^{\xi_3} |q(\xi_4)| ... \int_0^{\xi_{n-4}}\int_0^{\xi_{n-3}} \left|q(\xi_{n-2})\right| \int_0^{\xi_{n-2}}\int_0^{\xi_{n-1}} \left|q(\xi_n)\widetilde{X}_0^{(0)}(\xi_n)\right| d\xi_n d\xi_{n-1} d\xi_{n-2} d\xi_{n-3} ... d\xi_4 d\xi_3 d\xi_2 d\xi_1 \\
&\leq \int_0^x\int_0^{\xi_1} |q(\xi_2)| \int_0^{\xi_2}\int_0^{\xi_3} |q(\xi_4)| ... \int_0^{\xi_{n-4}}\int_0^{\xi_{n-3}} \left|q(\xi_{n-2})\right| \int_0^{\xi_{n-2}} \xi_{n-1} \max_{[0,x]} |q| \, d\xi_{n-1} d\xi_{n-2} d\xi_{n-3} ... d\xi_4 d\xi_3 d\xi_2 d\xi_1 \\
&= \int_0^x\int_0^{\xi_1} |q(\xi_2)| \int_0^{\xi_2}\int_0^{\xi_3} |q(\xi_4)| ... \int_0^{\xi_{n-4}}\int_0^{\xi_{n-3}} \max_{[0,x]} |q| \left|q(\xi_{n-2})\right| \frac{\xi_{n-2}^2}{2} d\xi_{n-2} d\xi_{n-3} ... d\xi_4 d\xi_3 d\xi_2 d\xi_1 \\
&\leq \int_0^x\int_0^{\xi_1} |q(\xi_2)| \int_0^{\xi_2}\int_0^{\xi_3} |q(\xi_4)| ... \int_0^{\xi_{n-4}}\int_0^{\xi_{n-3}} \left(\max_{[0,x]} |q|\right)^2 \frac{\xi_{n-2}^2}{2} d\xi_{n-2} d\xi_{n-3} ... d\xi_4 d\xi_3 d\xi_2 d\xi_1 \\
&= \int_0^x\int_0^{\xi_1} |q(\xi_2)| \int_0^{\xi_2}\int_0^{\xi_3} |q(\xi_4)| ... \int_0^{\xi_{n-4}} \left(\max_{[0,x]} |q|\right)^2 \frac{\xi_{n-3}^3}{3!} d\xi_{n-3} ... d\xi_4 d\xi_3 d\xi_2 d\xi_1 \\
&\leq ... \leq \int_0^x\int_0^{\xi_1} |q(\xi_2)| \int_0^{\xi_2}\int_0^{\xi_3} |q(\xi_4)| \left(\max_{[0,x]} |q|\right)^{\frac{n-4}{2}} \frac{\xi_4^{n-4}}{(n-4)!} d\xi_4 d\xi_3 d\xi_2 d\xi_1 \\
&\leq \int_0^x\int_0^{\xi_1} |q(\xi_2)| \int_0^{\xi_2} \left(\max_{[0,x]} |q|\right)^{\frac{n-2}{2}} \frac{\xi_3^{n-3}}{(n-3)!} d\xi_3 d\xi_2 d\xi_1 = \int_0^x\int_0^{\xi_1} |q(\xi_2)| \left(\max_{[0,x]} |q|\right)^{\frac{n-2}{2}} \frac{\xi_2^{n-2}}{(n-2)!} d\xi_2 d\xi_1 \\
&\leq \int_0^x \left(\max_{[0,x]} |q|\right)^{\frac{n}{2}} \frac{\xi_1^{n-1}}{(n-1)!} d\xi_1 = \left(\max_{[0,x]} |q|\right)^{\frac{n}{2}} \frac{x^n}{n!} \leq \left(\max_{[0,d]} |q|\right)^{\frac{n}{2}} \frac{d^n}{n!}, \qquad (20)
\end{aligned}
$$

Para n impar, n-1 es par, de manera que haciendo uso del caso (20) se tiene

$$\begin{aligned}
\left|\widetilde{X}_0^{(n)}(x)\right| &= \left|\int_0^x q(\xi_1)\widetilde{X}_0^{(n-1)}(\xi_1)d\xi_1\right| \leq \int_0^x |q(\xi_1)|\left|\widetilde{X}_0^{(n-1)}(\xi_1)\right|d\xi_1 \\
&\leq \int_0^x |q(\xi_1)|\left(\max_{[0,x]}|q|\right)^{\frac{n-1}{2}}\frac{\xi_1^{n-1}}{(n-1)!}d\xi_1 \\
&\leq \left(\max_{[0,x]}|q|\right)^{\frac{n+1}{2}}\int_0^x\frac{\xi_1^{n-1}}{(n-1)!}d\xi_1 = \left(\max_{[0,x]}|q|\right)^{\frac{n+1}{2}}\frac{x^n}{n!} \\
&\leq \left(\max_{[0,d]}|q|\right)^{\frac{n+1}{2}}\frac{d^n}{n!}
\end{aligned}$$

Para n impar y $x \in [0,d]$

$$\begin{aligned}
\left|X_0^{(n)}(x)\right| &= \left|\int_0^x X_0^{(n-1)}(\xi_1)d\xi_1\right| \leq \int_0^x\left|X_0^{(n-1)}(\xi_1)\right|d\xi_1 \leq \int_0^x\int_0^{\xi_1}\left|q(\xi_2)X_0^{(n-2)}(\xi_2)\right|d\xi_2 d\xi_1 \\
&\leq \int_0^x\int_0^{\xi_1}|q(\xi_2)|\int_0^{\xi_2}\left|X_0^{(n-3)}(\xi_3)\right|d\xi_3 d\xi_2 d\xi_1 \\
&\leq \int_0^x\int_0^{\xi_1}|q(\xi_2)|\int_0^{\xi_2}\int_0^{\xi_3}\left|q(\xi_4)X_0^{(n-4)}(\xi_4)\right|d\xi_4 d\xi_3 d\xi_2 d\xi_1 \leq ... \\
&\leq \int_0^x\int_0^{\xi_1}|q(\xi_2)|\int_0^{\xi_2}\int_0^{\xi_3}|q(\xi_4)|...\int_0^{\xi_{n-4}}\left|q(\xi_{n-3})\right|\int_0^{\xi_{n-3}\xi_{n-2}}\left|q(\xi_{n-1})\right|\int_0^{\xi_{n-1}}\left|X_0^{(0)}(\xi_n)\right|d\xi_n d\xi_{n-1}d\xi_{n-2}d\xi_{n-3}...d\xi_4 d\xi_3 d\xi_2 \\
&= \int_0^x\int_0^{\xi_1}|q(\xi_2)|\int_0^{\xi_2}\int_0^{\xi_3}|q(\xi_4)|...\int_0^{\xi_{n-4}}\left|q(\xi_{n-3})\right|\int_0^{\xi_{n-3}\xi_{n-2}}\left|q(\xi_{n-1})\right|\xi_{n-1}d\xi_{n-1}d\xi_{n-2}d\xi_{n-3}...d\xi_4 d\xi_3 d\xi_2 d\xi_1 \\
&\leq \int_0^x\int_0^{\xi_1}|q(\xi_2)|\int_0^{\xi_2}\int_0^{\xi_3}|q(\xi_4)|...\int_0^{\xi_{n-4}}\left|q(\xi_{n-3})\right|\int_0^{\xi_{n-3}\xi_{n-2}}\max_{[0,x]}|q|\,\xi_{n-1}d\xi_{n-1}d\xi_{n-2}d\xi_{n-3}...d\xi_4 d\xi_3 d\xi_2 d\xi_1 \\
&\leq \int_0^x\int_0^{\xi_1}|q(\xi_2)|\int_0^{\xi_2}\int_0^{\xi_3}|q(\xi_4)|...\int_0^{\xi_{n-4}}\left|q(\xi_{n-3})\right|\int_0^{\xi_{n-3}}\max_{[0,x]}|q|\,\frac{\xi_{n-2}^2}{2}d\xi_{n-2}d\xi_{n-3}...d\xi_4 d\xi_3 d\xi_2 d\xi_1 \\
&\leq \int_0^x\int_0^{\xi_1}|q(\xi_2)|\int_0^{\xi_2}\int_0^{\xi_3}|q(\xi_4)|...\int_0^{\xi_{n-4}}\left(\max_{[0,x]}|q|\right)^2\frac{\xi_{n-3}^3}{3!}d\xi_{n-3}...d\xi_4 d\xi_3 d\xi_2 d\xi_1 \\
&\leq ... \leq \int_0^x\int_0^{\xi_1}|q(\xi_2)|\int_0^{\xi_2}\int_0^{\xi_3}|q(\xi_4)|\left(\max_{[0,x]}|q|\right)^{\frac{n-5}{2}}\frac{\xi_4^{n-4}}{(n-4)!}d\xi_4 d\xi_3 d\xi_2 d\xi_1 \\
&\leq \int_0^x\int_0^{\xi_1}|q(\xi_2)|\int_0^{\xi_2}\left(\max_{[0,x]}|q|\right)^{\frac{n-3}{2}}\frac{\xi_3^{n-3}}{(n-3)!}d\xi_3 d\xi_2 d\xi_1 = \int_0^x\int_0^{\xi_1}|q(\xi_2)|\left(\max_{[0,x]}|q|\right)^{\frac{n-3}{2}}\frac{\xi_2^{n-2}}{(n-2)!}d\xi_2 d\xi_1 \\
&\leq \int_0^x\left(\max_{[0,x]}|q|\right)^{\frac{n-1}{2}}\frac{\xi_1^{n-1}}{(n-1)!}d\xi_1 = \left(\max_{[0,x]}|q|\right)^{\frac{n-1}{2}}\frac{x^n}{n!} \leq \left(\max_{[0,d]}|q|\right)^{\frac{n-1}{2}}\frac{d^n}{n!}.
\end{aligned}$$

Para n par, n-1 es impar y podemos hacer uso del caso anterior

$$\begin{aligned}\left|X_0^{(n)}(x)\right| &= \left|\int_0^x q(\xi_1)X_0^{(n-1)}(\xi_1)d\xi_1\right| \leq \int_0^x |q(\xi_1)|\left|X_0^{(n-1)}(\xi_1)\right|d\xi_1 \\ &\leq \int_0^x |q(\xi_1)|\left(\max_{[0,x]}|q|\right)^{\frac{n-2}{2}}\frac{\xi_1^{n-1}}{(n-1)!}d\xi_1 \leq \int_0^x \left(\max_{[0,x]}|q|\right)^{\frac{n}{2}}\frac{\xi_1^{n-1}}{(n-1)!}d\xi_1 \\ &= \left(\max_{[0,x]}|q|\right)^{\frac{n}{2}}\frac{x^n}{n!} \leq \left(\max_{[0,d]}|q|\right)^{\frac{n}{2}}\frac{d^n}{n!}\end{aligned}$$

De manera que

$$\left|\widetilde{X}_0^{(n)}(x)\right| \leq \begin{cases}\left(\max_{[0,d]}|q|\right)^{\frac{n}{2}}\frac{d^n}{n!} & \text{para n par} \\ \left(\max_{[0,d]}|q|\right)^{\frac{n+1}{2}}\frac{d^n}{n!} & \text{para n impar}\end{cases}$$

$$\left|X_0^{(n)}(x)\right| \leq \begin{cases}\left(\max_{[0,d]}|q|\right)^{\frac{n-1}{2}}\frac{d^n}{n!} & \text{para n impar} \\ \left(\max_{[0,d]}|q|\right)^{\frac{n}{2}}\frac{d^n}{n!} & \text{para n par}\end{cases}$$

o bien

$$\left|\widetilde{X}_0^{(2k)}(x)\right| \leq \left(\max_{[0,d]}|q|\right)^{k}\frac{d^{2k}}{(2k)!} = \frac{C^k}{(2k)!} \tag{21}$$

$$\left|\widetilde{X}_0^{(2k+1)}(x)\right| \leq \left(\max_{[0,d]}|q|\right)^{k+1}\frac{d^{2k+1}}{(2k+1)!} = \left(\max_{[0,d]}|q|\right)^{\frac{1}{2}}\frac{C^{\frac{2k+1}{2}}}{(2k+1)!} \tag{22}$$

$$\left|X_0^{(2k+1)}(x)\right| \leq \left(\max_{[0,d]}|q|\right)^{k}\frac{d^{2k+1}}{(2k+1)!} = \left(\max_{[0,d]}|q|\right)^{-\frac{1}{2}}\frac{C^{\frac{2k+1}{2}}}{(2k+1)!} \tag{23}$$

$$\left|X_0^{(2k)}(x)\right| \leq \left(\max_{[0,d]}|q|\right)^{k}\frac{d^{2k}}{(2k)!} = \frac{C^k}{(2k)!} \tag{24}$$

donde $x \in [0,d], k \in \mathbb{N}$ y $C = d^2 \max_{[0,d]}|q|$. Por tanto se tiene que las series numéricas

$$\sum_{k=0}^{\infty}\frac{C^k}{(2k)!},\ \sum_{k=0}^{\infty}\left(\max_{[0,d]}|q|\right)^{\frac{1}{2}}\frac{C^{\frac{2k+1}{2}}}{(2k+1)!},\ \sum_{k=0}^{\infty}\left(\max_{[0,d]}|q|\right)^{-\frac{1}{2}}\frac{C^{\frac{2k+1}{2}}}{(2k+1)!},\ \sum_{k=0}^{\infty}\frac{C^k}{(2k)!}$$

son series mayorantes de las series funcionales $\sum_{k=0}^{\infty}\widetilde{X}_0^{(2k)}(x), \sum_{k=0}^{\infty}\widetilde{X}_0^{(2k+1)}(x), \sum_{k=0}^{\infty}X_0^{(2k+1)}(x)$ y $\sum_{k=0}^{\infty}X_0^{(2k)}(x)$, respectivamente.

Por otro lado recordemos que

$$\cosh(t) = \frac{e^t+e^{-t}}{2} = \frac{\sum_{n=0}^{\infty}\frac{t^n}{n!}+\sum_{n=0}^{\infty}\frac{(-t)^n}{n!}}{2} = \frac{\sum_{n=0}^{\infty}\left(\frac{t^n}{n!}+\frac{(-t)^n}{n!}\right)}{2} = \sum_{k=0}^{\infty}\frac{t^{2k}}{(2k)!} \tag{25}$$

$$\sinh(t) = \frac{e^t-e^{-t}}{2} = \frac{\sum_{n=0}^{\infty}\frac{t^n}{n!}-\sum_{n=0}^{\infty}\frac{(-t)^n}{n!}}{2} = \frac{\sum_{n=0}^{\infty}\left(\frac{t^n}{n!}-\frac{(-t)^n}{n!}\right)}{2} = \sum_{k=0}^{\infty}\frac{t^{2k+1}}{(2k+1)!}. \tag{26}$$

Considerando $t = \sqrt{C}$ en las series anteriores se sigue que las series

$$\sum_{k=0}^{\infty} \frac{C^k}{(2k)!},\ \sum_{k=0}^{\infty} \left(\max_{[0,d]} |q| \right)^{\frac{1}{2}} \frac{C^{\frac{2k+1}{2}}}{(2k+1)!},\ \sum_{k=0}^{\infty} \left(\max_{[0,d]} |q| \right)^{-\frac{1}{2}} \frac{C^{\frac{2k+1}{2}}}{(2k+1)!}$$

convergen. Entonces por las desigualdades (21)-(24), la convergencia de las series anteriores y aplicando la prueba M de Weierstrass se tiene que las series de funciones

$$\sum_{k=0}^{\infty} \widetilde{X}_0^{(2k)}(x),\quad \sum_{k=0}^{\infty} \widetilde{X}_0^{(2k+1)}(x),\quad \sum_{k=0}^{\infty} X_0^{(2k+1)}(x),\quad \sum_{k=0}^{\infty} X_0^{(2k)}(x)$$

convergen uniformemente en $[0, d]$.
Obsérvese que como $q(x)$ y $X_0^{(0)}$son continuas en $[0, d]$ por el teorema fundamental del Cálculo

$$\begin{aligned} \widetilde{X}_0^{(1)}(x) &= \int_0^x \widetilde{X}_0^{(0)}(s) q(s) ds = \int_0^x q(s) ds \\ X_0^{(1)}(x) &= \int_0^x X_0^{(0)}(s) ds = x \end{aligned}$$

son continuas y derivables en $[0, d]$ y además

$$\begin{aligned} \frac{d\widetilde{X}_0^{(1)}(x)}{dx} &= q(x) \\ \frac{dX_0^{(1)}(x)}{dx} &= 1. \end{aligned}$$

Así, por inducción, se tiene que $\widetilde{X}_0^{(n)}(x)$ y $X_0^{(n)}(x)$ son continuas y derivables en $[0, d]$, donde

$$\begin{aligned} \frac{d\widetilde{X}_0^{(n)}(x)}{dx} &= \begin{cases} \widetilde{X}_0^{(n-1)}(x) q(x) & \text{si n es impar} \\ \widetilde{X}_0^{(n-1)}(x) dx & \text{si n es par} \end{cases} \\ \frac{dX_0^{(n)}(x)}{dx} &= \begin{cases} X_0^{(n-1)}(x) & \text{si n es impar} \\ X_0^{(n-1)}(x) q(x) & \text{si n es par} \end{cases} . \end{aligned}$$

Utilizando las fórmulas anteriores tenemos

$$\begin{aligned}\sum_{k=0}^{\infty}\frac{d\widetilde{X}_0^{(2k)}(x)}{dx} &= \frac{d\widetilde{X}_0^{(0)}(x)}{dx}+\frac{d\widetilde{X}_0^{(2)}(x)}{dx}+\frac{d\widetilde{X}_0^{(4)}(x)}{dx}+\frac{d\widetilde{X}_0^{(6)}(x)}{dx}+.... \qquad (27)\\ &= 0+\widetilde{X}_0^{(1)}(x)+\widetilde{X}_0^{(3)}(x)+\widetilde{X}_0^{(5)}(x)+...\\ &= \sum_{k=0}^{\infty}\widetilde{X}_0^{(2k+1)}(x)\\ \sum_{k=0}^{\infty}\frac{d\widetilde{X}_0^{(2k+1)}(x)}{dx} &= \frac{d\widetilde{X}_0^{(1)}(x)}{dx}+\frac{d\widetilde{X}_0^{(3)}(x)}{dx}+\frac{d\widetilde{X}_0^{(5)}(x)}{dx}+\frac{d\widetilde{X}_0^{(7)}(x)}{dx}+.... \qquad (28)\\ &= q(x)\widetilde{X}_0^{(0)}(x)+q(x)\widetilde{X}_0^{(2)}(x)+q(x)\widetilde{X}_0^{(4)}(x)+q(x)\widetilde{X}_0^{(6)}(x)+...\\ &= \sum_{k=0}^{\infty}q(x)\widetilde{X}_0^{(2k)}(x)\\ \sum_{k=0}^{\infty}\frac{dX_0^{(2k+1)}(x)}{dx} &= \frac{dX_0^{(1)}(x)}{dx}+\frac{dX_0^{(3)}(x)}{dx}+\frac{dX_0^{(5)}(x)}{dx}+\frac{dX_0^{(7)}(x)}{dx}+.... \qquad (29)\\ &= 1+X_0^{(2)}(x)+X_0^{(4)}(x)+X_0^{(6)}(x)+...\\ &= \sum_{k=0}^{\infty}X_0^{(2k)}(x)\\ \sum_{k=0}^{\infty}\frac{dX_0^{(2k)}(x)}{dx} &= \frac{dX_0^{(0)}(x)}{dx}+\frac{dX_0^{(2)}(x)}{dx}+\frac{dX_0^{(4)}(x)}{dx}+\frac{dX_0^{(6)}(x)}{dx}+.... \qquad (30)\\ &= 0+q(x)X_0^{(1)}(x)+q(x)X_0^{(3)}(x)+q(x)X_0^{(5)}(x)+...\\ &= \sum_{k=0}^{\infty}q(x)X_0^{(2k+1)}(x)\end{aligned}$$

Ahora para justificar la primera derivada de las series (19), debido a que $\widetilde{X}_0^{(n)}(x)$ y $X_0^{(n)}(x)$ son derivables en $[0,d]$ y $\sum_{k=0}^{\infty}\widetilde{X}_0^{(2k)}(x)$, $\sum_{k=0}^{\infty}X_0^{(2k+1)}(x)$, $\sum_{k=0}^{\infty}\frac{d\widetilde{X}_0^{(2k)}(x)}{dx}$, $\sum_{k=0}^{\infty}\frac{dX_0^{(2k+1)}(x)}{dx}$ convergen uniformemente en $[0,d]$, se tiene que

$$\begin{aligned}\frac{d\sum_{k=0}^{\infty}\widetilde{X}_0^{(2k)}(x)}{dx} &= \sum_{k=0}^{\infty}\frac{d\widetilde{X}_0^{(2k)}(x)}{dx} \qquad (31)\\ \frac{d\sum_{k=0}^{\infty}X_0^{(2k+1)}(x)}{dx} &= \sum_{k=0}^{\infty}\frac{dX_0^{(2k+1)}(x)}{dx}. \qquad (32)\end{aligned}$$

De modo que por (27),(29) y (31),(32) las derivadas de v_1 y v_2 son

$$v_1'(x)=\sum_{k=0}^{\infty}\widetilde{X}_0^{(2k+1)}(x) \qquad v_2'(x)=\sum_{k=0}^{\infty}X_0^{(2k)}(x). \qquad (33)$$

Para la segunda derivada de las series (19), nótese que por (27)-(30), (31) y (32) y el mismo argumento anterior se tiene

$$\begin{aligned}
\frac{d^2\sum_{k=0}^{\infty}\widetilde{X}_0^{(2k)}(x)}{dx^2} &= \frac{d\sum_{k=0}^{\infty}\frac{d\widetilde{X}_0^{(2k)}(x)}{dx}}{dx} = \frac{d\sum_{k=0}^{\infty}\widetilde{X}_0^{(2k+1)}(x)}{dx} = \sum_{k=0}^{\infty}\frac{d\widetilde{X}_0^{(2k+1)}(x)}{dx} \\
\frac{d^2\sum_{k=0}^{\infty}X_0^{(2k+1)}(x)}{dx^2} &= \frac{d\sum_{k=0}^{\infty}\frac{dX_0^{(2k+1)}(x)}{dx}}{dx} = \frac{d\sum_{k=0}^{\infty}X_0^{(2k)}(x)}{dx} = \sum_{k=0}^{\infty}\frac{dX_0^{(2k)}(x)}{dx}.
\end{aligned}$$

Por tanto ya podemos demostrar que v_1 y v_2 son soluciones de la ecuación (18), en efecto por (28), (30) y como $q(x)$ es continua en $[0,d]$ y las series v_1 y v_2 convergen uniformemente en $[0,d]$, se tiene

$$\begin{aligned}
-\frac{d^2v_1(x)}{dx^2}+q(x)v_1(x) &= -\frac{d^2\sum_{k=0}^{\infty}\widetilde{X}_0^{(2k)}(x)}{dx^2}+q(x)\sum_{k=0}^{\infty}\widetilde{X}_0^{(2k)}(x) = -\sum_{k=0}^{\infty}\frac{d\widetilde{X}_0^{(2k+1)}(x)}{dx}+q(x)\sum_{k=0}^{\infty}\widetilde{X}_0^{(2k)}(x) \\
&= -\sum_{k=0}^{\infty}q(x)\widetilde{X}_0^{(2k)}(x)+q(x)\sum_{k=0}^{\infty}\widetilde{X}_0^{(2k)}(x) = 0 \\
-\frac{d^2v_2(x)}{dx^2}+q(x)v_2(x) &= -\frac{d^2\sum_{k=0}^{\infty}X_0^{(2k+1)}(x)}{dx^2}+q(x)\sum_{k=0}^{\infty}X_0^{(2k+1)}(x) = -\sum_{k=0}^{\infty}\frac{dX_0^{(2k)}(x)}{dx}+q(x)\sum_{k=0}^{\infty}X_0^{(2k+1)}(x) \\
&= -\sum_{k=0}^{\infty}q(x)X_0^{(2k+1)}(x)+q(x)\sum_{k=0}^{\infty}X_0^{(2k+1)}(x) = 0.
\end{aligned}$$

Sólo falta por demostrar que v_1 y v_2 son linealmente independientes. Para esto basta con calcular el Wronskiano y mostrar que es distinto de cero en algún punto de $[0,d]$,

$$\begin{aligned}
W(v_1(x),v_2(x)) &= \begin{vmatrix} v_1(x) & v_2(x) \\ v_1'(x) & v_2'(x) \end{vmatrix} = v_1(x)v_2'(x)-v_2(x)v_1'(x) \\
&= \sum_{k=0}^{\infty}\widetilde{X}_0^{(2k)}(x)\sum_{k=0}^{\infty}\widetilde{X}_0^{(2k+1)}(x)-\sum_{k=0}^{\infty}X_0^{(2k+1)}(x)\sum_{k=0}^{\infty}X_0^{(2k)}(x),
\end{aligned}$$

evaluando el Wronskiano en 0 (ver la observación para los cálculos de $v_1(0), v_2(0), v_1'(0), v_2'(0)$) tenemos

$$W(v_1(0),v_2(0)) = v_1(0)v_2'(0)-v_2(0)v_1'(0) = 1\cdot 1-0\cdot 0 = 1.$$

Por el Lema 1 se tiene que $W(v_1(x),v_2(x)) \neq 0$ en $[0,d]$, y por el Lema 2 v_1 y v_2 son linealmente independientes. ■

Observación 1 *El Teorema 7 puede ser generalizado para el caso de la función $q(x)$ continua por partes en el intervalo $[0,d]$. En este caso la solución $v(x)$ de la ecuación (18) es de la clase $C^1[0,d]$ y no $C^2[0,d]$, por lo tanto la segunda derivada de $v(x)$ se debe considerar en el sentido de distribución (o sentido generalizado) (vea p.e. [19]).*
En efecto,

$$\begin{aligned}
v(x) &= c_1\sum_{k=0}^{\infty}\widetilde{X}_0^{(2k)}(x)+c_2\sum_{k=0}^{\infty}X_0^{(2k+1)}(x), \\
v'(x) &= c_1\sum_{k=0}^{\infty}\widetilde{X}_0^{(2k+1)}(x)+c_2\sum_{k=0}^{\infty}X_0^{(2k)}(x),
\end{aligned}$$

y por construcción las funciones $X_0^{(2k+1)}, \widetilde{X}_0^{(2k)} \in C^1[0,d]$ *y* $X_0^{(2k)}, \widetilde{X}_0^{(2k+1)} \in C[0,d]$. *Sean* $\{x_i\}_{i=1}^k$ *puntos de discontiniudad de salto finito de la función* $q(x)$, *en estos puntos* $v'(x)$ *es continua, pero no derivable. Por la definición de la derivada generalizada tenemos*

$$v''(x) = \{v''(x)\} + \sum_{i=1}^{k} [v'(x)]_{x_i} \delta(x - x_i),$$

donde $[v']_{x_i} = v'(x_i + 0) - v'(x_i - 0)$ *y es igual a cero por continuidad de* v' *en los puntos* $\{x_i\}$. *La expresión* $\{v''(x)\}$ *denota la derivada clasica de* $v'(x)$ *donde esta existe. Por lo tanto obtuvimos*

$$v''(x) = \{v''(x)\}.$$

En el ejemplo 2 se considera la aplicación numérica del método SPPS a la ecuación (18) con

$$q(x) = \begin{cases} -c^2 & 0 \le x < \frac{\pi}{2} \\ 0 & x = \frac{\pi}{2}, \\ c^2 & \frac{\pi}{2} < x \le \pi \end{cases},$$

se muestra que la solución $v(x)$ *en este caso es una función de clase* $C^1[0,\pi]$ *y que el método SPPS es aplicable con la misma eficiencia que para el caso* $q(x) \in C[0,\pi]$.

Observación 2 *Por definición* $\widetilde{X}_0^{(m)}(0) = 0$, $X_0^{(m)}(0) = 0$ *para* $m \in \mathbb{Z}^+$, *entonces haciendo uso de las definiciones de* v_1, v_2 *y de (33)*

$$\begin{aligned}
v_1(0) &= \sum_{k=0}^{\infty} \widetilde{X}_0^{(2k)}(0) = \widetilde{X}_0^{(0)}(0) + \widetilde{X}_0^{(2)}(0) + \widetilde{X}_0^{(4)}(0) + ... = 1 + 0 + 0 + \ldots = 1 \\
v_2(0) &= \sum_{k=0}^{\infty} X_0^{(2k+1)}(0) = X_0^{(1)}(0) + X_0^{(3)}(0) + X_0^{(5)}(0) + ... = 0 + 0 + 0 + \ldots = 0 \\
v_1'(0) &= \sum_{k=0}^{\infty} \widetilde{X}_0^{(2k+1)}(0) = 0 + \widetilde{X}_0^{(1)}(0) + \widetilde{X}_0^{(3)}(0) + ... = 0 + 0 + 0 + ... = 0 \\
v_2'(0) &= \sum_{k=0}^{\infty} X_0^{(2k)}(0) = X_0^{(0)}(0) + X_0^{(2)}(0) + X_0^{(4)}(0) + ... = 1 + 0 + 0 + ... = 1
\end{aligned}$$

Obsérvese que con los valores anteriores es posible calcular fácilmente las constantes c_1 *y* c_2 *de los problemas con valores en la frontera de tipo*

Dirichlet

$$\begin{aligned}
-\frac{dv(x)}{dx^2} + q(x)v(x) &= 0 \\
v(0) &= \alpha \\
v(d) &= \beta
\end{aligned}$$

Aquí, la solución $v(x) = c_1 v_1(x) + c_2 v_2(x)$ *debe satisfacer las condiciones de frontera, esto es*

$$\begin{aligned}
v(0) &= c_1 v_1(0) + c_2 v_2(0) = \alpha \\
v(d) &= c_1 v_1(d) + c_2 v_2(d) = \beta,
\end{aligned}$$

sustituyendo los valores de $v_1(0) = 1$ *y* $v_2(0) = 0$ *se tiene que*

$$\begin{aligned}
c_1 &= \alpha \\
c_2 &= \frac{\beta - \alpha v_1(d)}{v_2(d)}.
\end{aligned} \tag{34}$$

Neumann

$$\begin{aligned}
-\frac{dv(x)}{dx^2} + q(x)v(x) &= 0 \\
v'(0) &= \alpha \\
v'(d) &= \beta
\end{aligned}$$

La solución $v(x) = c_1 v_1(x) + c_2 v_2(x)$ *debe satisfacer las condiciones de frontera, esto es*

$$\begin{aligned} v'(0) &= c_1 v_1'(0) + c_2 v_2'(0) = \alpha \\ v'(d) &= c_1 v_1'(d) + c_2 v_2'(d) = \beta, \end{aligned}$$

sustituyendo los valores de $v_1'(0) = 0$ *y* $v_2'(0) = 1$ *se tiene que*

$$\begin{aligned} c_1 &= \frac{\beta - \alpha v_2'(d)}{v_1'(d)} \\ c_2 &= \alpha. \end{aligned} \tag{35}$$

Al implementar el método SPPS en Matlab es necesario especificar el valor de dos parámetros, el número de subintervalos en los que se divide el intervalo $[0, d]$, que se utilizan para calcular numéricamente las integrales $\widetilde{X}_0^{(n)}$ y $X_0^{(n)}$, y el número N, que corresponde a los primeros N+1 términos de las series (19). La elección de N se puede hacer de acuerdo a la precisión que se requiera, pues se puede obtener una N suficientemente grande para la cual el módulo del resto de las series (19) sea menor o igual a la precisión dada. Para esto, utilizando (21) y (23) se tiene la siguiente estimación de los términos

$$\begin{aligned} \left|\widetilde{X}_0^{(2k)}(x)\right| &\leq \left(\max_{[0,d]}|q|\right)^k \frac{d^{2k}}{(2k)!} = \frac{C^k}{(2k)!} \\ \left|X_0^{(2k+1)}(x)\right| &\leq \left(\max_{[0,d]}|q|\right)^k \frac{d^{2k+1}}{(2k+1)!} = \left(\max_{[0,d]}|q|\right)^{-\frac{1}{2}} \frac{C^{\frac{2k+1}{2}}}{(2k+1)!} \end{aligned}$$

donde $C = d^2 \max|q|$.

Así, si $v_{1,N} = \sum_{k=0}^{N} \widetilde{X}_0^{(2k)}$, $v_{2,N} = \sum_{k=0}^{N} X_0^{(2k+1)}$ y haciendo uso de la expansión en series (25),(26) de $\cosh\sqrt{C}$ y de $\sinh\sqrt{C}$ se tiene

$$\begin{aligned} |v_1 - v_{1,N}| &= \left|\sum_{k=0}^{\infty} \widetilde{X}_0^{(2k)} - \sum_{k=0}^{N} \widetilde{X}_0^{(2k)}\right| = \left|\sum_{k=N+1}^{\infty} \widetilde{X}_0^{(2k)}\right| \leq \left|\sum_{k=N+1}^{\infty} \frac{C^k}{(2k)!}\right| \\ &= \left|\sum_{k=0}^{\infty} \frac{C^k}{(2k)!} - \sum_{k=0}^{N} \frac{C^k}{(2k)!}\right| = \left|\cosh\sqrt{C} - \sum_{k=0}^{N} \frac{C^k}{(2k)!}\right| \qquad (36) \\ |v_2 - v_{2,N}| &= \left|\sum_{k=0}^{\infty} X_0^{(2k+1)} - \sum_{k=0}^{N} X_0^{(2k+1)}\right| = \left|\sum_{k=N+1}^{\infty} X_0^{(2k+1)}\right| \leq \left|\sum_{k=N+1}^{\infty} \left(\max_{[0,d]}|q|\right)^{-\frac{1}{2}} \frac{C^{\frac{2k+1}{2}}}{(2k+1)!}\right| \\ &= \left|\sum_{k=0}^{\infty} \left(\max_{[0,d]}|q|\right)^{-\frac{1}{2}} \frac{C^{\frac{2k+1}{2}}}{(2k+1)!} - \sum_{k=0}^{N} \left(\max_{[0,d]}|q|\right)^{-\frac{1}{2}} \frac{C^{\frac{2k+1}{2}}}{(2k+1)!}\right| \\ &= \left|\left(\max_{[0,d]}|q|\right)^{-\frac{1}{2}}\right| \left|\sum_{k=0}^{\infty} \frac{C^{\frac{2k+1}{2}}}{(2k+1)!} - \sum_{k=0}^{N} \frac{C^{\frac{2k+1}{2}}}{(2k+1)!}\right| \qquad (37) \\ &= \left(\max_{[0,d]}|q|\right)^{-\frac{1}{2}} \left|\sinh\sqrt{C} - \sum_{k=0}^{N} \frac{C^{\frac{2k+1}{2}}}{(2k+1)!}\right| \end{aligned}$$

De esta manera para una precisión $\varepsilon > 0$ dada se puede encontrar una N tal que $|v_1 - v_{1,N}|, |v_2 - v_{2,N}| < \varepsilon$. En efecto por la convergencia de las series (26),(25), dado $\varepsilon > 0$ existen N_1 y N_2 tales que para $m > N_1$, $n > N_2$ se cumple

$$\begin{aligned} \left|\sum_{k=0}^{m} \frac{C^k}{(2k)!} - \sum_{k=0}^{\infty} \frac{C^k}{(2k)!}\right| &= \left|\sum_{k=m+1}^{\infty} \frac{C^k}{(2k)!}\right| < \varepsilon \\ \left|\sum_{k=0}^{n} \left(\max_{[0,d]}|q|\right)^{-\frac{1}{2}} \frac{C^{\frac{2k+1}{2}}}{(2k+1)!} - \sum_{k=0}^{\infty} \left(\max_{[0,d]}|q|\right)^{-\frac{1}{2}} \frac{C^{\frac{2k+1}{2}}}{(2k+1)!}\right| &= \left|\sum_{k=n+1}^{\infty} \left(\max_{[0,d]}|q|\right)^{-\frac{1}{2}} \frac{C^{\frac{2k+1}{2}}}{(2k+1)!}\right| < \varepsilon, \end{aligned}$$

por lo tanto si tomamos $N = \max(N_1, N_2)$ se sigue que $|v_1 - v_{1,N}|, |v_2 - v_{2,N}| < \varepsilon$.

Ahora obsérvese que C depende de la longitud del intervalo considerado, de modo que entre más pequeño sea el intervalo las series

$$\sum_{k=0}^{\infty} \frac{C^k}{(2k)!} \qquad \text{y} \qquad \sum_{k=0}^{\infty} \frac{C^{\frac{2k+1}{2}}}{(2k+1)!}$$

convergen más rápido. Por lo anterior es preferible trabajar con intervalos pequeños, pues al calcular las estimaciones (36) y (37) el número de términos necesarios para satisfacer una precisión dada es pequeño y con esto se reduce el error de aproximación de $v_{1,N}$ y $v_{2,N}$. Es por esto que en [11] se propone el siguiente procedimiento para obtener mejores aproximaciones a las soluciones v_1 y v_2: se divide el intervalo inicial en $r-1$ subintervalos $[a_1, a_2], ..., [a_{r-1}, a_r]$ donde $a_1 = 0$ y $a_r = d$, se resuelve el problema de Cauchy asociado al primer subintervalo usando las condiciones iniciales ya conocidas $v_1(0) = 1$ y $v_1'(0) = 0$ para obtener una aproximación w_1 de v_1 en el primer subintervalo según la precisión requerida. Luego se calcula, haciendo uso de (27), la derivada $w_1'(a_2)$ y se utiliza junto con $w_1(a_2)$ como valores iniciales para resolver el problema de Cauchy del segundo subintervalo, de esta manera se prosigue hasta obtener la aproximación w_{r-1}. De esta manera se obtiene una aproximación a la solución v_1 en $[0, d]$ continua. Análogamente se calcula una aproximación para v_2 continua en $[0, d]$.

Ventajas del método SPPS sobre el método de Series de Potencias

Recordemos que para que una ecuación de la forma (18) tenga una solución en $[0, d]$ utilizando el método de Series de Potencias, es necesario que el potencial $q(x)$ sea analítico. El método SPPS se aplica a potenciales de clase $C[0, d]$ y se puede extender para funciones continuas por partes en $[0, d]$. Como se muestra en (34) y (35) es posible calcular los valores de las constantes c_1 y c_2 para los problemas de Dirichlet y Neumann de manera explícita, no obstante, en el caso de Series de Potencias una vez obtenida la solución general se tiene que calcular dos constantes a partir de los valores en la frontera, para lo cual no existe un método estándar y se tiene que implementar alguna técnica numérica.

En el aspecto numérico, el método SPPS posee la ventaja de que las $\widetilde{X}_0^{(2k)}(x)$, $X_0^{(2k+1)}(x)$ se calculan de una manera relativamente sencilla, pues sólo se integra recursivamente el potencial $q(x)$ o la variable de integración, según sea el caso. Sin embargo en el método de Series de Potencias es necesario trabajar cada problema por separado y tratar de encontrar una fórmula general para los coeficientes de la serie (solución), la cuál puede llegar a ser complicada. Otra ventaja del método SPPS es que la estimación del resto de las series v_1 y v_2 es fácil de calcular, pues siempre depende del resto de las series de Taylor de $\sinh(\sqrt{C})$ y $\cosh(\sqrt{C})$ donde $C = d^2 \max|q|$, en cambio para el método de Series de Potencias, dicha estimación puede llegar a ser díficil de calcular dependiendo de cada problema, pues no se cuenta con una estimación general como en SPPS. En general implementar el método de Serie de Potencias puede llegar a ser complicado y tardado, pues se tiene que tratar cada problema por separado, primero se debe de asegurar la analiticidad del potencial $q(x)$ en el intervalo $[0, d]$, expresarlo en forma de series, calcular los coeficientes de la solución, obtener los valores de las constantes a partir de los valores en la frontera y calcular una cota para estimar el resto de la solución en serie para poder truncar la serie de acuerdo a la precisión requerida. En este aspecto el método SPPS ofrece la ventaja en que es sencillo de implementar numéricamente y no es necesario tratar cada ejemplo por separado.

8. Ejemplos de Prueba

8.1. Comparación de soluciones numéricas y exactas de problemas con valores en la frontera

En algunos de los ejemplos presentados se aplicó la reducción del teorema 2, obteniendo ecuaciones de la forma

$$u'' + q(x)u = 0,$$

pues para utilizar SPPS es necesario que la ecuación diferencial del problema tenga esta forma.

Se escogieron ejemplos de prueba en los cuales la solución exacta esta dada de manera explícita. Para cada ejemplo se consideran dos tipos de gráficas, las primeras presentan soluciones exactas para algunos valores de los parámetros del ejemplo correspondiente y el comportamiento del coeficiente $q(x)$. Las soluciones numéricas correspondientes a bvp4c y a SPPS se comparan con la solución exacta. En el segundo conjunto de gráficas se presentan los errores absoluto y relativo entre la solución exacta y cada una de las soluciones numéricas. En ciertos casos donde alguno de estos errores no fue significativo se omitió en las gráficas.

Debido a que la solución que brinda el método SPPS consiste de series, al implementar este método en Matlab es necesario truncarlas para un cierto N (que corresponde a los primeros N+1 términos de la serie). En los siguientes ejemplos se varía ésta N a fin de obtener diferentes aproximaciones. También es posible considerar distintas cantidades de puntos que dividen al intervalo principal formando subintervalos en los que las integrales $X^{(n)}(x)$ y $\widetilde{X}^{(n)}(x)$ se aproximan por medio de la función *fnint*, de modo que en general entre más puntos se tomen es mejor la aproximación y por consiguiente menor el error, aunque la cantidad de puntos y la exactitud dependen de la longitud del intervalo y del ejemplo en cuestión.

Por otro lado para el método bvp4c existe una forma de mejorar la exactitud de la solución, esto se logra utilizando la función `bvpset` y añadiendo en cada uno de sus argumentos 'AbsTol', 'ReTol' el valor 1e-k, donde k es entero positivo. En caso de que no se considere esta opción `bvp4c` toma por defecto el valor 1e-6 para AbsTol y 1e-3 para RelTol. Otra propiedad que se puede modificar y añadir como argumento en `bvpset` para obtener mejor exactitud es la cantidad de puntos malla ('`NMax`'), por defecto `bvp4c` toma 5000 puntos. En este trabajo no se modificó el valor de `NMax` en la mayoría de los ejemplos.

En general se consideran 2 conjuntos de valores para cada parámetro del problema. Para cada conjunto y cada método se presentan una serie de gráficas, utilizando bvp4c en general la primera representa el error obtenido con los valores de tolerancias por defecto y las siguientes son obtenidas con diferentes valores de las tolerancias, en caso de que los valores por defecto no calculen una solución no se consideran. Para SPPS se presentan gráficas donde se varía la cantidad de puntos y términos de las series. En algunos casos se escoge la cantidad mínima de términos de tal forma que si se incrementa ésta el orden del error obtenido es el mismo. Existe una función de Matlab que sirve para calcular el tiempo que tarda cierta porción de código en ejecutarse, ésta fue utilizada para determinar cuántos segundos tarda cada programa para calcular la solución con SPPS o Bvp4c, la solución exacta y graficar el error deseado. Esta cantidad de segundos aparece en los pies de las gráficas como Cputime.

8.2. Condiciones de Dirichlet

Supóngase se tiene el siguiente problema con condiciones de Dirichlet

$$\begin{aligned} y'' + q(x)y &= 0 \\ y(a) &= \alpha \\ y(b) &= \beta \end{aligned} \tag{38}$$

donde $a, b, \alpha, \beta \in \mathbb{R}$.

Para utilizar la función bvp4c se aplica la sustitución $y_1 = y \quad$ y $\quad y_2 = y'$ en (38) y se obtiene el sistema

$$\begin{cases} y_1' = y_2 \\ y_2' = -q(x)y_1 \end{cases} \qquad y_1(a) = \alpha,\ y_1(b) = \beta \tag{39}$$

Note que una solución de (38) es solución de (39) y cualquier solución de (39) es solución de (38).
Este sistema se utiliza en los siguientes ejemplos para calcular una solucion numérica con bvp4c y utilizarla junto con la solución exacta para calcular los errores absoluto y relativo.

Ejemplo 1

Solución Exacta

Considérese el problema

$$\begin{aligned} y'' + c^2 y &= 0 \\ y(a) &= \alpha \\ y(b) &= \beta \end{aligned} \tag{40}$$

donde $c(\neq 0), a, b, \alpha, \beta \in \mathbb{R}$, con $a < b$.
Se sabe que la ecuación diferencial en (40) tiene como solución general a $y_g = c_1 \cos(cx) + c_2 \sin(cx)$, donde c_1 y c_2 son constantes arbitrarias. Entonces se debe cumplir

$$\begin{aligned} y_g(a) &= c_1 \cos(ca) + c_2 \sin(ca) = \alpha \\ y_g(b) &= c_1 \cos(cb) + c_2 \sin(cb) = \beta \end{aligned} \tag{41}$$

Por los Teoremas 4 y 5 el problema (40) tiene solución única si y sólo si

$$\Delta = \begin{vmatrix} \cos(ca) & \sin(ca) \\ \cos(cb) & \sin(cb) \end{vmatrix} = \cos(ca)\sin(cb) - \sin(ca)\cos(cb) = \sin(cb - ca) \neq 0.$$

Pero $\Delta = 0$ *sii* $c(b-a) = k\pi$, $k \in \mathbb{Z}$. Supóngase que $\Delta \neq 0$. Aplicando la regla de Cramer a (41) se obtienen los coeficientes

$$c_1 = \frac{\begin{vmatrix} \alpha & \sin(ca) \\ \beta & \sin(cb) \end{vmatrix}}{\Delta} = \frac{\alpha \sin(cb) - \beta \sin(ca)}{\sin(cb - ca)}, \qquad c_2 = \frac{\begin{vmatrix} \cos(ca) & \alpha \\ \cos(cb) & \beta \end{vmatrix}}{\Delta} = \frac{\beta \cos(ca) - \alpha \cos(cb)}{\sin(cb - ca)}.$$

Entonces la solución exacta de (40) es

$$y_p = \frac{\alpha \sin(cb) - \beta \sin(ca)}{\sin(cb - ca)} \cos(cx) + \frac{\beta \cos(ca) - \alpha \cos(cb)}{\sin(cb - ca)} \sin(cx), \tag{42}$$

siempre que $\sin(cb - ca) \neq 0$.
A continuación se presentan las gráficas de algunas soluciones exactas para los valores $\alpha = 0$, $\beta = 1$, $a = 0$, $b = \sqrt{2}, c = 1, 3, 5, 7, 9, 11$.

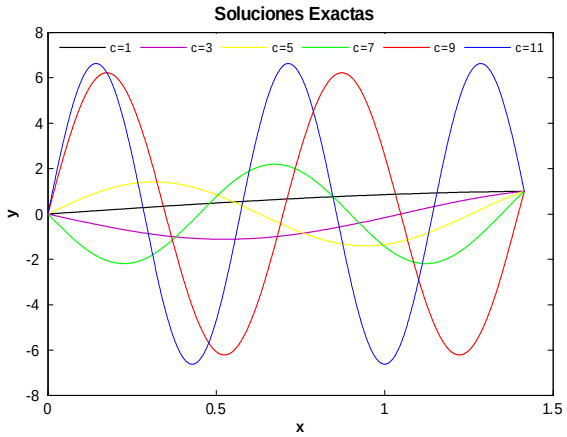


Fig. 1

Soluciones Numéricas

Las siguientes figuras muestran el error absoluto entre la solución de cada método y la solución exacta para los valores de $c = 1, ..., 13$, considerando $\alpha = 0$, $\beta = 1$, $a = 0$, $b = \sqrt{2}$ en (40).

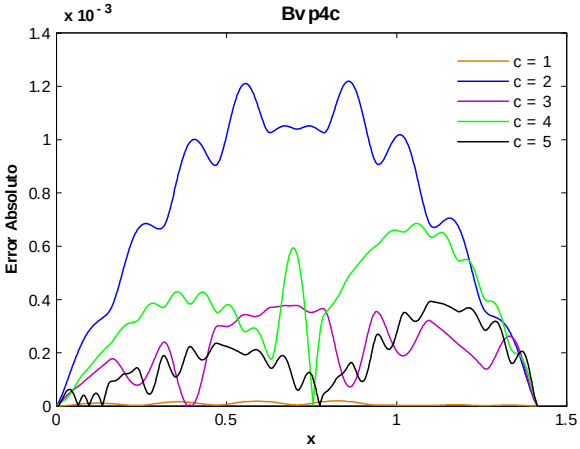


Fig 2. Error Absoluto de la solución Bvp4c para el ejemplo 1 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 1.25 s.

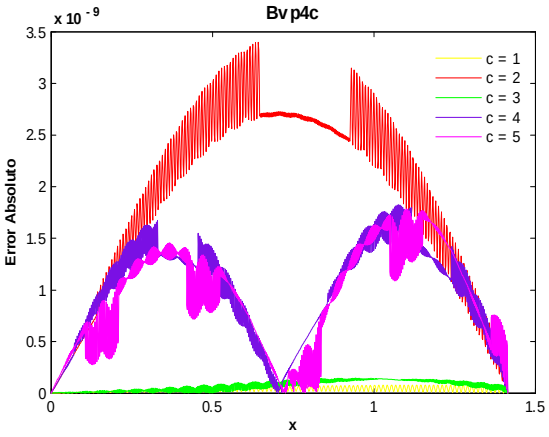


Fig 3. Error Absoluto de la solución Bvp4c para el ejemplo 1 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta y relativa de 1e-8.
Cputime = 1.3 s.

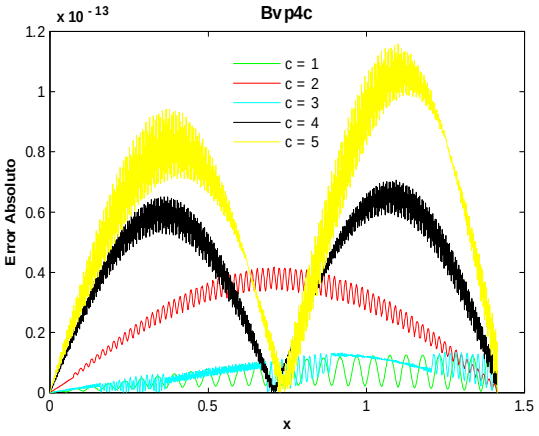


Fig 4. Error Absoluto de la solución Bvp4c para el ejemplo 1 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta y relativa de 1e-12.
Cputime = 7.19 s.

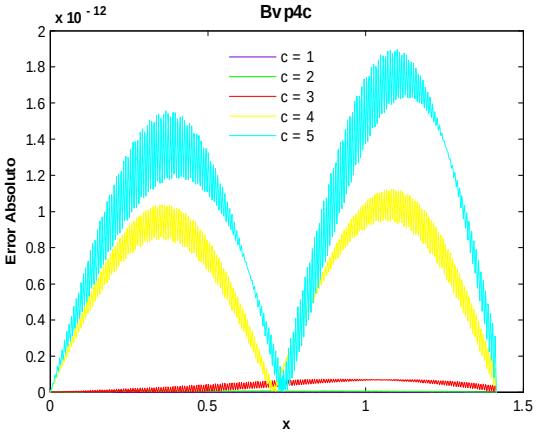


Fig 5. Error Absoluto de la solución Bvp4c para el ejemplo 1 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta y relativa de 1e-14.
Cputime = 51.31 s.

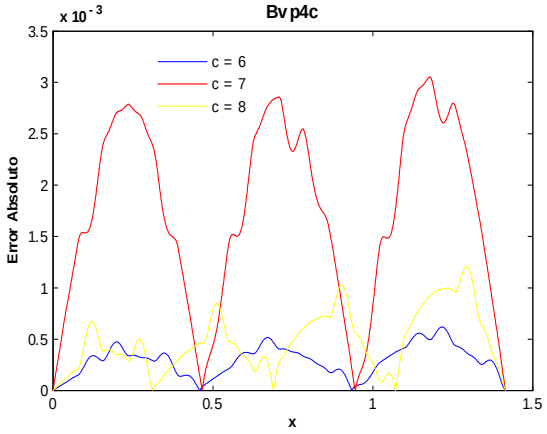


Fig 6. Error Absoluto de la solución Bvp4c para el ejemplo 1 con los valores del parámetro c=6,7,8 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 0.4s.

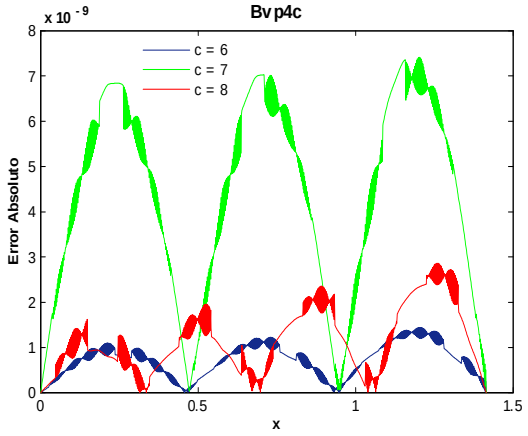


Fig 7. Error Absoluto de la solución Bvp4c para el ejemplo 1 con los valores del parámetro c=6,7,8 y tolerancias absoluta y relativa de 1e-8. Cputime = 1.51 s.

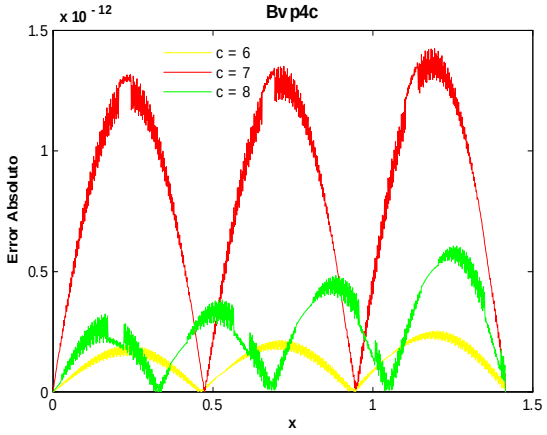


Fig 8. Error Absoluto de la solución Bvp4c para el ejemplo 1 con los valores del parámetro c=6,7,8 y tolerancias absoluta y relativa de 1e-12. Cputime = 5.97 s.

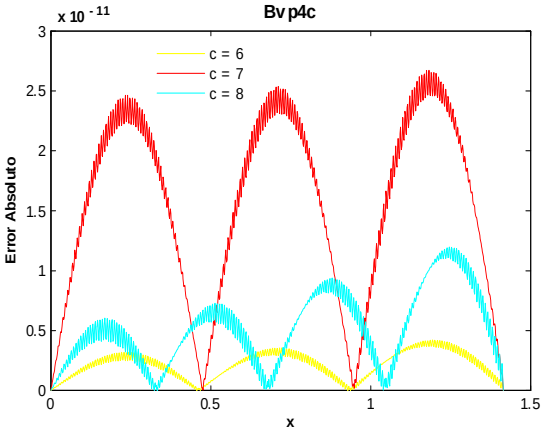


Fig 9. Error Absoluto de la solución Bvp4c para el ejemplo 1 con los valores del parámetro c=6,7,8 y tolerancias absoluta y relativa de 1e-14. Cputime = 7.55 s.

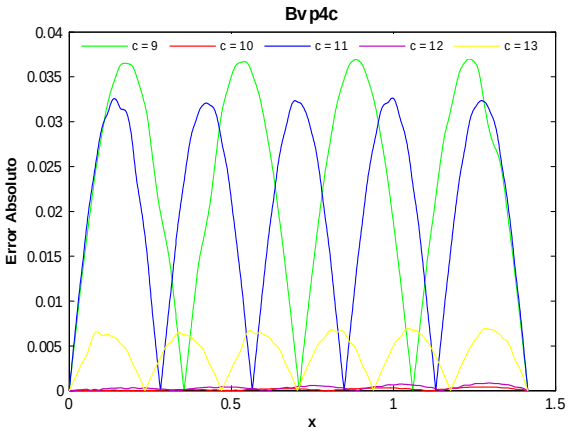


Fig 10. Error Absoluto de la solución Bvp4c para el ejemplo 1 con los valores del parámetro c=9,10,11,12,13 y tolerancias absoluta 1e-6 y relativa de 1e-3.
Cputime = 0.51 s.

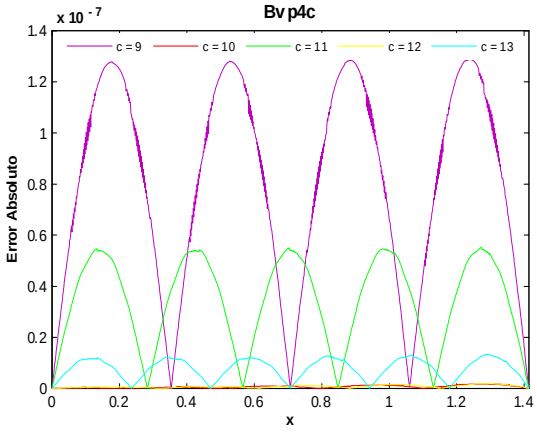


Fig 11. Error Absoluto de la solución Bvp4c para el ejemplo 1 con los valores del parámetro c=9,10,11,12,13 y tolerancias absoluta y relativa de 1e-8.
Cputime = 4.37 s.

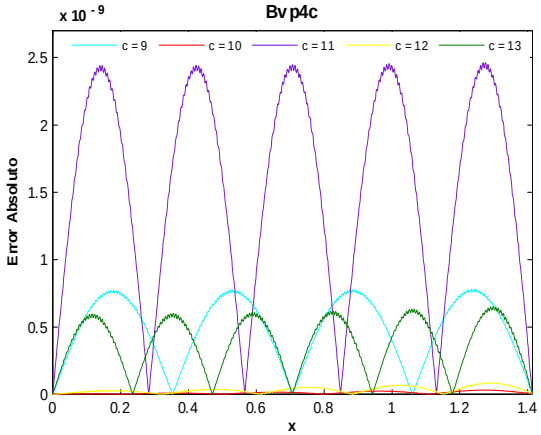


Fig 12. Error Absoluto de la solución Bvp4c para el ejemplo 1 con los valores del parámetro c=9,10,11,12,13 y tolerancias absoluta y relativa de 1e-12.
Cputime = 6.24 s.

Como se puede ver en las anteriores figuras, al disminuir los valores de las tolerancias absoluta y relativa de $1e-6$ y $1e-3$ a $1e-12$ y $1e-12$ respectivamente, el orden del error calculado con la solución bvp4c mejoró. Considerando valores menores, el orden empeora, por ejemplo las figuras 2,3,4 muestran que para $c = 4$ el orden del error absoluto es de $-3, -9, -13$ con valores de tolerancias absoluta y relativa de $1e-6$ y $1e-3$, $1e-8$ y $1e-8$, $1e-12$ y $1e-12$, respectivamente. Tomando $1e-14$ y $1e-14$ el orden empeora a -12 y tarda $51.31s$. En este ejemplo entre más exactitud se pida a la solución, bvp4c toma más tiempo en calcularla, aunque como se ve en las figuras 5 y 9, el orden no necesariamente es mejor que con menor exactitud. Como se puede ver en la figura 12 para el valor de tolerancias $1e-12$ y $c = 11$ el error empeora y al exigir mayor exactitud, por ejemplo $1e-14$, el orden se mantiene.

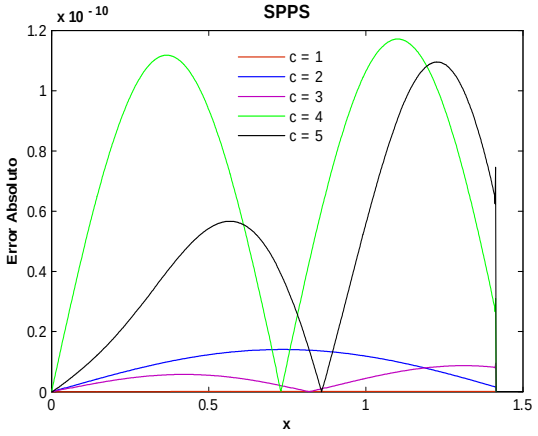


Fig 13. Error Absoluto de la solución SPPS para el ejemplo 1 con los valores del parámetro c=1,2,3,4,5 considerando 20 potencias y 1000 puntos.
Cputime = 27.41 s.

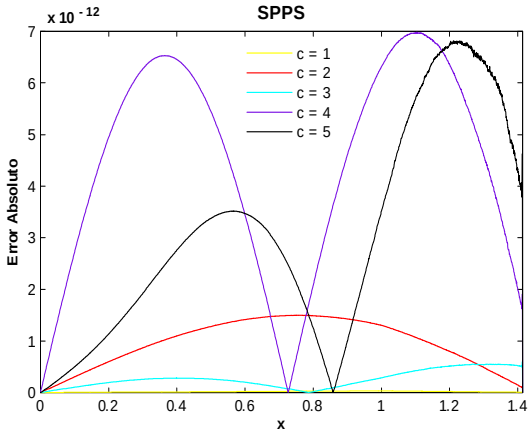


Fig 14. Error Absoluto de la solución SPPS para el ejemplo 1 con los valores del parámetro c=1,2,3,4,5 considerando 20 potencias y 2000 puntos.
Cputime = 54.01 s.

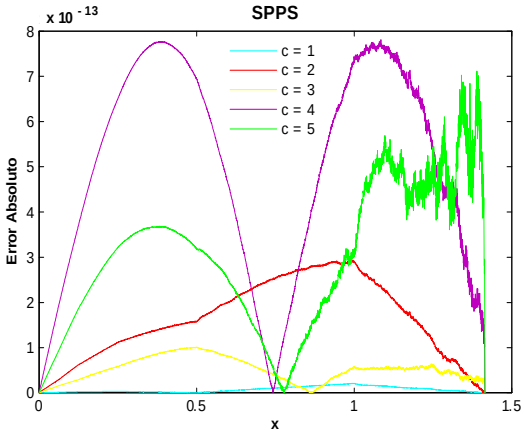


Fig 15. Error Absoluto de la solución SPPS para el ejemplo 1 con los valores del parámetro c=1,2,3,4,5 considerando 20 potencias y 4000 puntos.
Cputime = 106.86 s.

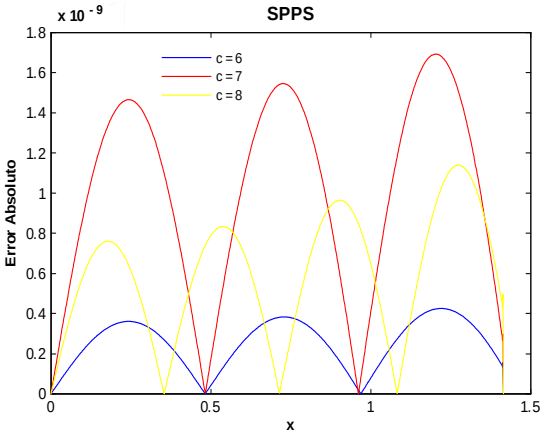


Fig 16. Error Absoluto de la solución SPPS para el ejemplo 1 con los valores del parámetro c=6,7,8 considerando 25 potencias y 1000 puntos.
Cputime = 20.72 s.

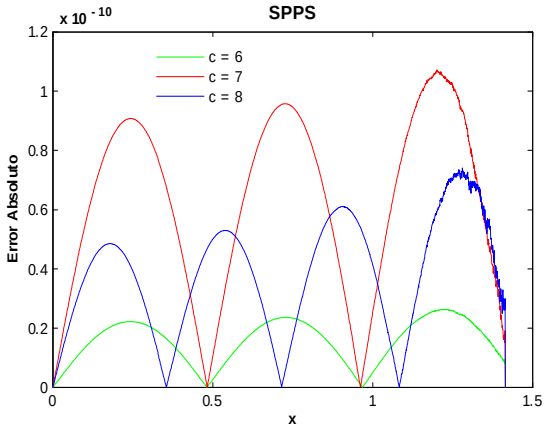


Fig 17. Error Absoluto de la solución SPPS para el ejemplo 1 con los valores del parámetro c=6,7,8 considerando 25 potencias y 2000 puntos.
Cputime = 40.73 s.

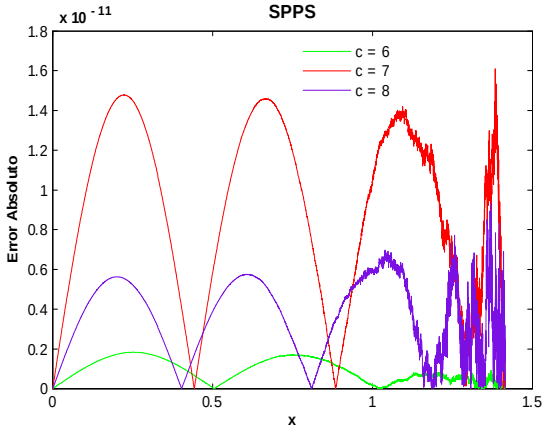


Fig 18. Error Absoluto de la solución SPPS para el ejemplo 1 con los valores del parámetro c=6,7,8 considerando 25 potencias y 3100 puntos.
Cputime = 62.23 s.

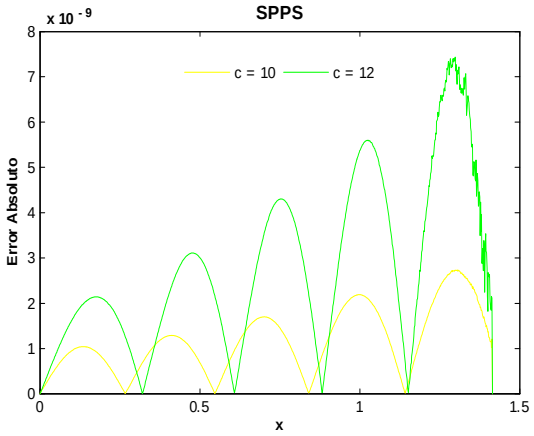


Fig 19. Error Absoluto de la solución SPPS para el ejemplo 1 con los valores del parámetro c=10,12 considerando 35 potencias y 1000 puntos. Cputime = 18.71 s.

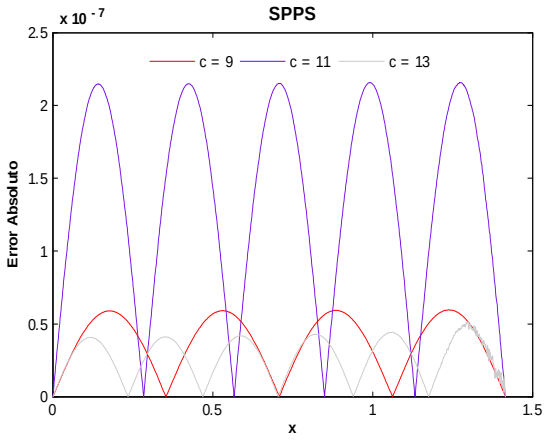


Fig 20. Error Absoluto de la solución SPPS para el ejemplo 1 con los valores del parámetro c=9,11,13 considerando 35 potencias y 1000 puntos. Cputime = 27.4 s.

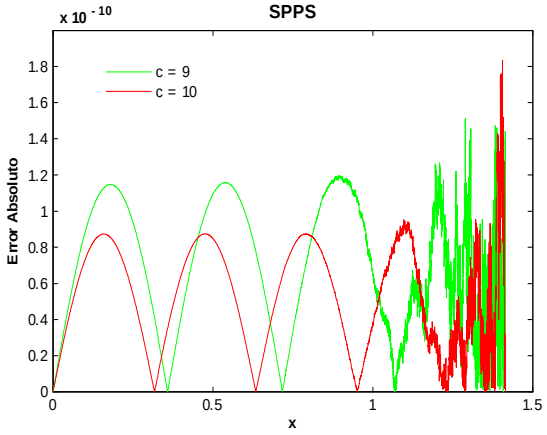


Fig 21. Error Absoluto de la solución SPPS para el ejemplo 1 con los valores del parámetro c=9,10 considerando 35 potencias y 4000 puntos. Cputime = 74.67 s.

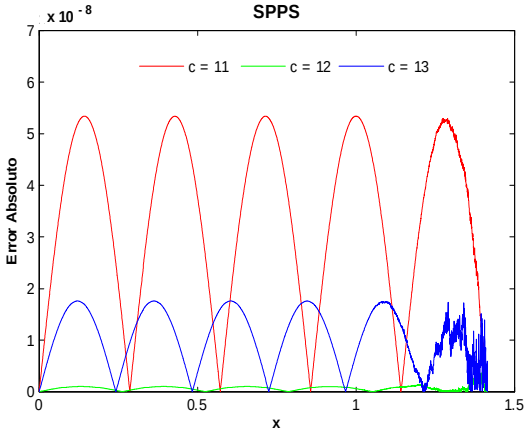


Fig 22. Error Absoluto de la solución SPPS para el ejemplo 1 con los valores del parámetro c=11,12,13 considerando 35 potencias y 4000 puntos. Cputime = 111.53 s.

En general en este ejemplo usando SPPS, al aumentar la cantidad de puntos el orden del error absoluto mejoró. Por ejemplo para $c = 4$ las figuras 13,14 y 15 muestran que el orden calculado con la solución SPPS mejora al aumentar la cantidad de puntos de -10 (con 20 potencias y 1000 puntos) hasta -13(con 20 potencias y 4000 puntos). Cabe señalar que el número de potencias es tal que si se aumenta, el orden obtenido con esta nueva cantidad es igual, esto siempre que se considere el mismo número de puntos. Al aumentar los puntos es posible aumentar las potencias y obtener un mejor orden. Aunque en el caso de las figuras 18 y 22 al aumentar el número de puntos no mejoró el orden del error.

En ambos métodos conforme el parámetro c crece, el orden del error empeora, de modo que es necesario mejorar la exactitud para obtener mejores órdenes. Para bvp4c se disminuyen los valores de las tolerancias absoluta y relativa y para SPPS se incrementa el número de puntos.

Determinante tiende a cero

Recuérdese que (40) tiene solución única si y sólo si el determinante Δ de (41) es no nulo. Entonces es interesante saber cómo se comporta cada método cuando el determinante Δ tiende a cero. Puesto que

$$\Delta = \sin(cb - ca),$$

$\Delta = 0$ si y sólo si $c(b-a) = k\pi, k \in \mathbb{Z}$. Considérese los valores del intervalo y de frontera $\alpha = 0, \beta = 1, a = 0, b = \pi$. Entonces para nuestro propósito tómese valores de c cercanos a algún entero tales como $c = 3.0001, 7.0001, 3.00001, 7.00001$. De esta manera Δ es cercano a cero.

Las siguientes gráficas contienen el error absoluto obtenido con las soluciones Bvp4c y SPPS con los valores anteriores.

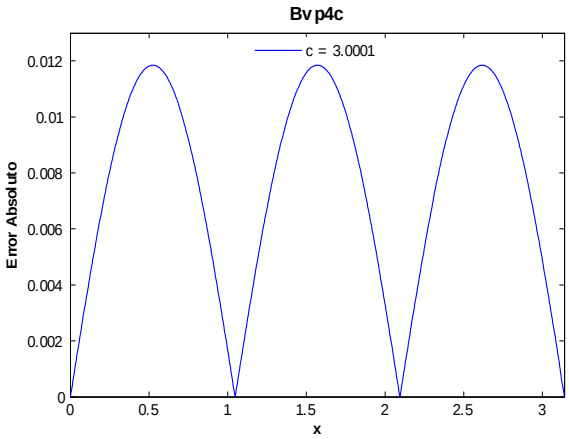


Fig 23. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=3.0001 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 1.6 s.

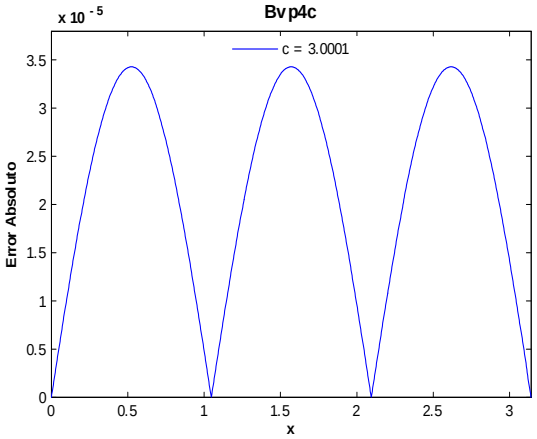


Fig 24. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=3.0001 y tolerancias absoluta de 1e-8 y relativa de 1e-10. Cputime = 4.53 s.

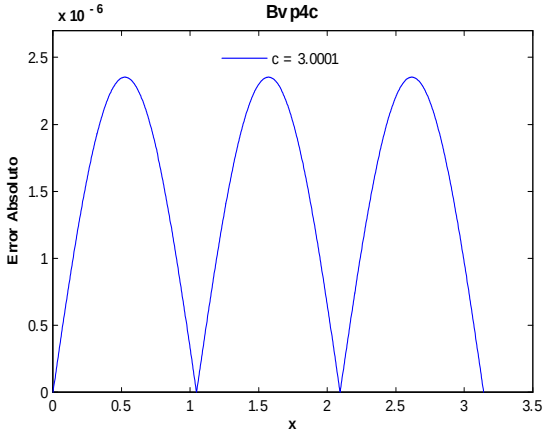


Fig 25. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=3.0001 y tolerancias absoluta de 1e-10 y relativa de 1e-12. Cputime = 4.87 s.

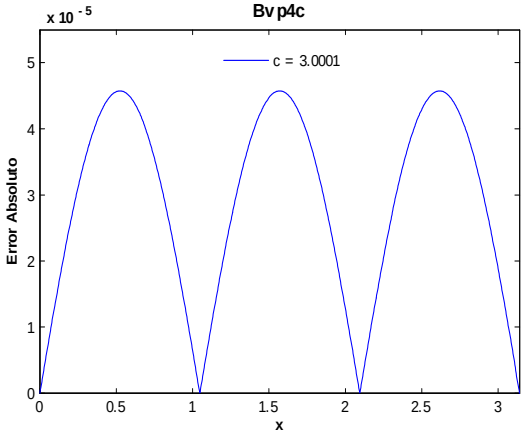


Fig 26. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=3.0001 y tolerancias absoluta de 1e-12 y relativa de 1e-14. Cputime = 2.17 s.

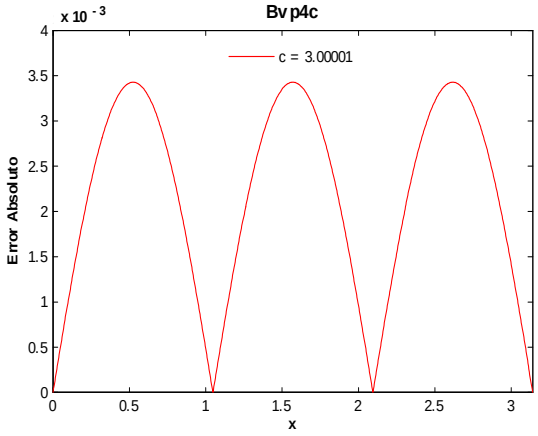


Fig 27. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=3.00001 y tolerancias absoluta de 1e-8 y relativa de 1e-10. Cputime = 5.16 s.

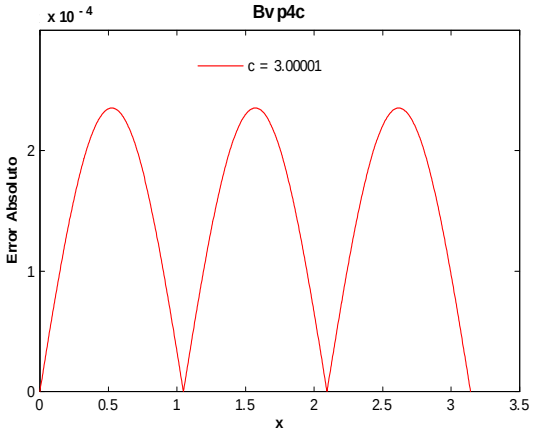


Fig 28. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=3.00001 y tolerancias absoluta de 1e-10 y relativa de 1e-12. Cputime = 4.76 s.

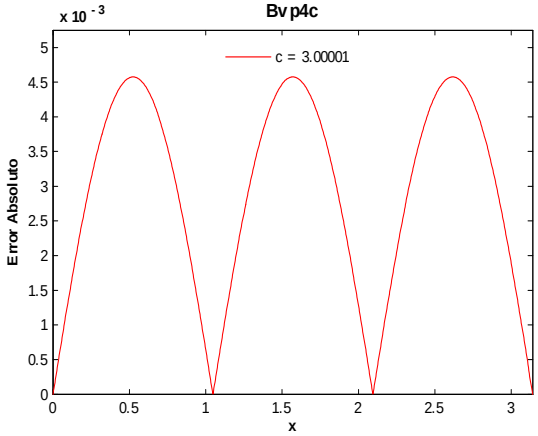


Fig 29. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=3.00001 y tolerancias absoluta de 1e-12 y relativa de 1e-14. Cputime = 2.14 s.

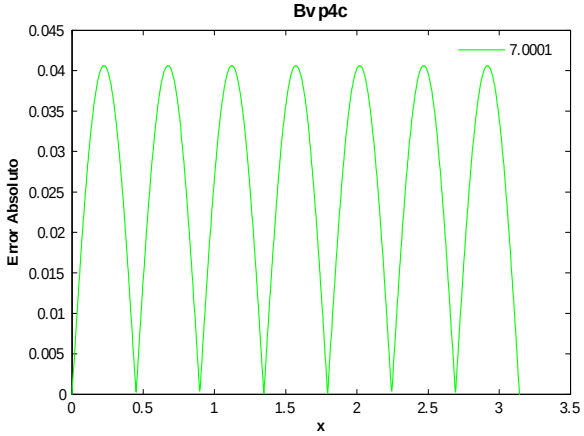


Fig 30. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=7.0001 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 4.01 s.

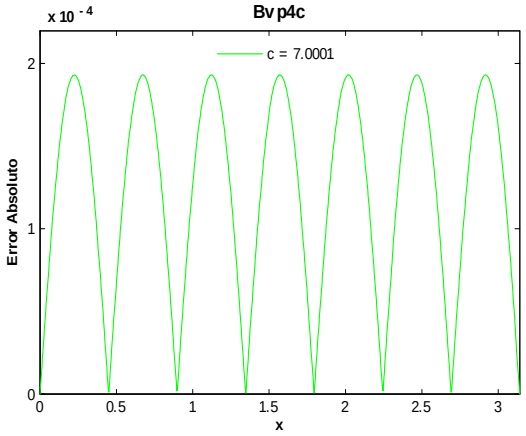


Fig.31. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=7.0001 y tolerancias absoluta de 1e-8 y relativa de 1e-10. Cputime = 4.59 s.

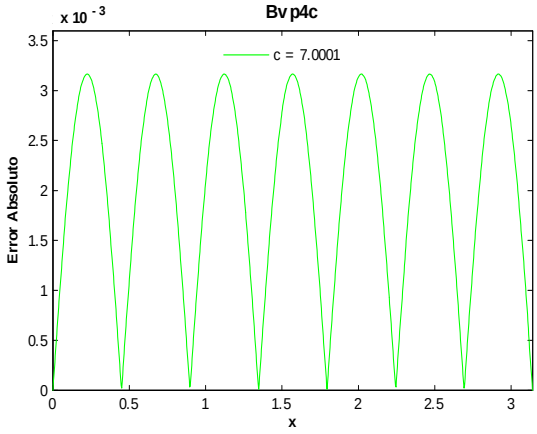


Fig.32 Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=7.0001 y tolerancias absoluta de 1e-10 y relativa de 1e-12. Cputime = 1.69 s.

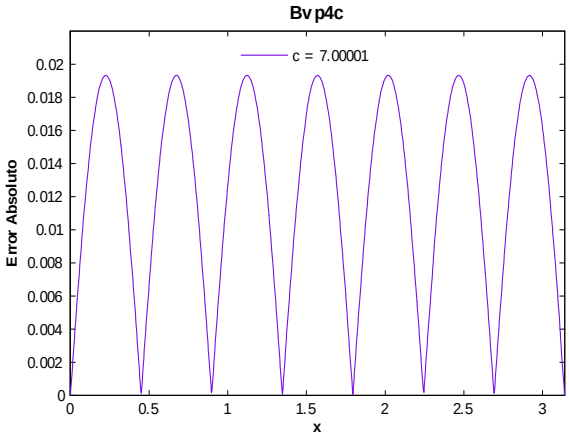


Fig 33. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=7.00001 y tolerancias absoluta de 1e-8 y relativa de 1e-10. Cputime = 3.75 s.

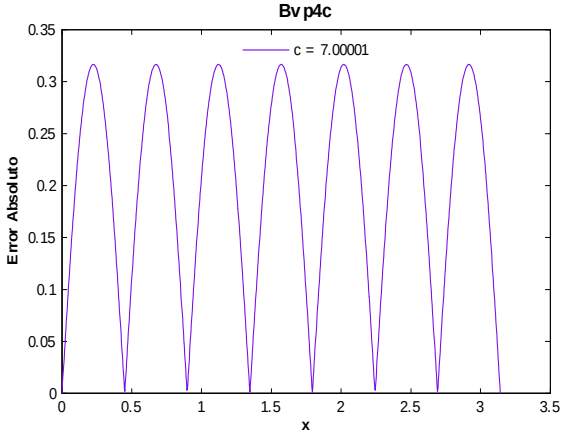


Fig 34. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=7.00001 y tolerancias absoluta de 1e-10 y relativa de 1e-12. Cputime = 1.79 s.

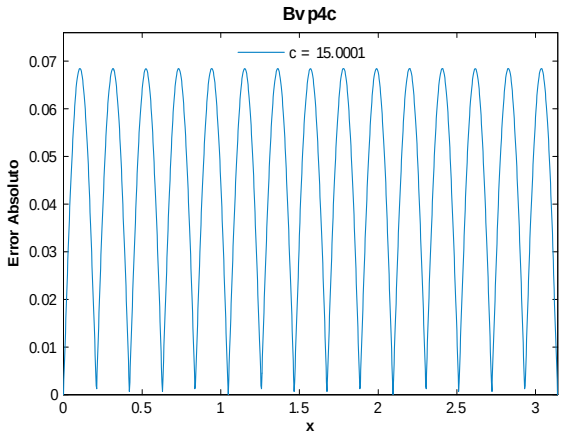


Fig 35. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=15.0001 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 6.42 s.

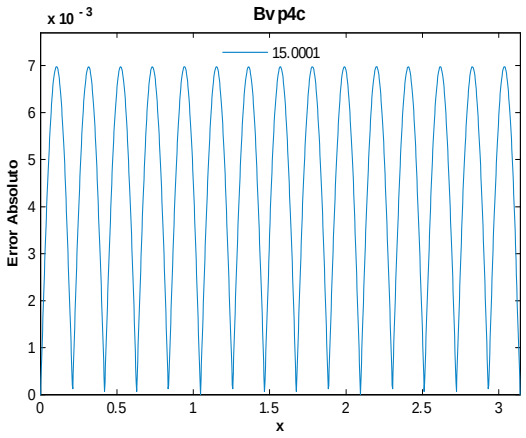


Fig 36. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=15.0001 y tolerancias absoluta de 1e-8 y relativa de 1e-10. Cputime = 4.78 s.

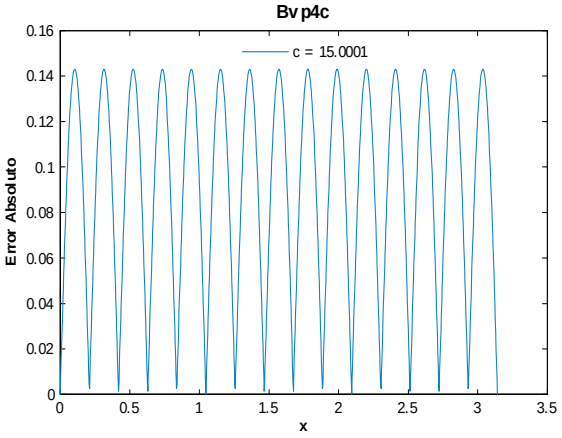


Fig 37. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=15.0001 y tolerancias absoluta de 1e-10 y relativa de 1e-12. Cputime = 1.54 s.

En este caso no se consideraron los valores de tolerancias por defecto pues no calcularon una solución. Como muestran las figuras 23-25 al disminuir los valores de las tolerancias absoluta y relativa el orden calculado con bvp4c mejora. Aunque las figuras 26, 29, 32, 34 y 37 muestran que puede empeorar al tomar valores muy pequeños en las tolerancias. Al hacer tender c a un entero, el orden empeora, por ejemplo la figura 33 muestra que para $c = 7.00001$ el orden obtenido fue de -2, mientras que para $c = 7.0001$ se obtuvo un orden de -4. Cuando se toma el valor $c = 15.00001$ bvp4c ya no calcula una solución.

Una manera de mejorar el orden obtenido para este ejemplo es incrementar el valor de los puntos malla (`NMax`). A continuación se presenta una tabla que contiene la comparación de los órdenes obtenidos al considerar distintos valores de `NMax` para los valores de $c = 7.00001, 15.0001, 15.00001, 15.000001$ incluyendo el caso presentado en las anteriores figuras, es decir cuando `NMax` $= 5000$.

Valor de c	Orden del Error	Valor de NMax	AbsTol, RelTol
7.00001	-2	5000	$1e-8, 1e-10$
	-5	25000	$1e-10, 1e-12$
15.0001	-3	5000	$1e-8, 1e-10$
	-5	10000	$1e-8, 1e-10$
15.00001	-3	130000	$1e-8, 1e-10$
15.000001	-3	35000	$1e-10, 1e-12$

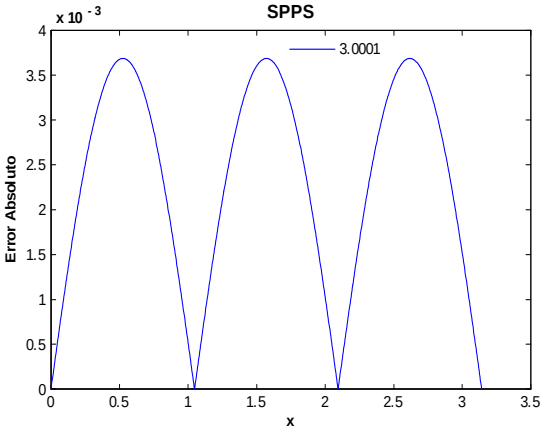


Fig 38. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=3.0001 considerando 25 potencias y 1000 puntos. Cputime = 6.83 s.

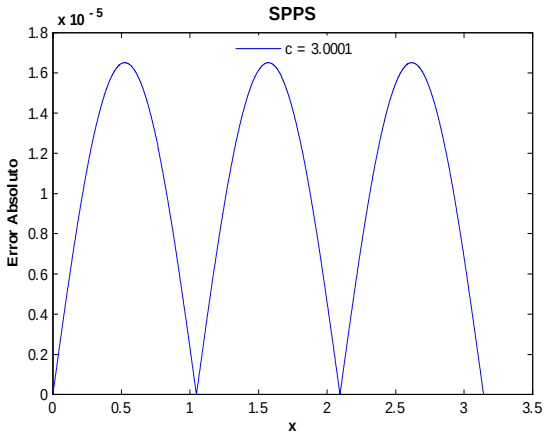


Fig 39. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=3.0001 considerando 25 potencias y 4000 puntos. Cputime = 25.9 s.

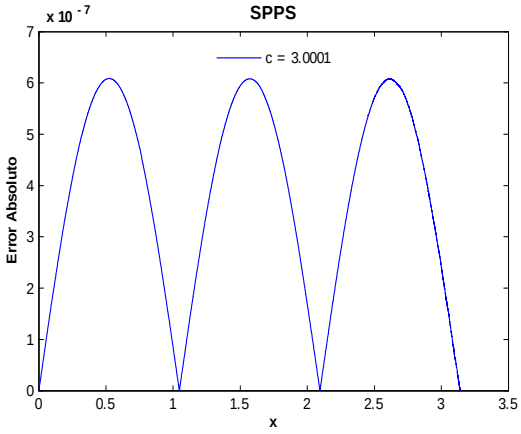


Fig 40. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=3.0001 considerando 30 potencias y 9000 puntos. Cputime = 71.58s.

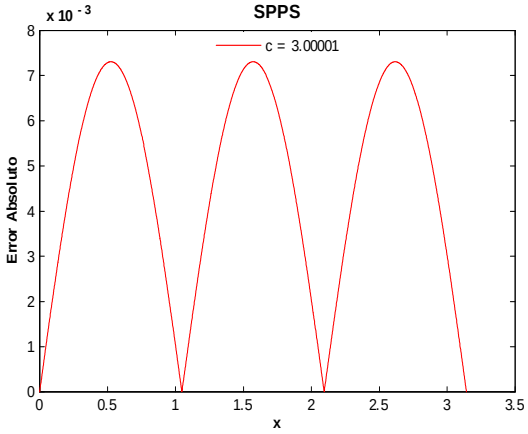


Fig 41. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=3.00001 considerando 25 potencias y 2500 puntos. Cputime = 16.26 s.

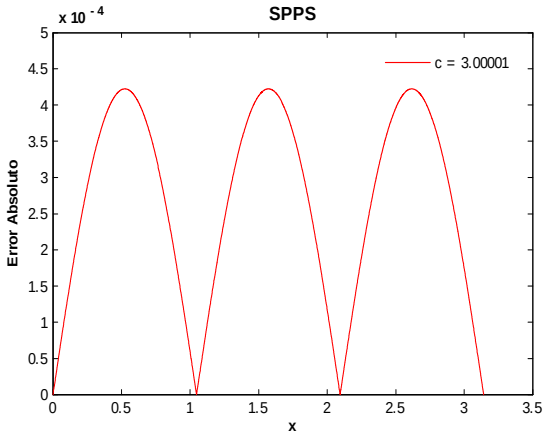


Fig 42. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=3.00001 considerando 25 potencias y 5000 puntos. Cputime = 39.17s.

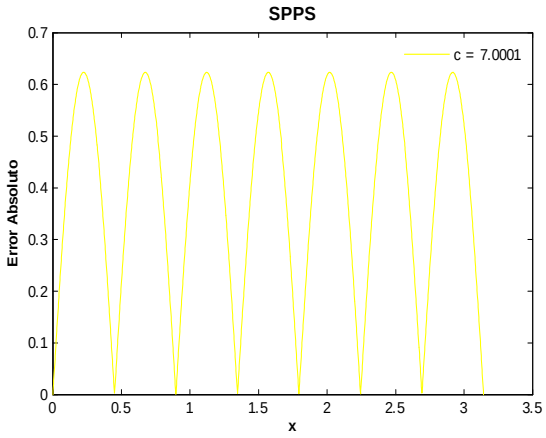


Fig 43. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=7.0001 considerando 40 potencias y 4000 puntos. Cputime = 41.04 s.

En las figuras 38-40 podemos ver que aumentando la cantidad de puntos mejora el orden del error, el mejor orden obtenido fue de -7 . Incluso cuando $c = 3.00001$ se obtuvo un orden de -4 con 25 potencias y 5000 puntos. Pero cuando se incrementa el parámetro c, como se ve en la figura 42 el orden del error es de -1 para $c = 7.0001$. Si se toma $c = 15.0001$, no se obtiene una solución.

Ahora consideraremos los siguientes valores de los parámetros $\alpha = 0, \beta = 1, a = 0, b = 1/2$, con el objetivo de saber como se comportan las soluciones obtenidas con SPPS y bvp4c al disminuir la longitud del intervalo. Con éstos valores para que el determinante

$$\Delta = \sin(cb - ca),$$

sea cero se debe cumplir $c = 2k\pi$, $k \in \mathbb{Z}$. De modo que consideraremos los valores $c = 2.0001\pi, 6.0001\pi$, $12.0001\pi, 2.00001\pi, 6.00001\pi, 12.00001\pi$. Por la sección 7 sabemos que el método SPPS obtiene mejores aproximaciones en intervalos pequeños, por lo se espera que la implementación en Matlab se comporte de forma análoga.

Para bvp4c utilizaremos los valores de tolerancias $1e-8, 1e-10, 1e-12, 1e-14$.

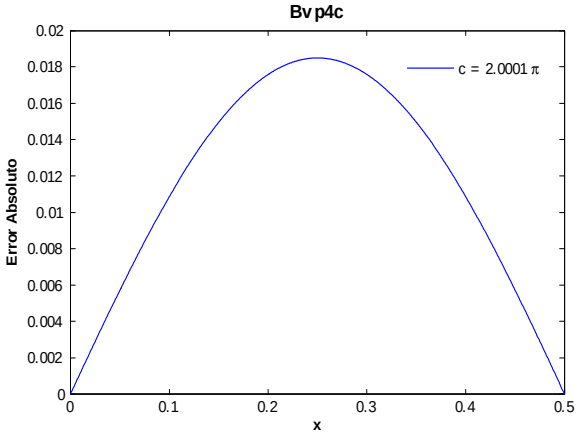


Fig 44. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=2.0001π y tolerancias absoluta y relativa de 1e-8.
Cputime = 0.62 s.

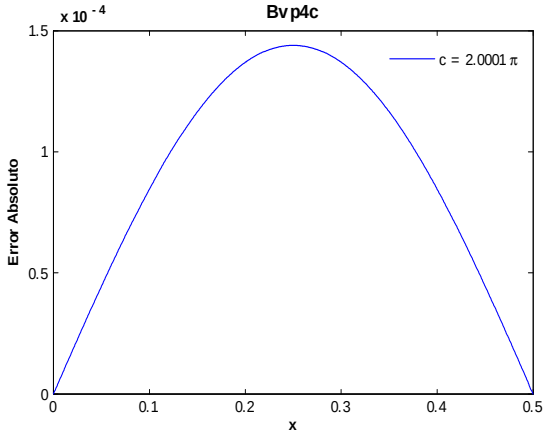


Fig 45. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=2.0001π y tolerancias absoluta y relativa de 1e-10.
Cputime = 2.88 s.

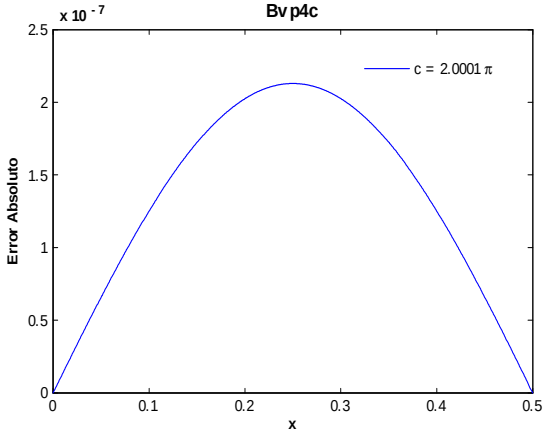


Fig 46. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=2.0001π y tolerancias absoluta y relativa de 1e-12.
Cputime = 25.1 s.

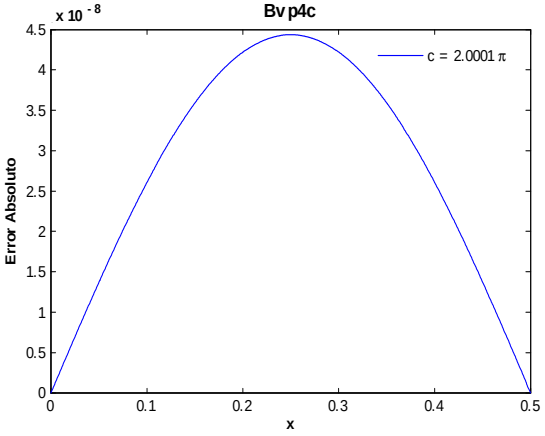


Fig 47. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=2.0001π y tolerancias absoluta y relativa de 1e-14.
Cputime = 5.14 s.

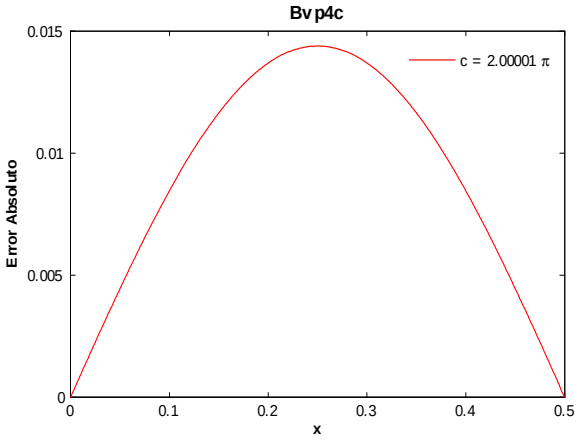


Fig 48. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=2.00001π y tolerancias absoluta y relativa de 1e-10.
Cputime = 2.34 s.

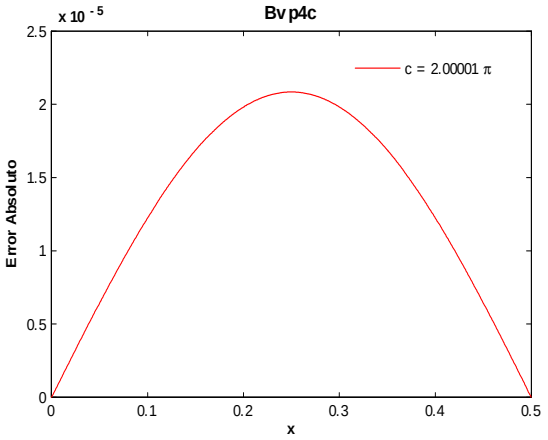


Fig 49. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=2.00001π y tolerancias absoluta y relativa de 1e-12.
Cputime = 25.06 s.

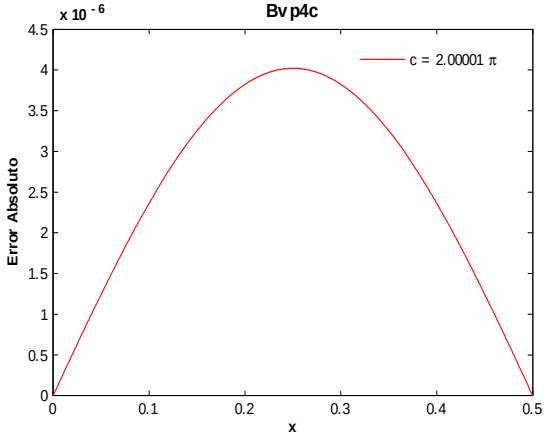


Fig 50. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=2.00001π y tolerancias absoluta y relativa de 1e-14.
Cputime = 5.92 s.

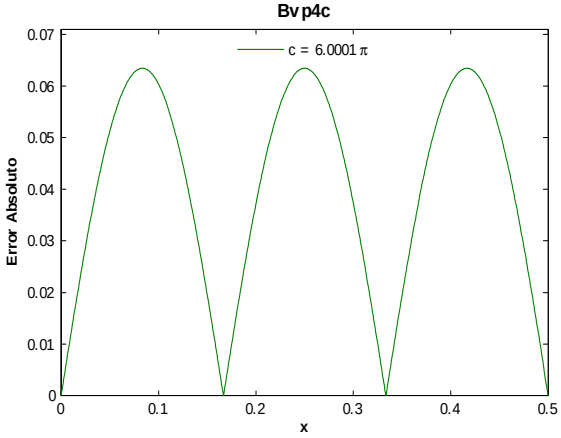


Fig 51. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=6.0001π y tolerancias absoluta y relativa de 1e-8.
Cputime = 1.47 s.

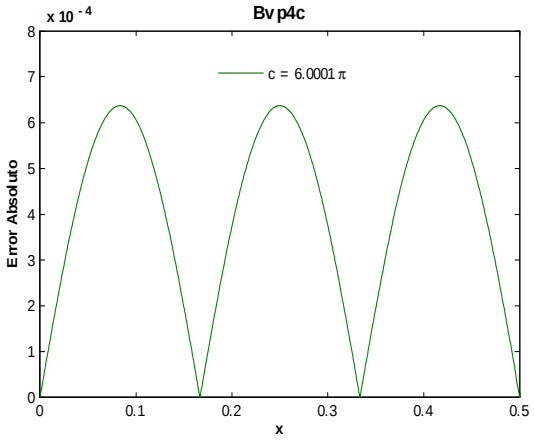


Fig 52. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=6.0001π y tolerancias absoluta y relativa de 1e-10.
Cputime = 7.24 s.

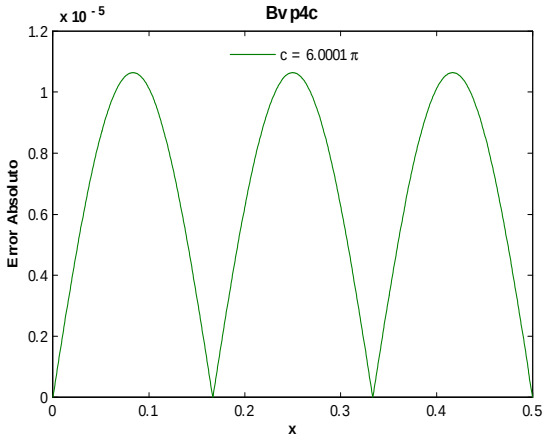


Fig 53. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=6.0001π y tolerancias absoluta y relativa de 1e-12.
Cputime = 5.52 s.

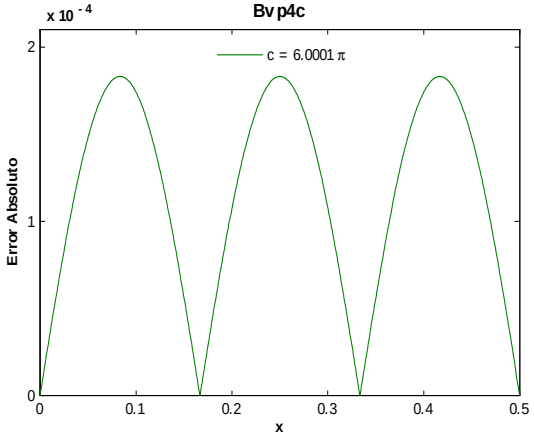


Fig 54. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=6.0001π y tolerancias absoluta y relativa de 1e-14.
Cputime = 2.35 s.

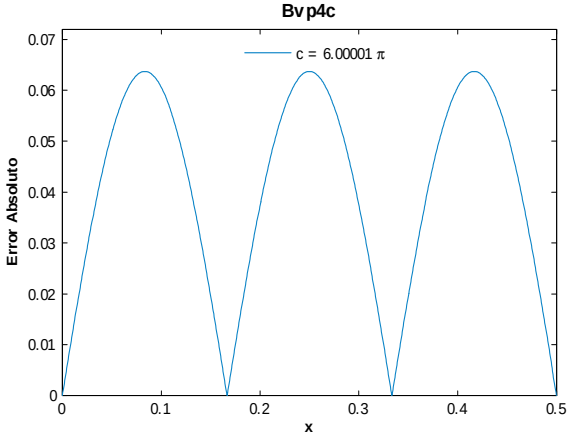


Fig 55. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=6.00001π y tolerancias absoluta y relativa de 1e-10.
Cputime = 5.79 s.

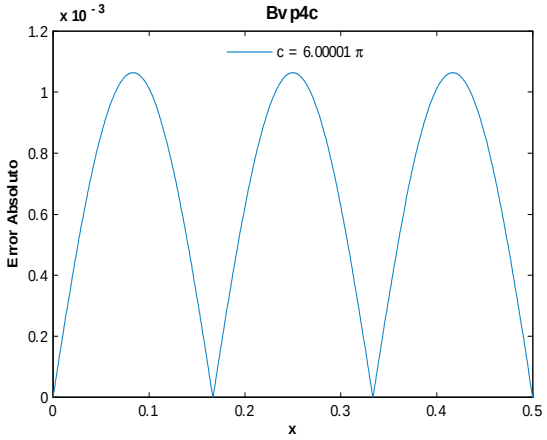


Fig 56. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=6.00001π y tolerancias absoluta y relativa de 1e-12.
Cputime = 4.5 s.

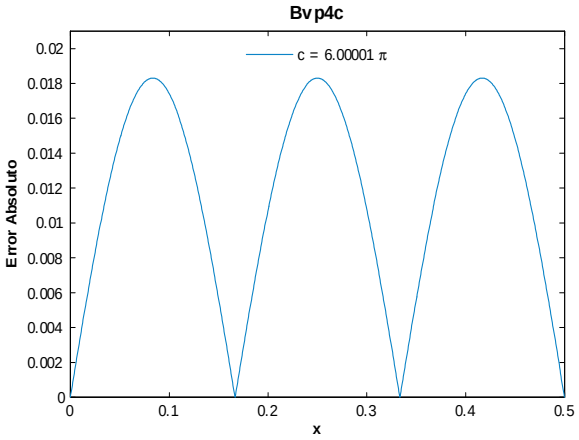


Fig 57. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=6.00001π y tolerancias absoluta y relativa de 1e-14.
Cputime = 2.06 s.

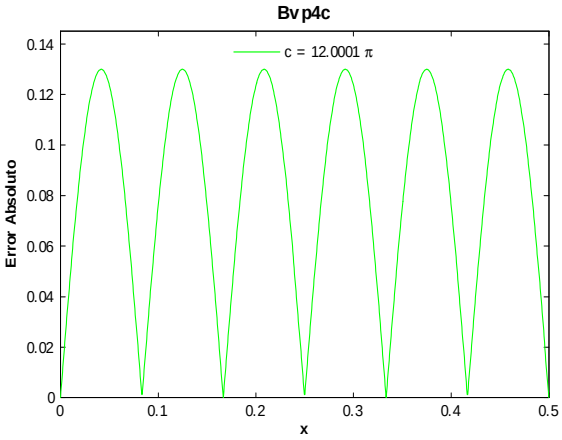


Fig 58. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=12.0001π y tolerancias absoluta y relativa de 1e-8.
Cputime = 2.85 s.

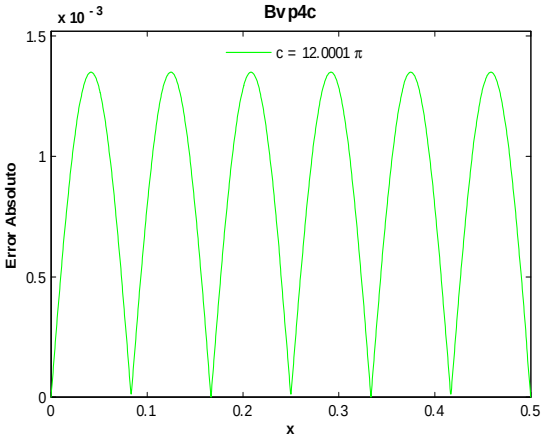


Fig 59. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=12.0001π y tolerancias absoluta y relativa de 1e-10.
Cputime = 14.87 s.

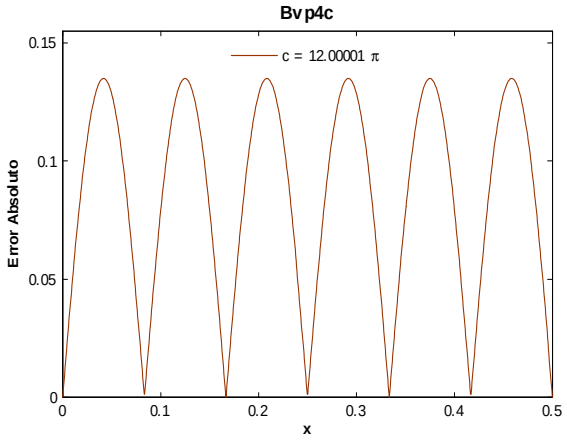


Fig 60. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=12.00001π y tolerancias absoluta y relativa de 1e-10.
Cputime = 7.84 s.

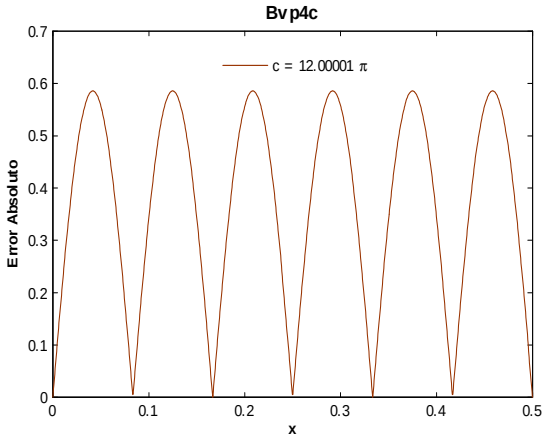


Fig 61. Error Absoluto de la solución Bvp4c para el ejemplo 1 con el valor del parámetro c=12.00001π y tolerancias absoluta y relativa de 1e-12.
Cputime = 1.93 s.

En este ejemplo los valores de tolerancias por defecto no calcularon una solución para ningún valor de c, es por eso que no se incluyeron. En algunos casos tampoco se incluyó el valor $1e-8$ pues no se obtenía una solución. En general el orden del error mejoró al disminuir los valores de las tolerancias, aunque como muestran las figuras 54 y 57 puede llegar a empeorar al tomar valores pequeños como $1e-14$. Al hacer tender c a un múltiplo par de π el orden del error empeora. Comparando este ejemplo con el anterior en donde se consideró el intervalo de longitud π y valores $c = 3.0001, 7.0001, 15.0001, 3.00001, 7.00001$, se ve que en este caso bvp4c obtuvo mejores órdenes de error aún cuando los valores aquí considerados fueron mayores que los anteriores, sólo que la longitud del intervalo fue menor.

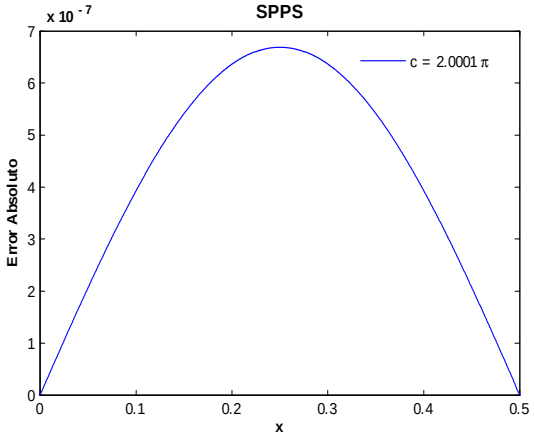


Fig 62. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=2.0001π considerando 20 potencias y 2000 puntos.
Cputime = 10.42 s.

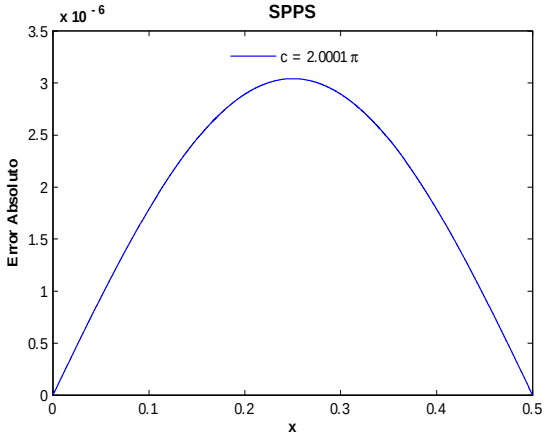


Fig 63. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=2.0001π considerando 20 potencias y 5000 puntos.
Cputime = 25.28 s.

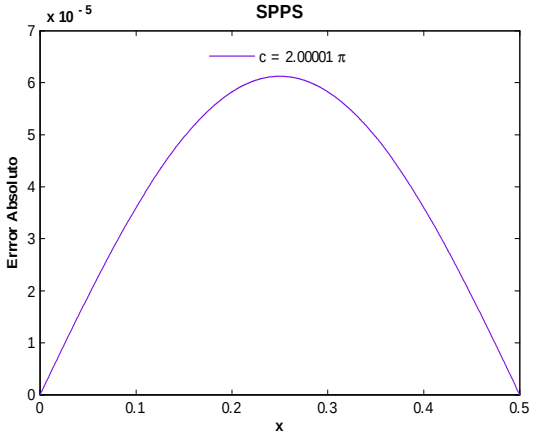


Fig 64. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=2.00001π considerando 20 potencias y 2000 puntos.
Cputime = 10.68 s.

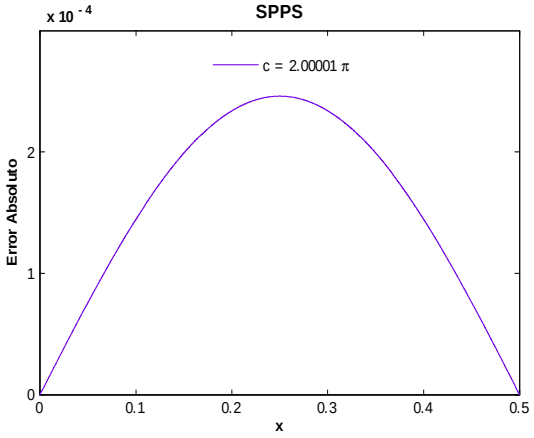


Fig 65. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=2.00001π considerando 20 potencias y 5000 puntos.
Cputime = 25.58 s.

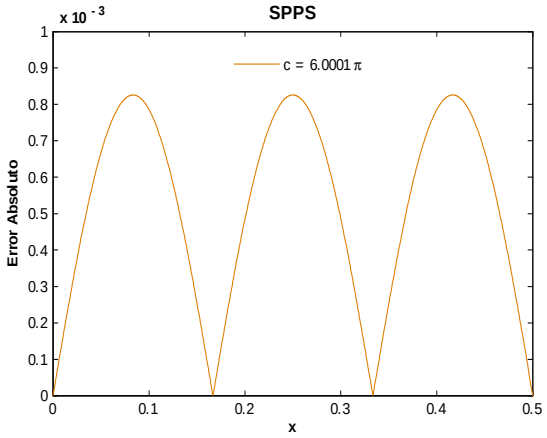


Fig 66. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=6.0001π considerando 40 potencias y 2000 puntos.
Cputime = 20.49 s.

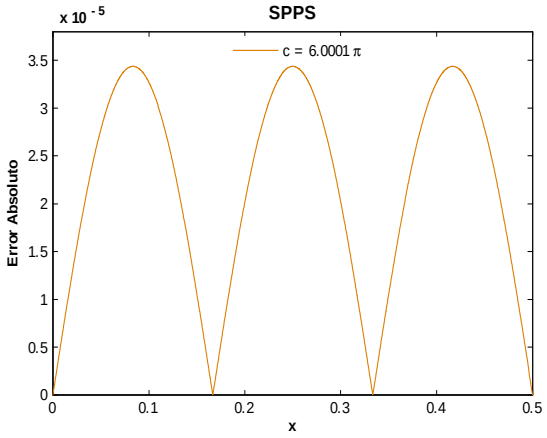


Fig 67. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=6.0001π considerando 40 potencias y 5000 puntos.
Cputime = 50.64 s.

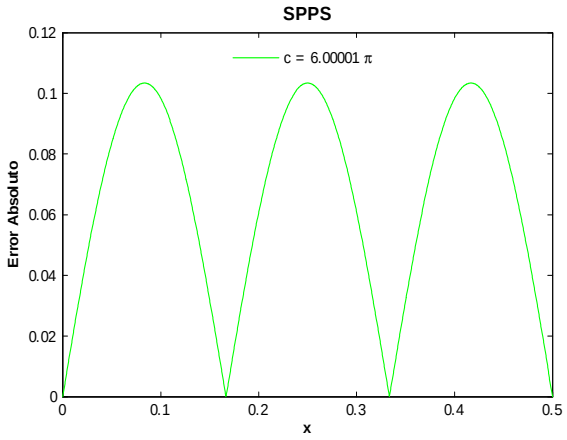


Fig 68. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=6.00001π considerando 40 potencias y 2000 puntos. Cputime = 20.6 s.

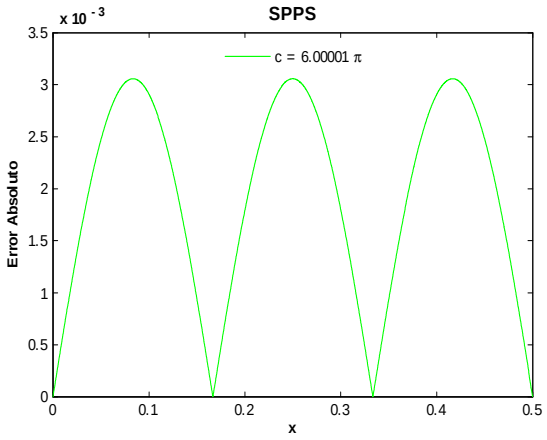


Fig 69. Error Absoluto de la solución SPPS para el ejemplo 1 con el valor del parámetro c=6.00001π considerando 40 potencias y 5000 puntos. Cputime = 50.44 s.

Nótese que en este caso la longitud del intervalo es de $\frac{1}{2}$ y que el parámetro c es grande en comparación con el ejemplo anterior, donde se considera un intervalo de longitud π y $c = 3.0001, 7.0001, 15.0001, 3.00001,$ 7.00001. Esto es, el valor de c se aumentó pero la longitud del intervalo se disminuyó, y se obtuvieron mejores órdenes para valores de c grandes. En efecto, para el caso $[0, \pi]$ el mejor orden obtenido para $c = 3.0001$ fue de -7, para $c = 3.00001$ fue de -4 y para $c = 7.0001$ fue de -1, en cambio para el intervalo $[0, 1/2]$, para $c = 2.0001\pi$ se obtuvo un mejor orden de -7, para $c = 2.00001\pi$ fue de -5, para $c = 6.0001\pi$ fue de -5, para $c = 6.00001\pi$ fue de -3 y para $c = 12.0001$ fue de -2. Claramente la aproximación fue mejor para el intervalo $[0, 1/2]$, aún considerando valores más grandes del parámetro c.

Obsérvese que la estimación de las soluciones particulares v_1 y v_2 obtenida con (36) y (37) para $c = 2.0001\pi, 2.00001\pi$ es de $N = 10$, para obtener un orden de error menor o igual a -11. Para $c = 6.0001\pi$, 6.00001π basta con tomar $N = 20$ para obtener un orden de error igual o menor a -11. Para $c = 12.0001\pi$, 12.00001π es necesario tomar $N = 40$ para obtener un orden de error igual o menor a -19. Como se puede ver la convergencia es muy rápida para este caso. Una de las razones por la que no se obtuvo los órdenes esperados es por el error númerico generado al calcular las integrales $\widetilde{X}_0^{(n)}$ y $X_0^{(n)}$. Otra razón importante es que en general por la definición del método SPPS para un problema de tipo Dirichlet en un intervalo $[0, b]$, las constantes c_1 y c_2 de la solución general $y = c_1 v_1 + c_2 v_2$ estan dadas por

$$\begin{aligned} c_1 &= \alpha \\ c_2 &= \frac{\beta - \alpha v_1(b)}{v_2(b)} \end{aligned}$$

y $\Delta = v_1(0)v_2(b) - v_1(b)v_2(0) = v_2(b)$, lo cuál hace que la constante c_2 tienda a infinito cuando el determinante Δ tiende a cero. De manera que en este ejemplo aún cuando se pueda obtener una buena aproximación de las soluciones v_1 y v_2, al hacer tender el deteminante a cero la constante c_2 hace que la solución general $y = c_1 v_1 + c_2 v_2$ pierda precisión.

Ejemplo 2

Solución Exacta

Considérese el problema

$$\begin{aligned} y'' + q(x)y &= 0 \\ y(0) &= \alpha \\ y(b) &= \beta \end{aligned} \tag{43}$$

donde $c(\neq 0), b, \alpha, \beta \in \mathbb{R}$, y q está dada por

$$q(x) = \begin{cases} -c^2 & 0 \leq x < \frac{b}{2} \\ 0 & x = \frac{b}{2} \\ c^2 & \frac{b}{2} < x \leq b. \end{cases}$$

Llamaremos y_1, y_2 a las soluciones de la ecuacion diferencial de (43) en $[0, \frac{b}{2})$ y $(\frac{b}{2}, b]$ respectivamente. Así, si $0 \leq x < \frac{b}{2}$ se tiene

$$\begin{aligned} y_1(x) &= c_1 e^{cx} + c_2 e^{-cx} \\ y_1(0) &= c_1 + c_2 = \alpha \end{aligned}$$

donde c_1 y c_2 son constantes arbitrarias. De donde

$$\begin{aligned} y_1(x) &= (\alpha - c_2) e^{cx} + c_2 e^{-cx} \\ y_1'(x) &= c(\alpha - c_2) e^{cx} - c c_2 e^{-cx}. \end{aligned}$$

Si $\frac{b}{2} < x \leq b$ entonces para k_1 y k_2 constantes arbitrarias

$$\begin{aligned} y_2(x) &= k_1 \cos(cx) + k_2 \sin(cx) \\ y_2(b) &= k_1 \cos(cb) + k_2 \sin(cb) = \beta, \end{aligned}$$

por lo tanto, suponiendo que $\cos(cb) \neq 0$, i.e. $cb \neq (2k+1)\pi/2$, $k \in \mathbb{Z}$, tenemos

$$\begin{aligned} y_2(x) &= \frac{\beta - k_2 \sin(cb)}{\cos(cb)} \cos(cx) + k_2 \sin(cx) \\ y_2'(x) &= -c \frac{\beta - k_2 \sin(cb)}{\cos(cb)} \sin(cx) + c k_2 \cos(cx). \end{aligned}$$

Para obtener la solución de (43) de manera que ésta sea continua y con derivada continua añadimos las condiciones de conjugación en $\frac{b}{2}$:

$$\begin{aligned} y_1\left(\frac{b}{2}\right) &= y_2\left(\frac{b}{2}\right) \\ y_1'\left(\frac{b}{2}\right) &= y_2'\left(\frac{b}{2}\right). \end{aligned}$$

Entonces se tiene

$$\begin{aligned} (\alpha - c_2) e^{\frac{cb}{2}} + c_2 e^{-\frac{cb}{2}} &= \frac{\beta - k_2 \sin(cb)}{\cos(cb)} \cos\left(\frac{cb}{2}\right) + k_2 \sin\left(\frac{cb}{2}\right) \\ c(\alpha - c_2) e^{\frac{cb}{2}} - c c_2 e^{-\frac{cb}{2}} &= -c \frac{\beta - k_2 \sin(cb)}{\cos(cb)} \sin\left(\frac{cb}{2}\right) + c k_2 \cos\left(\frac{cb}{2}\right). \end{aligned}$$

Reescribiendo las ecuaciones anteriores

$$\begin{aligned} 2\sinh\left(\frac{cb}{2}\right) c_2 - \left[\tan(cb)\cos\left(\frac{cb}{2}\right) - \sin\left(\frac{cb}{2}\right)\right] k_2 &= -\frac{\beta \cos\left(\frac{cb}{2}\right)}{\cos(cb)} + \alpha e^{\frac{cb}{2}} \\ 2\cosh\left(\frac{cb}{2}\right) c_2 + \left[\tan(cb)\sin\left(\frac{cb}{2}\right) + \cos\left(\frac{cb}{2}\right)\right] k_2 &= \frac{\beta \sin\left(\frac{cb}{2}\right)}{\cos(cb)} + \alpha e^{\frac{cb}{2}}. \end{aligned} \tag{44}$$

Por los Teoremas 4 y 5 el problema (43) tiene solución única si y sólo si

$$\Delta = \begin{vmatrix} 2\sinh\left(\frac{cb}{2}\right) & -\tan(cb)\cos\left(\frac{cb}{2}\right) + \sin\left(\frac{cb}{2}\right) \\ 2\cosh\left(\frac{cb}{2}\right) & \tan(cb)\sin\left(\frac{cb}{2}\right) + \cos\left(\frac{cb}{2}\right) \end{vmatrix} \neq 0.$$

Supongáse que $\Delta \neq 0$. Aplicando la regla de Cramer a (44)

$$\begin{aligned} c_2 &= \frac{\begin{vmatrix} -\frac{\beta\cos\left(\frac{cb}{2}\right)}{\cos(cb)} + \alpha e^{\frac{cb}{2}} & -\tan(cb)\cos\left(\frac{cb}{2}\right) + \sin\left(\frac{cb}{2}\right) \\ \frac{\beta\sin\left(\frac{cb}{2}\right)}{\cos(cb)} + \alpha e^{\frac{cb}{2}} & \tan(cb)\sin\left(\frac{cb}{2}\right) + \cos\left(\frac{cb}{2}\right) \end{vmatrix}}{\Delta} \\ &= \frac{-\frac{\beta}{\cos(cb)} + \alpha e^{\frac{cb}{2}}\left[\tan(cb)\left(\sin\left(\frac{cb}{2}\right) + \cos\left(\frac{cb}{2}\right)\right) + \left(\cos\left(\frac{cb}{2}\right) - \sin\left(\frac{cb}{2}\right)\right)\right]}{\Delta} \\ k_2 &= \frac{\begin{vmatrix} 2\sinh\left(\frac{cb}{2}\right) & -\frac{\beta\cos\left(\frac{cb}{2}\right)}{\cos(cb)} + \alpha e^{\frac{cb}{2}} \\ 2\cosh\left(\frac{cb}{2}\right) & \frac{\beta sen\left(\frac{cb}{2}\right)}{\cos(cb)} + \alpha e^{\frac{cb}{2}} \end{vmatrix}}{\Delta} \\ &= \frac{\frac{2\beta}{\cos(cb)}\left[\sin\left(\frac{cb}{2}\right)\sinh\left(\frac{cb}{2}\right) + \cos\left(\frac{cb}{2}\right)\cosh\left(\frac{cb}{2}\right)\right] - 2\alpha}{\Delta} \end{aligned} \tag{45}$$

Entonces la solución exacta de (43) es

$$y = \begin{cases} (\alpha - c_2)e^{cx} + c_2 e^{-cx} & 0 \leq x \leq \frac{b}{2} \\ \frac{\beta - k_2\sin(cb)}{\cos(cb)}\cos(cx) + k_2\sin(cx) & \frac{b}{2} < x \leq b \end{cases} \tag{46}$$

donde c_2 y k_2 están dadas en (45).

Soluciones Numéricas

En este ejemplo se resolverá el problema (43) en los intervalos $[0, \pi]$ y $\left[0, \frac{1}{2}\right]$ a fin de analizar el comportamiento de las soluciones utilizando el método SPPS y la función bvp4c en intervalos de distinta longitud.

I *Intervalo* $[0, \pi]$

Para poder usar la solución exacta (46) en este intervalo basta que se cumpla

$$\Delta = \begin{vmatrix} 2\sinh\left(\frac{c\pi}{2}\right) & -\left[\tan(cb)\cos\left(\frac{cb}{2}\right) - \sin\left(\frac{cb}{2}\right)\right] \\ 2\cosh\left(\frac{c\pi}{2}\right) & \tan(cb)\sin\left(\frac{cb}{2}\right) + \cos\left(\frac{cb}{2}\right) \end{vmatrix} \neq 0.$$

Por simplicidad tomemos a $c = 1, 2, 3, 4, 5$. Así tendremos

$$\Delta = 2\sinh\left(\frac{c\pi}{2}\right)\cos\left(\frac{c\pi}{2}\right) - 2\cosh\left(\frac{c\pi}{2}\right)\sin\left(\frac{c\pi}{2}\right).$$

Nótese que como c es entero positivo

$$\Delta = \begin{cases} 2\sinh\left(\frac{c\pi}{2}\right) & c\ par \\ -2\cosh\left(\frac{c\pi}{2}\right)\sin\left(\frac{c\pi}{2}\right) & c\ impar \end{cases}$$

de donde se sigue que $\Delta \neq 0$.

A continuación se muestran gráficas de algunos potenciales $q(x)$ y soluciones exactas, así como las de los errores absoluto y relativo obtenidas con las soluciones de los métodos Bvp4c y SPPS. Los valores de los parámetros utilizados son: $\alpha = 0, \beta = 3, a = 0, b = \pi, c = 1, 2, 3, 4, 5$.

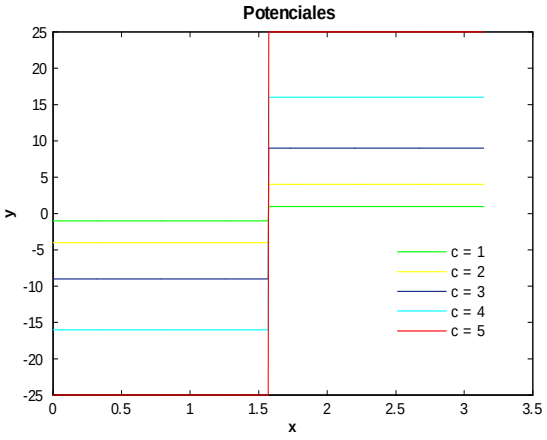


Fig. 70

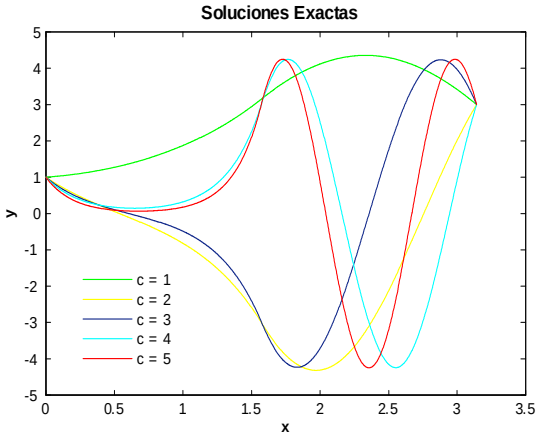


Fig.71

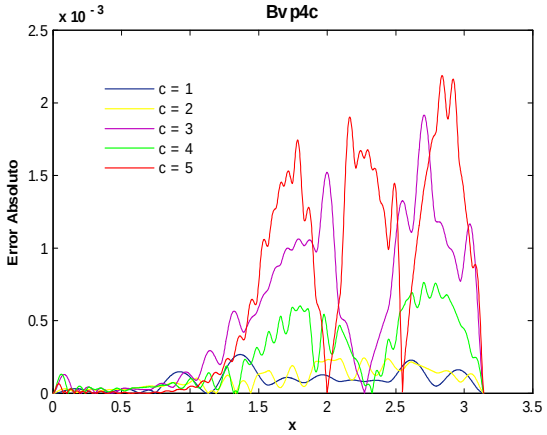


Fig.72. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta 1e-6 y relativa de 1e-3.
Cputime =1.36 s.

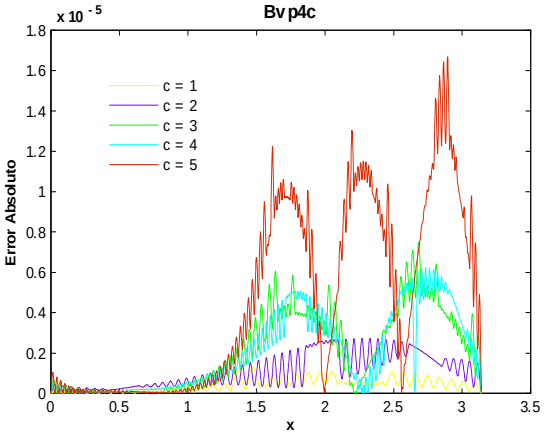


Fig. 73. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta y relativa de 1e-5.
Cputime =2.85 s.

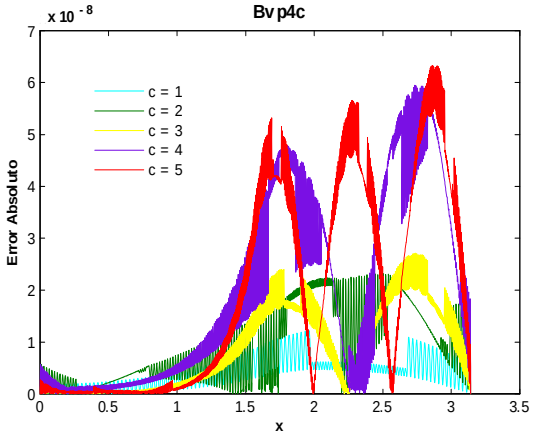


Fig. 74. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta y relativa de 1e-7.
Cputime =10.59 s.

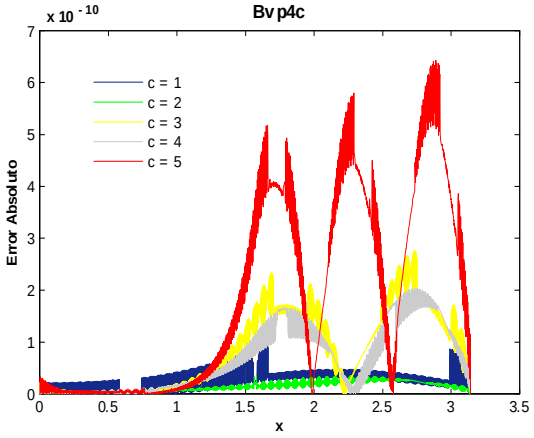


Fig. 75. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta y relativa de 1e-9.
Cputime = 2865.96 s.

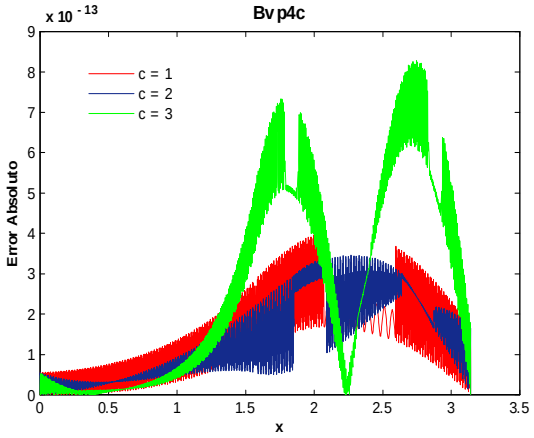


Fig. 76. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=1,2,3 y tolerancias absoluta y relativa de 1e-11.
Cputime =54.62 s.

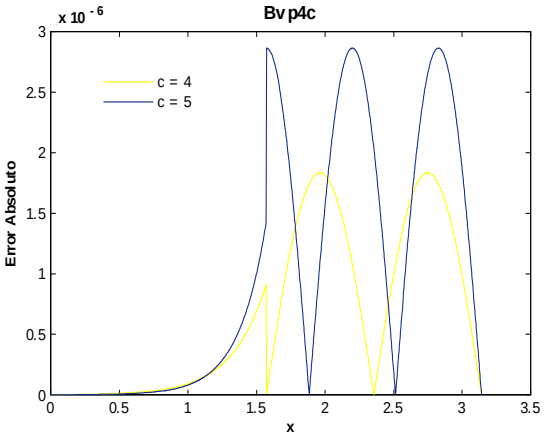


Fig. 77. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=4,5 y tolerancias absoluta y relativa de 1e-11.
Cputime =4.08 s.

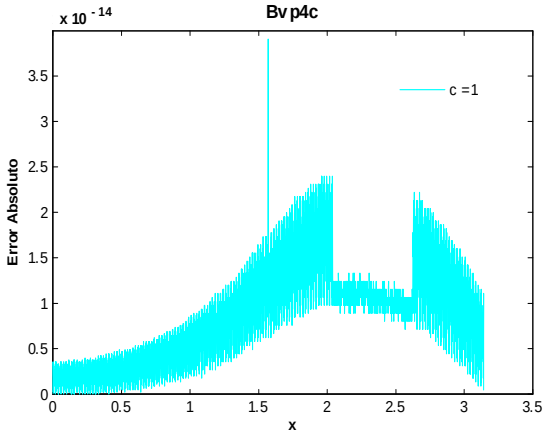


Fig. 78. Error Absoluto de la solución Bvp4c para el ejemplo 2 con el valor del parámetro c=1 y tolerancias absoluta y relativa de 1e-12.
Cputime =14.72 s.

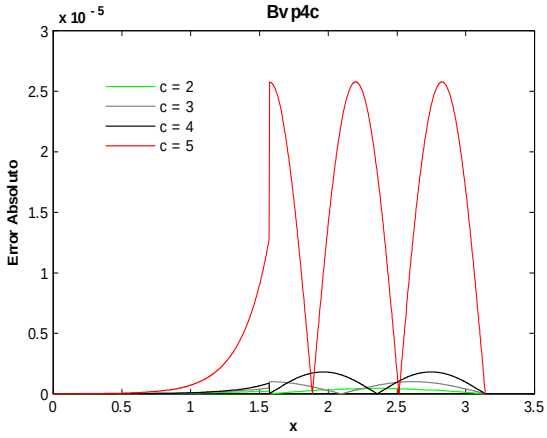


Fig. 79. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=2,3,4,5 y tolerancias absoluta y relativa de 1e-12.
Cputime =7.15 s.

Utilizando bvp4c, como se muestra en las figuras 72-75, al aumentar las tolerancias absoluta y relativa de $1e-6$ y $1e-3$ hasta $1e-9$ y $1e-9$ respectivamente el orden del error mejora de -3 a -10, y si se toman valores de $1e-12$ en ambas tolerancias el orden empeora siendo igual a -5, vea Fig. 79. A partir del valor $1e-12$ para AbsTol y RelTol se obtiene el mismo orden -5. Note que para calcular el error presentado en la figura 75 el programa tomó 2865.96 segundos, el cuál fue muy grande en comparación con los otros casos. Esto se debe a que bvp4c forma la malla inicial y checa en cada subintervalo si se cumple (17) con los valores de tolerancias dados, si no se cumple escoge otra malla y vuelve hacer lo mismo hasta que en todos los subintervalos se satisfaga (17). Entre más intentos realice tarda más. En este caso bvp4c hizo varias selecciones de malla con cerca de 5000 puntos.

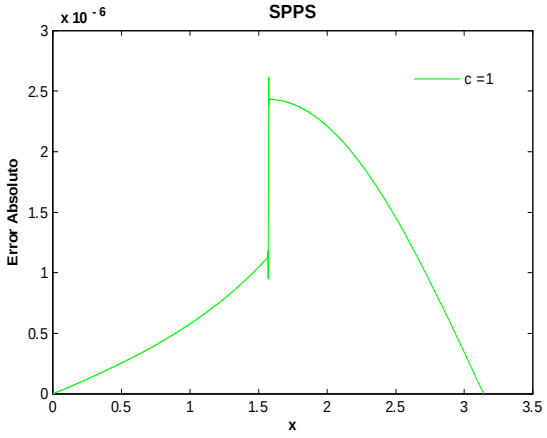


Fig. 80. Error Absoluto de la solución SPPS para el ejemplo 2 con el valor del parámetro c=1 considerando 45 potencias y 2000 puntos.
Cputime = 23.48 s.

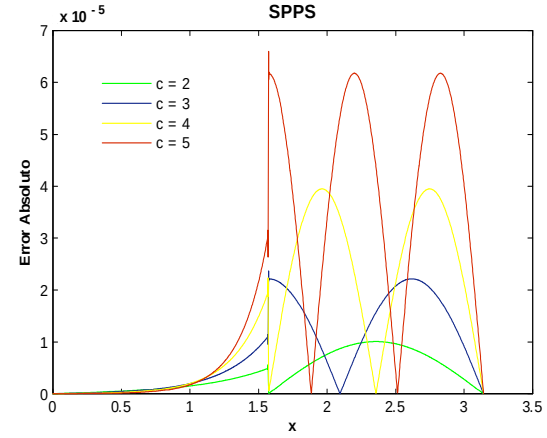


Fig.81. Error Absoluto de la solución SPPS para el ejemplo 2 con los valores del parámetro c=2,3,4,5 considerando 45 potencias y 2000 puntos.
Cputime = 93.42 s.

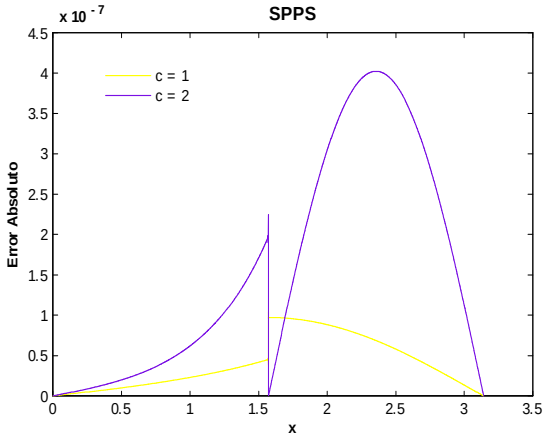


Fig. 82. Error Absoluto de la solución SPPS para el ejemplo 2 con los valores del parámetro c=1,2 considerando 45 potencias y 10000 puntos.
Cputime = 228.36 s.

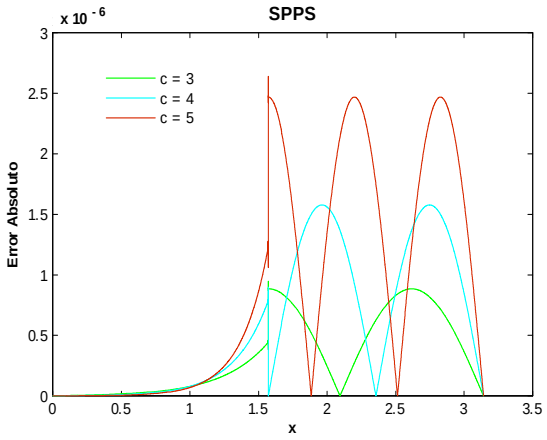


Fig. 83. Error Absoluto de la solución SPPS para el ejemplo 2 con los valores del parámetro c=3,4,5 considerando 45 potencias y 10000 puntos.
Cputime = 345.82 s.

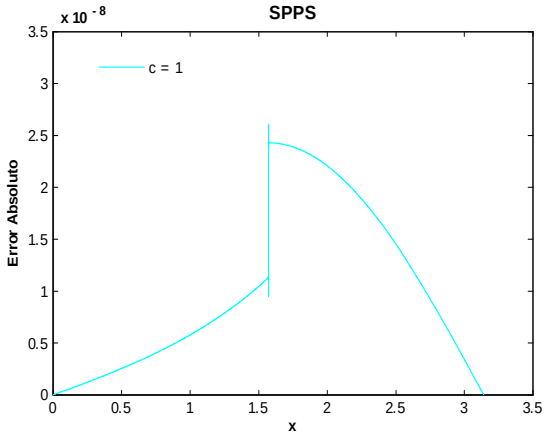


Fig. 84. Error Absoluto de la solución SPPS para el ejemplo 2 con el valor del parámetro c=1 considerando 45 potencias y 20000 puntos.
Cputime = 228.86 s.

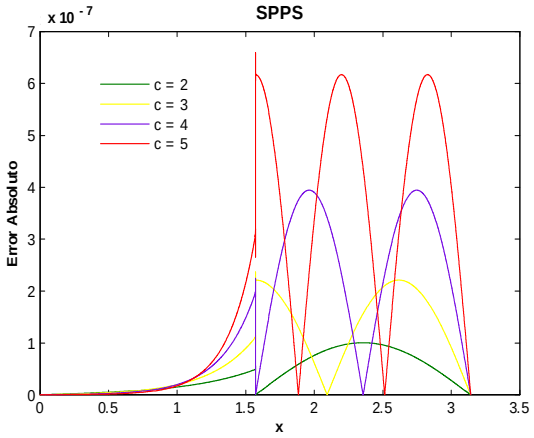


Fig. 85. Error Absoluto de la solución SPPS para el ejemplo 2 con los valores del parámetro c=2,3,4,5 considerando 45 potencias y 20000 puntos.
Cputime = 909.23 s.

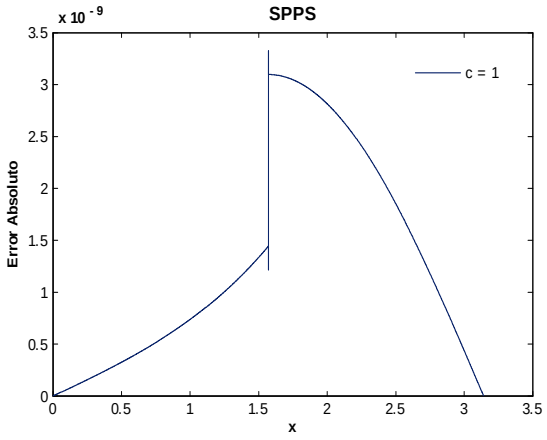


Fig. 86. Error Absoluto de la solución SPPS para el ejemplo 2 con el valor del parámetro c=1 considerando 45 potencias y 56000 puntos.
Cputime = 638.06 s.

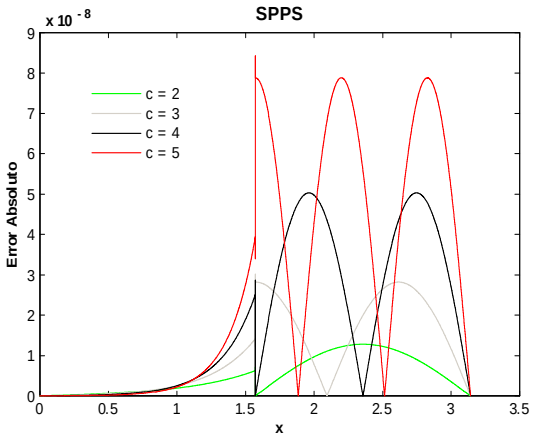


Fig. 87. Error Absoluto de la solución SPPS para el ejemplo 2 con los valores del parámetro c=2,3,4,5 considerando 45 potencias y 56000 puntos.
Cputime = 2561.81 s.

Para SPPS el aumento de puntos de 2000 a 56000 mejoró el orden del error de -5 a -8 para $c = 2, 3, 4, 5$, pero para $c = 1$ los órdenes fueron de -6 a -9 . El incremento de potencias a partir de 45 no representó una mejora en el orden. Como la solución exacta tiene ceros, no se consideró el error relativo pues para valores de la solución cercanos a cero el error se hace grande, de modo que no es significativo. Note que aquí el potencial es una función continua por partes, la cual no representó problema para ambos métodos, pues obtuvieron ordenes de error absoluto de hasta -8 para SPPS y de -10 para Bvp4c.

II *Intervalo* $\left[0, \frac{1}{2}\right]$

A continuación se muestran gráficas de algunos potenciales $q(x)$ y soluciones exactas, así como las del error absoluto obtenidas con las soluciones Bvp4c y SPPS. Se consideraron los valores de los parámetros: $c = 3, 5, 7, 13, 15, 17, 19, 21, 23, 25$, $a = 0, b = 1/2, \alpha = 1, \beta = 3$.

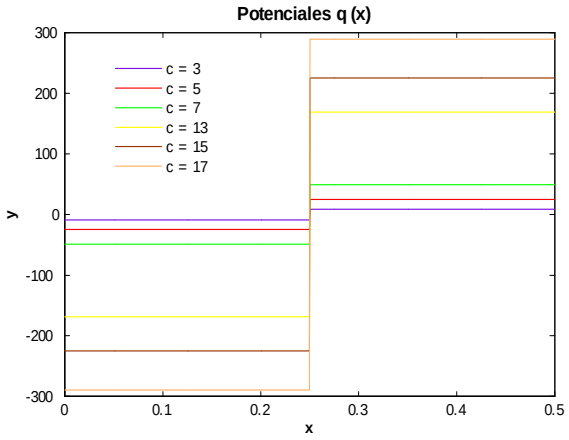


Fig. 88

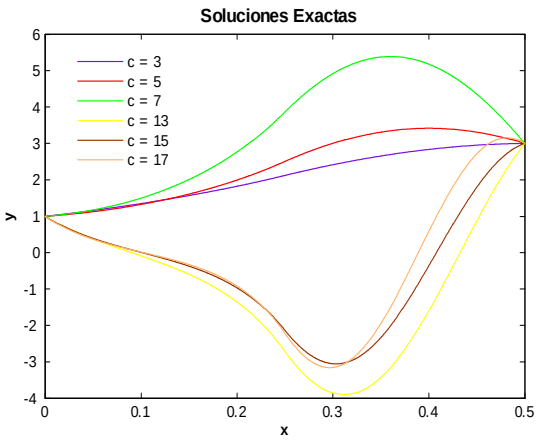


Fig. 89

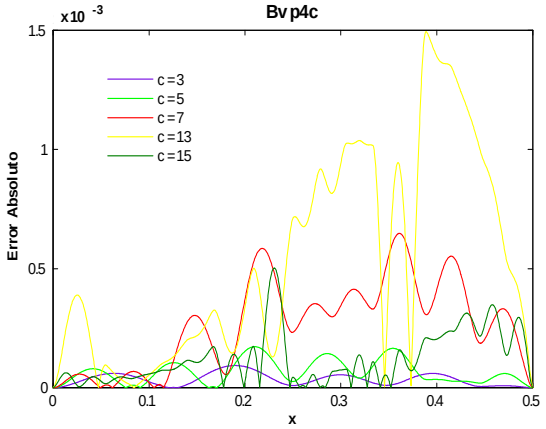


Fig. 90. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=3,5,7,13,15 y tolerancias absoluta 1e-6 y relativa de 1e-3.
Cputime = 1.63 s.

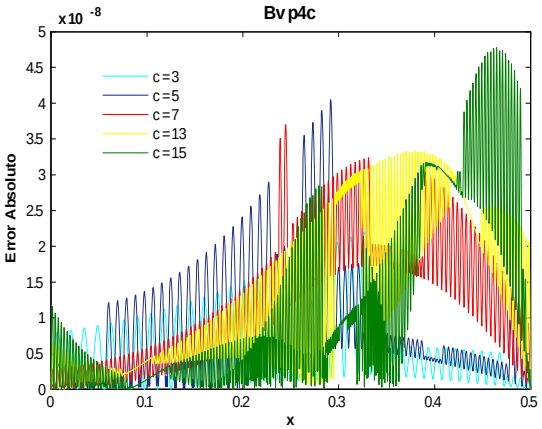


Fig. 91. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=3,5,7,13,15 y tolerancias absoluta y relativa de 1e-7.
Cputime = 5.97 s.

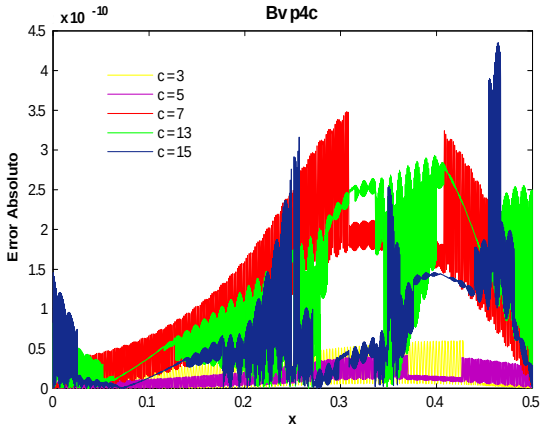


Fig. 92. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=3,5,7,13,15 y tolerancias absoluta y relativa de 1e-9.
Cputime = 2142.18 s.

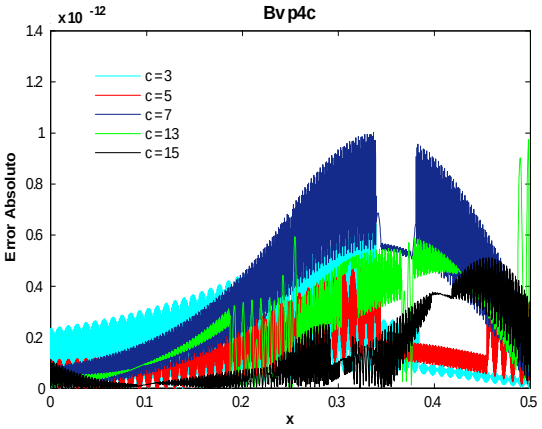


Fig. 93. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=3,5,7,13,15 y tolerancias absoluta y relativa de 1e-11.
Cputime = 73.72 s.

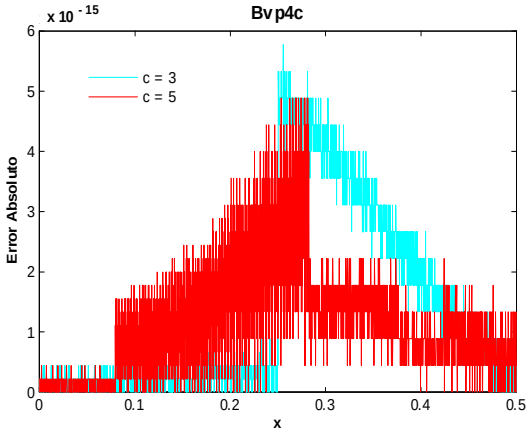


Fig. 94. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=3,5 y tolerancias absoluta y relativa de 1e-13.
Cputime = 108.11 s.

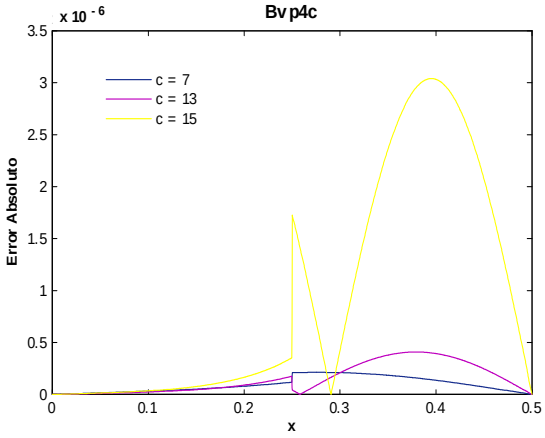


Fig. 95. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=7,13,15 y tolerancias absoluta y relativa de 1e-13.
Cputime = 16.13 s.

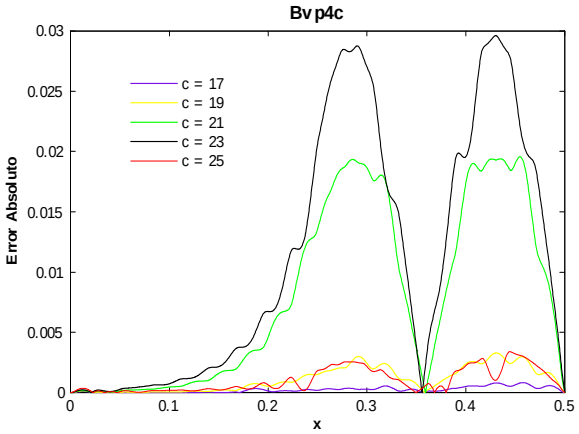


Fig. 96. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=17,19,21,23,25 y tolerancias absoluta 1e-6 y relativa de 1e-3.
Cputime = 1.02 s.

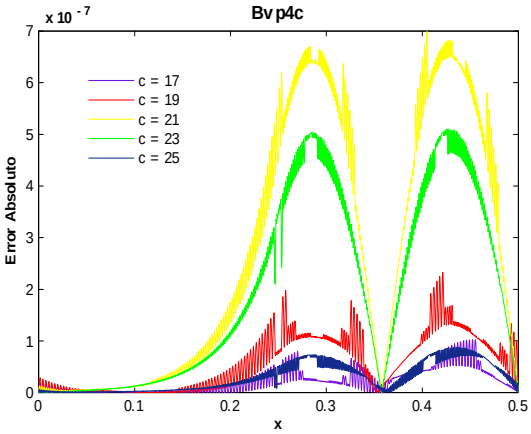


Fig. 97. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=17,19,21,23,25 y tolerancias absoluta y relativa de 1e-7.
Cputime = 11.79 s.

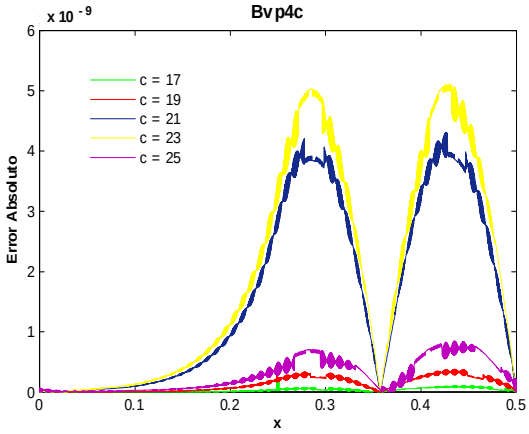


Fig. 98. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=17,19,21,23,25 y tolerancias absoluta y relativa de 1e-9.
Cputime = 3949.81 s.

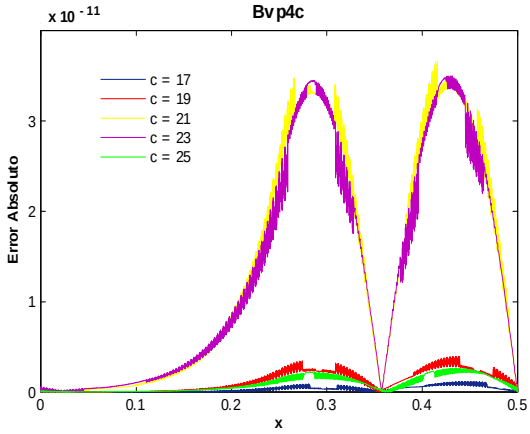


Fig. 99. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=17,19,21,23,25 y tolerancias absoluta y relativa de 1e-11.
Cputime = 122.35 s.

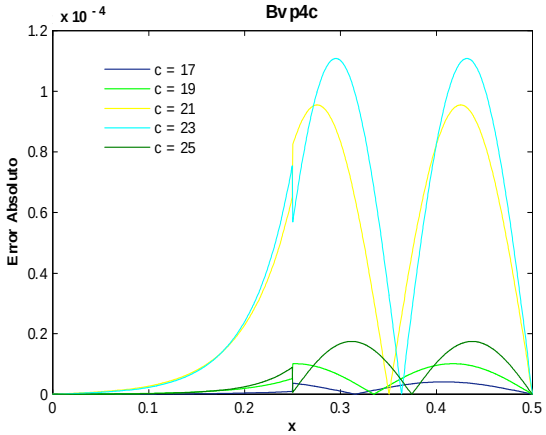


Fig. 100. Error Absoluto de la solución Bvp4c para el ejemplo 2 con los valores del parámetro c=17,19,21,23,25 y tolerancias absoluta y relativa de 1e-13.
Cputime = 11.89 s.

Utilizando `bvp4c`, como se muestra en las figuras 90-93, al aumentar las tolerancias absoluta y relativa de $1e-6$ y $1e-3$ hasta $1e-11$ y $1e-11$ respectivamente el orden del error mejora de -3 a -12, y si se toman valores de $1e-13$ en ambas tolerancias el orden empeora siendo igual a -6, vea fig. 95. Note que para calcular el error presentado en la figura 98 el programa tomó 3948.81 s, el cuál fue muy grande en comparación con las otros casos, aunque el orden obtenido no fue el mejor. De forma análoga para el segundo conjunto de valores de c el orden fue de -2 a -11 con tolerancias de $1e-6$ y $1e-3$ hasta $1e-11$ y $1e-11$ respectivamente, ver figs. 96-99. Al tomar $1e-13$ el orden obtenido es -4, ver fig.100.

Obsérvese que en este caso la longitud del intervalo fue de $\frac{1}{2}$ y el mejor orden fue de -11 para $c=25$, mientras que en el caso anterior el intervalo fue de longitud π y el mejor orden obtenido fue de -10 para $c=5$. Entonces bvp4c obtuvo una mejor aproximación a la solución exacta en un intervalo de longitud pequeña aún cuando se consideró valores grandes del parámetro c.

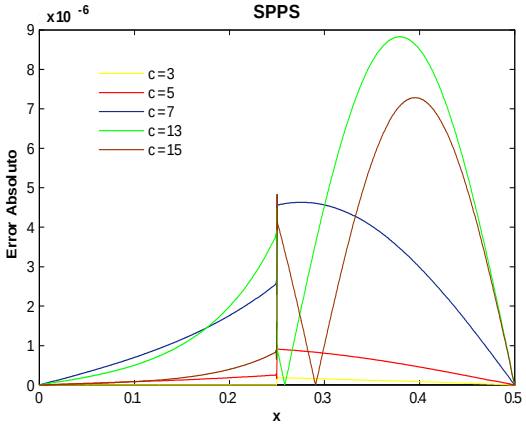


Fig. 101. Error Absoluto de la solución SPPS para el ejemplo 2 con los valores del parámetro c=3,5,7,13,15 considerando 15 potencias y 2000 puntos.
Cputime = 78.47 s.

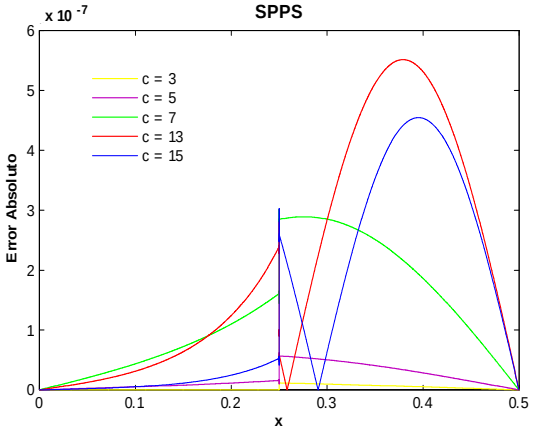


Fig. 102. Error Absoluto de la solución SPPS para el ejemplo 2 con los valores del parámetro c=3,5,7,13,15 considerando 15 potencias y 8000 puntos.
Cputime = 308.08 s.

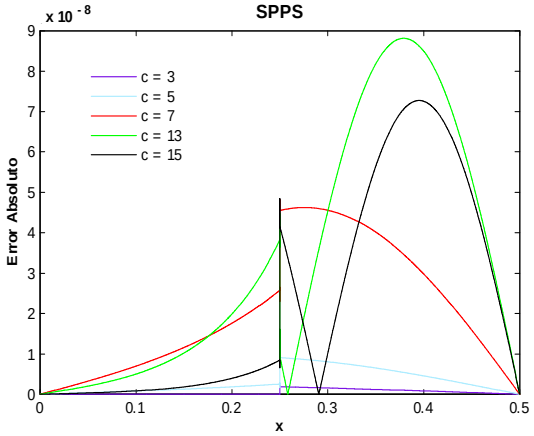


Fig. 103. Error Absoluto de la solución SPPS para el ejemplo 2 con los valores del parámetro c=3,5,7,13,15 considerando 15 potencias y 20000 puntos.
Cputime = 764.09 s.

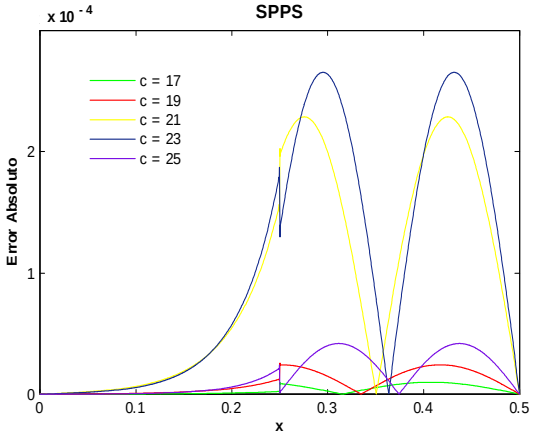


Fig. 104. Error Absoluto de la solución SPPS para el ejemplo 2 con los valores del parámetro c=17,19,21,23,25 considerando 15 potencias y 2000 puntos.
Cputime = 79.51 s.

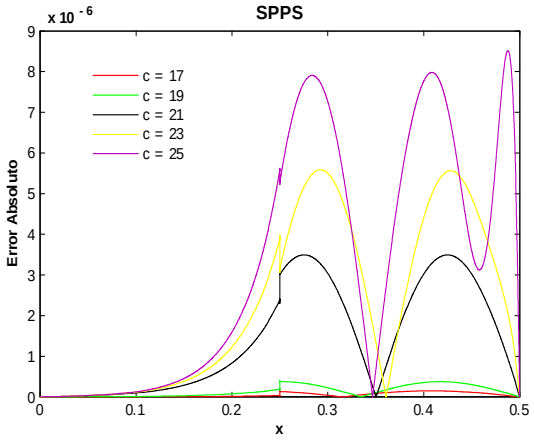


Fig. 105. Error Absoluto de la solución SPPS para el ejemplo 2 con los valores del parámetro c=17,19,21,23,25 considerando 15 potencias y 8000 puntos.
Cputime = 303.677 s.

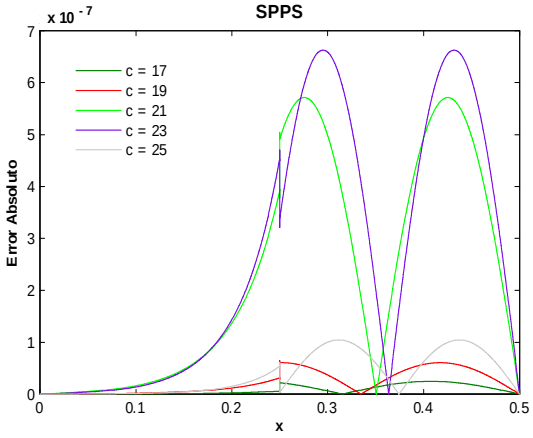


Fig. 106. Error Absoluto de la solución SPPS para el ejemplo 2 con los valores del parámetro c=17,19,21,23,25 considerando 30 potencias y 20000 puntos.
Cputime = 1520.49 s.

Como la solución exacta tiene ceros, no se consideró el error relativo pues para valores de la solución cercanos a cero éste error se hace grande, de modo que no es significativo. Para SPPS al aumentar puntos el orden del error mejoró, para $c = 15$ fue de -6 (fig. 101, con 2000 puntos, 15 potencias) hasta -8 (fig. 103, con 20000 puntos, 15 potencias). Note que en este ejemplo se pueden obtener mejores órdenes al considerar más puntos. La longitud del intervalo usada en este ejemplo fue de $\frac{1}{2}$ y se obtuvo un mejor orden de -7 para $c = 25$, vea fig. 106, con 20000 puntos y 30 potencias, mientras que para el caso anterior se consideró el intervalo de longitud π y se obtuvo un orden de -7 para $c = 5$, con 20000 puntos y 45 potencias. Esto quiere decir que la implementación de SPPS en Matlab obtuvo mejores aproximaciones a la solución exacta en un intervalo de menor longitud, aún cuando se consideraron valores del parámetro más grandes.

Obsérvese que el potencial es una función continua por partes, la cual no representó problema para ambos métodos, pues obtuvieron órdenes de error absoluto de hasta -7 para SPPS y de -11 para Bvp4c con $c = 17, 19, 21, 23, 25$.

Ejemplo 3

Solución Exacta

Considérese el problema con valores en la frontera

$$\begin{aligned} -y'' + (c^2x^2 + c)y &= 0 \\ y(a) &= \alpha \\ y(b) &= \beta \end{aligned} \tag{47}$$

donde $c(\neq 0), \alpha, \beta, a, b \in \mathbb{R}$, $a < b$.

La ecuación diferencial en (47) tiene como solución general a $y_g(x) = e^{\frac{cx^2}{2}} \left(c_1 + c_2 \int_0^x e^{-ct^2} dt \right)$ donde c_1 y c_2 son constantes arbitrarias. Entonces se debe cumplir

$$\begin{aligned} y_g(a) &= e^{\frac{ca^2}{2}} \left(c_1 + c_2 \int_0^a e^{-ct^2} dt \right) = \alpha \\ y_g(b) &= e^{\frac{cb^2}{2}} \left(c_1 + c_2 \int_0^b e^{-ct^2} dt \right) = \beta. \end{aligned} \tag{48}$$

Por los Teoremas 4 y 5 el problema (47) tiene solución única si y sólo si

$$\Delta = \begin{vmatrix} 1 & \int_0^a e^{-ct^2}dt \\ 1 & \int_0^b e^{-ct^2}dt \end{vmatrix} = \int_0^b e^{-ct^2}dt - \int_0^a e^{-ct^2}dt \neq 0. \tag{49}$$

Pero nótese que

$$\int_0^b e^{-ct^2}dt - \int_0^a e^{-ct^2}dt = 0 \quad sii \quad a = b$$

de modo que el problema (47) satisface (49). Resolviendo el sistema (48) mediante la regla de Cramer:

$$c_1 = \frac{\begin{vmatrix} \alpha e^{-\frac{ca^2}{2}} & \int_0^a e^{-ct^2}dt \\ \beta e^{-\frac{cb^2}{2}} & \int_0^b e^{-ct^2}dt \end{vmatrix}}{\Delta} = \frac{\alpha e^{-\frac{ca^2}{2}} \int_0^b e^{-ct^2}dt - \beta e^{-\frac{cb^2}{2}} \int_0^a e^{-ct^2}dt}{\Delta}$$

$$c_2 = \frac{\begin{vmatrix} 1 & \alpha e^{-\frac{ca^2}{2}} \\ 1 & \beta e^{-\frac{cb^2}{2}} \end{vmatrix}}{\Delta} = \frac{\beta e^{-\frac{cb^2}{2}} - \alpha e^{-\frac{ca^2}{2}}}{\Delta}$$

De donde se concluye que la solución de (47) es:

$$y_p = e^{\frac{cx^2}{2}} \left(\frac{\alpha e^{-\frac{ca^2}{2}} \int_0^b e^{-ct^2}dt - \beta e^{-\frac{cb^2}{2}} \int_0^a e^{-ct^2}dt}{\Delta} + \frac{\beta e^{-\frac{cb^2}{2}} - \alpha e^{-\frac{ca^2}{2}}}{\Delta} \int_0^x e^{-ct^2}dt \right)$$

Las siguientes gráficas muestran algunas soluciones exactas y potenciales para los valores $\alpha = 0$, $\beta = 1$, $a = 0$, $b = 1$, $c = 1, 4, 7, 10, 12, 15$.

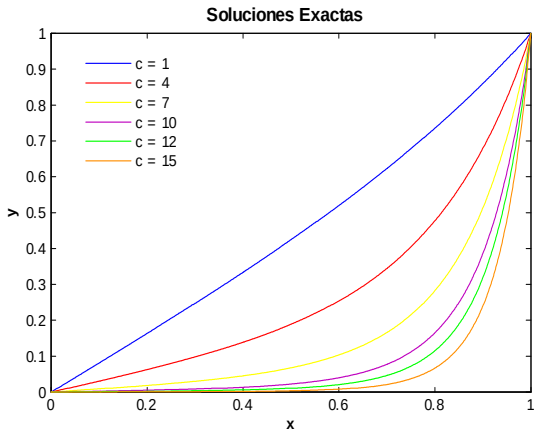


Fig. 107

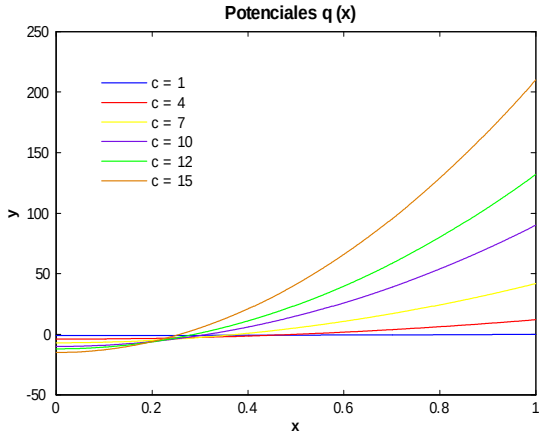


Fig. 108

Soluciones Numéricas

Los valores de los parámetros en (47) usados en ambas soluciones numéricas para obtener los errores absoluto y relativo son los siguientes: $\alpha = 0$, $\beta = 1$, $a = 0$, $b = 1$, $c = 1, 3, 5, 7, 9, 11, 13, 15, 17, 19$. A continuación se presentan gráficas considerando distintos valores de los parámetros de exactitud de ambos métodos.

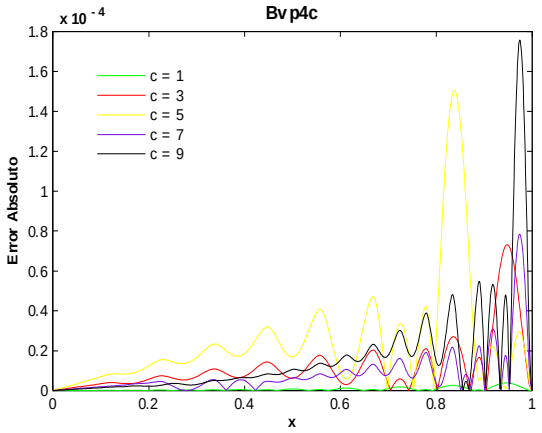


Fig. 109. Error Absoluto de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 3.15 s.

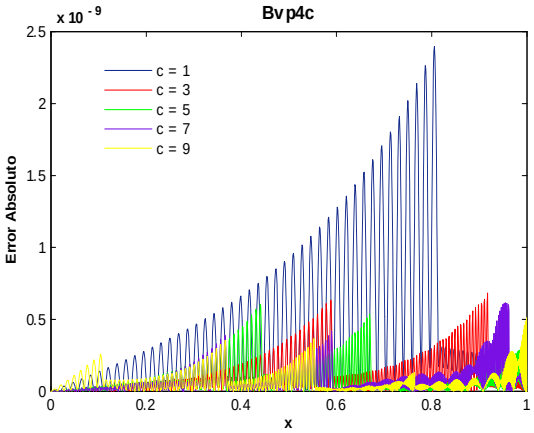


Fig. 110.Error Absoluto de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-7 y relativa de 1e-8. Cputime = 3.64 s.

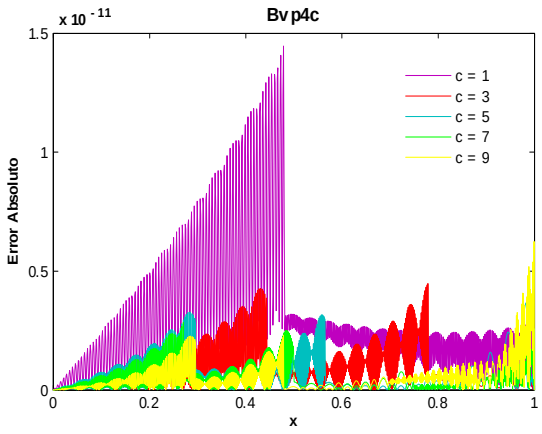


Fig. 111. Error Absoluto de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-9 y relativa de 1e-10. Cputime = 4.81 s.

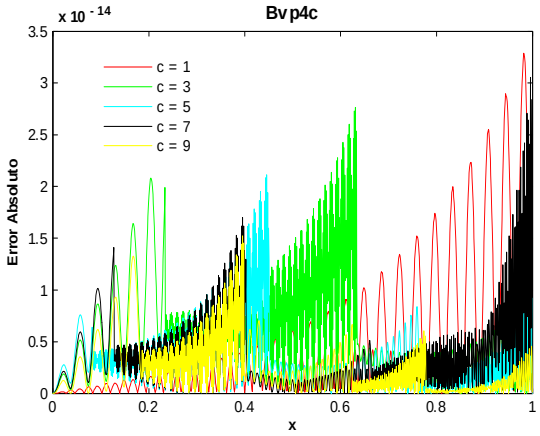


Fig. 112. Error Absoluto de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-11 y relativa de 1e-12. Cputime = 9.15 s.

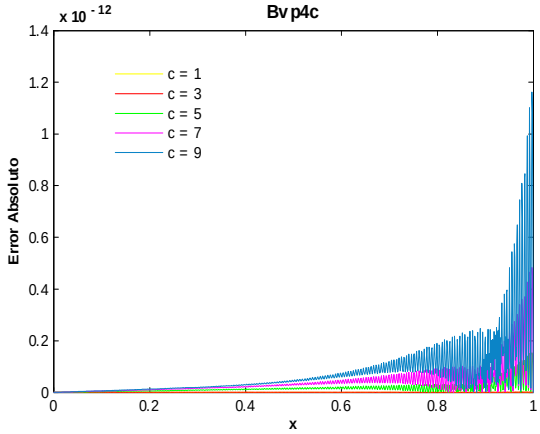


Fig. 113. Error Absoluto de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-13 y relativa de 1e-14.
Cputime = 18.36 s.

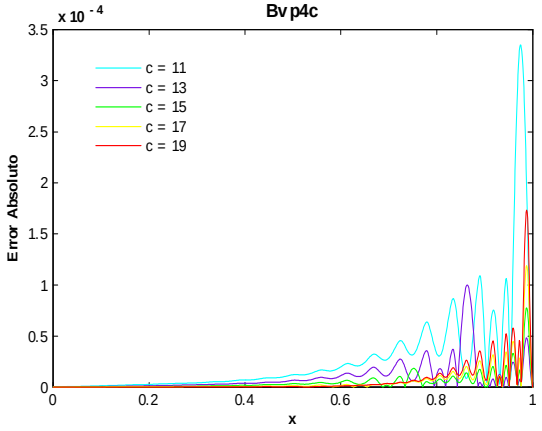


Fig. 114. Error Absoluto de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 3.32 s.

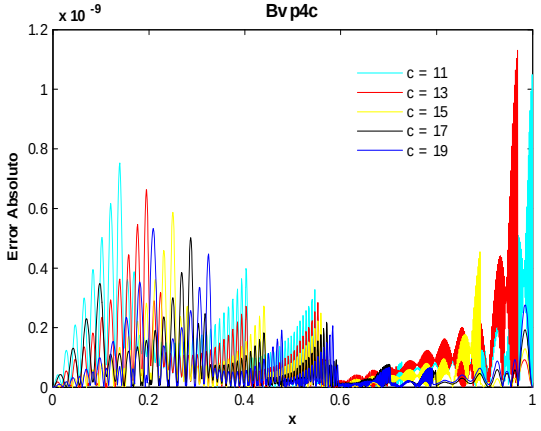


Fig. 115. Error Absoluto de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 y tolerancias absoluta de 1e-7 y relativa de 1e-8.
Cputime = 4.19 s.

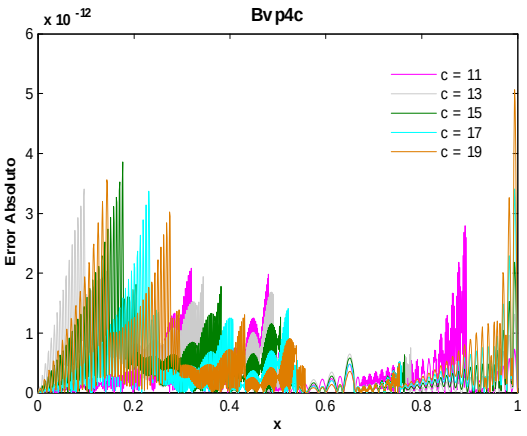


Fig. 116. Error Absoluto de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 y tolerancias absoluta de 1e-9 y relativa de 1e-10.
Cputime = 6.46 s.

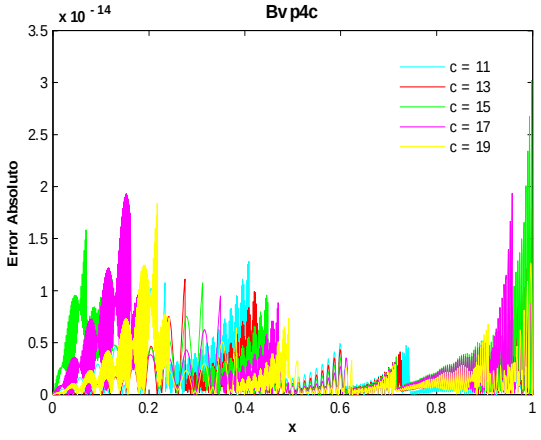


Fig. 117. Error Absoluto de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 y tolerancias absoluta de 1e-11 y relativa de 1e-12.
Cputime = 13.47 s.

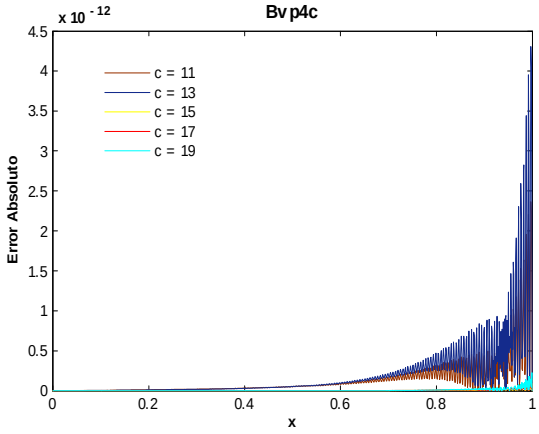


Fig. 118. Error Absoluto de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 y tolerancias absoluta de 1e-13 y relativa de 1e-14.
Cputime = 23.97 s.

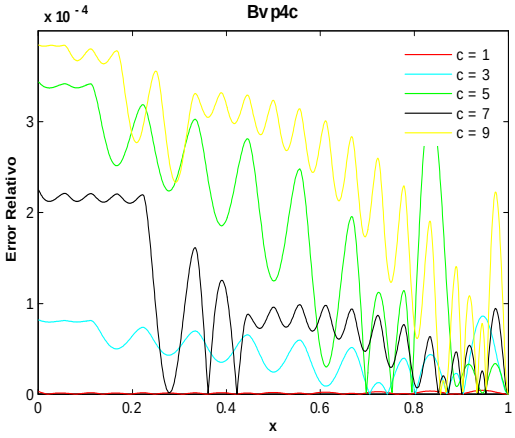


Fig. 119. Error Relativo de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 3.04 s.

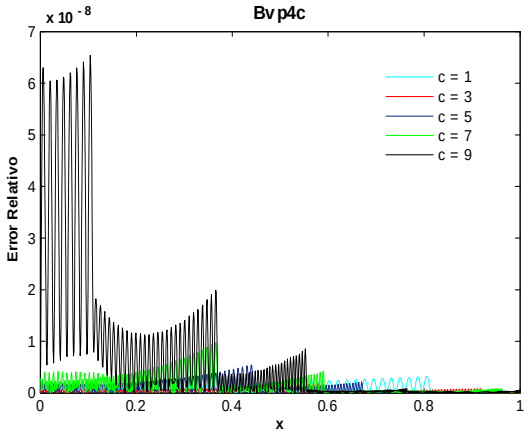


Fig. 120. Error Relativo de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-7 y relativa de 1e-8. Cputime = 3.83 s.

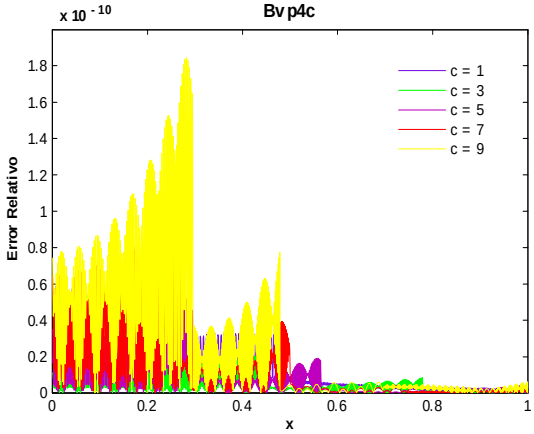


Fig. 121. Error Relativo de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-9 y relativa de 1e-10. Cputime = 4.86 s.

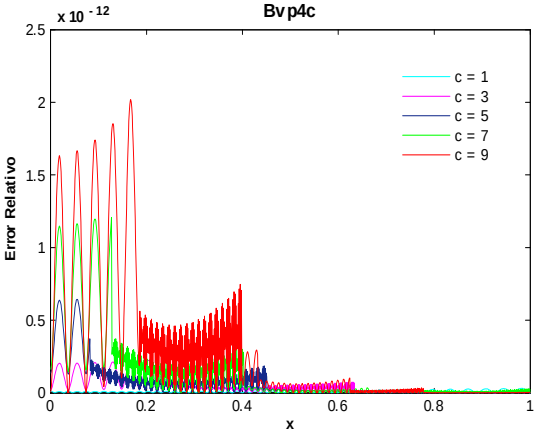


Fig. 122. Error Relativo de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-11 y relativa de 1e-12. Cputime = 9.48 s.

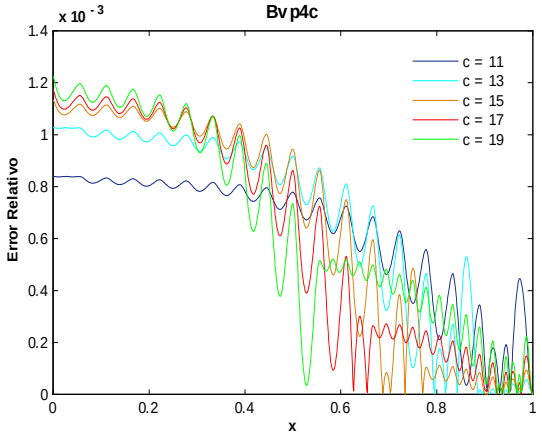


Fig. 123. Error Relativo de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 3.17 s.

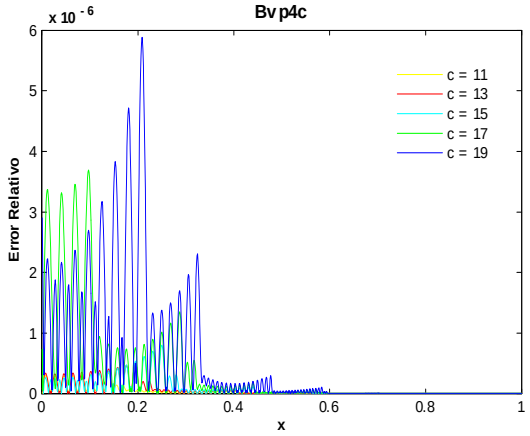


Fig. 124. Error Relativo de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 y tolerancias absoluta de 1e-7 y relativa de 1e-8.
Cputime = 4.39 s.

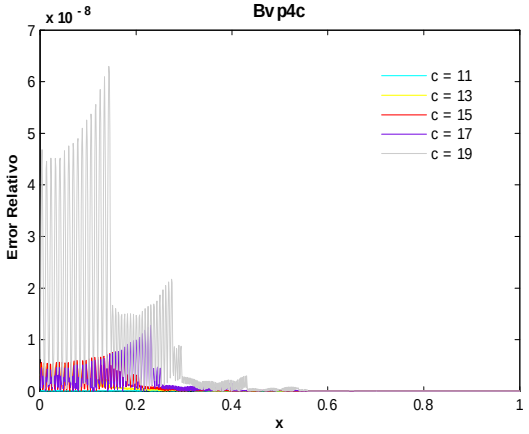


Fig. 125. Error Relativo de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 y tolerancias absoluta de e-9 y relativa de 1e-10.
Cputime = 6.59 s.

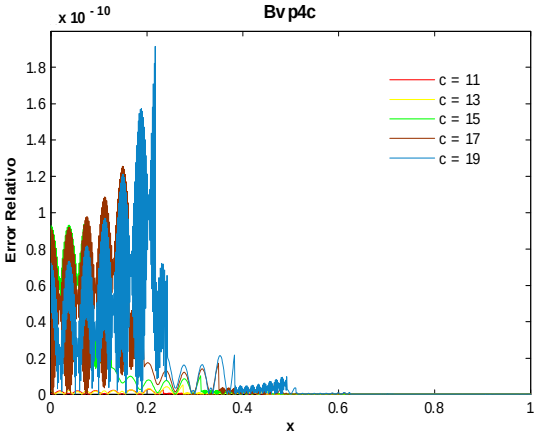


Fig. 126. Error Relativo de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 y tolerancias absoluta de 1e-11 y relativa de 1e-12.
Cputime = 14.9 s.

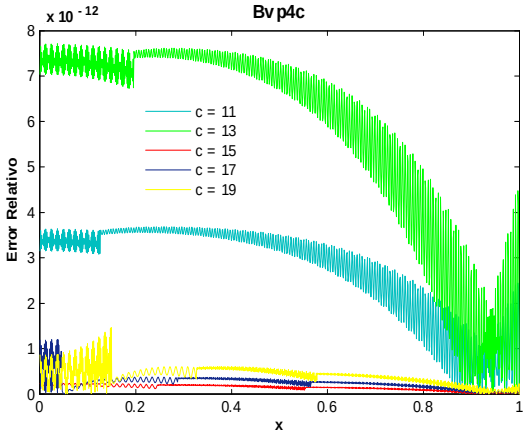


Fig.127 . Error Relativo de la solución Bvp4c para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 y tolerancias absoluta de 1e-13 y relativa de 1e-14.
Cputime = 25.77 s.

Como se puede ver en las figuras 109-112 y 114-117, para bvp4c, el orden del error absoluto mejoró de -4 a -14 al variar las tolerancias absoluta y relativa de $1e-6$ y $1e-3$ hasta $1e-11$ y $1e-12$ respectivamente. Al considerar los valores $AbsTol = 1e-13, \mathrm{Re}\, lTol = 1e-14$ el orden empeora tomando el valor de -12, vea figs. 113 y 118. El orden del error relativo disminuyó de -4(con $AbsTol = 1e-6, \mathrm{Re}\, lTol = 1e-3$) a -12(con $AbsTol = 1e-11$ y $\mathrm{Re}\, lTol = 1e-12$) para $c = 1, 3, 5, 7, 9$, vea figs.119-122. Para $c = 11, 13, 15, 17, 19$ (figs.123-127) el orden del error relativo fue de -3(con $AbsTol = 1e-6, \mathrm{Re}\, lTol = 1e-3$) a -12(con $AbsTol = 1e-13$ y $\mathrm{Re}\, lTol = 1e-14$).

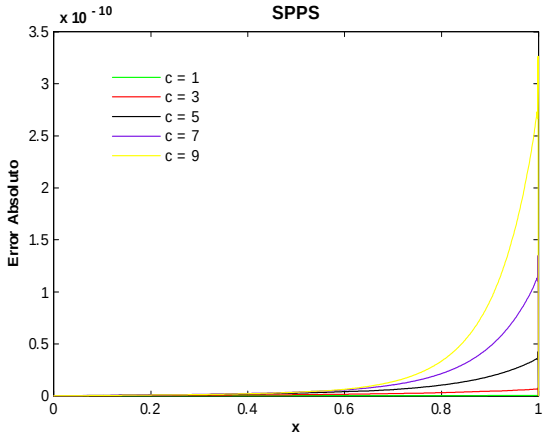


Fig. 128. Error Absoluto de la solución SPPS para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 considerando 15 potencias y 1000 puntos.
Cputime = 22.91 s.

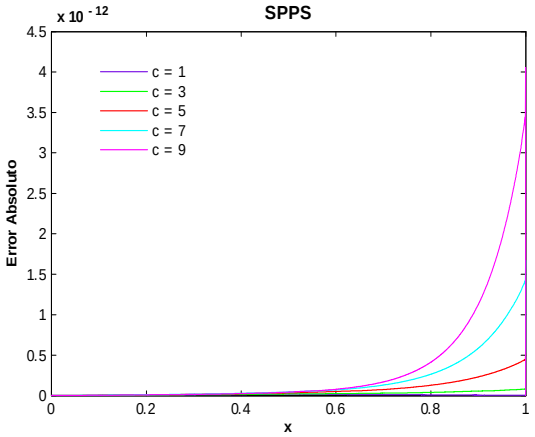


Fig. 129. Error Absoluto de la solución SPPS para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 considerando 15 potencias y 3000 puntos.
Cputime = 61.96 s.

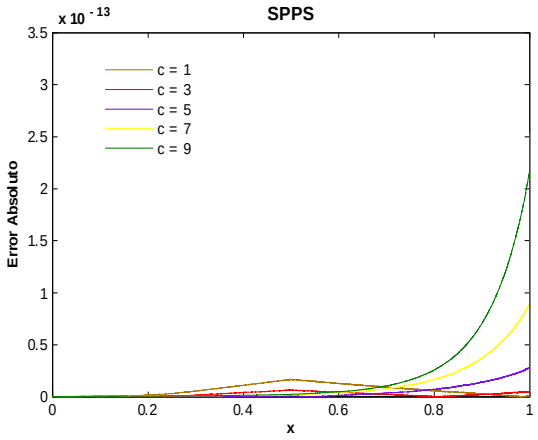


Fig. 130. Error Absoluto de la solución SPPS para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 considerando 15 potencias y 6000 puntos.
Cputime = 119.92 s.

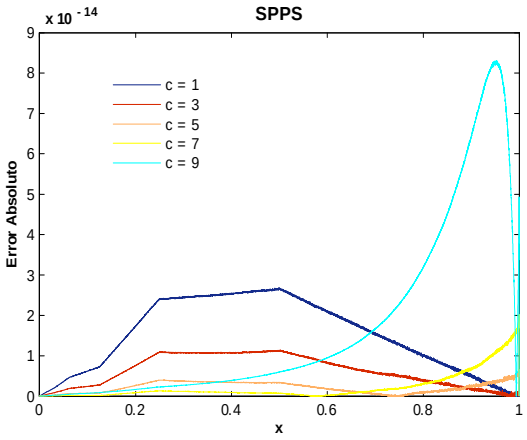


Fig. 131. Error Absoluto de la solución SPPS para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 considerando 15 potencias y 9000 puntos.
Cputime = 178.22 s.

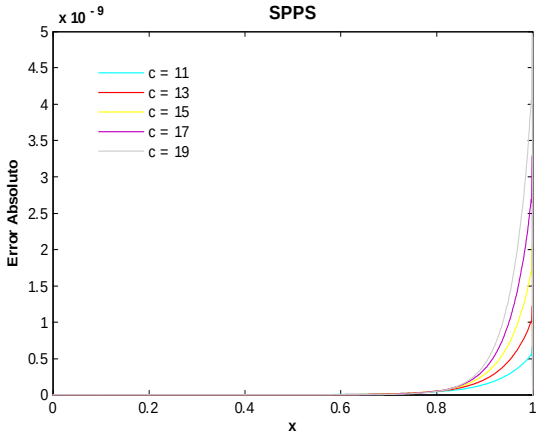


Fig. 132. Error Absoluto de la solución SPPS para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 considerando 25 potencias y 1000 puntos.
Cputime = 36.45 s.

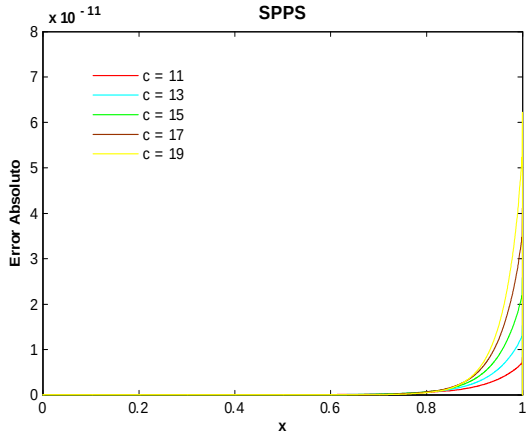


Fig. 133. Error Absoluto de la solución SPPS para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 considerando 25 potencias y 3000 puntos.
Cputime = 101.49 s.

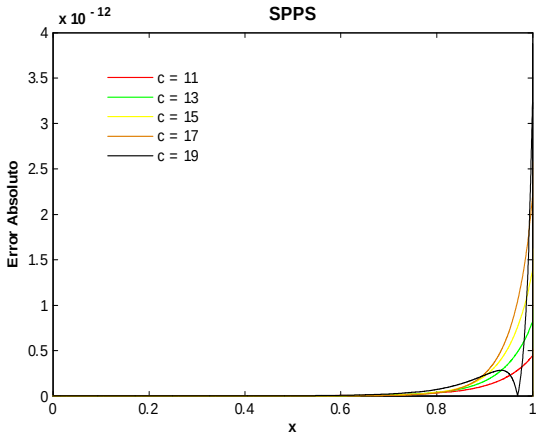


Fig. 134. Error Absoluto de la solución SPPS para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 considerando 25 potencias y 6000 puntos.
Cputime = 198.91 s.

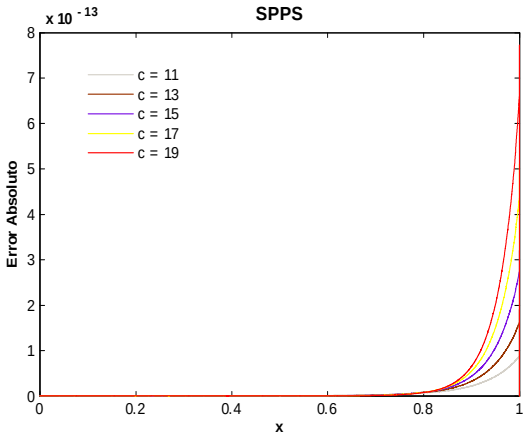


Fig. 135. Error Absoluto de la solución SPPS para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 considerando 25 potencias y 9000 puntos.
Cputime = 295.8 s.

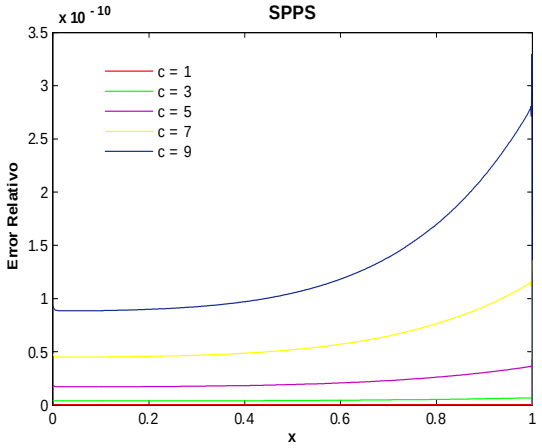


Fig. 136. Error Relativo de la solución SPPS para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 considerando 20 potencias y 1000 puntos.
Cputime = 29.59 s.

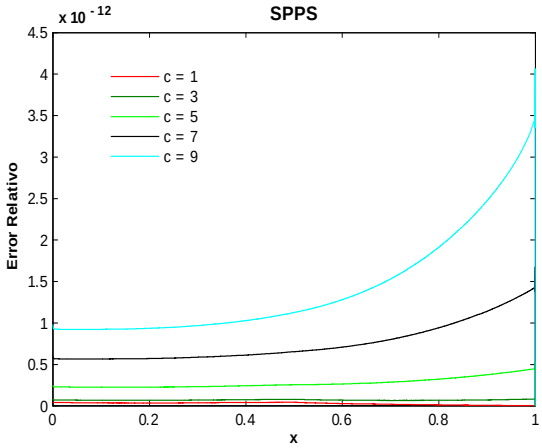


Fig. 137. Error Relativo de la solución SPPS para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 considerando 20 potencias y 3000 puntos.
Cputime = 81.38 s.

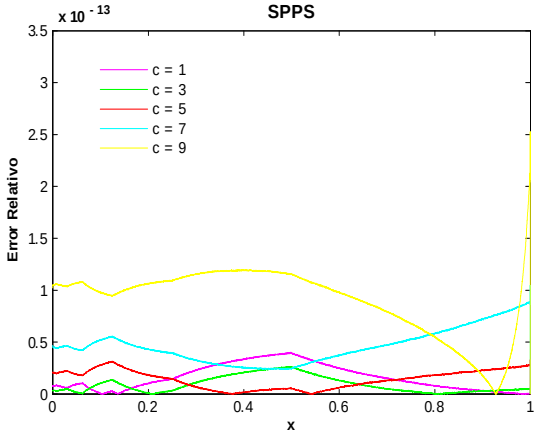


Fig. 138. Error Relativo de la solución SPPS para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 considerando 20 potencias y 6000 puntos.
Cputime = 159.97 s.

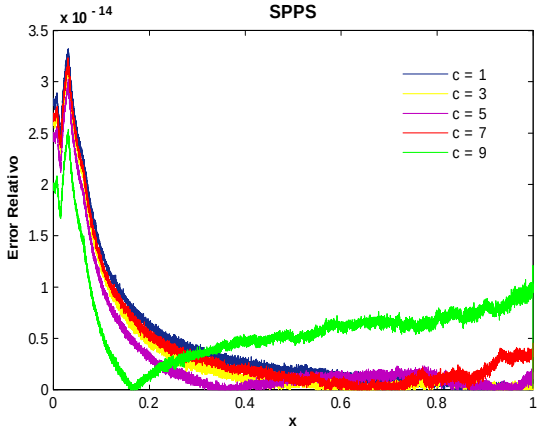


Fig. 139. Error Relativo de la solución SPPS para el ejemplo 3 con los valores del parámetro c=1,3,5,7,9 considerando 20 potencias y 13000 puntos.
Cputime = 337.57 s.

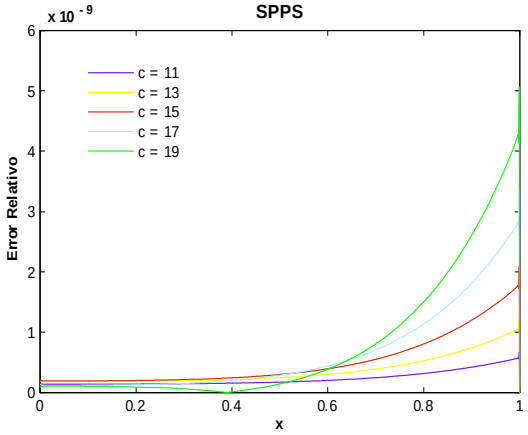


Fig. 140. Error Relativo de la solución SPPS para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 considerando 25 potencias y 1000 puntos.
Cputime = 36.77 s.

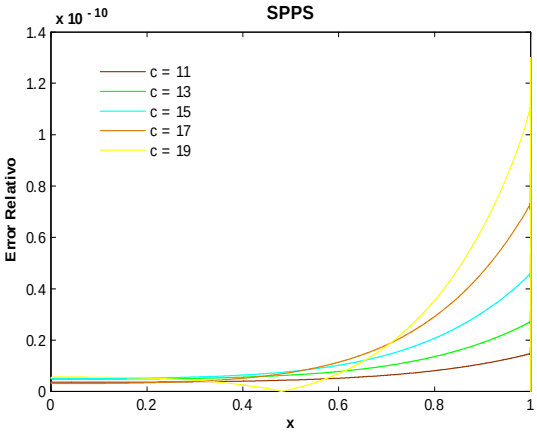


Fig. 141. Error Relativo de la solución SPPS para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 considerando 25 potencias y 2500 puntos.
Cputime = 85.97 s.

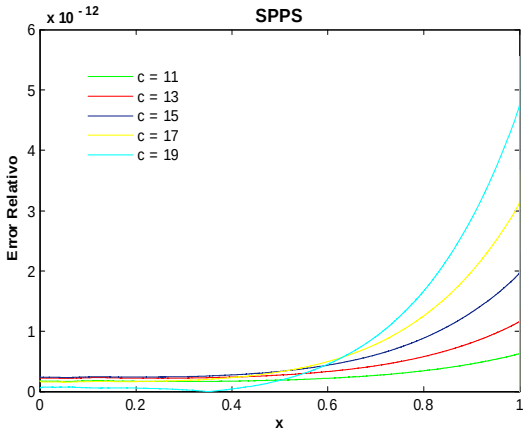


Fig. 142. Error Relativo de la solución SPPS para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 considerando 25 potencias y 5500 puntos. Cputime = 182.5 s.

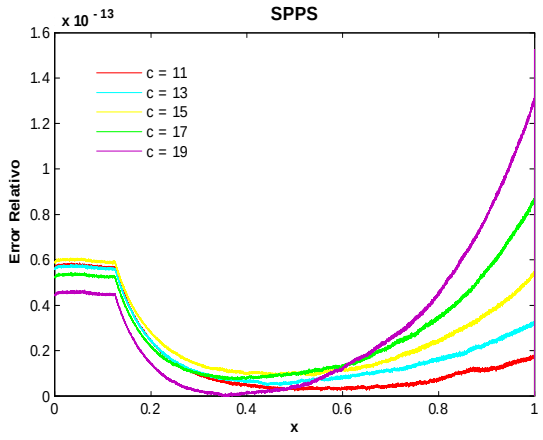


Fig. 143. Error Relativo de la solución SPPS para el ejemplo 3 con los valores del parámetro c=11,13,15,17,19 considerando 25 potencias y 13500 puntos. Cputime = 438.74 s.

Utilizando SPPS para $c = 9$ el orden del error absoluto fue de -10, con sólo 15 potencias y 1000 puntos, a -14, con 15 potencias y 9000 puntos, ver figs. 128-131.

Para $c = 19$ fue necesario tomar 25 potencias para obtener órdenes del error absoluto de -9 con 1000 puntos, a -13, con 9000 puntos, ver figs. 132-135. En cuanto al error relativo, para $c = 9$ se consideraron 20 potencias para las cuáles el orden del error fue de -10 con 1000 puntos, a -14 con 13000 puntos, ver figs. 136-139. Para $c = 19$ el orden fue de -9, con 1000 puntos y 25 potencias, a -13, con 13500 puntos y 25 potencias, ver figs. 140-143.

Ejemplo 4

Solución Exacta

Considérese el problema con valores en la frontera

$$\begin{aligned} y'' - c^2 y &= 0 \\ y(a) &= \alpha \\ y(b) &= \beta \end{aligned} \tag{50}$$

donde $c(\neq 0), \alpha, \beta, a, b \in \mathbb{R}$, $a < b$.
Se sabe que la solución general de la ecuación diferencial en (50) es $y_g = c_1 e^{cx} + c_2 e^{-cx}$ donde c_1 y c_2 son constantes arbitrarias. Entonces se debe cumplir

$$\begin{aligned} y_g(a) &= c_1 e^{ca} + c_2 e^{-ca} = \alpha \\ y_g(b) &= c_1 e^{cb} + c_2 e^{-cb} = \beta. \end{aligned} \tag{51}$$

Por los Teoremas 4 y 5 el problema (50) tiene solución única si y sólo si

$$\Delta = \begin{vmatrix} e^{ca} & e^{-ca} \\ e^{cb} & e^{-cb} \end{vmatrix} = e^{c(a-b)} - e^{-c(a-b)} \neq 0 \tag{52}$$

Pero nótese que
$e^{c(a-b)} - e^{-c(a-b)} = 0 \quad sii \quad e^{2c(a-b)} = 1 \quad sii \quad 2c(a-b) = 0 \quad sii \quad c = 0 \;\; ó \;\; a = b$, de modo que el

problema (50) satisface (52).
Resolviendo el sistema (51) mediante la regla de Cramer:

$$c_1 = \frac{\begin{vmatrix} \alpha & e^{-ca} \\ \beta & e^{-cb} \end{vmatrix}}{\Delta} = \frac{\alpha e^{-cb} - \beta e^{-ca}}{e^{c(a-b)} - e^{-c(a-b)}} \qquad c_2 = \frac{\begin{vmatrix} e^{ca} & \alpha \\ e^{cb} & \beta \end{vmatrix}}{\Delta} = \frac{\beta e^{ca} - \alpha e^{cb}}{e^{c(a-b)} - e^{-c(a-b)}}$$

De donde se concluye que la solución de (50) es:

$$y_p = \frac{\alpha e^{-cb} - \beta e^{-ca}}{e^{c(a-b)} - e^{-c(a-b)}} e^{cx} + \frac{\beta e^{ca} - \alpha e^{cb}}{e^{c(a-b)} - e^{-c(a-b)}} e^{-cx}$$

Las siguientes gráficas muestran algunas soluciones exactas con valores $\alpha = 2$, $\beta = 1$, $a = 0$, $b = 1$, $c = 2, 5, 8, 10, 13, 15$.

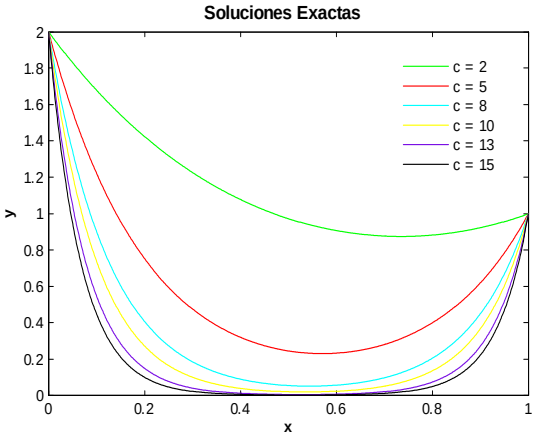


Fig. 94

Soluciones Numéricas

A continuación se presentan las gráficas obtenidas para los errores absoluto y relativo con los valores de parámetros $\alpha = 2$, $\beta = 1$, $a = 0$, $b = 1$, $c = 1, 2, 3, 4, 5, 6, 7, 8, 9$.

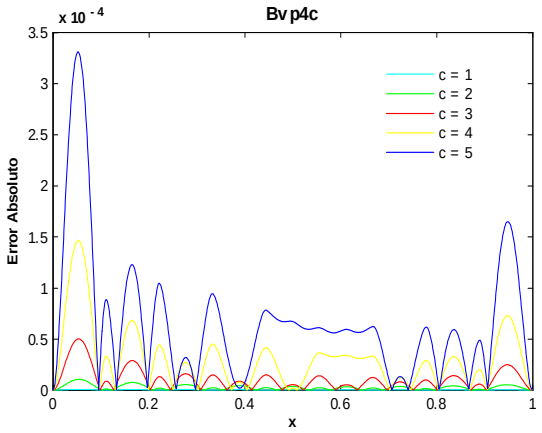


Fig. 144. Error Absoluto de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 0.37 s.

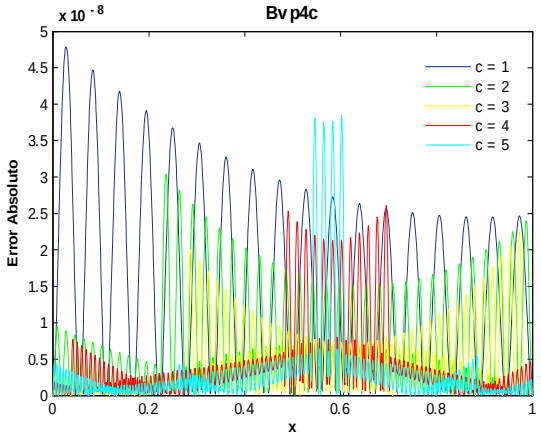


Fig. 145. Error Absoluto de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta de 1e-6 y relativa de 1e-8.
Cputime = 0.54 s.

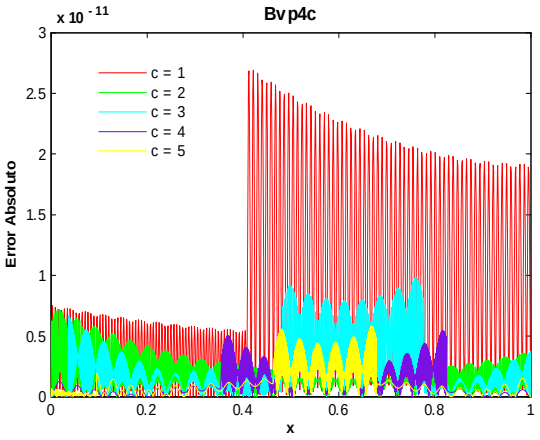


Fig. 146. Error Absoluto de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta de 1e-9 y relativa de 1e-11. Cputime = 1.71 s.

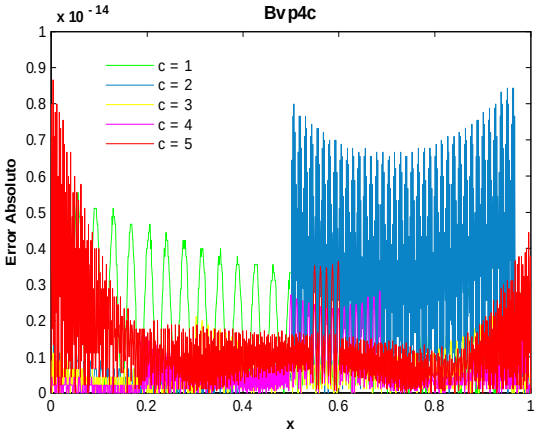


Fig. 147. Error Absoluto de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta de 1e-12 y relativa de 1e-14. Cputime = 22.49 s.

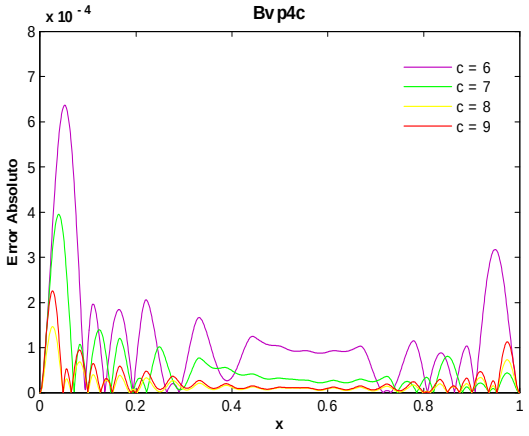


Fig. 148. Error Absoluto de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=6,7,8,9 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 0.39 s.

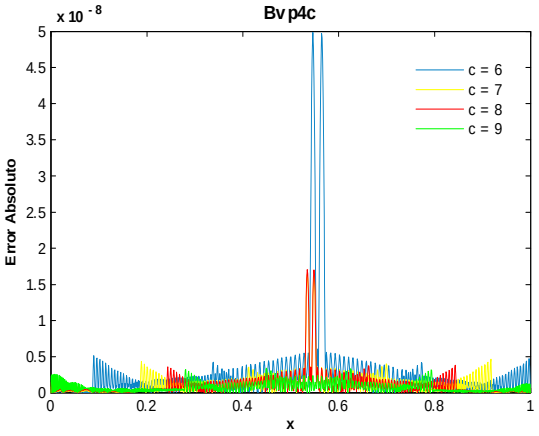


Fig.149. Error Absoluto de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=6,7,8,9 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 0.81 s.

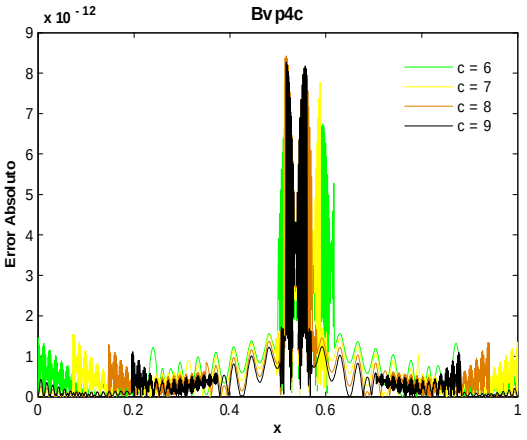


Fig. 150. Error Absoluto de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=6,7,8,9 y tolerancias absoluta de 1e-9 y relativa de 1e-11. Cputime = 3.36 s.

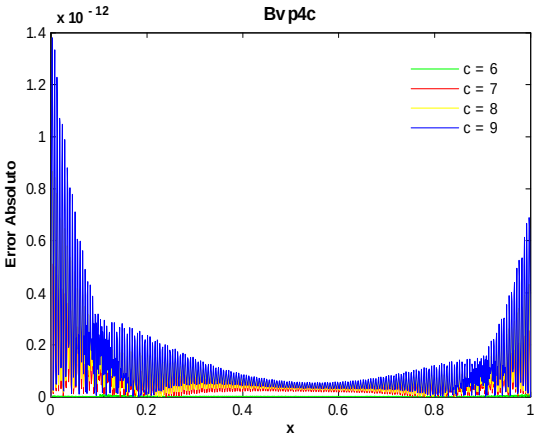


Fig. 151. Error Absoluto de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=6,7,8,9 y tolerancias absoluta de 1e-12 y relativa de 1e-14. Cputime = 13.05 s.

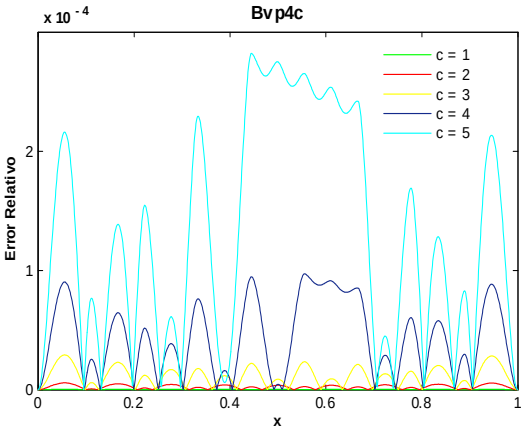


Fig. 152. Error Relativo de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 0.35 s.

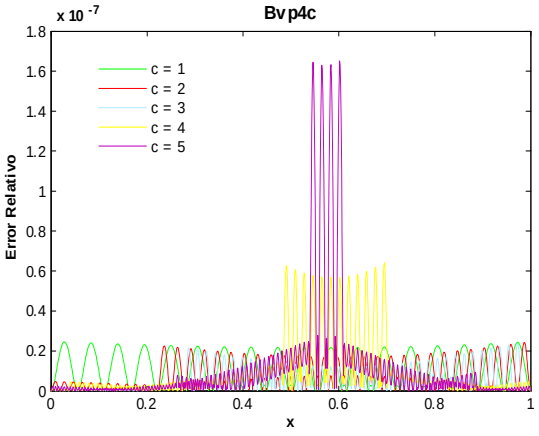


Fig. 153. Error Relativo de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 0.53 s.

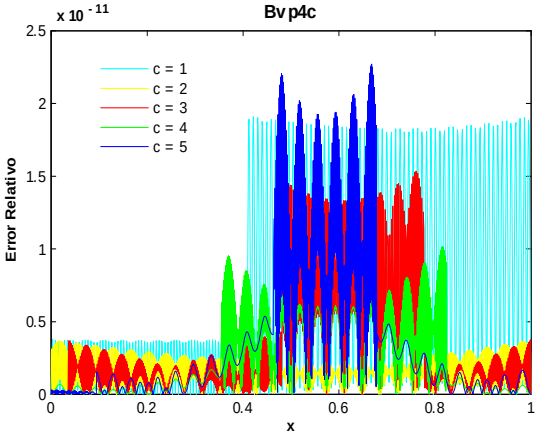


Fig. 154. Error Relativo de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta de 1e-9 y relativa de 1e-11. Cputime = 1.7 s.

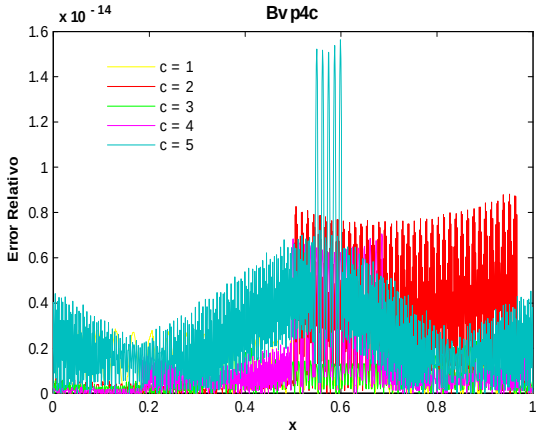


Fig. 155. Error Relativo de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 y tolerancias absoluta de 1e-12 y relativa de 1e-14. Cputime = 22.62 s.

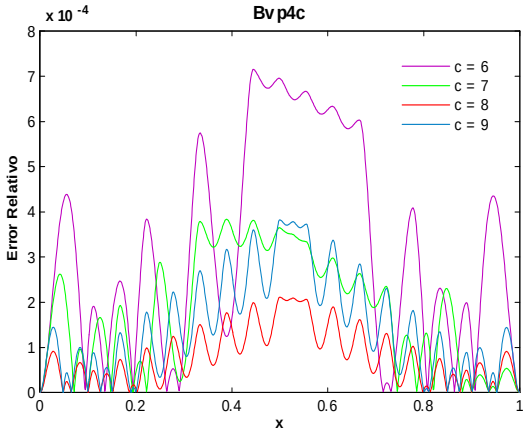


Fig. 156. Error Relativo de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=6,7,8,9 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 0.38 s.

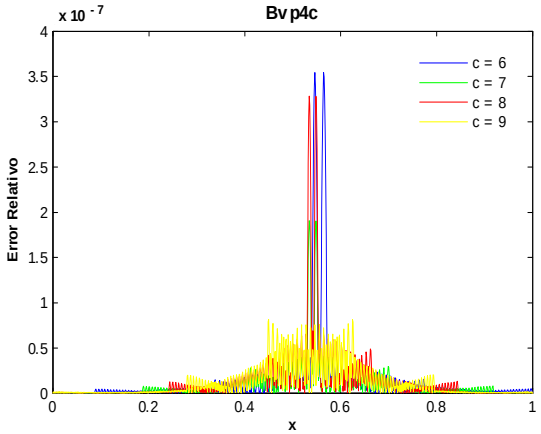


Fig. 157. Error Relativo de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=6,7,8,9 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 0.81 s.

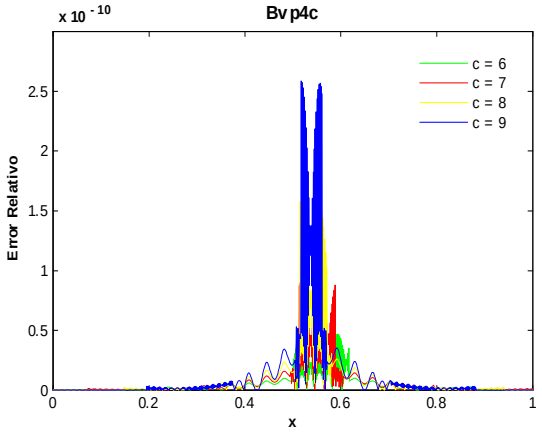


Fig. 158. Error Relativo de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=6,7,8,9 y tolerancias absoluta de 1e-9 y relativa de 1e-11. Cputime = 3.37 s.

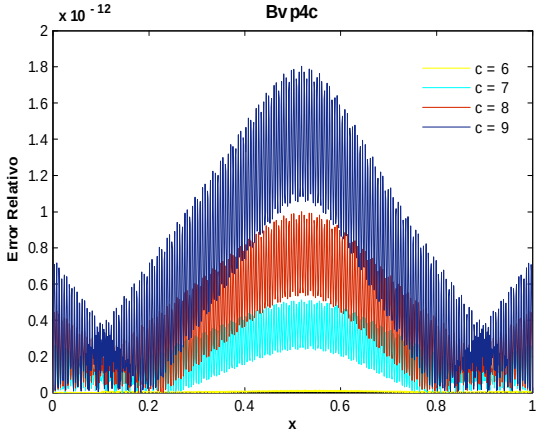


Fig. 159. Error Relativo de la solución Bvp4c para el ejemplo 4 con los valores del parámetro c=6,8,7,9 y tolerancias absoluta de 1e-12 y relativa de 1e-14. Cputime = 13.07 s.

Como se puede ver en las anteriores figuras, los órdenes de los errores absoluto y relativo utilizando bvp4c para $c = 1, 2, 3, 4, 5$ fueron de -4 (para valores $AbsTol = 1e-6, \mathrm{Re}\, lTol = 1e-3$) ver figs. 144 y 152, hasta -14 (con $AbsTol = 1e-12, \mathrm{Re}\, lTol = 1e-14$) ver figs. 147 y 155. Para $c = 6, 7, 8, 9$ los órdenes fueron de -4 hasta -12, con los mismos valores de tolerancias anteriores, ver figs. 148, 156 y 151,159. Al considerar valores de $1e-14$ o menores en ambas tolerancias el orden obtenido no mejoró.

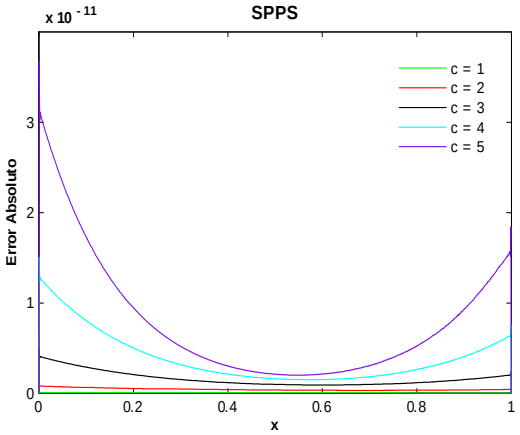


Fig. 160. Error Absoluto de la solución SPPS para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 considerando 15 potencias y 1000 puntos. Cputime = 20.29 s.

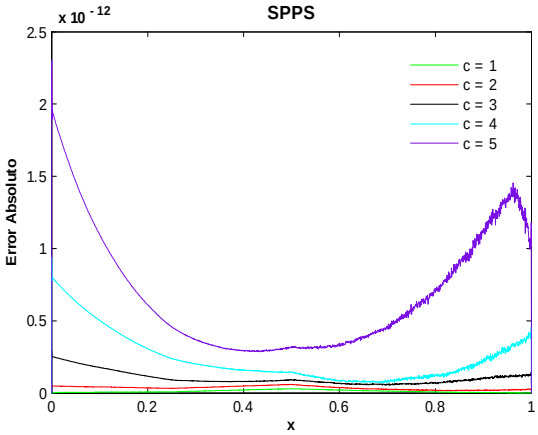


Fig. 161. Error Absoluto de la solución SPPS para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 considerando 15 potencias y 2000 puntos. Cputime = 40.14 s.

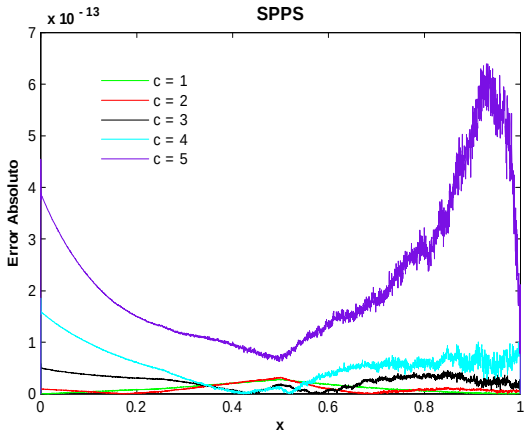


Fig. 162. Error Absoluto de la solución SPPS para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 considerando 15 potencias y 3000 puntos.
Cputime = 59.06 s.

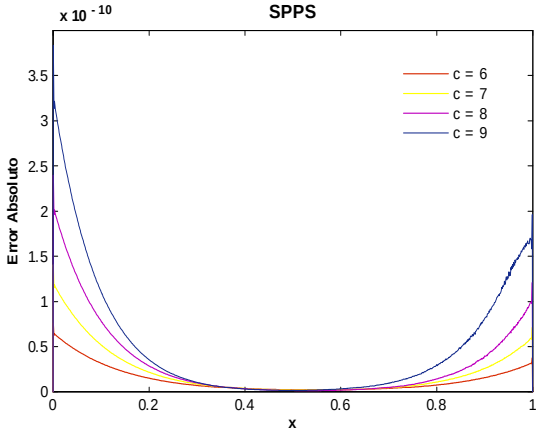


Fig.163. Error Absoluto de la solución SPPS para el ejemplo 4 con los valores del parámetro c=6,7,8,9 considerando 25 potencias y 1000 puntos.
Cputime = 27.23 s.

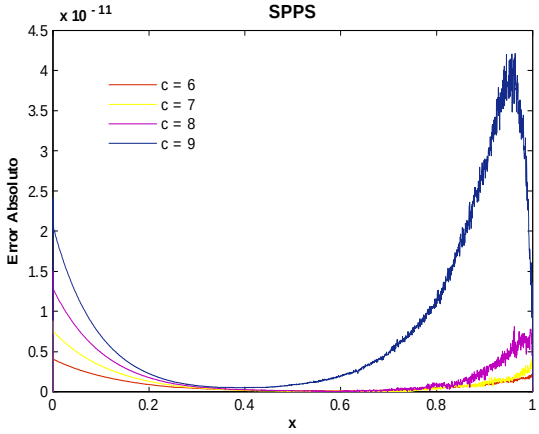


Fig. 164. Error Absoluto de la solución SPPS para el ejemplo 4 con los valores del parámetro c=6,7,8,9 considerando 25 potencias y 2000 puntos.
Cputime = 53.34s.

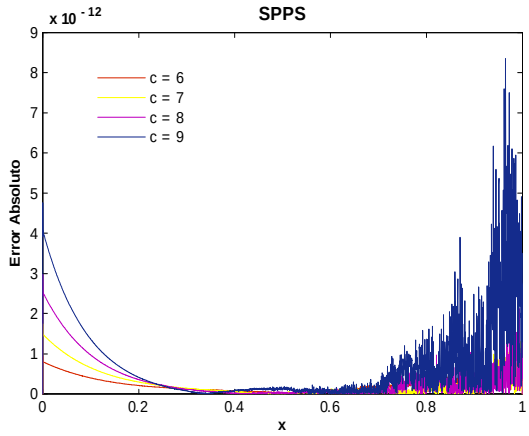


Fig. 165. Error Absoluto de la solución SPPS para el ejemplo 4 con los valores del parámetro c=6,7,8,9 considerando 25 potencias y 3000 puntos.
Cputime = 78.59 s.

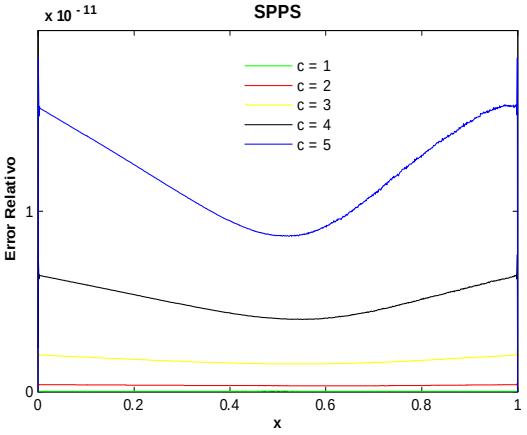


Fig. 166. Error Relativo de la solución SPPS para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 considerando 15 potencias y 1000 puntos.
Cputime = 20.34 s.

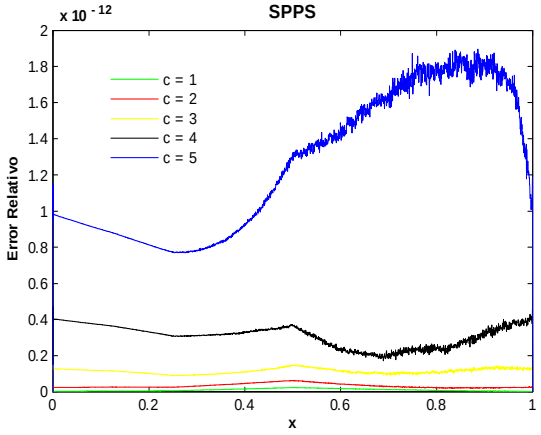


Fig. 167. Error Relativo de la solución SPPS para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 considerando 15 potencias y 2000 puntos.
Cputime = 39.53 s.

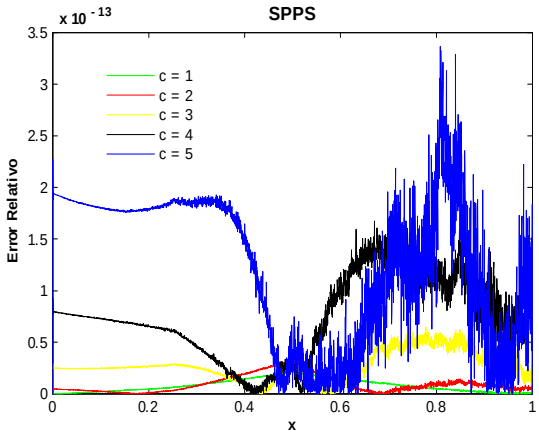


Fig.168. Error Relativo de la solución SPPS para el ejemplo 4 con los valores del parámetro c=1,2,3,4,5 considerando 20 potencias y 3000 puntos. Cputime = 78.94 s.

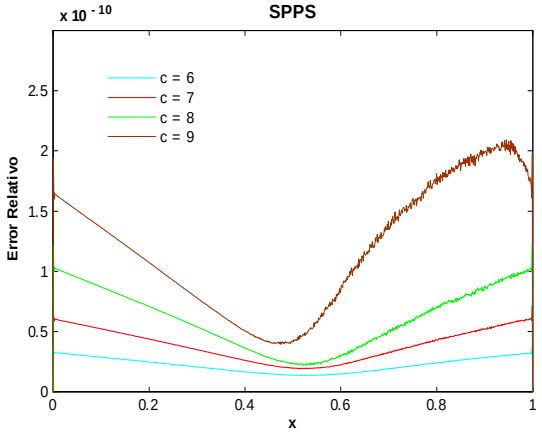


Fig. 169. Error Relativo de la solución SPPS para el ejemplo 4 con los valores del parámetro c=6,7,8,9 considerando 20 potencias y 1000 puntos. Cputime = 21.67 s.

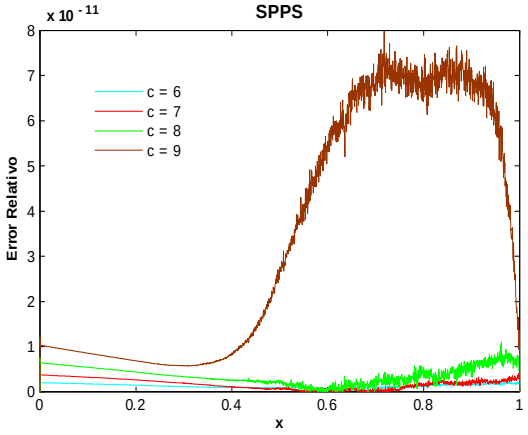


Fig. 170. Error Relativo de la solución SPPS para el ejemplo 4 con los valores del parámetro c=6,7,8,9 considerando 20 potencias y 2000 puntos. Cputime = 42.44 s.

En cuanto a SPPS el número de potencias utilizado fue escogido de tal manera que si ésta cantidad se aumenta dejando fijo el número de puntos, el orden obtenido no mejora. Así, por ejemplo para el error absoluto y $c = 6, 7, 8, 9$ bastó con tomar 25 potencias para obtener órdenes desde -10 con 1000 puntos hasta -12 con 3000 puntos, ver figs. 163-165, de manera que al incrementar las potencias el orden seguía siendo el mismo en cada caso. Para el error Relativo si $c = 1, 2, 3, 4, 5$, al considerar 20 potencias el orden fue de -11 con 1000 puntos a -13 con 3000, ver figs. 166-168. Al incrementar a $c = 6, 7, 8, 9$ y utilizando 20 potencias sólo se obtuvo órdenes de -10 con 1000 puntos y de -11 con 2000 puntos, ver figs.169 y 170, al aumentar potencias y términos no se obtuvo un mejor orden.

Al considerar un intervalo de longitud menor a la de $[0, 1]$ con los mismos valores de los parámetros el método SPPS obtiene una mejor aproximación a la solución exacta, pues las estimaciones (36) y (37) tienden a cero más rápido y en consecuencia los primeros N+1 términos aproximan mejor la solución . Para ilustrar lo anterior considérese la figura 170, se pueden tomar los intervalos [0, 0.25] y [0, 0.5] y mantener los mismos valores de los demás parámetros, esto es, $\alpha = 2$, $\beta = 1$, $c = 6, 7, 8, 9$. Nótese que para estos dos casos, la solución exacta descrita anteriormente sigue teniendo sentido, pues el determinante

$\Delta = e^{c(a-b)} - e^{-c(a-b)} \neq 0$ para los valores $a = 0, b = 0.25, 0.5, c = 6, 7, 8, 9$. Al calcular el error relativo obtenemos las siguientes gráficas.

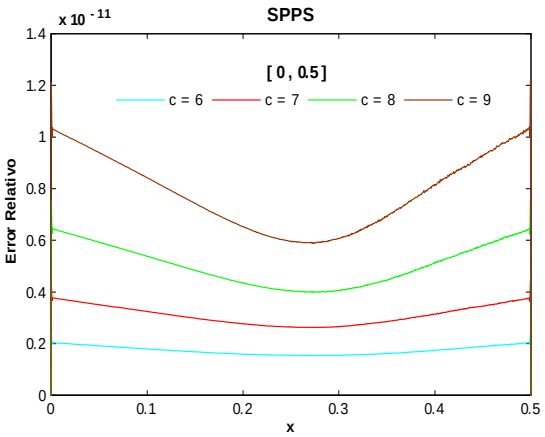


Fig. 171. Error Relativo de la solución SPPS para el ejemplo 4 con los valores del parámetro c=6,7,8,9 considerando 20 potencias y 1000 puntos.
Cputime = 21.83 s.

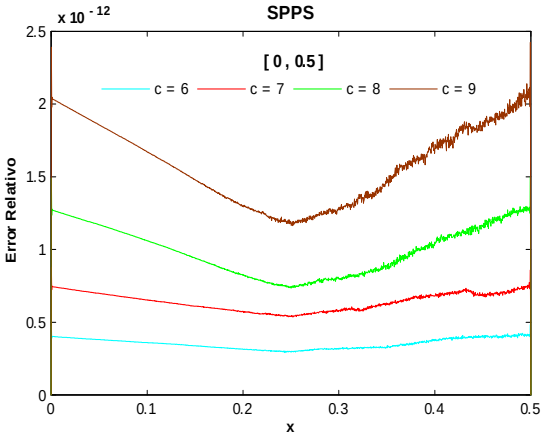


Fig.172. Error Relativo de la solución SPPS para el ejemplo 4 con los valores del parámetro c=6,7,8,9 considerando 20 potencias y 1500 puntos.
Cputime = 32.12 s.

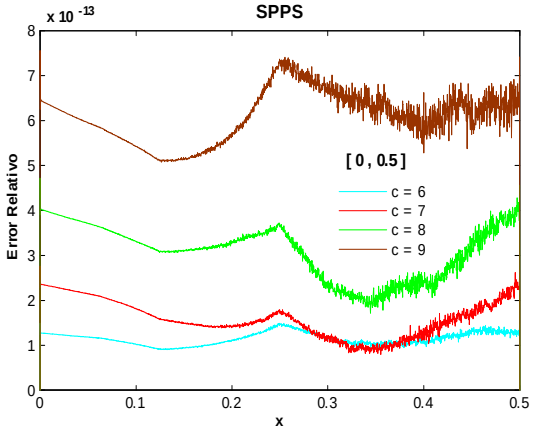


Fig. 173. Error Relativo de la solución SPPS para el ejemplo 4 con los valores del parámetro c=6,7,8,9 considerando 20 potencias y 2000 puntos.
Cputime = 42.87 s.

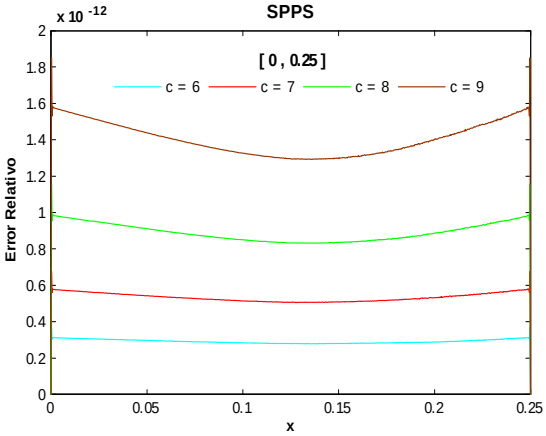


Fig.174. Error Relativo de la solución SPPS para el ejemplo 4 con los valores del parámetro c=6,7,8,9 considerando 20 potencias y 800 puntos.
Cputime = 17.02 s.

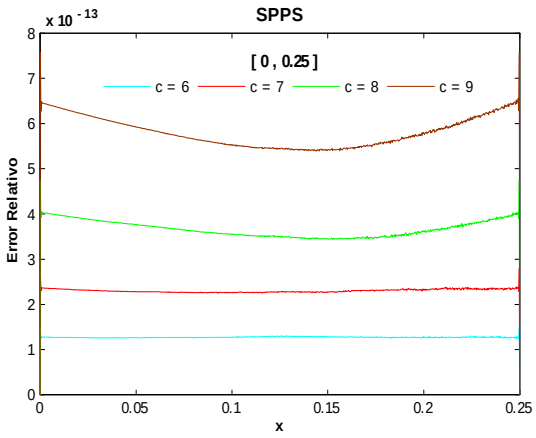


Fig. 175. Error Relativo de la solución SPPS para el ejemplo 4 con los valores del parámetro c=6,7,8,9 considerando 20 potencias y 100 puntos. Cputime = 21.01 s.

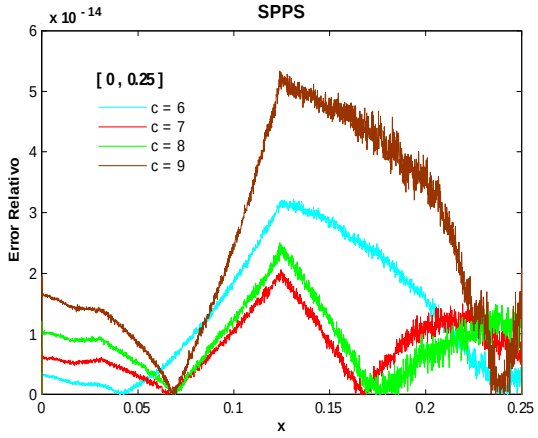


Fig. 176. Error Relativo de la solución SPPS para el ejemplo 4 con los valores del parámetro c=6,7,8,9 considerando 20 potencias y 2500 puntos. Cputime = 51.81 s.

Como se puede ver en las anteriores figuras, el orden del error relativo mejoró hasta tomar el valor de -13 con 2000 puntos y 20 potencias, ver fig.173, para el intervalo $[0,\ 0.5]$. Y se obtuvo el orden de -14 para el intervalo $[0.\ 0.25]$ con 2500 puntos y 20 potencias, ver fig. 176.

Nótese que esto no resuelve el problema inicial en todo el intervalo $[0, 1]$, sólo fue un ejemplo en el cuál se muestra la mejora que tiene el orden del error si se toma intervalos de longitud menor a la del inicial. La forma de resolver el problema original con mayor exactitud se explica en la sección 7, donde se divide el intervalo original en dos o más subintervalos y se resuelven problemas de Cauchy asociados a cada subintervalo.

Ejemplo 5

Solución Exacta

Considérese el problema con valores en la frontera

$$\begin{aligned} y'' - k[k\sin^2(cx) + c\cos(cx)]y &= 0 \\ y(a) &= \alpha \\ y(b) &= \beta \end{aligned} \tag{53}$$

donde $c, k \neq 0, \alpha, \beta, a, b \in \mathbb{R}$, $a < b$.
De acuerdo a [18] la solución general de la ecuación diferencial en (53) es $y_g = c_1 e^{-\frac{k}{c}\cos(cx)} + c_2 e^{-\frac{k}{c}\cos(cx)} F(x)$ donde c_1 y c_2 son constantes arbitrarias y $F(x) = \int e^{\frac{2k}{c}\cos(cx)} dx$. Entonces se debe cumplir

$$\begin{aligned} y_g(a) &= c_1 e^{-\frac{k}{c}\cos(ca)} + c_2 e^{-\frac{k}{c}\cos(ca)} F(a) = \alpha \\ y_g(b) &= c_1 e^{-\frac{k}{c}\cos(cb)} + c_2 e^{-\frac{k}{c}\cos(cb)} F(b) = \beta, \end{aligned} \tag{54}$$

Por los Teoremas 4 y 5, el problema (53) tiene solución única si y sólo si

$$\Delta = \begin{vmatrix} e^{-\frac{k}{c}\cos(ca)} & e^{-\frac{k}{c}\cos(ca)} F(a) \\ e^{-\frac{k}{c}\cos(cb)} & e^{-\frac{k}{c}\cos(cb)} F(b) \end{vmatrix} = e^{-\frac{k}{c}[\cos(ca)+\cos(cb)]}[F(b) - F(a)] \neq 0. \tag{55}$$

Usando el hecho de que $k \neq 0$ nótese que $e^{-\frac{k}{c}[\cos(ca)+\cos(cb)]}[F(b)-F(a)] = 0 \quad \Leftrightarrow \quad F(b)-F(a) = 0$. Supongamos se cumple (55). Resolviendo el sistema (54) mediante la regla de Cramer:

$$c_1 = \frac{\begin{vmatrix} \alpha & e^{-\frac{k}{c}\cos(ca)}F(a) \\ \beta & e^{-\frac{k}{c}\cos(cb)}F(b) \end{vmatrix}}{\Delta} = \frac{\alpha e^{-\frac{k}{c}\cos(cb)}F(b) - \beta e^{-\frac{k}{c}\cos(ca)}F(a)}{e^{-\frac{k}{c}[\cos(ca)+\cos(cb)]}[F(b)-F(a)]}$$

$$c_2 = \frac{\begin{vmatrix} e^{-\frac{k}{c}\cos(ca)} & \alpha \\ e^{-\frac{k}{c}\cos(cb)} & \beta \end{vmatrix}}{\Delta} = \frac{\beta e^{-\frac{k}{c}\cos(ca)} - \alpha e^{-\frac{k}{c}\cos(cb)}}{e^{-\frac{k}{c}[\cos(ca)+\cos(cb)]}[F(b)-F(a)]}$$

De donde se concluye que la solución de (53) es:

$$y_p = \frac{\alpha e^{-\frac{k}{c}\cos(cb)}F(b) - \beta e^{-\frac{k}{c}\cos(ca)}F(a)}{e^{-\frac{k}{c}[\cos(ca)+\cos(cb)]}[F(b)-F(a)]}e^{-\frac{k}{c}\cos(cx)} + \frac{\beta e^{-\frac{k}{c}\cos(ca)} - \alpha e^{-\frac{k}{c}\cos(cb)}}{e^{-\frac{k}{c}[\cos(ca)+\cos(cb)]}[F(b)-F(a)]}e^{-\frac{k}{c}\cos(cx)}F(x)$$

donde $F(x) = \int e^{\frac{2k}{c}\cos(cx)}dx$.

A continuación se presentan las gráficas de algunos potenciales y soluciones exactas para los valores de parámetros $k = 3, 15, 25$ y $c = 2, 15$.

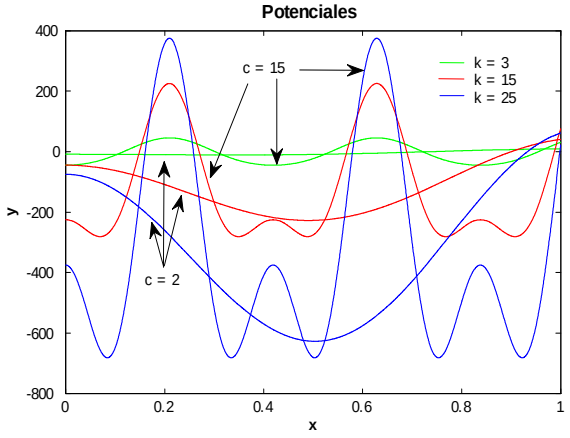


Fig. 177

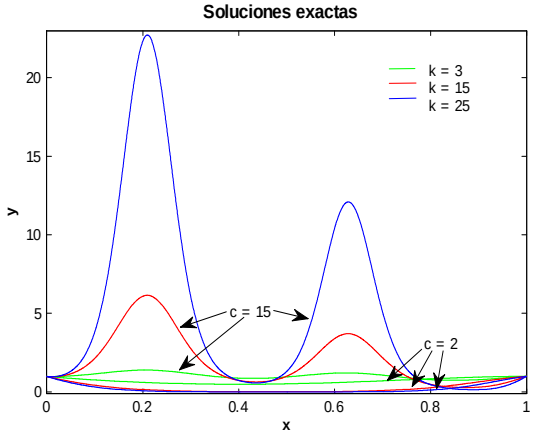


Fig. 178

Soluciones Numéricas

A continuación se presentan las gráficas del error relativo calculado con las soluciones de los métodos SPPS y bvp4c para los valores de los parámetros $\alpha = 1, \beta = 1, a = 0, b = 1, k = 3, 15, 25, c = 2n+1, n = 0, 1, ..., 12$.

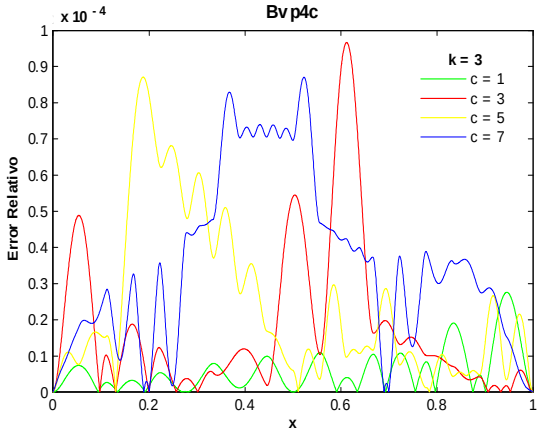


Fig. 179. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=3, c=1,3,5,7 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 1.85 s.

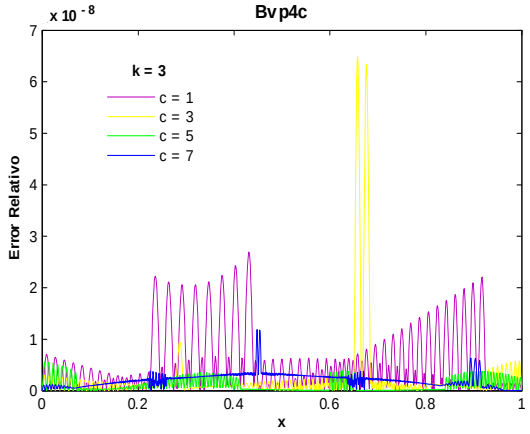


Fig. 180. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=3, c=1,3,5,7 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 1.99 s.

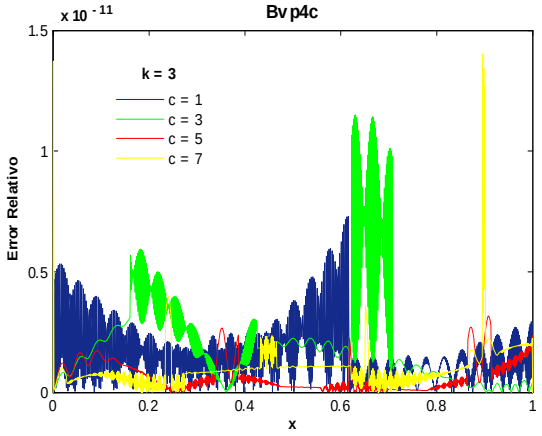


Fig. 181. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=3, c=1,3,5,7 y tolerancias absoluta de 1e-9 y relativa de 1e-11. Cputime = 4.33 s.

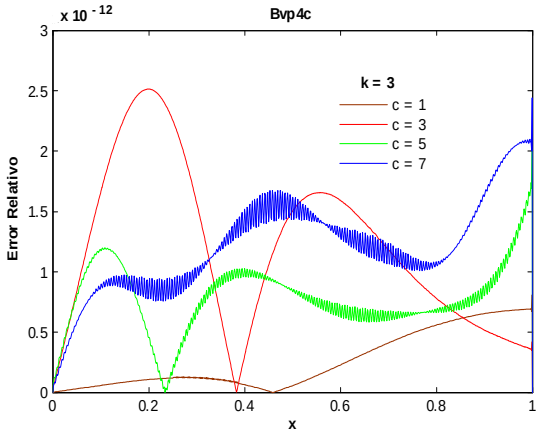


Fig. 182. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=3, c=1,3,5,7 y tolerancias absoluta de 1e-12 y relativa de 1e-14. Cputime = 15.91 s.

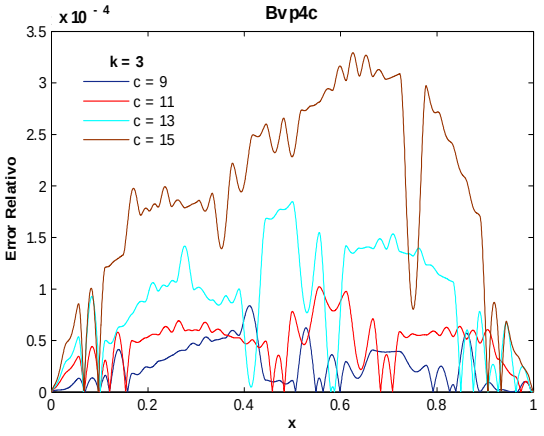


Fig. 183. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=3, c=9,11,13,15 y la tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 1.98 s.

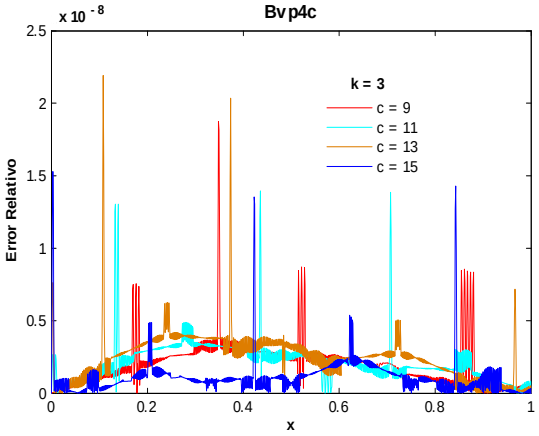


Fig. 184. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=3, c=9,11,13,15 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 3.89 s

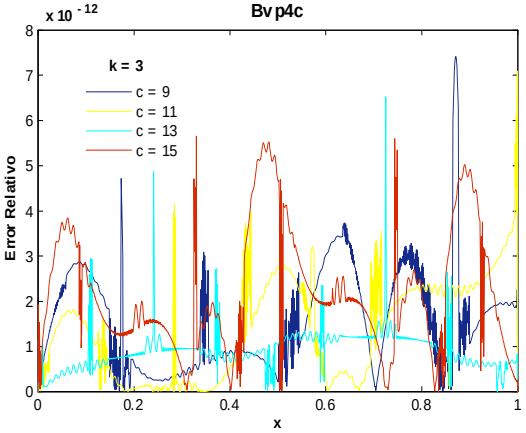


Fig. 185. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=3, c=9,11,13,15 y tolerancias absoluta de 1e-9 y relativa de 1e-11. Cputime = 7.33 s.

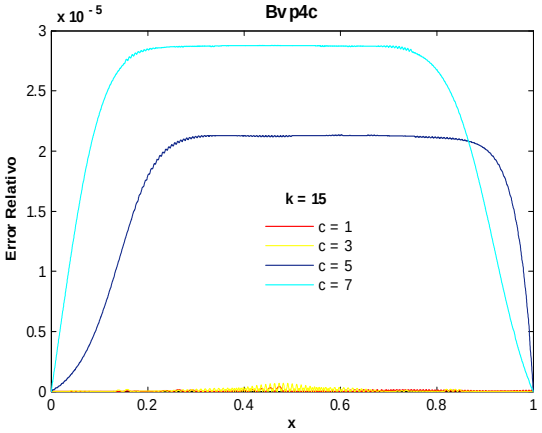


Fig. 186. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=15, c=1,3,5,7 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 2.36 s.

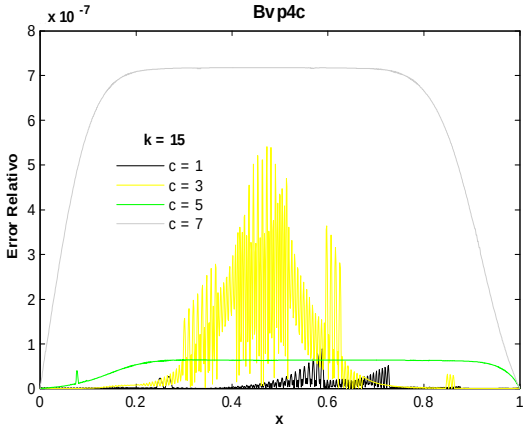


Fig. 187. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=15, c=1,3,5,7 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 4.16 s.

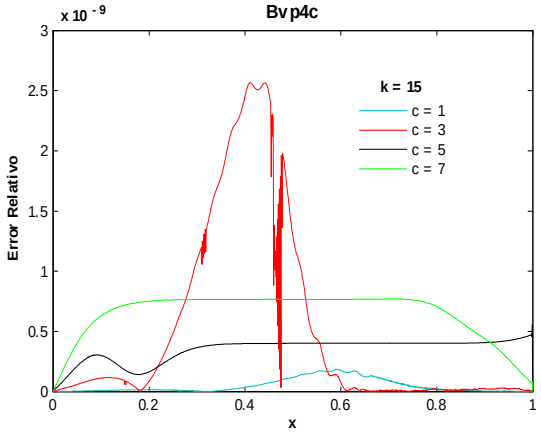


Fig. 188. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=15, c=1,3,5,7 y tolerancias absoluta de 1e-9 y relativa de 1e-11. Cputime = 12.9 s.

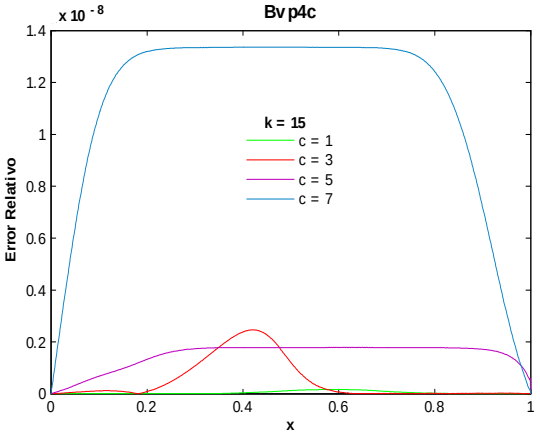


Fig. 189. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=15, c=1,3,5,7 y tolerancias absoluta de 1e-12 y relativa de 1e-14. Cputime = 13.56 s.

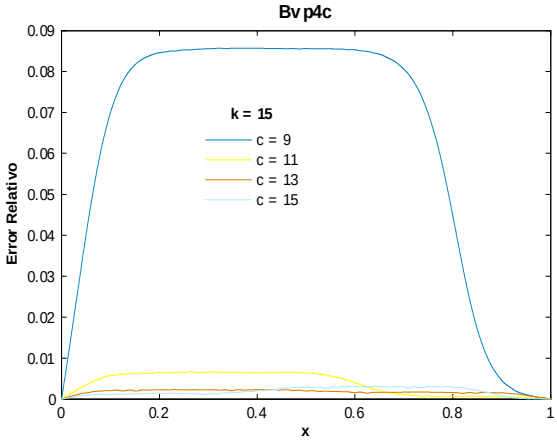


Fig. 190. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=15, c=9,11,13,15 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 2.53 s.

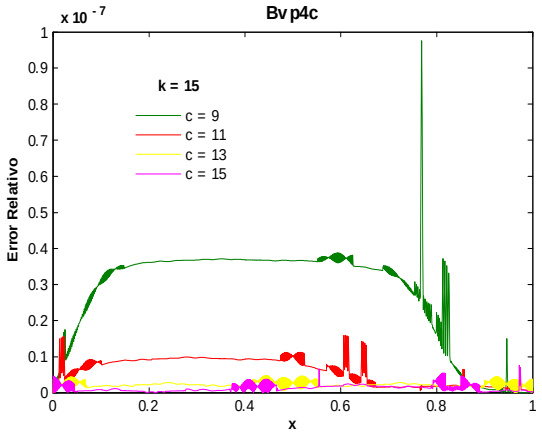


Fig. 191. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=15, c=9,11,13,15 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 4.72 s.

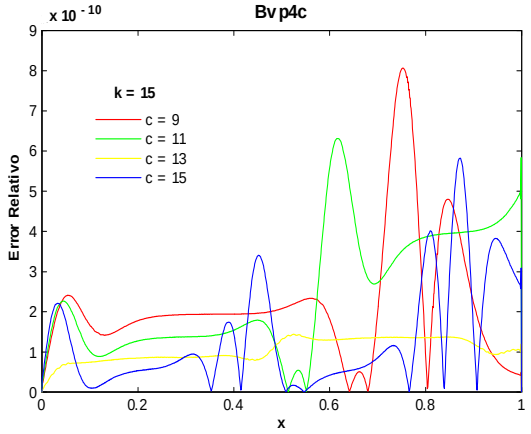


Fig. 192. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=15, c=9,11,13,15 y tolerancias absoluta de 1e-9 y relativa de 1e-11. Cputime = 11.16 s.

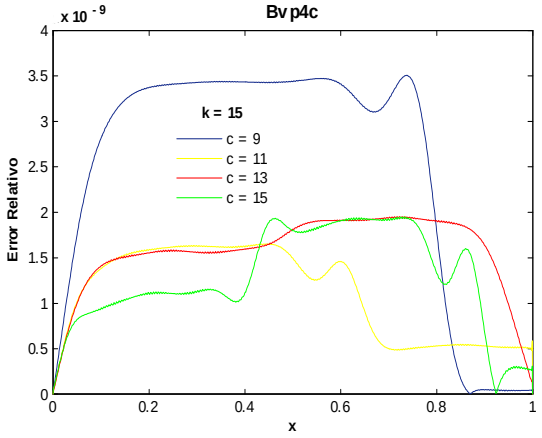


Fig. 193. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=15, c=9,11,13,15 y tolerancias absoluta de 1e-12 y relativa de 1e-14. Cputime = 6.69 s.

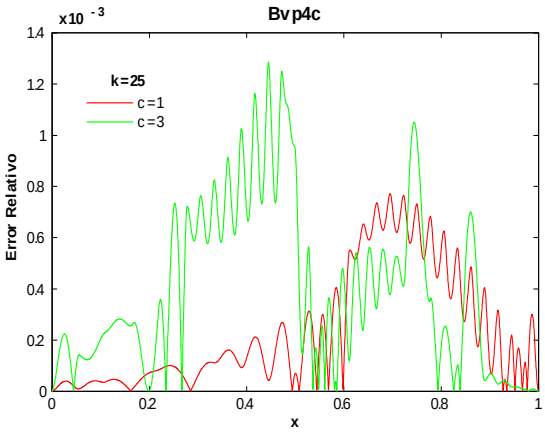


Fig. 194. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=25, c=1,3 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 1.43 s.

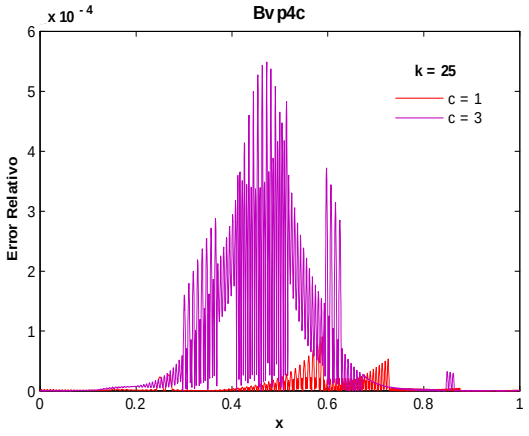


Fig. 195. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=25, c=1,3 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 1.65 s.

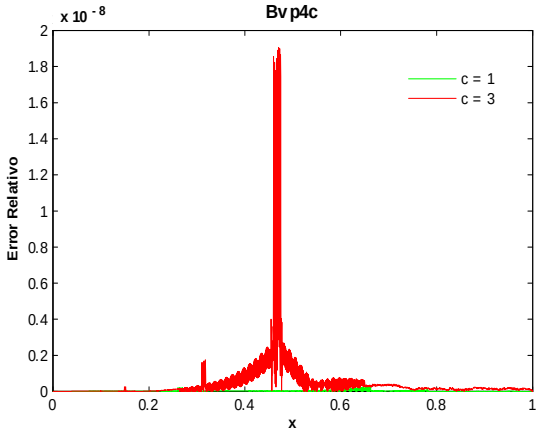


Fig. 196. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=25, c=1,3 y tolerancias absoluta de 1e-9 y relativa de 1e-11. Cputime = 1.85 s.

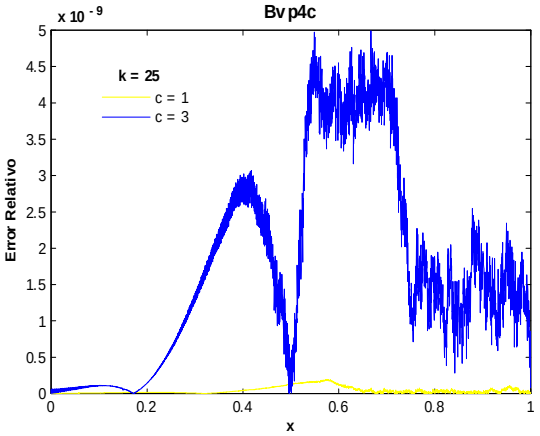


Fig. 197. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=25, c=1,3 y tolerancias absoluta de 1e-12 y relativa de 1e-14. Cputime = 1.85 s.

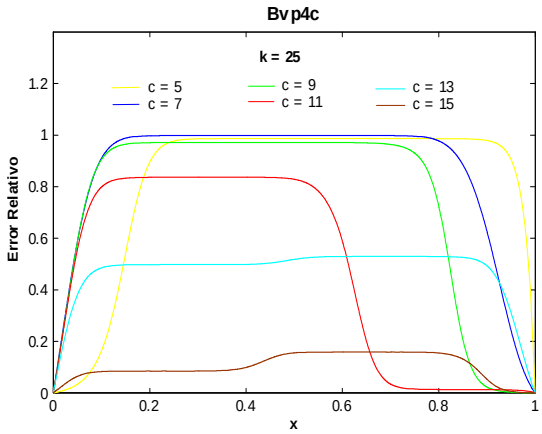


Fig. 198. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=25, c=5,7,9,11,13,15 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 7.88 s.

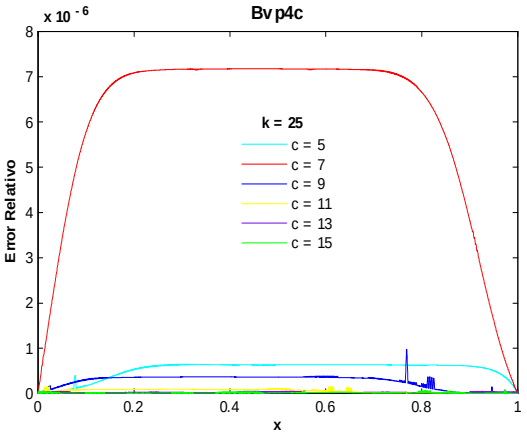


Fig. 199. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=25, c=5,7,9,11,13,15 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 8.73 s.

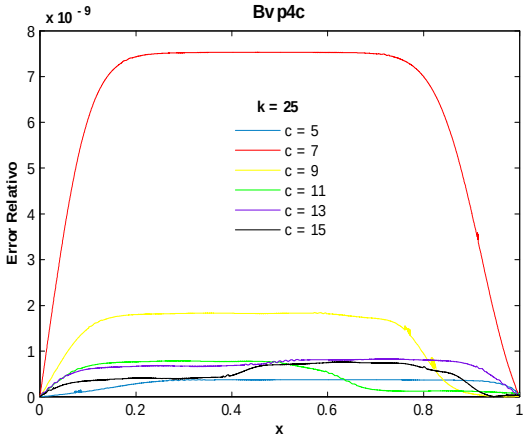


Fig. 200. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=25, c=5,7,9,11,13,15 y tolerancias absoluta de 1e-9 y relativa de 1e-11. Cputime = 13.42 s.

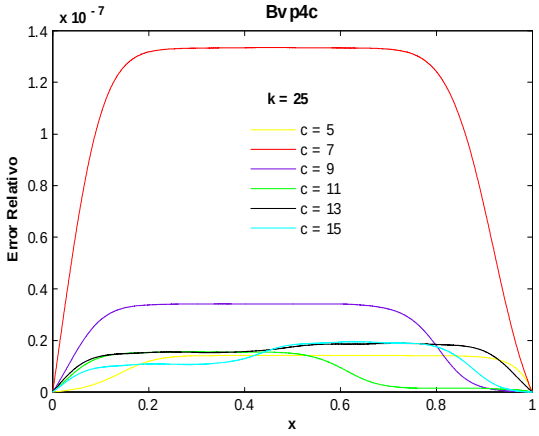


Fig. 201. Error Relativo de la solución Bvp4c para el ejemplo 5 con los valores de los parámetros k=25, c=5,7,9,11,13,15 y tolerancias absoluta de 1e-12 y relativa de 1e-14. Cputime = 9.11s.

Las figuras anteriores muestran que al disminuir el valor de las tolerancias absoluta y relativa, el orden del error disminuye, esto es, se obtiene una mejor aproximación de la solución exacta. Aunque al tomar valores muy pequeños en las tolerancias el orden del error llega a aumentar. Por ejemplo en las figs. 183-185 se observa que para $k = 3$ y $c = 15$ el orden del error fue de -4 para valores $AbsTol = 1e-6$, $RelTol = 1e-3$ hasta -12 al tomar $AbsTol = 1e-9, RelTol = 1e-11$. Para $k = 15$ y $c = 15$ el orden obtenido con tolerancias por defecto es de -2 (ver fig.190) y al tomar $AbsTol = 1e-9$, $RelTol = 1e-11$ se obtiene -10 (ver fig. 192). Si $k = 25$ y $c = 15$ las tolerancias por defecto nos dan un orden de -1 (ver fig. 198) y considerando $AbsTol = 1e-9$, $RelTol = 1e-11$ obtenemos -9 (ver fig. 200). Note que conforme k y c crecen el orden del error empeora. Las figuras 189,193,201 muestran que al utilizar tolerancias muy pequeñas el orden del error puede empeorar.

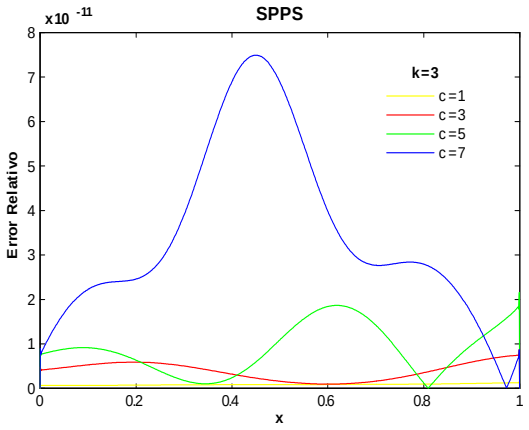


Fig. 202. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=3, c=1,3,5,7 considerando 15 potencias y 1000 puntos. Cputime=16.81 s

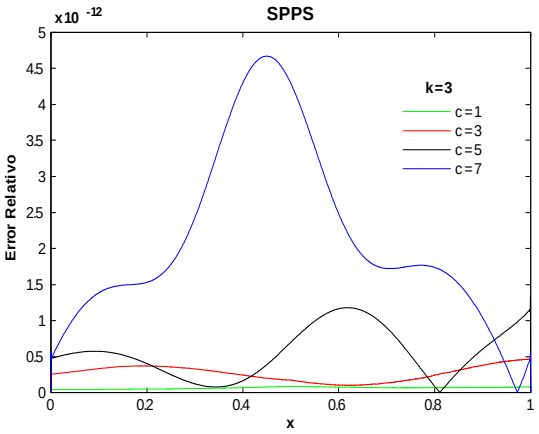


Fig. 203. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=3, c=1,3,5,7 considerando 15 potencias y 2000 puntos. Cputime = 32.22 s.

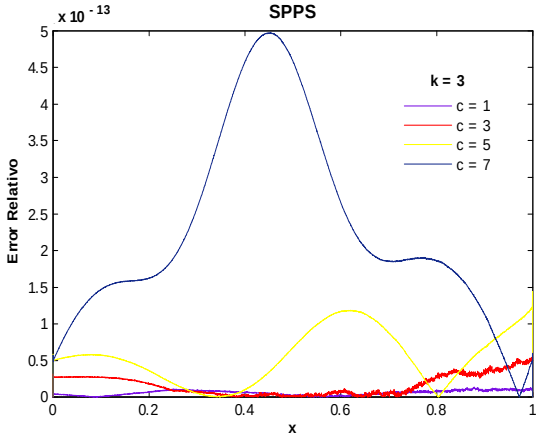


Fig. 204. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=3, c=1,3,5,7 considerando 15 potencias y 3500 puntos. Cputime=52.22 s

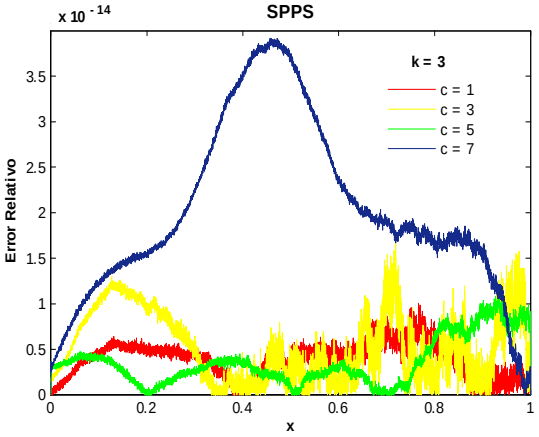


Fig.205. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=3, c=1,3,5,7 considerando 15 potencias y 7000 puntos. Cputime=108.78 s

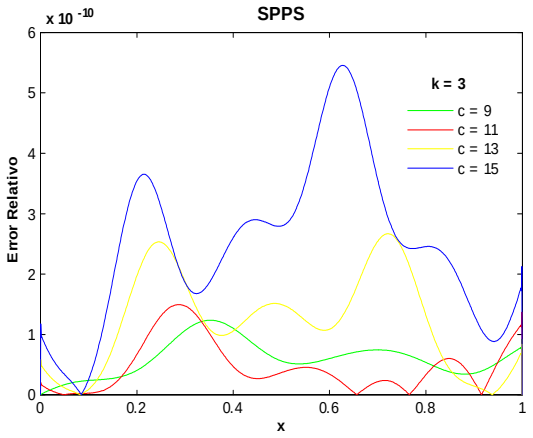


Fig. 206. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=3, c=9,11,13,15 considerando 15 potencias y 1000 puntos. Cputime = 16.82 s

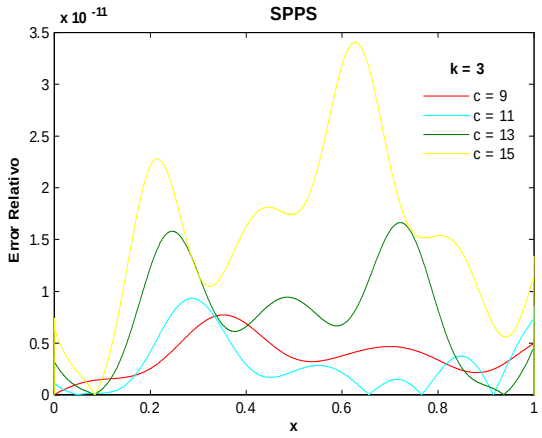


Fig. 207. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=3, c=9,11,13,15 considerando 15 potencias y 2000 puntos. Cputime = 32.29 s.

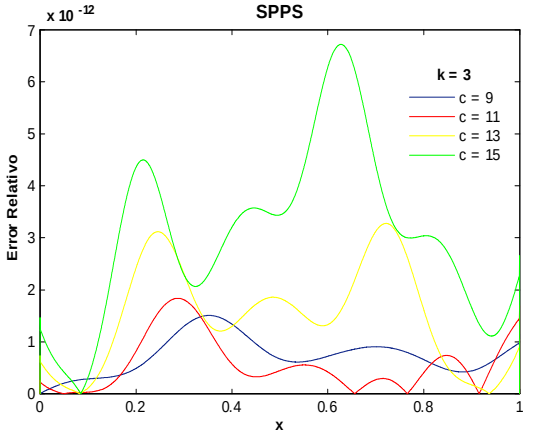


Fig. 208. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=3, c=9,11,13,15 considerando 15 potencias y 3000 puntos. Cputime= 47.56 s

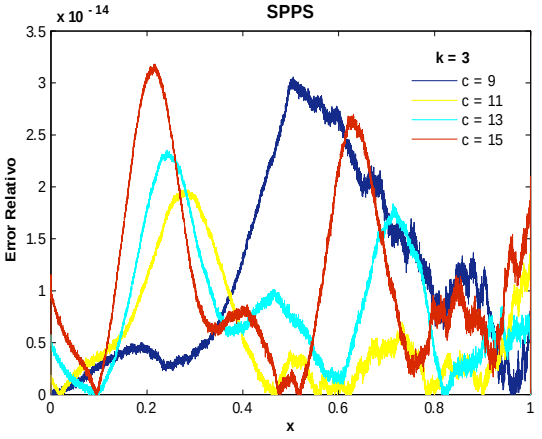


Fig. 209. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=3, c=9,11,13,15 considerando 15 potencias y 10100 puntos. Cputime= 156.47 s

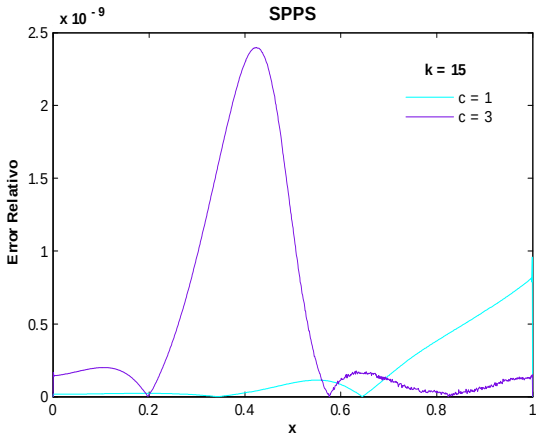


Fig. 210. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=15, c=1,3 considerando 25 potencias y 1000 puntos. Cputime = 14 s

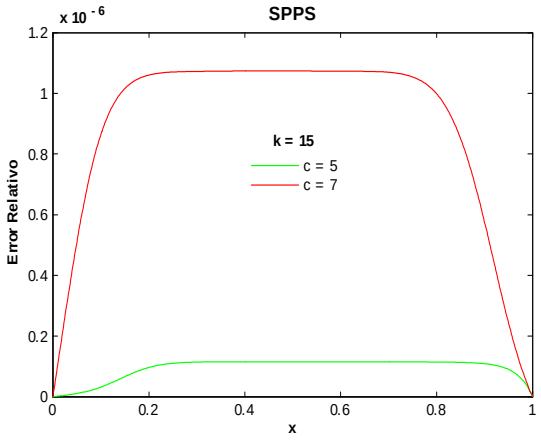


Fig. 211. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=15, c=5,7 considerando 25 potencias y 1000 puntos. Cputime = 14.07 s

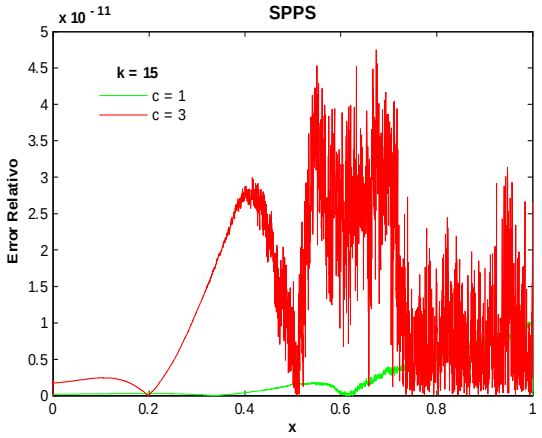


Fig. 212. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=15, c=1,3 considerando 25 potencias y 3000 puntos. Cputime = 39.49 s

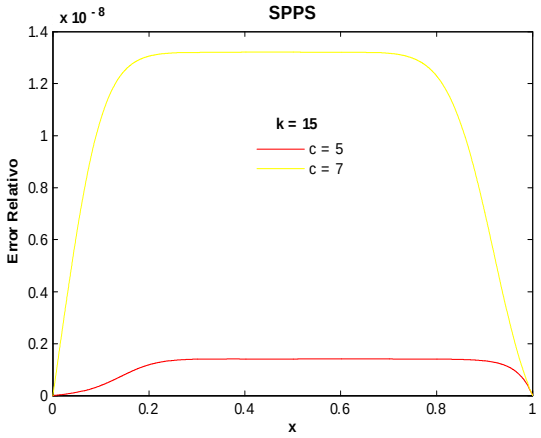


Fig. 213. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=15, c=5,7 considerando 25 potencias y 3000 puntos. Cputime = 39.46 s

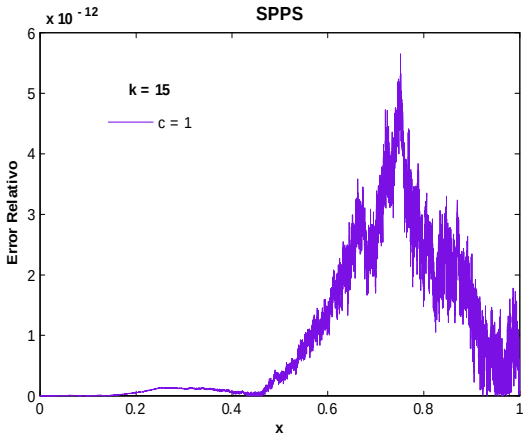


Fig. 214. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=15, c=1 considerando 25 potencias y 9000 puntos. Cputime = 58.08 s

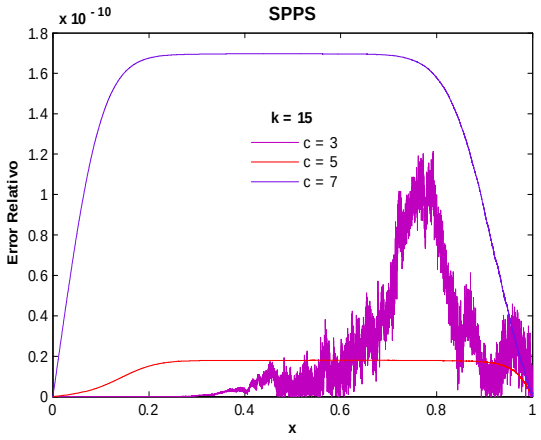


Fig. 215. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=15, c=3,5,7 considerando 25 potencias y 9000 puntos. Cputime = 173.43 s

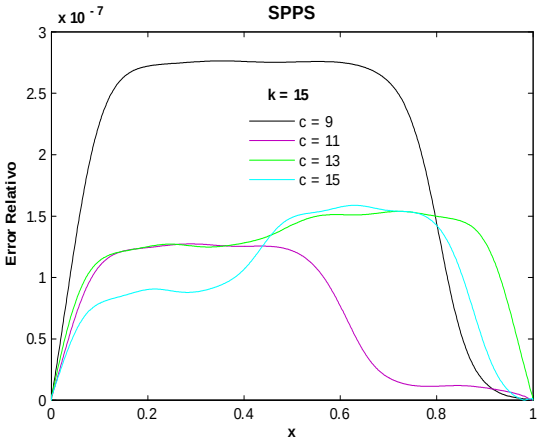


Fig. 216. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=15, c=9,11,13,15 considerando 30 potencias y 1000 puntos. Cputime = 32.65 s.

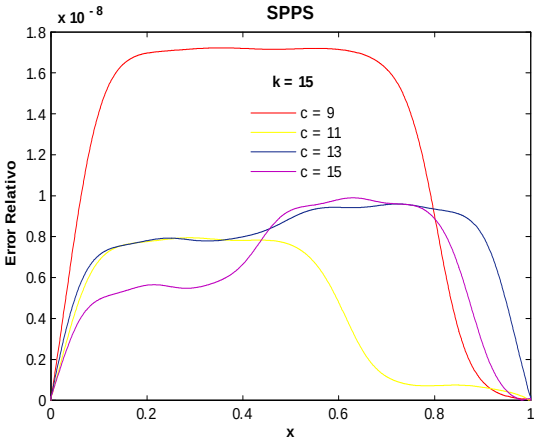


Fig. 217. Error Relativo de la solución SPPS para el ejemplo 5con los valores de los parámetros k=15, c=9,11,13,15 considerando 30 potencias y 2000 puntos. Cputime = 63.17 s.

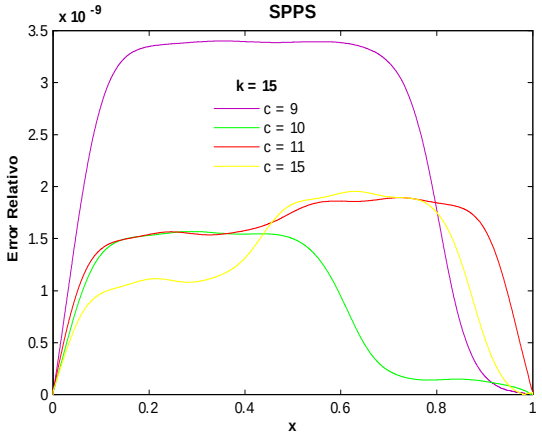


Fig. 218. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=15, c=9,11,13,15 considerando 30 potencias y 3000 puntos. Cputime = 92.94 s.

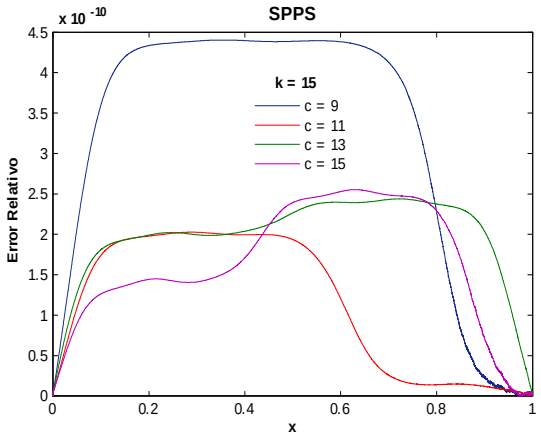


Fig. 219. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=15, c=9,11,13,15 considerando 30 potencias y 5000 puntos. Cputime = 154.64 s.

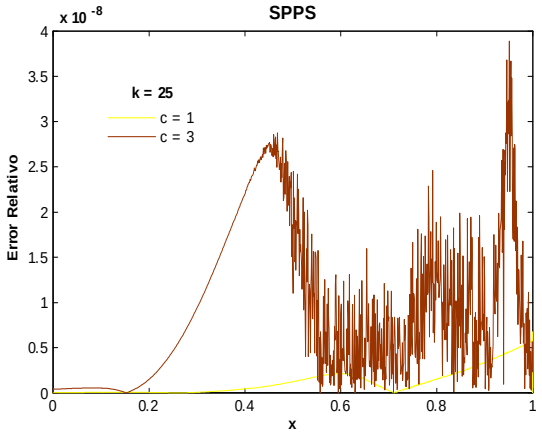


Fig. 220. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=25, c=1,3 considerando 30 potencias y 1000 puntos. Cputime = 16.61 s.

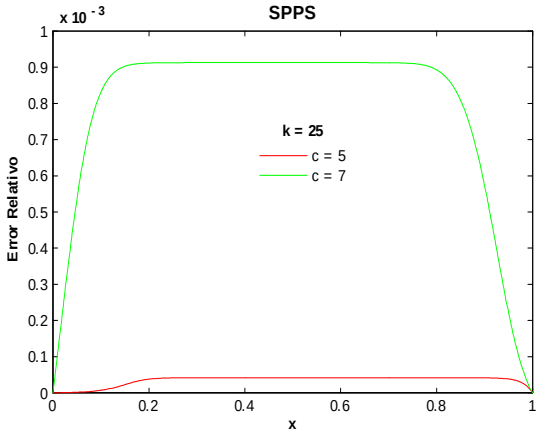


Fig. 221. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=25, c=5,7 considerando 30 potencias y 1000 puntos. Cputime = 16.59 s.

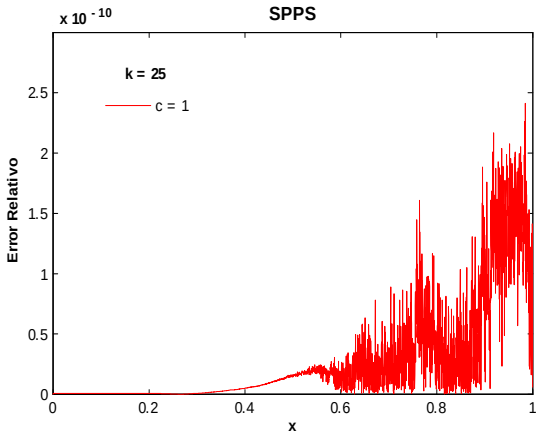


Fig. 222. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=25, c=1 considerando 30 potencias y 3000 puntos. Cputime = 23.82 s.

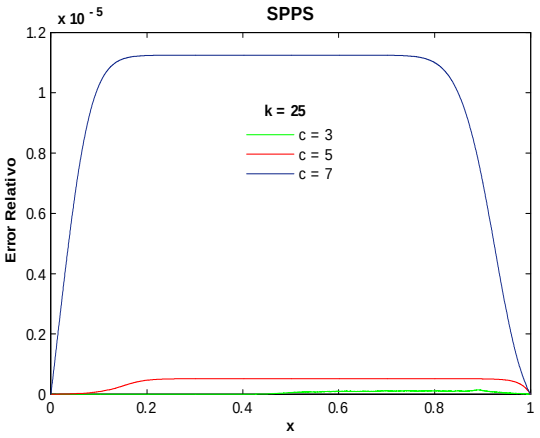


Fig. 223. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=25, c=3,5,7 considerando 30 potencias y 3000 puntos. Cputime = 70.3 s.

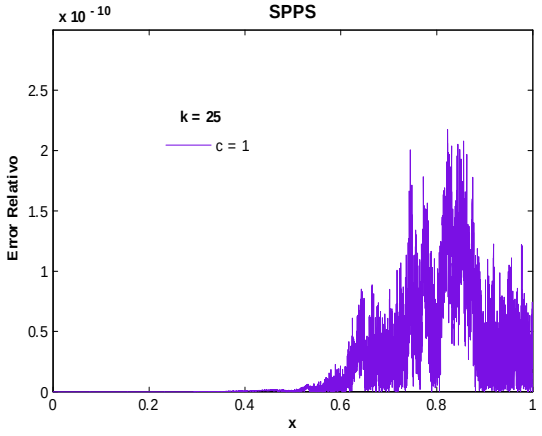


Fig. 224. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=25, c=1 considerando 30 potencias y 7000 puntos. Cputime = 54.26 s.

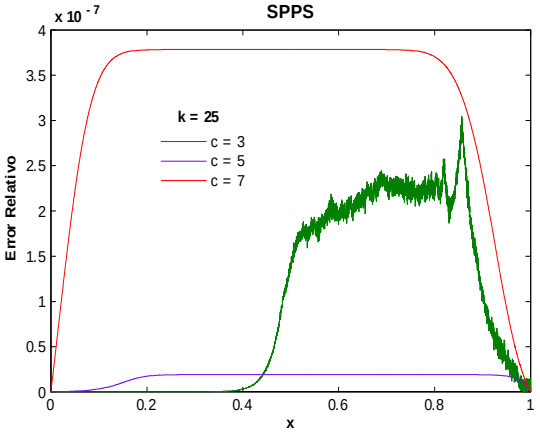


Fig. 225. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=25, c=3,5,7 considerando 30 potencias y 7000 puntos. Cputime = 161.45 s.

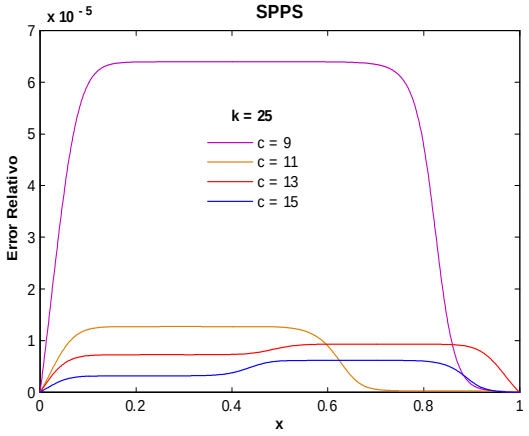


Fig. 226. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=25, c=9,11,13,15 considerando 30 potencias y 1000 puntos. Cputime = 33.88 s.

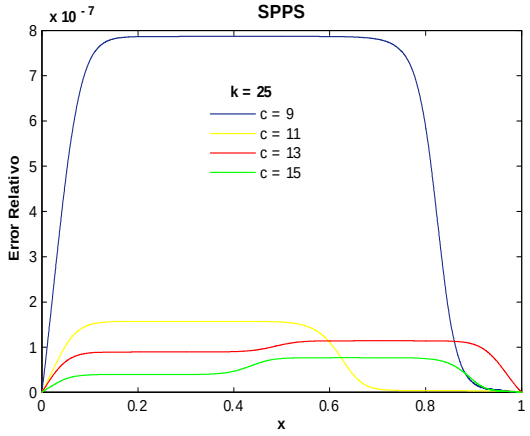


Fig. 227. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=25, c=9,11,13,15 considerando 30 potencias y 3000 puntos. Cputime = 157.81 s.

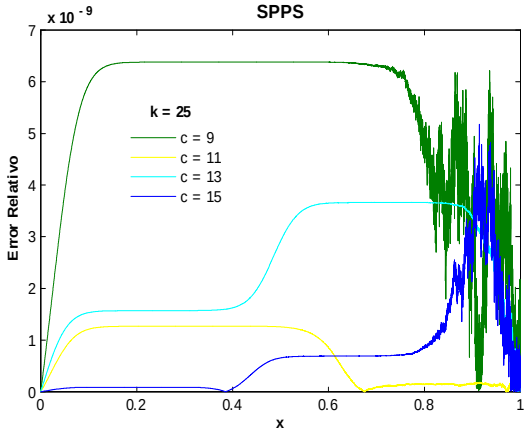


Fig. 228. Error Relativo de la solución SPPS para el ejemplo 5 con los valores de los parámetros k=25, c=9,11,13,15 considerando 30 potencias y 10000 puntos. Cputime = 312.24 s.

Utilizando SPPS para $k = 3$ y $c = 15$ basta con tomar 15 potencias para obtener órdenes del error relativo desde -10, con 1000 puntos, hasta -14, con 10100 puntos, ver figs. 206-209. Al incrementar c y k es necesario aumentar la cantidad de potencias para obtener mejores aproximaciones, pues el orden del error empeora, por ejemplo en la figuras 216-219 se muestra que para $k = 15$ y $c = 15$ se obtienen órdenes de error de -7 con 1000 puntos, hasta -10 con 5000 puntos, todos considerando 30 potencias. Para $k = 25$ y $c = 15$ el orden del error fue de -5 con 1000 puntos, hasta -9 con 10000 puntos, ver figs. 226-228, la cantidad de potencias consideradas fue 30. En todos los casos el número de potencias usado fue tal que al aumentar éste y dejar fija la cantidad de puntos el orden se preserva.

Ejemplo 6

Solución Exacta

Considérese el problema con valores en la frontera

$$\begin{aligned} y'' + c\left[\frac{1}{2} + \left(\frac{1}{2} - c\right)\tan^2\left(\frac{x}{2}\right)\right] y &= 0 \\ y(a) &= \alpha \\ y(b) &= \beta \end{aligned} \tag{56}$$

donde $c \neq 0, \alpha, \beta, a, b \in \mathbb{R}$, $a < b$.
Según [18] la solución general de la ecuación diferencial en (56) es $y_g = c_1 \cos^{2c}\left(\frac{x}{2}\right) + c_2 F(x)\cos^{2c}\left(\frac{x}{2}\right)$ donde c_1 y c_2 son constantes arbitrarias y $F(x) = \int \frac{dx}{\cos^{4c}\left(\frac{x}{2}\right)}$. Entonces se debe cumplir

$$\begin{aligned} y_g(a) &= c_1 \cos^{2c}\left(\frac{a}{2}\right) + c_2 F(a)\cos^{2c}\left(\frac{a}{2}\right) = \alpha \\ y_g(b) &= c_1 \cos^{2c}\left(\frac{b}{2}\right) + c_2 F(b)\cos^{2c}\left(\frac{b}{2}\right) = \beta. \end{aligned} \tag{57}$$

Por losTeoremas 4 y 5, el problema (56) tiene solución única si y sólo si

$$\Delta = \begin{vmatrix} \cos^{2c}\left(\frac{a}{2}\right) & F(a)\cos^{2c}\left(\frac{a}{2}\right) \\ \cos^{2c}\left(\frac{b}{2}\right) & F(b)\cos^{2c}\left(\frac{b}{2}\right) \end{vmatrix} = \cos^{2c}\left(\frac{a}{2}\right)\cos^{2c}\left(\frac{b}{2}\right)[F(b) - F(a)] \neq 0 \tag{58}$$

Nótese que
$\cos^{2c}\left(\frac{a}{2}\right)\cos^{2c}\left(\frac{b}{2}\right)[F(b) - F(a)] = 0 \quad \Leftrightarrow \quad a = (2k+1)\pi$ ó $b = (2k+1)\pi$, $k \in \mathbb{Z}$ ó $F(b) = F(a)$.

Supongamos se cumple (58). Resolviendo el sistema (57) mediante la regla de Cramer:

$$\begin{aligned} c_1 &= \frac{\begin{vmatrix} \alpha & F(a)\cos^{2c}\left(\frac{a}{2}\right) \\ \beta & F(b)\cos^{2c}\left(\frac{b}{2}\right) \end{vmatrix}}{\Delta} = \frac{\alpha F(b)\cos^{2c}\left(\frac{b}{2}\right) - \beta F(a)\cos^{2c}\left(\frac{a}{2}\right)}{\cos^{2c}\left(\frac{a}{2}\right)\cos^{2c}\left(\frac{b}{2}\right)[F(b) - F(a)]} \\ c_2 &= \frac{\begin{vmatrix} \cos^{2c}\left(\frac{a}{2}\right) & \alpha \\ \cos^{2c}\left(\frac{b}{2}\right) & \beta \end{vmatrix}}{\Delta} = \frac{\beta\cos^{2c}\left(\frac{a}{2}\right) - \alpha\cos^{2c}\left(\frac{b}{2}\right)}{\cos^{2c}\left(\frac{a}{2}\right)\cos^{2c}\left(\frac{b}{2}\right)[F(b) - F(a)]} \end{aligned}$$

Por lo tanto la solución de (56) es:

$$y_p = \frac{\alpha F(b)\cos^{2c}\left(\frac{b}{2}\right) - \beta F(a)\cos^{2c}\left(\frac{a}{2}\right)}{\cos^{2c}\left(\frac{a}{2}\right)\cos^{2c}\left(\frac{b}{2}\right)[F(b) - F(a)]} e^{-\frac{k}{c}\cos(cx)} + \frac{\beta\cos^{2c}\left(\frac{a}{2}\right) - \alpha\cos^{2c}\left(\frac{b}{2}\right)}{\cos^{2c}\left(\frac{a}{2}\right)\cos^{2c}\left(\frac{b}{2}\right)[F(b) - F(a)]} e^{-\frac{k}{c}\cos(cx)} F(x)$$

donde $F(x) = \int \frac{dx}{\cos^{4c}\left(\frac{x}{2}\right)}$.

A continuación se presentan las gráficas de algunas soluciones exactas y potenciales con valores de parámetros $\alpha = 2, \beta = 1, a = 0, b = 1, c = 2, 6, 9, 12, 15, 18, 22, 26, 31, 38, 42$.

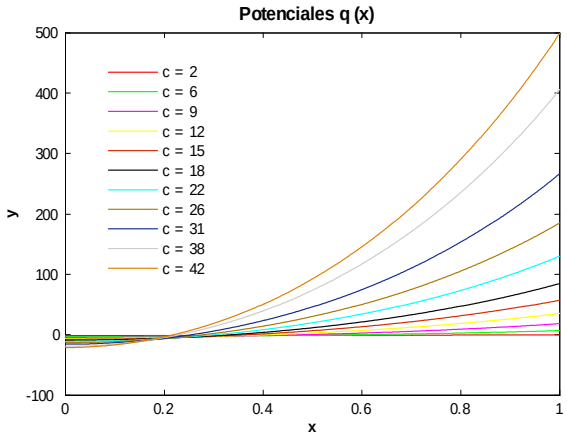


Fig. 229

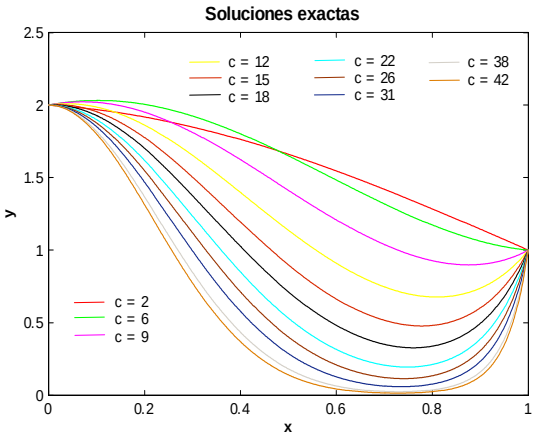


Fig. 230

Soluciones Numéricas

Las siguientes gráficas contienen los errores absoluto y relativo calculados con las soluciones de los métodos SPPS y Bvp4c para los siguientes valores de los parámetros: $\alpha = 2, \beta = 1, a = 0, b = 1, c = 4, 8, 11, 16, 20, 27, 32, 36, 39, 42, 48, 52, 54, 58$.

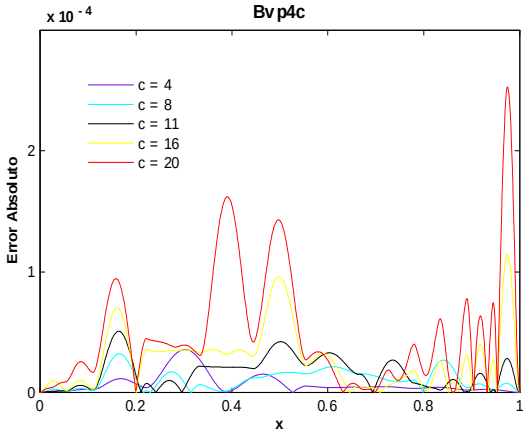


Fig. 231. Error absoluto de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 81.92 s.

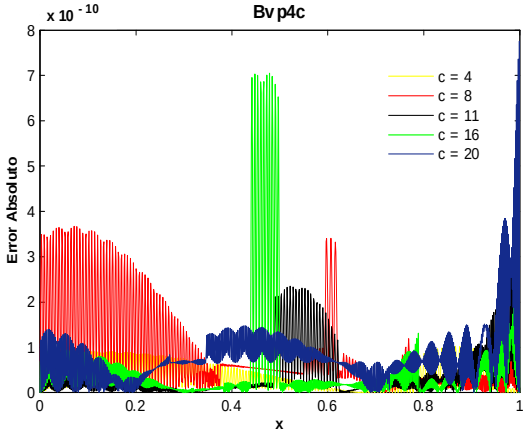


Fig. 232. Error absoluto de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 y tolerancias absoluta y relativa de 1e-8.
Cputime = 82. 06 s.

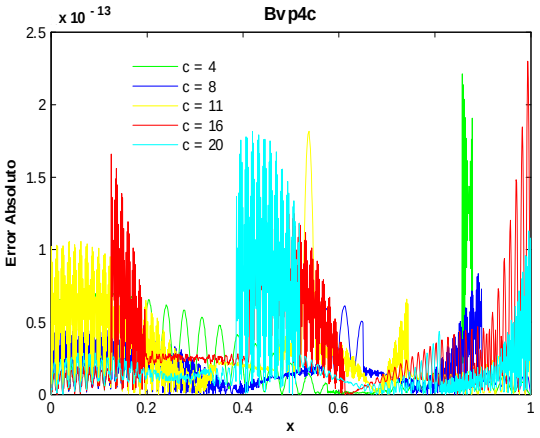


Fig. 233. Error absoluto de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 y tolerancias absoluta y relativa de 1e-11.
Cputime = 94.66 s.

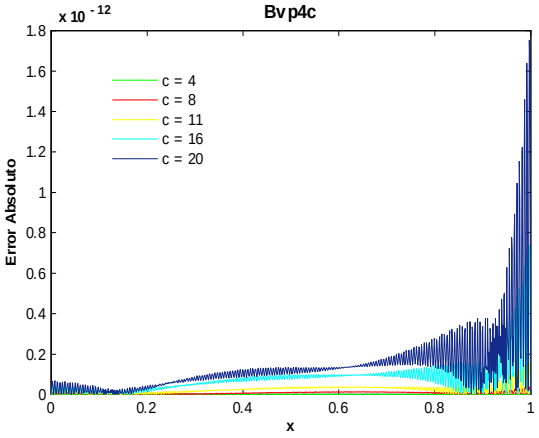


Fig. 234. Error absoluto de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 y tolerancias absoluta y relativa de 1e-14.
Cputime = 106.72 s.

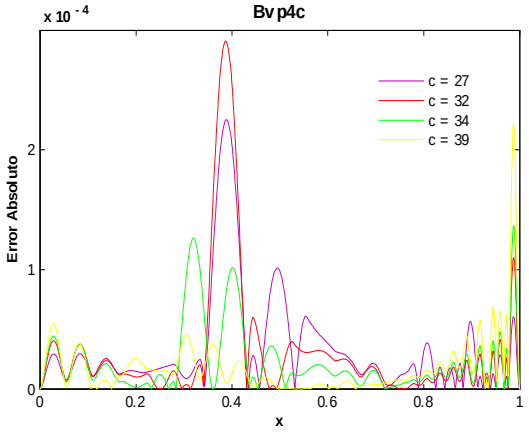


Fig. 235. Error absoluto de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=27,32,34,39 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 494.25 s.

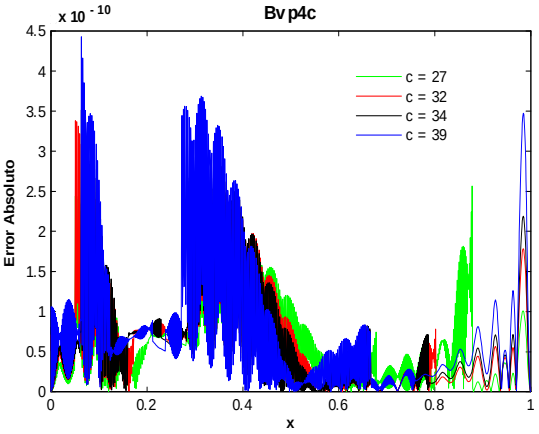


Fig. 236. Error absoluto de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=27,32,34,39 y tolerancias absoluta y relativa de 1e-8.
Cputime = 547.42 s.

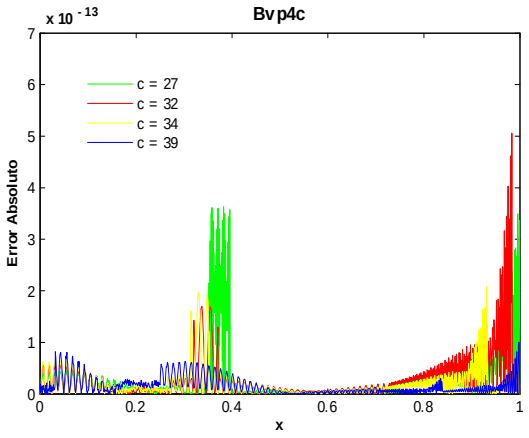


Fig. 237. Error absoluto de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=27,32,34,39 y tolerancias absoluta y relativa de 1e-11.
Cputime = 502.56 s.

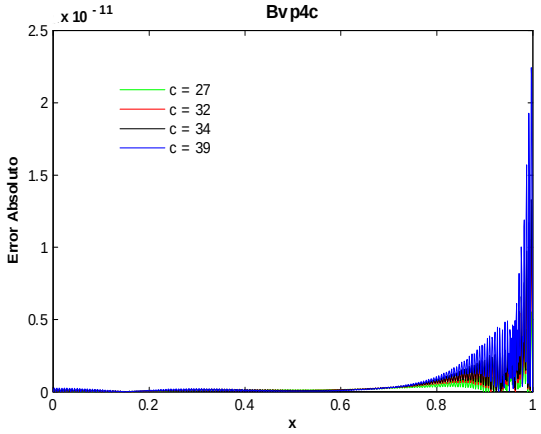


Fig. 238. Error absoluto de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=27,32,34,39 y tolerancias absoluta y relativa de 1e-14.
Cputime = 558.41 s.

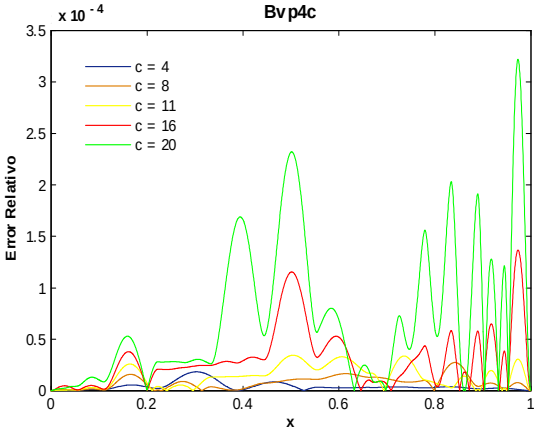


Fig. 239. Error relativo de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 81.65 s.

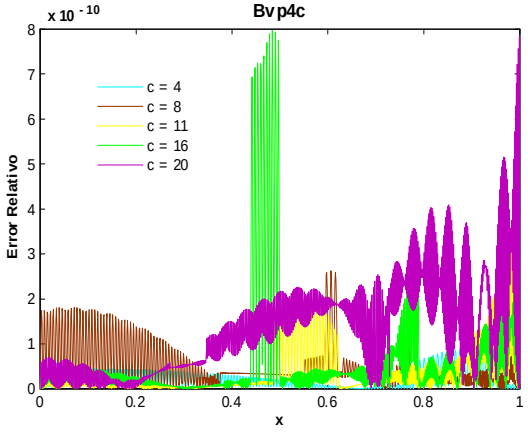


Fig. 240. Error relativo de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 y tolerancias absoluta y relativa de 1e-8.
Cputime = 82.47 s.

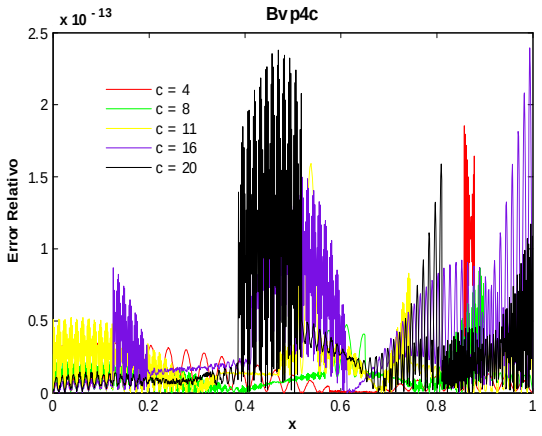


Fig. 241. Error relativo de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 y tolerancias absoluta y relativa de 1e-11.
Cputime = 86.44 s.

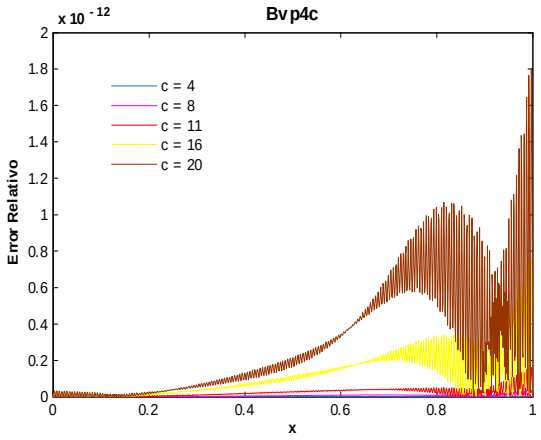


Fig. 242. Error relativo de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 y tolerancias absoluta y relativa de 1e-14.
Cputime = 96.32 s.

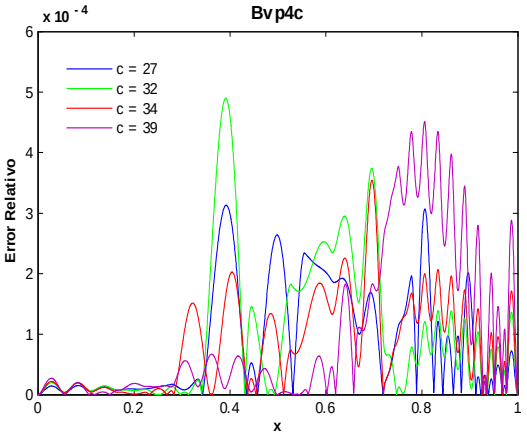


Fig. 243. Error relativo de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=27,32,34,39 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 496.03 s.

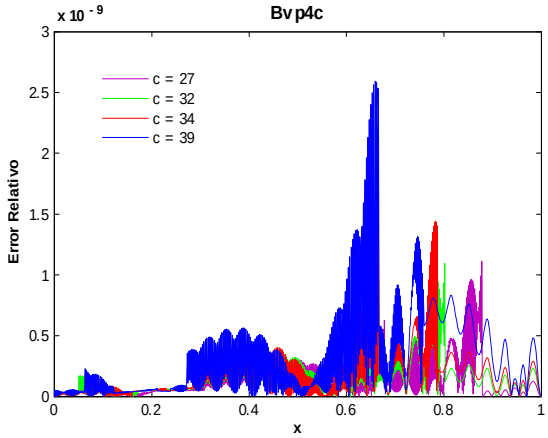


Fig. 244. Error relativo de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=27,32,34,39 y tolerancias absoluta y relativa de 1e-8.
Cputime = 495.45 s.

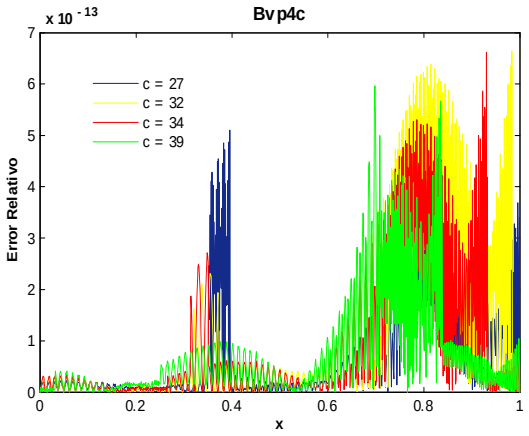


Fig. 245. Error relativo de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=27,32,34,39 y tolerancias absoluta y relativa de 1e-11.
Cputime = 502.48 s.

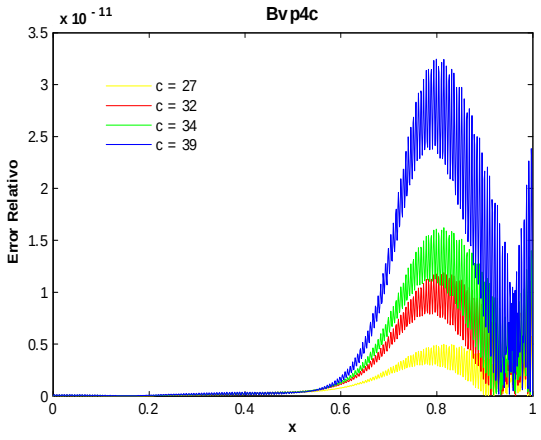


Fig. 246. Error relativo de la solución Bvp4c para el ejemplo 6 con los valores del parámetro c=27,32,34,39 y tolerancias absoluta de 1e-14 y relativa de 1e-14.
Cputime = 502.69 s.

Utilizando bvp4c, las figuras anteriores muestran que al disminuir los valores de las tolerancias el orden de los errores mejora, aunque para valores muy pequeños de tolerancias el orden empeora. Por ejemplo para el error absoluto y relativo con $c = 39$ y valores de tolerancias $AbsTol = 1e-6, RelTol = 1e-3$ se obtuvo un orden de -4 y con $AbsTol = RelTol = 1e-11$ se obtuvo un orden de -13. Al considerar valores de $1e-14$ o menores en las tolerancias el orden fue de -11. El tiempo máximo que tardó el programa fue de 558.41 s, al calcular el error absoluto para los valores de $c = 27, 32, 34, 39$ con tolerancias $AbsTol = RelTol = 1e-14$, note que no fue el mejor orden obtenido.

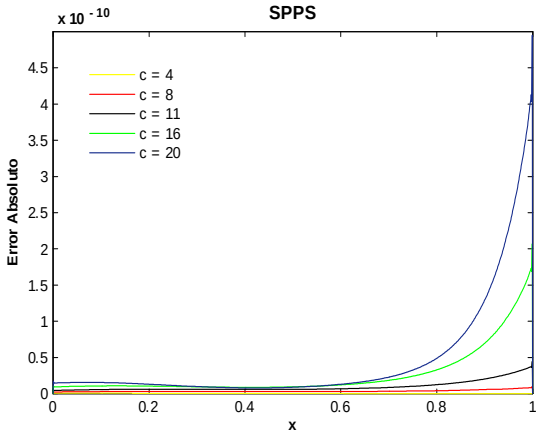


Fig. 247. Error absoluto de la solución SPPS para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 considerando 15 potencias y 1000 puntos.
Cputime = 95.94s.

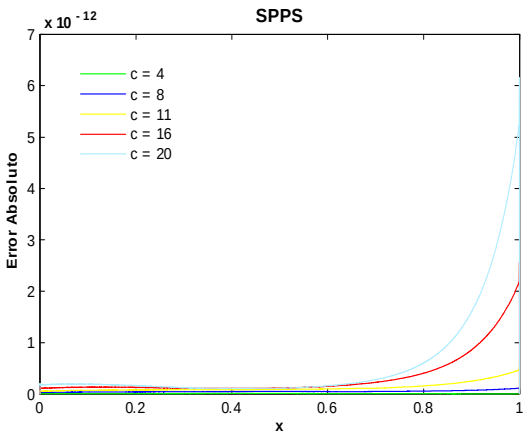


Fig. 248. Error absoluto de la solución SPPS para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 considerando 15 potencias y 3000 puntos.
Cputime = 134.36s.

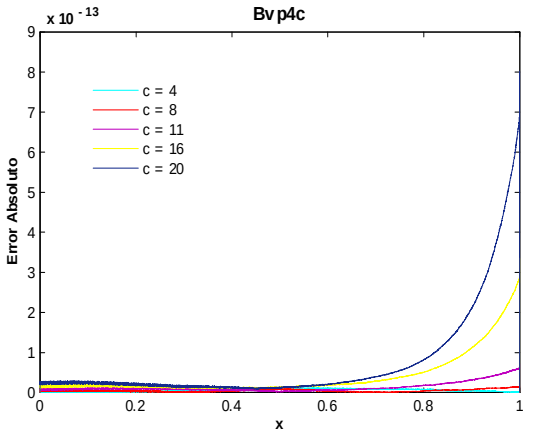


Fig. 249. Error absoluto de la solución SPPS para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 considerando 15 potencias y 5000 puntos.
Cputime = 174.79 s.

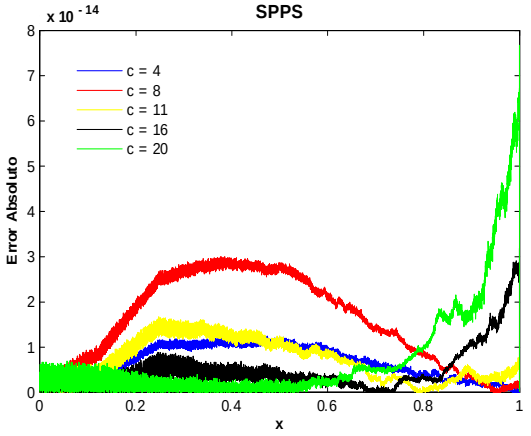


Fig. 250. Error absoluto de la solución SPPS para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 considerando 15 potencias y 9000 puntos.
Cputime = 254.83s.

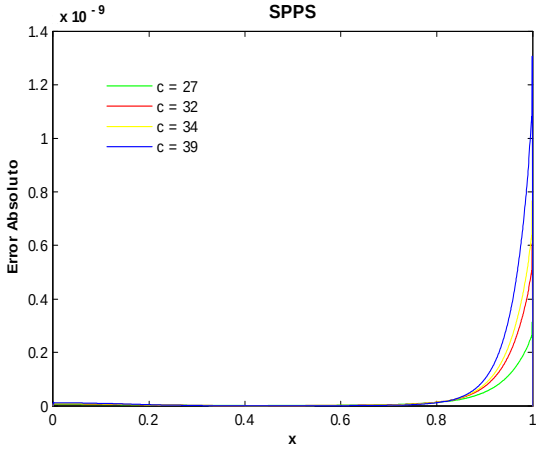


Fig. 251. Error absoluto de la solución SPPS para el ejemplo 6 con los valores del parámetro c=27,32,34,39 considerando 30 potencias y 1500 puntos.
Cputime = 510.55s.

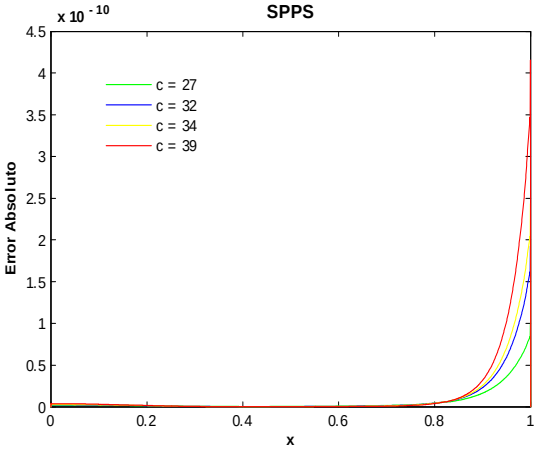


Fig. 252. Error absoluto de la solución SPPS para el ejemplo 6 con los valores del parámetro c=27,32,34,39 considerando 30 potencias y 2000 puntos.
Cputime = 525.68s.

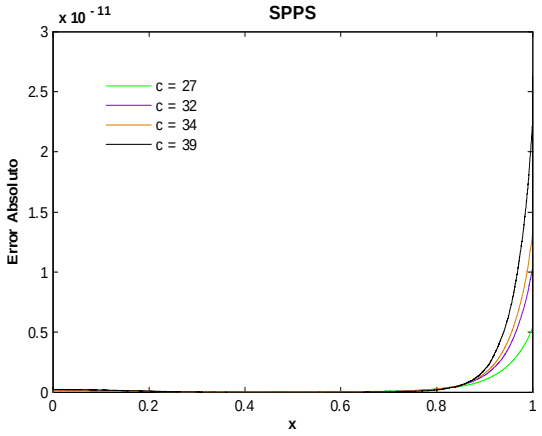


Fig. 253. Error absoluto de la solución SPPS para el ejemplo 6 con los valores del parámetro c=27,32,34,39 considerando 30 potencias y 4000 puntos.
Cputime = 601.62 s.

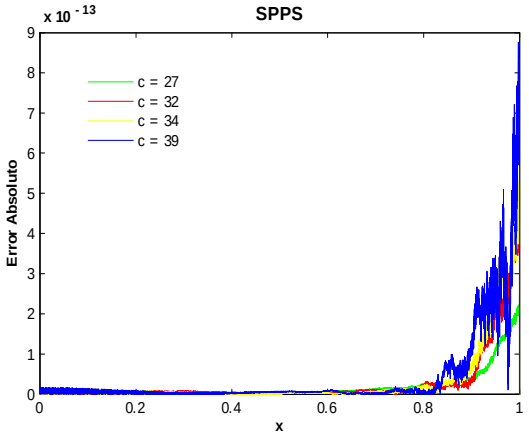


Fig. 254. Error absoluto de la solución SPPS para el ejemplo 6 con los valores del parámetro c=27,32,34,39 considerando 30 potencias y 9000 puntos.
Cputime = 718.15s.

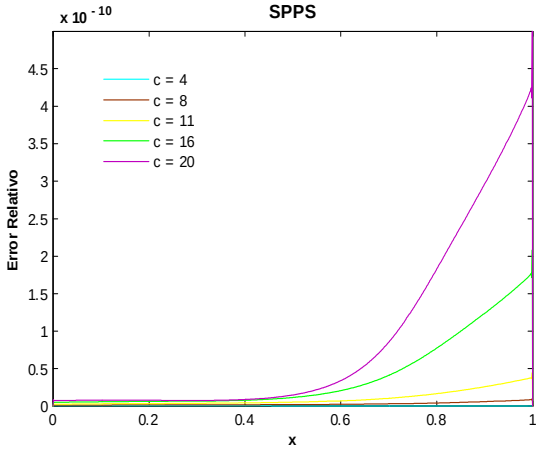


Fig. 255. Error relativo de la solución SPPS para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 considerando 30 potencias y 1000 puntos.
Cputime = 118.13 s.

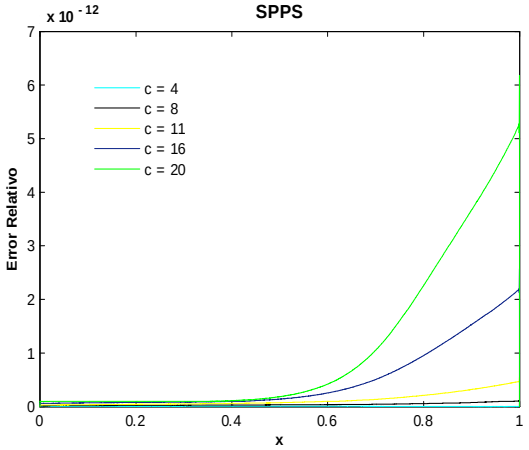


Fig. 256. Error relativo de la solución SPPS para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 considerando 30 potencias y 3000 puntos.
Cputime = 194.05 s.

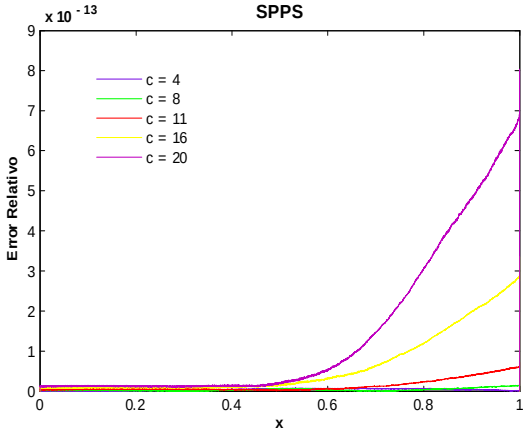


Fig. 257. Error relativo de la solución SPPS para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 considerando 30 potencias y 5000 puntos.
Cputime = 268.59 s.

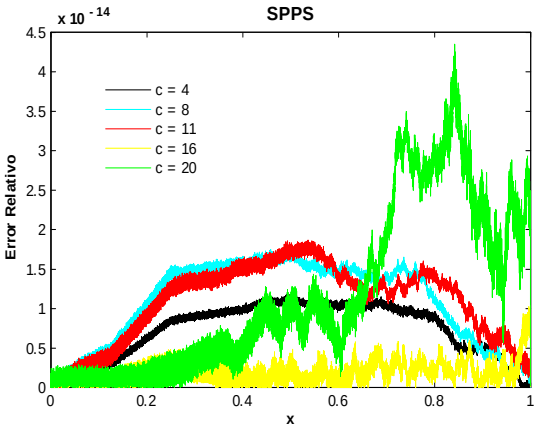


Fig. 258. Error relativo de la solución SPPS para el ejemplo 6 con los valores del parámetro c=4,8,11,16,20 considerando 30 potencias y 12000 puntos.
Cputime = 533.60 s.

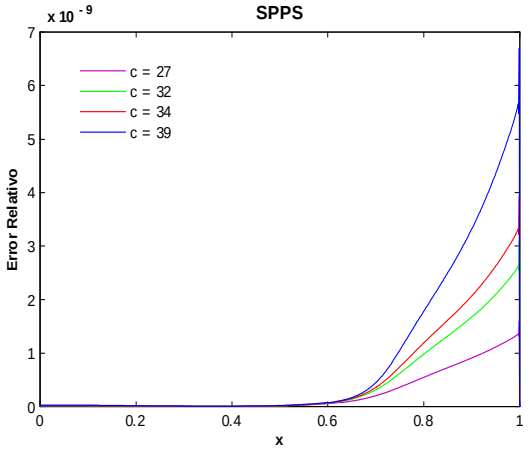


Fig. 259. Error relativo de la solución SPPS para el ejemplo 6 con los valores del parámetro c=27,32,34,39 considerando 30 potencias y 1000 puntos.
Cputime = 489.27 s.

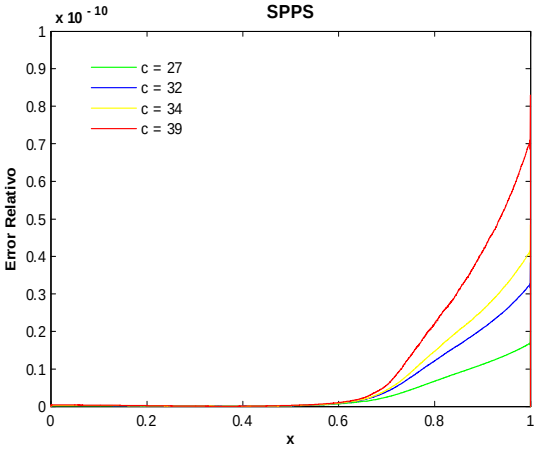


Fig. 260. Error relativo de la solución SPPS para el ejemplo 6 con los valores del parámetro c=27,32,34,39 considerando 30 potencias y 3000 puntos.
Cputime = 573.38 s.

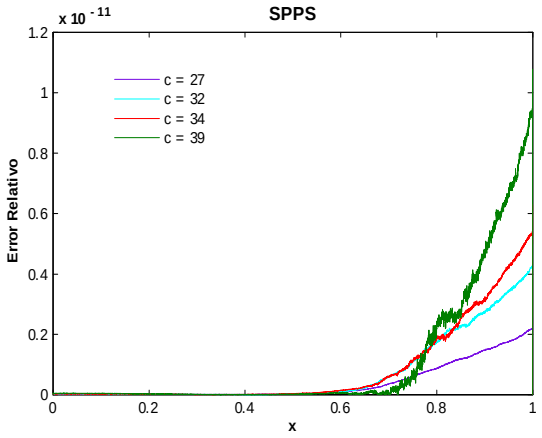


Fig. 261. Error relativo de la solución SPPS para el ejemplo 6 con los valores del parámetro c=27,32,34,39 considerando 30 potencias y 5000 puntos. Cputime = 715.63 s.

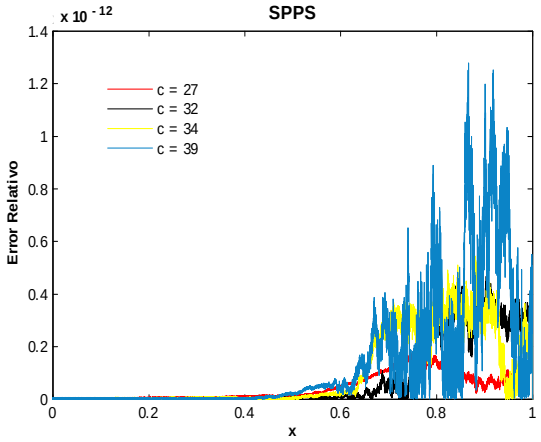


Fig. 262. Error relativo de la solución SPPS para el ejemplo 6 con los valores del parámetro c=27,32,34,39 considerando 30 potencias y 10000 puntos. Cputime = 748.23s.

Las figuras anteriores muestran que al incrementar el número de subintervalos de $[0,1]$ y las potencias, el orden de los errores absoluto y relativo mejora. La cantidad de potencias aquí consideradas fue tal que a partir de éste número al incrementarlo el orden obtenido fue el mismo. Por ejemplo, ver figs. 251-254, para $c = 39$ el orden del error absoluto considerando 1500 puntos y 30 potencias fue de -9, y tomando 9000 puntos y 30 potencias se obtuvo un orden de -13. Para el error relativo y $c = 39$ al tomar 1000 puntos y 30 potencias el orden fue de -9, considerando 10000 puntos y 30 potencias el orden obtenido fue de -12, ver figs. 259-262 . El tiempo máximo fue de 748.23, al calcular el error relativo para $c = 27, 32, 34, 39$ utilizando 10000 puntos y 30 potencias.

Ejemplo 7

Solución Exacta

Considérese el problema con valores en la frontera

$$\begin{aligned} y'' + \left(c^2\cos^4(x) - \frac{2}{\cos^2(x)} + 1\right) y &= 0 \\ y(a) &= \alpha \\ y(b) &= \beta \end{aligned} \tag{59}$$

donde $c \neq 0, \alpha, \beta, a, b \in \mathbb{R}$, $a < b$.
De acuerdo a [18] la solución general de la ecuación diferencial en (59) es $y_g = \frac{1}{\cos(x)}\left[c_1\sin(cv) + c_2\cos(cv)\right]$ donde c_1 y c_2 son constantes arbitrarias y $v(x) = \frac{1}{2}\left[\sin(x)\cos(x) + x\right]$. Entonce se debe cumplir

$$\begin{aligned} y_g(a) &= c_1\frac{\sin\left(cv(a)\right)}{\cos(a)} + c_2\frac{\cos(cv(a))}{\cos(a)} = \alpha \\ y_g(b) &= c_1\frac{\sin(cv(b))}{\cos(b)} + c_2\frac{\cos(cv(b))}{\cos(b)} = \beta. \end{aligned} \tag{60}$$

Por los Teoremas 4 y 5 el problema (59) tiene solución única si y sólo si

$$\Delta = \begin{vmatrix} \frac{\sin(cv(a))}{\cos(a)} & \frac{\cos(cv(a))}{\cos(a)} \\ \frac{\sin(cv(b))}{\cos(b)} & \frac{\cos(cv(b))}{\cos(b)} \end{vmatrix} = \frac{\sin\left(\frac{ac}{2} - \frac{bc}{2} + \frac{c\cos(a)\sin(a)}{2} - \frac{c\cos(b)\sin(b)}{2}\right)}{\cos(a)\cos(b)} \neq 0. \tag{61}$$

Note que

$\Delta = 0 \quad \Leftrightarrow \quad c(a - b + \cos a \sin(a) - \cos(b)\sin(b)) = 2k\pi, \;\; k \in \mathbb{Z}.$

Supongamos se cumple (61). Resolviendo (60) mediante la regla de Cramer:

$$c_1 = \frac{\begin{vmatrix} \alpha & \frac{\cos(cv(a))}{\cos(a)} \\ \beta & \frac{\cos(cv(b))}{\cos(b)} \end{vmatrix}}{\Delta} = \frac{\alpha\cos(cv(b))\cos(a) - \beta\cos(cv(b))\cos(b)}{\sin\left(\frac{ac}{2} - \frac{bc}{2} + \frac{c\cos(a)\sin(a)}{2} - \frac{c\cos(b)\sin(b)}{2}\right)} \tag{62}$$

$$c_2 = \frac{\begin{vmatrix} \frac{\sin(cv(a))}{\cos(a)} & \alpha \\ \frac{\sin(cv(b))}{\cos(b)} & \beta \end{vmatrix}}{\Delta} = \frac{\beta\sin(cv(a))\cos(b) - \alpha\sin(cv(b))\cos(a)}{\sin\left(\frac{ac}{2} - \frac{bc}{2} + \frac{c\cos(a)\sin(a)}{2} - \frac{c\cos(b)\sin(b)}{2}\right)}$$

De donde se sigue que la solución de (59) es:

$$y_p = \frac{1}{\cos(x)}\left[c_1\sin(cv) + c_2\cos(cv)\right] \tag{63}$$

donde $v(x) = \frac{1}{2}\left[\sin(x)\cos(x) + x\right]$, y c_1, c_2 están dadas en (62).

A continuación se presentan gráficas de soluciones y sus respectivos potenciales $q(x)$ para los valores de los parámetros $c = 5, 9, 11, 15, 20, 23, 29, 33$ y $a = 0, b = 1, \alpha = 2, \beta = 3$.

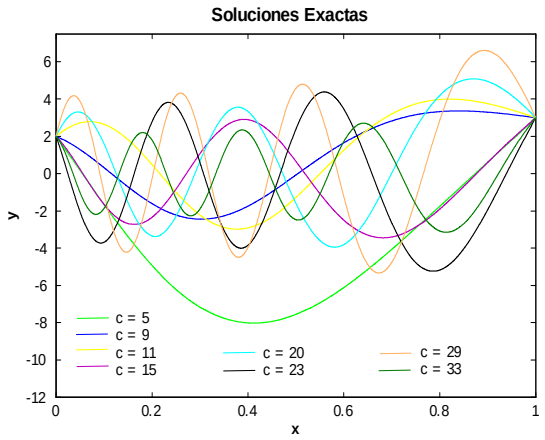


Fig. 263

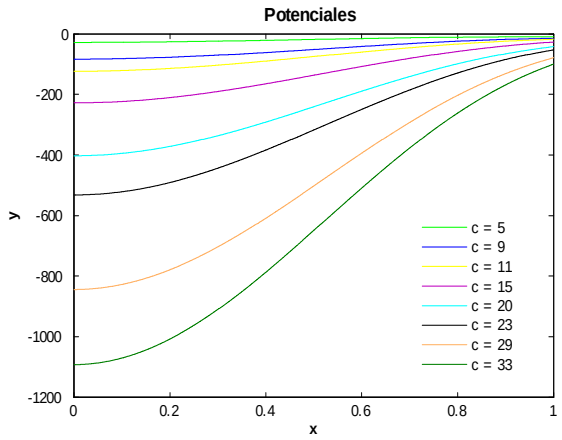


Fig. 264

Soluciones Numéricas

Las siguientes gráficas muestran el error absoluto entre la solución obtenida por bvp4c y la solución exacta (63), para los valores de los parámetros $c = 5, 9, 12, 15, 18, 21, 25, 31, 37, 41, 45, 49, 52, 57$, $a = 0, b = 1, \alpha = 2, \beta = 3$.

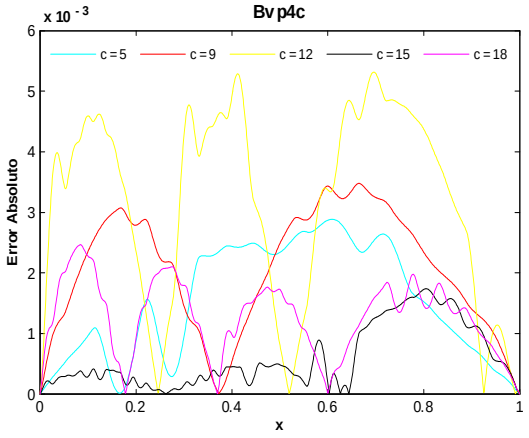


Fig. 265. Error absoluto de la solución Bvp4c para el ejemplo 7 con los valores del parámetro c=5,9,12,15,18 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 1.38 s.

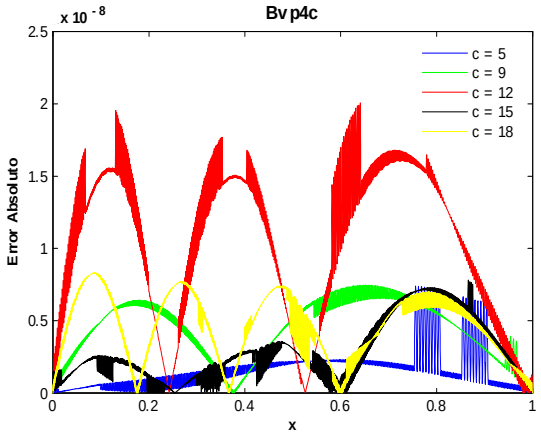


Fig. 266. Error absoluto de la solución Bvp4c para el ejemplo 7 con los valores del parámetro c=5,9,12,15,18 y tolerancias absoluta de 1e-7 y relativa de 1e-8.
Cputime = 3.47 s.

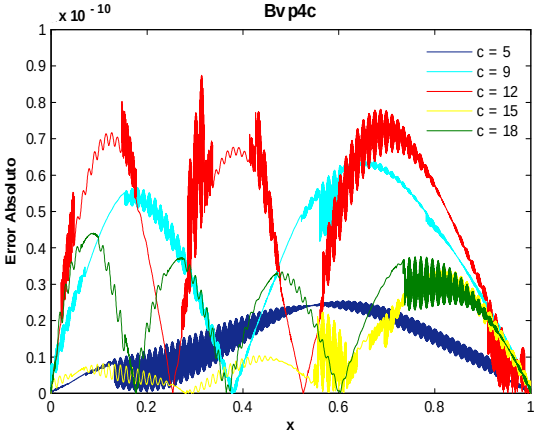


Fig. 267. Error absoluto de la solución Bvp4c para el ejemplo 7 con los valores del parámetro c=5,9,12,15,18 y tolerancias absoluta de 1e-9 y relativa de 1e-10.
Cputime = 9.47 s.

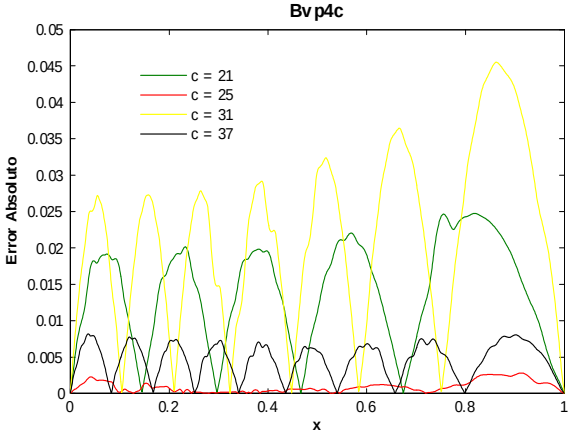


Fig. 268. Error absoluto de la solución Bvp4c para el ejemplo 7 con los valores del parámetro c=21,25,31,37 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 1.36 s.

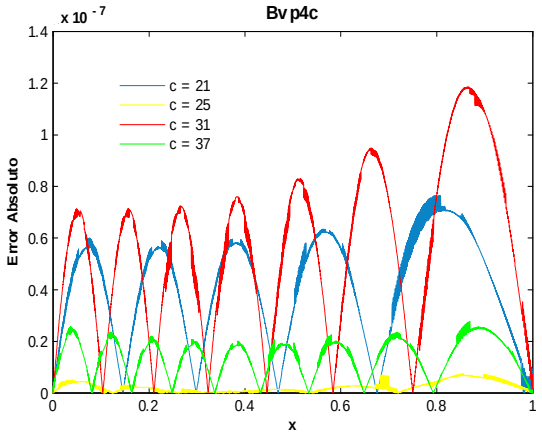


Fig. 269. Error absoluto de la solución Bvp4c para el ejemplo 7 con los valores del parámetro c=21,25,31,37 y tolerancias absoluta de 1e-7 relativa de 1e-8.
Cputime = 5.64 s.

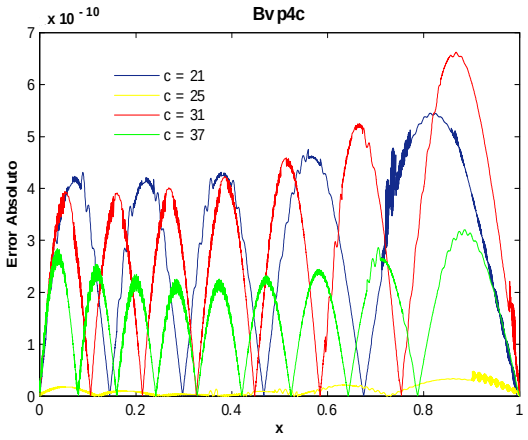


Fig. 270. Error absoluto de la solución Bvp4c para el ejemplo 7 con los valores del parámetro c=21,25,31,37 y tolerancias absoluta de 1e-9 y relativa de 1e-10.
Cputime = 13.32 s.

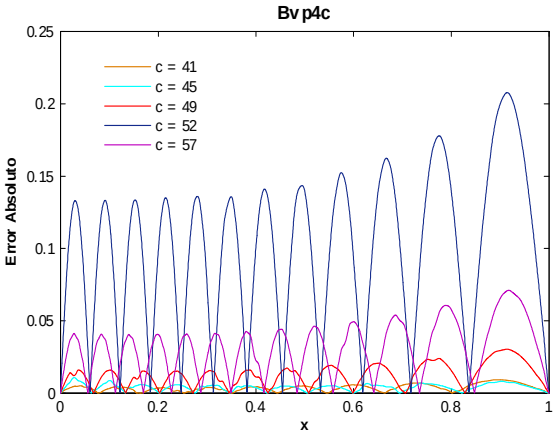


Fig. 271. Error absoluto de la solución Bvp4c para el ejemplo 7 con los valores del parámetro c=41,45,49,52,57 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 1.65 s.

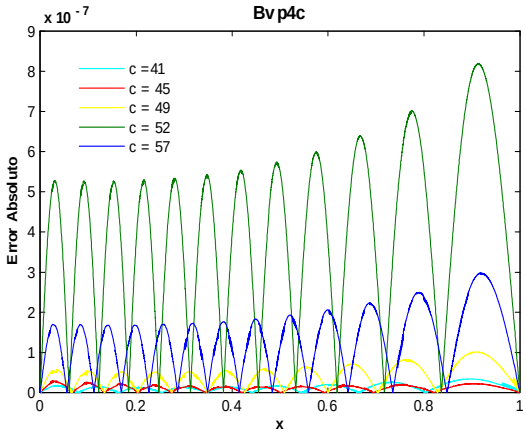


Fig. 272. Error absoluto de la solución Bvp4c para el ejemplo 7 con los valores del parámetro c=41,45,49,52,57 y tolerancias absoluta de 1e-7 y relativa de 1e-8.
Cputime = 10.78 s.

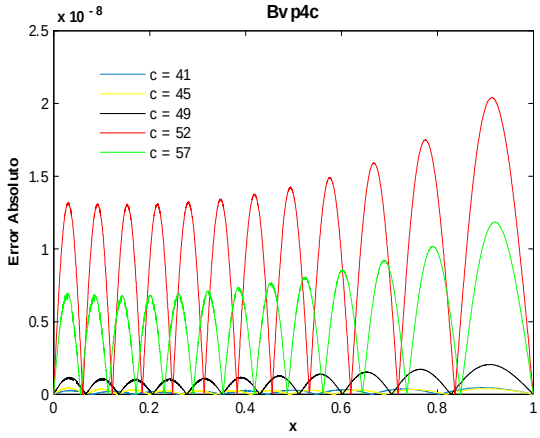


Fig. 273. Error absoluto de la solución Bvp4c para el ejemplo 7 con los valores del parámetro c=41,45,49,52,57 y tolerancias absoluta de 1e-9 y relativa de 1e-10.
Cputime = 13.24 s.

Como se puede ver en las figuras anteriores el orden del error absoluto calculado con la solución bvp4c mejora al incrementar los valores de las tolerancias absoluta y relativa. Para $c = 12, 31$ el mejor orden de error obtenido fue de -10 tomando $AbsTol = 1e-9$ y $RelTol = 1e-10$, ver figs. 267 y 270. Para $c = 52$ el mejor orden obtenido fue de -8 considerando las tolerancias $AbsTol = 1e-9$ y $RelTol = 1e-10$, ver fig.273. Al considerar valores menores en las tolerancias para el caso de la figura 267 el orden es el mismo. Para el caso de las figuras 270 y 273 el orden aumenta a -8 y -7, respectivamente, al tomar tolerancias más pequeñas.

Las siguientes gráficas muestran el error absoluto entre la solución SPPS y la solución exacta (63), para los valores de los parámetros $c = 5, 9, 12, 15, 18, 21, 25, 31, 37, 41, 45, 49, 52, 57$, $a = 0, b = 1, \alpha = 2, \beta = 3$.

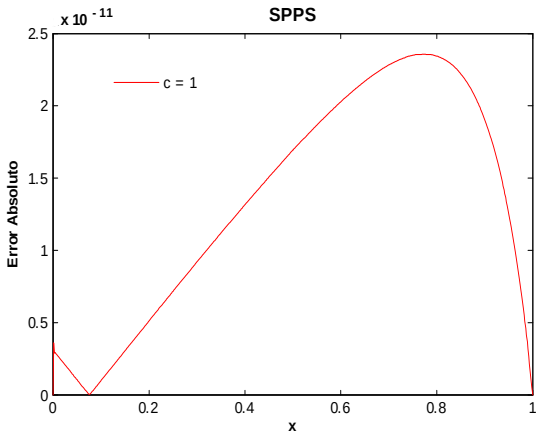


Fig.274. Error absoluto de la solución SPPS para el ejemplo 7 con los valores del parámetro c=1 considerando 10 potencias y 600 puntos.
Cputime = 2.87 s.

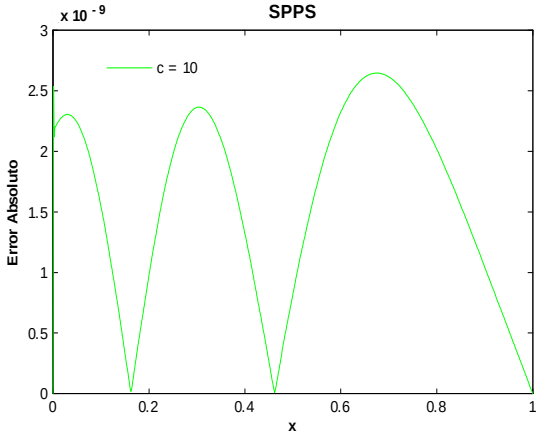


Fig.275. Error absoluto de la solución SPPS para el ejemplo 7 con los valores del parámetro c=20 considerando 20 potencias y 700 puntos.
Cputime = 5.45 s.

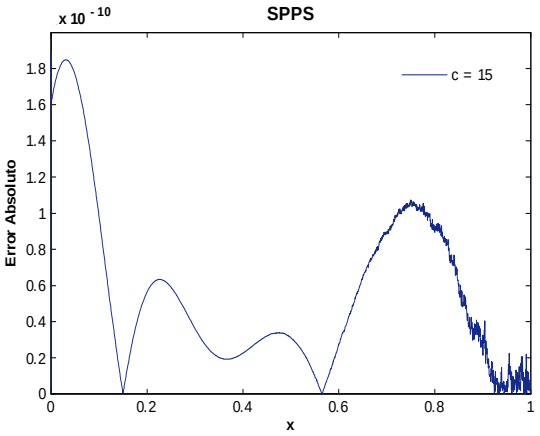


Fig.276. Error absoluto de la solución SPPS para el ejemplo 7 con los valores del parámetro c=15 considerando 25 potencias y 2000 puntos. Cputime = 16.57 s.

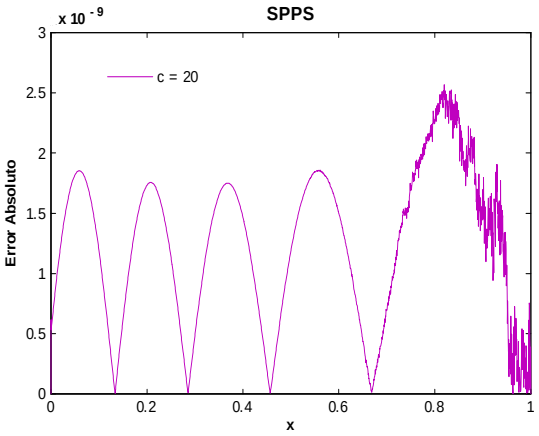


Fig.277. Error absoluto de la solución SPPS para el ejemplo 7 con los valores del parámetro c=20 considerando 60 potencias y 2001 puntos. Cputime = 38.49 s.

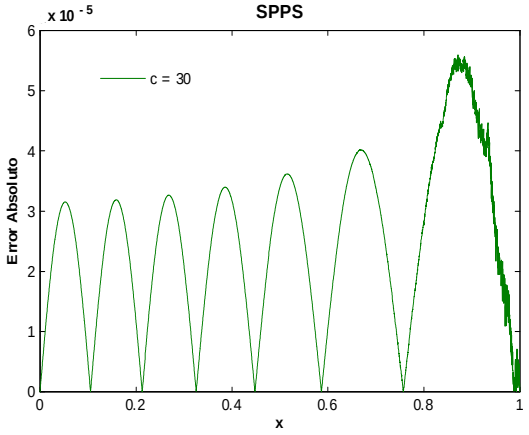


Fig.278. Error absoluto de la solución SPPS para el ejemplo 7 con los valores del parámetro c=30 considerando 90 potencias y 5000 puntos. Cputime = 138.6 s.

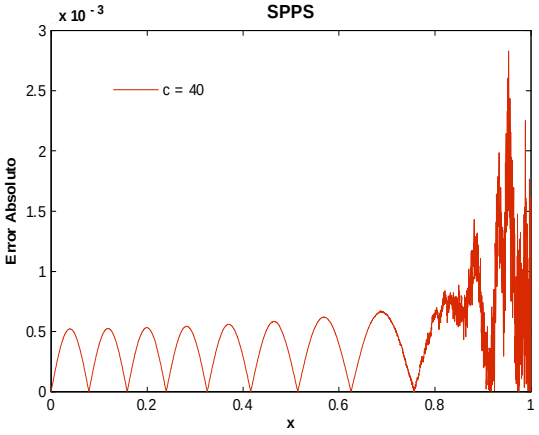


Fig.279 Error absoluto de la solución SPPS para el ejemplo 7 con los valores del parámetro c=40 considerando 90 potencias y 4000 puntos. Cputime = 112.52 s.

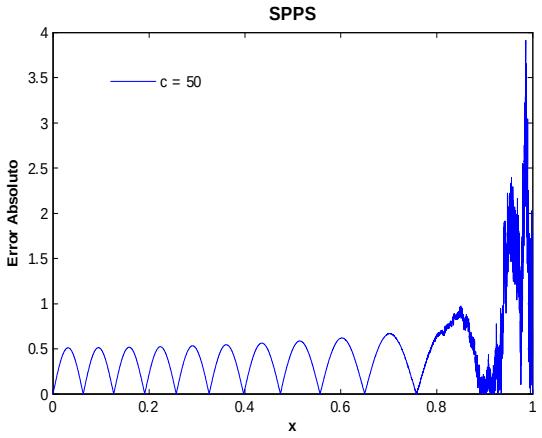


Fig.280. Error absoluto de la solución SPPS para el ejemplo 7 con los valores del parámetro c=50 considerando 200 potencias y 9000 puntos. Cputime = 552.8 s.

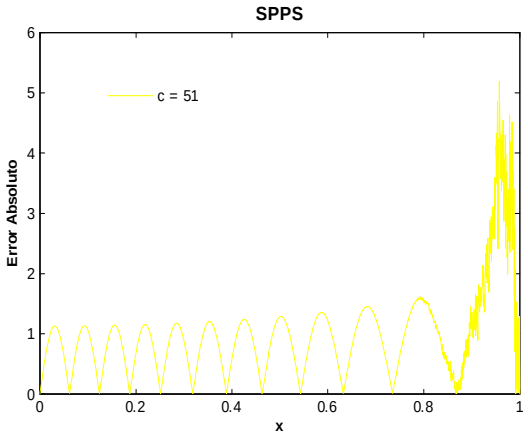


Fig.281. Error absoluto de la solución SPPS para el ejemplo 7 con los valores del parámetro c=51 considerando 75 potencias y 2000 puntos. Cputime = 47.7 s.

Las figuras anteriores muestran que conforme crece el parámetro c el orden del error absoluto incrementa, incluso no obteniéndose una solución, como es el caso de $c = 50, 51$, ver figs. 280,281.

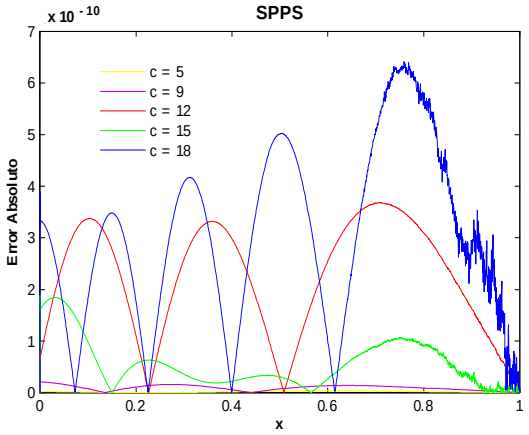


Fig.282. Error absoluto de la solución SPPS para el ejemplo 7 con los valores del parámetro c=5,9,12,15,18 considerando 30 potencias y 2000 puntos. Cputime = 96.39 s.

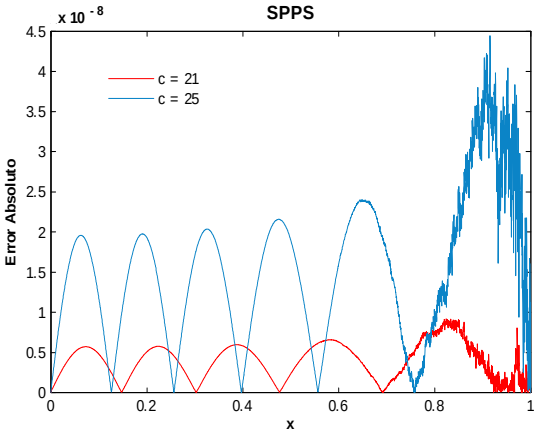


Fig.283. Error absoluto de la solución SPPS para el ejemplo 7 con los valores del parámetro c=21,25 considerando 40 potencias y 3000 puntos. Cputime = 74.55 s.

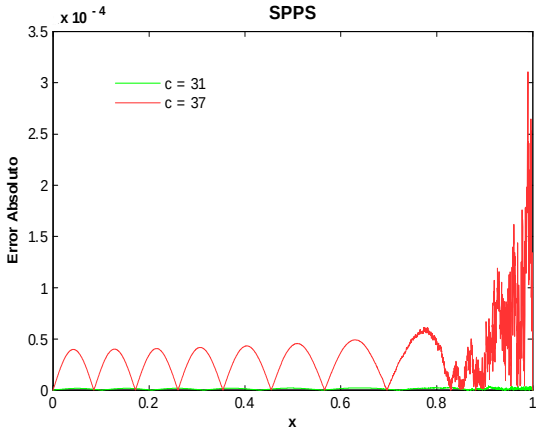


Fig.284. Error absoluto de la solución SPPS para el ejemplo 7 con los valores del parámetro c=31,37 considerando 40 potencias y 3000 puntos. Cputime = 74.52 s.

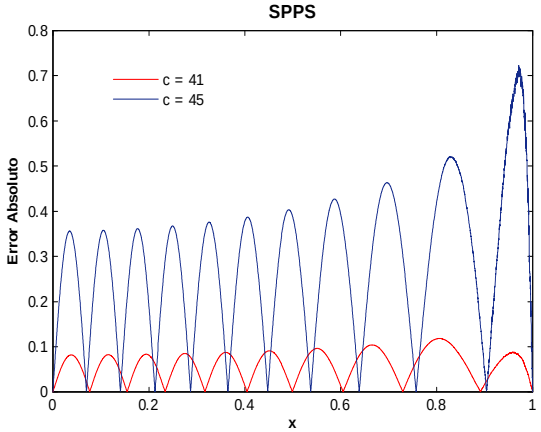


Fig.285. Error absoluto de la solución SPPS para el ejemplo 7 con los valores del parámetro c=41,45 considerando 40 potencias y 5000 puntos. Cputime = 123.07 s.

El Dr. Michael Porter ,Investigador y Profesor del Departamento de Matemáticas de Cinvestav, ha programado el método SPPS en Mathematica. Este software tiene una ventaja sobre Matlab, pues permite trabajar con una gran cantidad de dígitos decimales. El Dr. Porter amablemente nos proporcionó los resultados obtenidos usando la implementación del método SPPS en Mathematica para distintos valores de c utilizados en este ejemplo, con 100 dígitos de precisión y la cantidad de dígitos por defecto. A continuación se presentan estos resultados y los obtenidos con Matlab.

c	**N**	**Puntos**	**Error - Matlab**	**Error - Mathematica**
1	5	600	$2.5e-11$	$7.10543e-15$
10	10	700	$3e-9$	$5.62439e-13$
15	12	2000	$1.8e-10$	$2.12075e-11$
20	60	2001	$3e-9$	$3.77026e-10$
30	90	5000	$6e-5$	$8.37592e-6$
40	90	4000	$3e-3$	0.00135199
50	200	9000	4	0.923426
51	75	2000	6	1.38932

c	**N**	**Puntos**	**Error - Matlab**	**Error - Mathematica (100 dígitos)**
50	100	5000	3.5	$2.02505e-13$
50	200	9000	4	$6.21725e-15$
60	50	4000	3	$3.26339e-12$
60	100	5000	3.5	$8.55427e-13$

Como se puede ver en las anteriores tablas, cuando se utiliza Mathematica con la cantidad de dígitos decimales por defecto, se obtiene mejores órdenes del error que Matlab. Nótese que no se puede mejorar el orden del error añadiendo mas puntos o potencias, éste es un problema de precisión de máquina pues se calculan cientos de integrales iteradas de forma numérica lo cual hace que se pierda toda la precisión original. Cuando se consideran 100 dígitos decimales la diferencia es muy grande entre los resultados obtenidos por Matlab y Mathematica, este último obteniendo órdenes muy buenos incluso mejores a los obtenidos con bvp4c.

8.2.1. Condiciones de Neumann

Supongáse se tiene el siguiente problema con condiciones de Neumann

$$\begin{aligned} y'' + q(x)y &= 0 \\ y'(a) &= \alpha \\ y'(b) &= \beta \end{aligned} \tag{64}$$

donde $\alpha, \beta, a, b \in \mathbb{R}$. Para utilizar la función bvp4c se aplica la sustitución $y_1 = y \quad$ y $\quad y_2 = y'$ en (64) y se obtiene el sistema:

$$\begin{cases} y_1' = y_2 \\ y_2' = -q(x)y_1 \end{cases} \qquad y_2(a) = \alpha,\ y_2(b) = \beta \tag{65}$$

Note que cada solución de (65) es solución de (64) y cualquier solución de (65) es solución de (64). Este sistema de ecuaciones se resuelve con bvp4c y se obtiene una solución numérica de (64).

Ejemplo 8

Solución Exacta

Considérese el problema

$$\begin{aligned} y'' + c^2 y &= 0 \\ y'(a) &= \alpha \\ y'(b) &= \beta \end{aligned} \tag{66}$$

donde $c(\neq 0), a, b, \alpha, \beta \in \mathbb{R}$, con $a < b$.
Se sabe que la ecuación diferencial en (66) tiene como solución general a $y_g = c_1 \cos(cx) + c_2 \sin(cx)$, donde c_1 y c_2 son constantes arbitrarias. Entonces se debe cumplir

$$\begin{aligned} y_g'(a) &= -c_1 c \sin(ca) + c_2 c \cos(ca) = \alpha \\ y_g'(b) &= -c_1 c \sin(cb) + c_2 c \cos(cb) = \beta \end{aligned} \tag{67}$$

Pero por los Teoremas 4 y 5 el problema (66) tiene solución única si y sólo si

$$\Delta = \begin{vmatrix} -c\sin(ca) & c\cos(ca) \\ -c\sin(cb) & c\cos(cb) \end{vmatrix} = -c^2 \sin(ca)\cos(cb) + c^2 \sin(cb)\cos(ca) = c^2 \sin(cb - ca) \neq 0.$$

Note que $\Delta = 0$ *sii* $c = 0$ ó $c(b - a) = k\pi, k \in \mathbb{Z}$. Suponiendo que se cumple $\Delta \neq 0$ y aplicando la regla de Cramer a (67) se obtienen los coeficientes

$$c_1 = \frac{\begin{vmatrix} \alpha & c\cos(ca) \\ \beta & c\cos(cb) \end{vmatrix}}{\Delta} = \frac{\alpha\cos(cb) - \beta\cos(ca)}{c\sin(cb - ca)}, \qquad c_2 = \frac{\begin{vmatrix} -c\sin(ca) & \alpha \\ -c\sin(cb) & \beta \end{vmatrix}}{\Delta} = \frac{\alpha\sin(cb) - \beta\sin(ca)}{c\sin(cb - ca)}$$

Entonces la solución exacta de (66) es

$$y_p = \frac{\alpha\cos(cb) - \beta\cos(ca)}{c\sin(cb - ca)}\cos cx + \frac{\alpha\sin(cb) - \beta\sin(ca)}{c\sin(cb - ca)}\sin cx \tag{68}$$

siempre que $\sin(cb - ca) \neq 0$.
A continuación se presentan gráficas de algunas soluciones exactas para los valores $\alpha = 1$, $\beta = 2$, $a = 0$, $b = 1, c = 4, 6, 8, 11, 13, 15$.

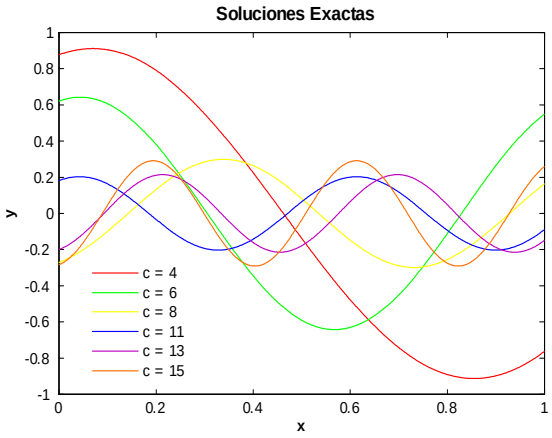


Fig.286

Soluciones Numéricas

Las siguientes figuras muestran el error absoluto entre la solución de cada método y la solución exacta, para los valores de $c = 2, 4, 6, 8, 10, 12, 14, 18, 20$, considerando $\alpha = 1$, $\beta = 2$, $a = 0$, $b = 1$ en (66).

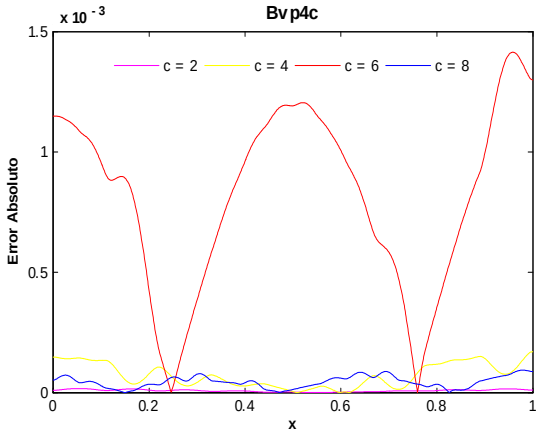


Fig.287. Error Absoluto de la solución Bvp4c para el ejemplo 8 con los valores del parámetro c=2,4,6,8 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 0.39 s.

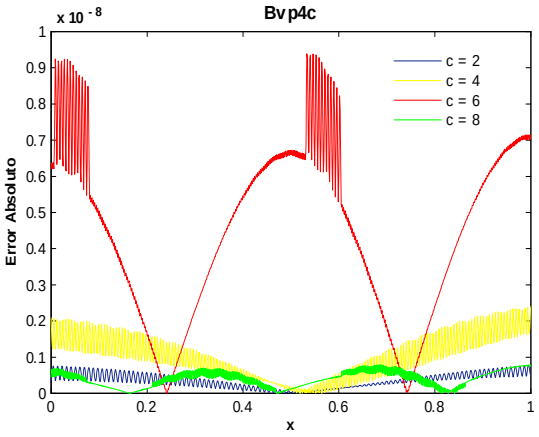


Fig.288. Error Absoluto de la solución Bvp4c para el ejemplo 8 con los valores del parámetro c=2,4,6,8 y tolerancias absoluta de 1e-7 y relativa de 1e-8. Cputime = 0.81 s.

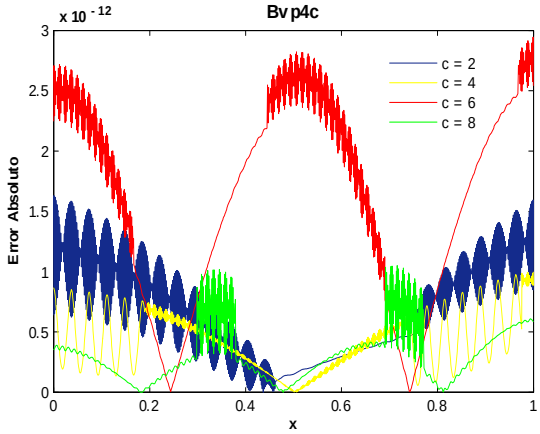


Fig.289. Error Absoluto de la solución Bvp4c para el ejemplo 8 con los valores del parámetro c=2,4,6,8 y tolerancias absoluta de 1e-10 y relativa de 1e-11. Cputime = 3.45 s.

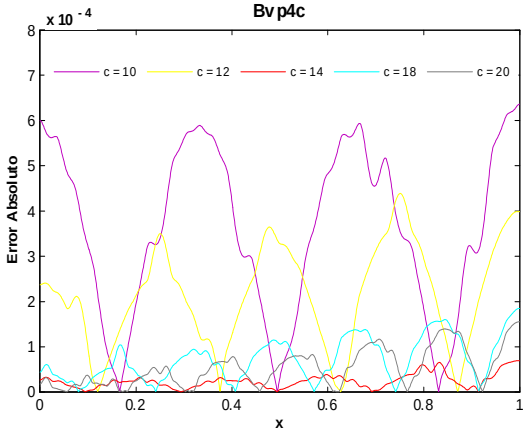


Fig.290. Error Absoluto de la solución Bvp4c para el ejemplo 8 con los valores del parámetro c=10,12,14,18,20 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 1.83 s.

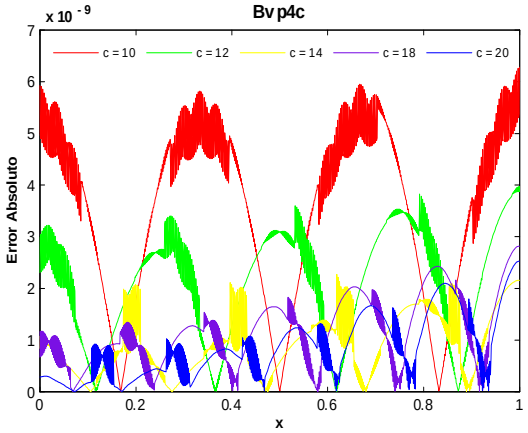


Fig.291. Error Absoluto de la solución Bvp4c para el ejemplo 8 con los valores del parámetro c=10,12,14,18,20 y tolerancias absoluta de 1e-7 y relativa de 1e-8. Cputime = 2.15 s.

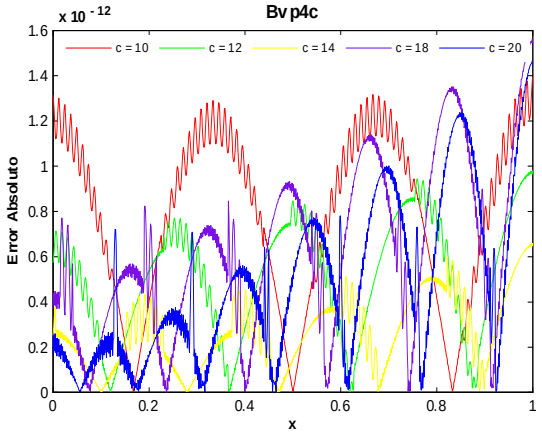


Fig.292. Error Absoluto de la solución Bvp4c para el ejemplo 8 con los valores del parámetro c=10,12,14,18,20 y tolerancias absoluta de 1e-10 y relativa de 1e-11. Cputime = 7.94 s.

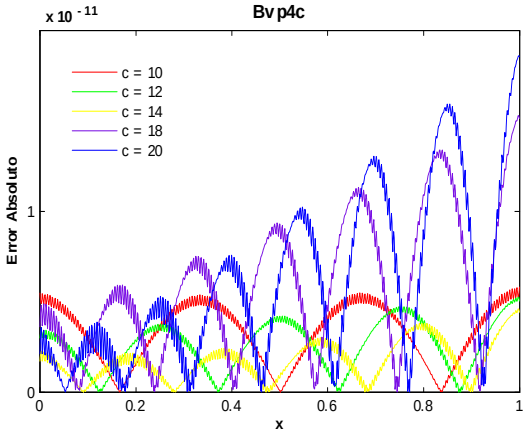


Fig.293. Error Absoluto de la solución Bvp4c para el ejemplo 8 con los valores del parámetro c=10,12,14,18,20 y tolerancias absoluta de 1e-13 y relativa de 1e-14. Cputime = 10.57 s.

Las anteriores figuras nos muestran que al utilizar bvp4c el orden del error absoluto para $c = 6$ fue -3, con valores de tolerancias AbsTol=1e-6,RelTol=1e-3, hasta -12 con AbsTol=1e-10, RelTol=1e-11, ver figs. 287-289. Para $c = 10$ fue de -4 hasta -12, utilizando los valores de tolerancias anteriores. Al tomar valores menores en las tolerancias, el orden del error empeora. Obsérvese que el tiempo máximo en este ejemplo, para obtener el error absoluto con $c = 10, 12, 14, 18, 20$, fue de 10.57 s, que en realidad no fue el mejor orden obtenido.

En la implementación de SPPS en Matlab, es necesario especificar el potencial, que en este caso es $q(x) = -c^2$, el número de subintervalos y el de términos N. Supongamos se quiere saber la cantidad de términos N suficiente para asegurar que $|v_1 - v_{1,N}|$ y $|v_2 - v_{2,N}|$ sean menor a $\varepsilon = 1e-5, 1e-9, 1e-12$ y $1e-15$. Para esto hacemos uso de las desigualdades (36) y (37), con $d = 1, q = -c^2$, es decir, buscamos las N tales que

$$\begin{aligned} \left|\cosh(c) - \sum_{k=0}^{N} \frac{c^{2k}}{(2k)!}\right| &< \varepsilon \\ \left|\sinh(c) - \sum_{k=0}^{N} \frac{c^{2k+1}}{(2k+1)!}\right| &< \varepsilon. \end{aligned} \tag{69}$$

Por ejemplo para $c = 6$ y $\varepsilon = 1e-5, 1e-9, 1e-12$ y $1e-15$ utilizando (69) se obtienen los valores de $N = 11, 14, 16, 18$ respectivamente. En efecto

$$\begin{aligned}
\left|\cosh(6) - \sum_{k=0}^{11} \frac{6^{2k}}{(2k)!}\right| &= 8.081007337068513e-006 < 1e-5 \\
\left|\sinh(6) - \sum_{k=0}^{11} \frac{6^{2k+1}}{(2k+1)!}\right| &= 1.931214853811980e-006 < 1e-5 \\
\left|\cosh(6) - \sum_{k=0}^{14} \frac{6^{2k}}{(2k)!}\right| &= 8.646736660011811e-10 < 1e-9 \\
\left|\sinh(6) - \sum_{k=0}^{14} \frac{6^{2k+1}}{(2k+1)!}\right| &= 1.669206994847627e-10 < 1e-9 \\
\left|\cosh(6) - \sum_{k=0}^{16} \frac{6^{2k}}{(2k)!}\right| &= 9.947598300641403e-13 < 1e-12 \\
\left|\sinh(6) - \sum_{k=0}^{16} \frac{6^{2k+1}}{(2k+1)!}\right| &= 1.136868377216160e-13 < 1e-12 \\
\left|\cosh(6) - \sum_{k=0}^{18} \frac{6^{2k}}{(2k)!}\right| &= 7.266814239107258e-16 < 1e-15 \\
\left|\sinh(6) - \sum_{k=0}^{18} \frac{6^{2k+1}}{(2k+1)!}\right| &= 1.116690121177443e-16 < 1e-15
\end{aligned}$$

A continuación se presentan las gráficas del error absoluto obtenidas con la solución SPPS para las N calculadas anteriormente.

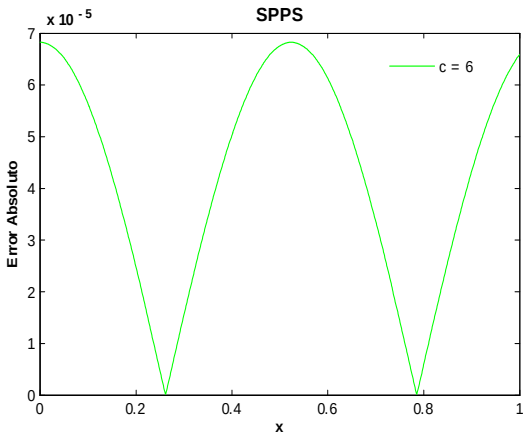


Fig.294. Error absoluto de la solución SPPS para el ejemplo 8 con los valores del parámetro c=6 considerando 11 potencias y 1000 puntos. Cputime = 3.18 s.

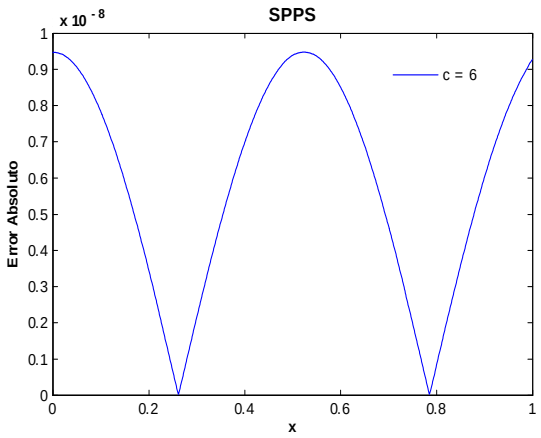


Fig.295. Error absoluto de la solución SPPS para el ejemplo 8 con el valor del parámetro c=6 considerando 14 potencias y 1000 puntos. Cputime = 3.89 s.

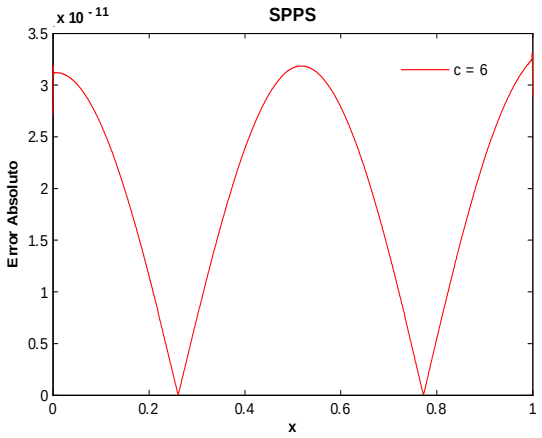


Fig.296. Error absoluto de la solución SPPS para el ejemplo 8 con el valor del parámetro c=6 considerando 16 potencias y 1500 puntos. Cputime = 6.44 s.

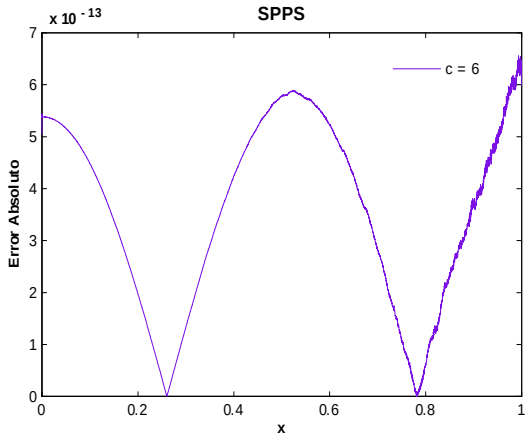


Fig.297. Error absoluto de la solución SPPS para el ejemplo 8 con el valor del parámetro c=6 considerando 18 potencias y 5000 puntos. Cputime = 23.04 s.

Como se puede observar en las anteriores figuras, al requerir más exactitud es necesario tomar más términos en las series (19), lo cual hace que se tenga que integrar más veces, lo que provoca que el error numérico generado al calcular las aproximaciones de v_1 y v_2 crezca, haciendo que no se cumpla $|v_1 - v_{1,N}|, |v_2 - v_{2,N}| < \varepsilon$. Note que este problema no es propio del método SPPS si no del hecho de la complejidad de las integrales en $\widetilde{X}^{(n)}(x)$ y $X^{(n)}(x)$, pues se tienen que calcular numéricamente generando cierto error de aproximación.

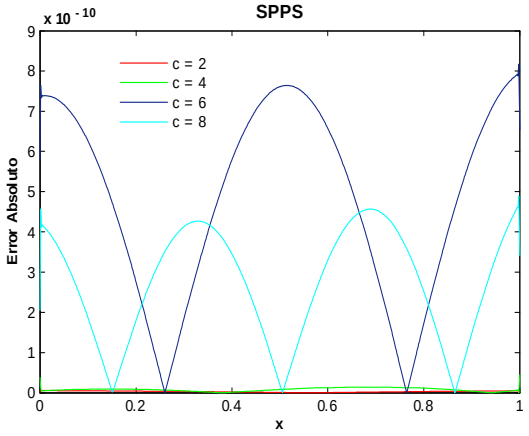


Fig.298. Error relativo de la solución SPPS para el ejemplo 8 con los valores del parámetro c=2,4,6,8 considerando 20 potencias y 600 puntos. Cputime = 13.36 s.

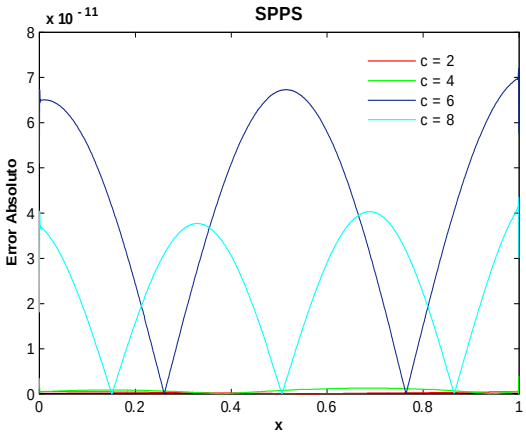


Fig.299. Error relativo de la solución SPPS para el ejemplo 8 con los valores del parámetro c=2,4,6,8 considerando 20 potencias y 1100 puntos. Cputime = 23.76 s.

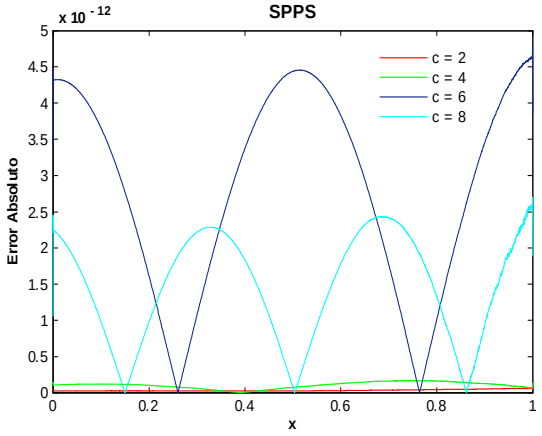


Fig.300. Error relativo de la solución SPPS para el ejemplo 8 con los valores del parámetro c=2,4,6,8 considerando 20 potencias y 2200 puntos.
Cputime = 45.44 s.

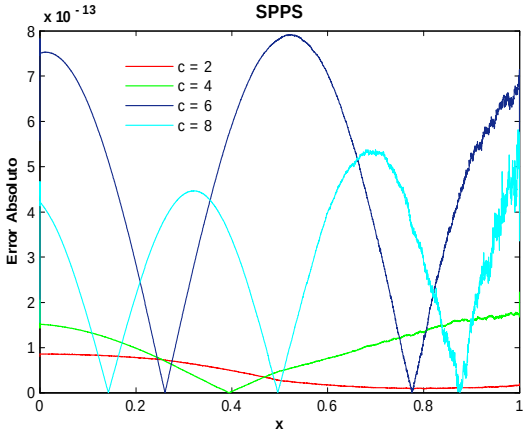


Fig.301. Error relativo de la solución SPPS para el ejemplo 8 con los valores del parámetro c=2,4,6,8 considerando 20 potencias y 3200 puntos.
Cputime = 67.09 s.

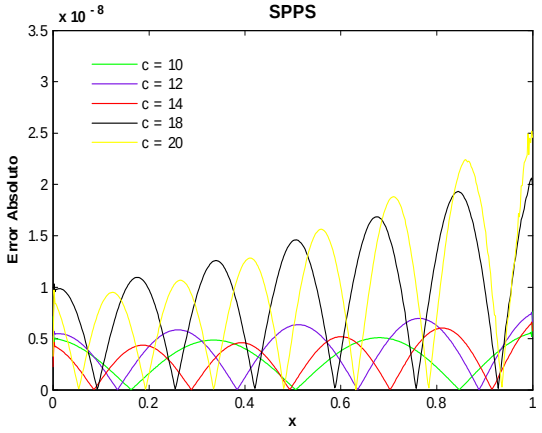


Fig.302. Error relativo de la solución SPPS para el ejemplo 8 con los valores del parámetro c=10,12,14,18,20 considerando 35 potencias y 500 puntos.
Cputime = 24.09 s.

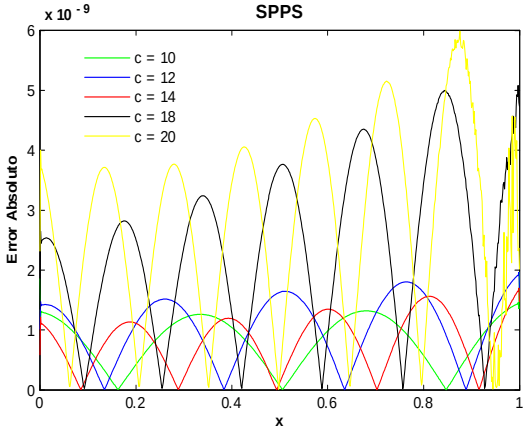


Fig.303. Error relativo de la solución SPPS para el ejemplo 8 con los valores del parámetro c=10,12,14,18,20 considerando 35 potencias y 700 puntos.
Cputime = 32.75 s.

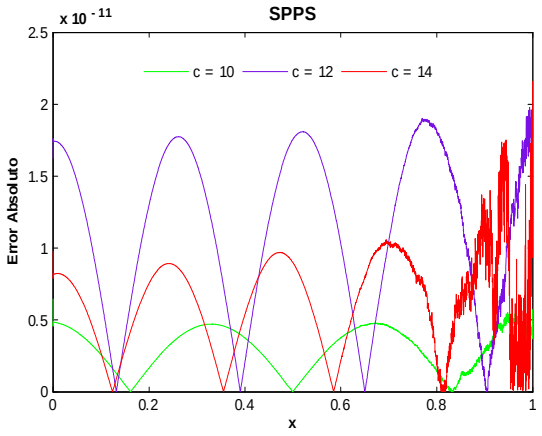


Fig.304. Error relativo de la solución SPPS para el ejemplo 8 con los valores del parámetro c=10,12,14 considerando 35 potencias y 3000 puntos. Cputime = 81.04 s.

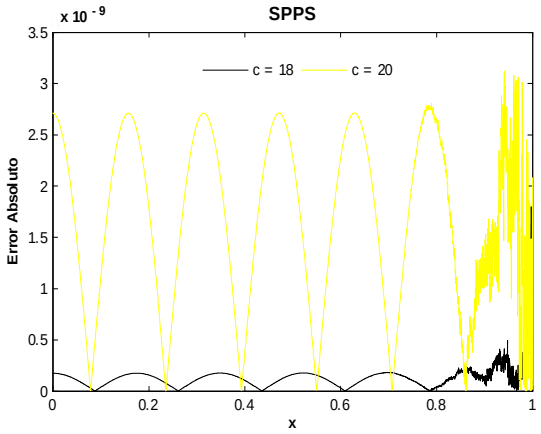


Fig.305. Error relativo de la solución SPPS para el ejemplo 8 con los valores del parámetro c=18,20 considerando 35 potencias y 3000 puntos. Cputime = 53.91 s.

Utilizando SPPS, las figuras 298-301 muestran que para $c = 6, 8$ y 20 potencias el orden del error absoluto fue de -10, con 600 puntos, hasta -13 con 3200 puntos. Para $c = 18, 20$ la cantidad de potencias se incrementó a 35 y se obtuvo un orden de -8 con 500 puntos y de -9 con 3000 puntos, ver figs. 303 y 305. El mayor tiempo utilizado en este ejemplo, 81.04 s, fue al calcular el error absoluto para $c = 10, 12, 14$ con 35 potencias y 3000 puntos.

Ejemplo 9

Solución Exacta

Considérese el problema con valores en la frontera

$$\begin{aligned} -y'' + (c^2x^2 + c)y &= 0 \\ y'(a) &= \alpha \\ y'(b) &= \beta \end{aligned} \tag{70}$$

donde $c(\neq 0), \alpha, \beta, a, b \in \mathbb{R}$, $a < b$.

La ecuación diferencial en (70) tiene como solución general a $y_g(x) = e^{\frac{cx^2}{2}} \left(c_1 + c_2 \int_0^x e^{-ct^2} dt \right)$ donde c_1 y c_2 son constantes arbitrarias. Entonces se debe cumplir

$$\begin{aligned} y_g'(a) &= cae^{\frac{ca^2}{2}} c_1 + \left(cae^{\frac{ca^2}{2}} \int_0^a e^{-ct^2} dt + e^{-\frac{ca^2}{2}} \right) c_2 = \alpha \\ y_g'(b) &= cbe^{\frac{cb^2}{2}} c_1 + \left(cbe^{\frac{cb^2}{2}} \int_0^b e^{-ct^2} dt + e^{-\frac{cb^2}{2}} \right) c_2 = \beta. \end{aligned} \tag{71}$$

Por los Teoremas 4 y 5 el problema (70) tiene solución única si y sólo si

$$\begin{aligned} \Delta &= \begin{vmatrix} cae^{\frac{ca^2}{2}} & cae^{\frac{ca^2}{2}} \int_0^a e^{-ct^2} dt + e^{-\frac{ca^2}{2}} \\ cbe^{\frac{cb^2}{2}} & cbe^{\frac{cb^2}{2}} \int_0^b e^{-ct^2} dt + e^{-\frac{cb^2}{2}} \end{vmatrix} \\ &= c^2 abe^{\frac{c(a^2+b^2)}{2}} \left[\int_0^b e^{-ct^2} dt - \int_0^a e^{-ct^2} dt \right] + c \left[ae^{\frac{c(a^2-b^2)}{2}} - be^{\frac{c(b^2-a^2)}{2}} \right] \neq 0 \end{aligned} \tag{72}$$

Suponiendo que lo anterior se cumple, resolviendo el sistema (57) mediante la regla de Cramer:

$$\begin{aligned}
c_1 &= \frac{\begin{vmatrix} \alpha & cae^{\frac{ca^2}{2}}\int_0^a e^{-ct^2}dt + e^{-\frac{ca^2}{2}} \\ \beta & cbe^{\frac{cb^2}{2}}\int_0^b e^{-ct^2}dt + e^{-\frac{cb^2}{2}} \end{vmatrix}}{\Delta} \\
&= \frac{\alpha\left[cbe^{\frac{cb^2}{2}}\int_0^b e^{-ct^2}dt + e^{-\frac{cb^2}{2}}\right] - \beta\left[cae^{\frac{ca^2}{2}}\int_0^a e^{-ct^2}dt + e^{-\frac{ca^2}{2}}\right]}{c^2abe^{\frac{c(a^2+b^2)}{2}}\left[\int_0^b e^{-ct^2}dt - \int_0^a e^{-ct^2}dt\right] + c\left[ae^{\frac{c(a^2-b^2)}{2}} - be^{\frac{c(b^2-a^2)}{2}}\right]} \\
c_2 &= \frac{\begin{vmatrix} cae^{\frac{ca^2}{2}} & \alpha \\ cbe^{\frac{cb^2}{2}} & \beta \end{vmatrix}}{\Delta} \\
&= \frac{a\beta e^{\frac{ca^2}{2}} - b\alpha e^{\frac{cb^2}{2}}}{cabe^{\frac{c(a^2+b^2)}{2}}\left[\int_0^b e^{-ct^2}dt - \int_0^a e^{-ct^2}dt\right] + ae^{\frac{c(a^2-b^2)}{2}} - be^{\frac{c(b^2-a^2)}{2}}}
\end{aligned} \tag{73}$$

Así, se concluye que la solución de (70) es:

$$y_p = e^{\frac{cx^2}{2}}\left(c_1 + c_2\int_0^x e^{-ct^2}dt\right) \tag{74}$$

donde c_1 y c_2 están definidas en (73). Las siguientes gráficas muestran algunas soluciones exactas para los valores de los parámetros $\alpha = 1$, $\beta = -1$, $a = 0$, $b = 1$, $c = 1, 3, 6, 8, 11, 14$.

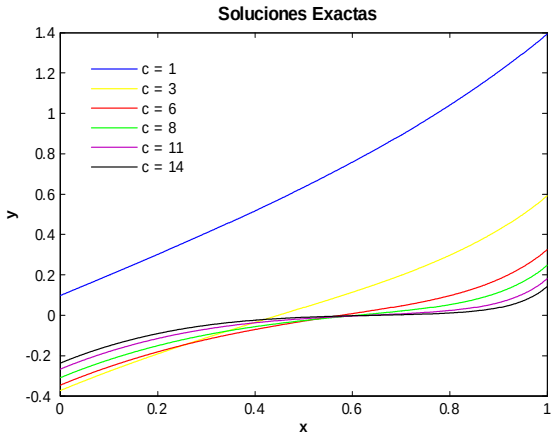


Fig. 306

Soluciones Numéricas

Los valores de los parámetros en (70) usados en ambos métodos son los siguientes: $\alpha = 1$, $\beta = -1$, $a = 0$, $b = 1$, $c = 2, 4, 6, 8, 10, 12, 14, 16, 18, 20$.

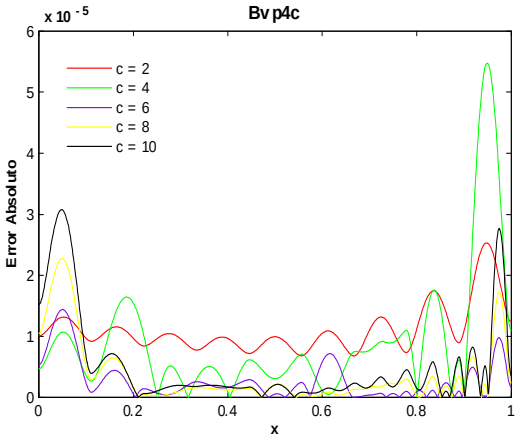


Fig.307. Error Absoluto de la solución Bvp4c para el ejemplo 9 con los valores del parámetro c=2,4,6,8,10 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 3.07 s.

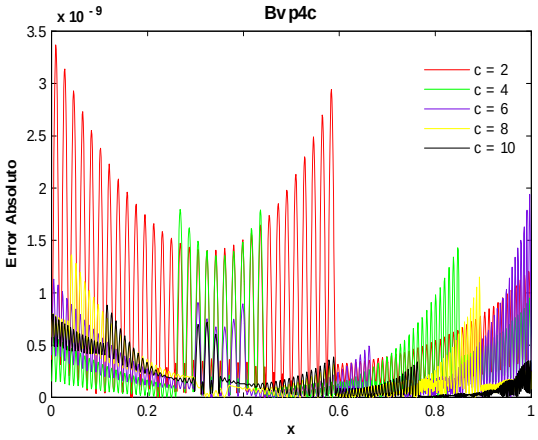


Fig.308. Error Absoluto de la solución Bvp4c para el ejemplo 9 con los valores del parámetro c=2,4,6,8,10 y tolerancias absoluta y relativa de 1e-7.
Cputime = 3.4 s.

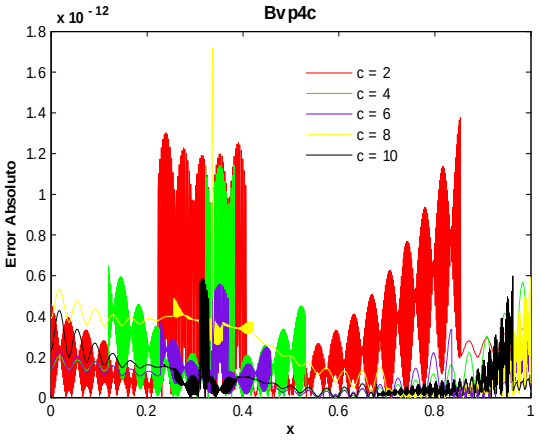


Fig.309. Error Absoluto de la solución Bvp4c para el ejemplo 9 con los valores del parámetro c=2,4,6,8,10 y tolerancias absoluta de y relativa de 1e-10.
Cputime = 5.41 s.

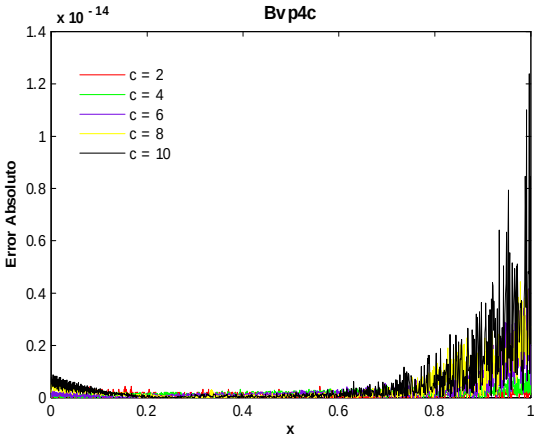


Fig.310. Error Absoluto de la solución Bvp4c para el ejemplo 9 con los valores del parámetro c=2,4,6,8,10 y tolerancias absoluta de y relativa de 1e-13.
Cputime = 11.27 s.

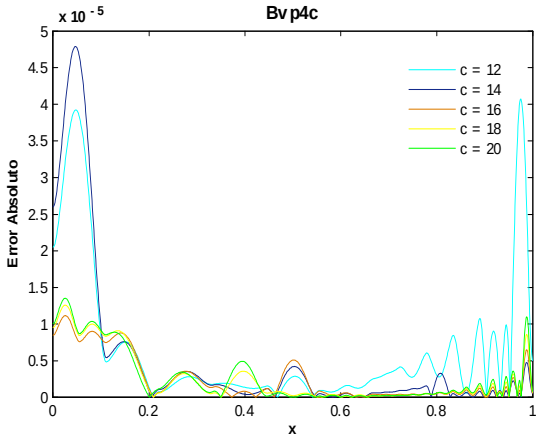


Fig.311. Error Absoluto de la solución Bvp4c para el ejemplo 9 con los valores del parámetro c=12,14,16,18,20 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 3.17 s.

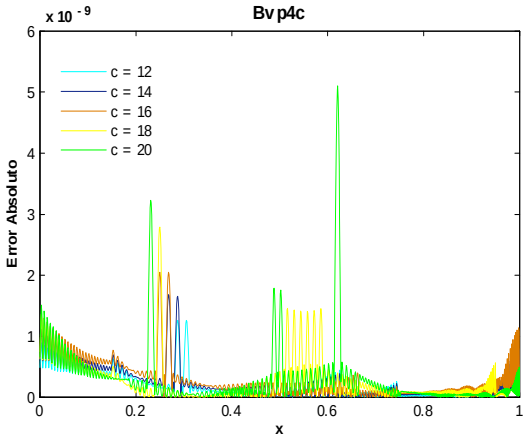


Fig.312. Error Absoluto de la solución Bvp4c para el ejemplo 9 con los valores del parámetro c=12,14,16,18,20 y tolerancias absoluta y relativa de 1e-7.
Cputime = 3.62 s.

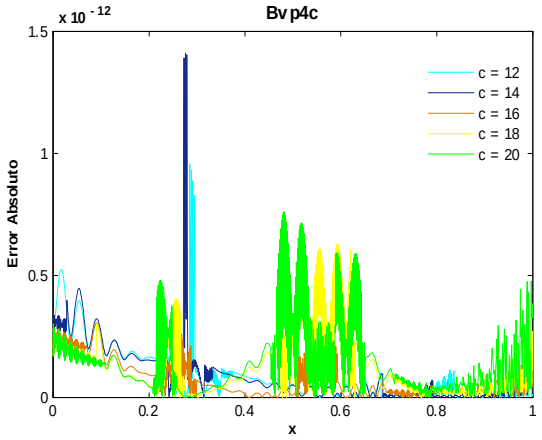


Fig.313. Error Absoluto de la solución Bvp4c para el ejemplo 9 con los valores del parámetro c=12,14,16,18,20 y tolerancias absoluta y relativa de 1e-10.
Cputime = 6.95 s.

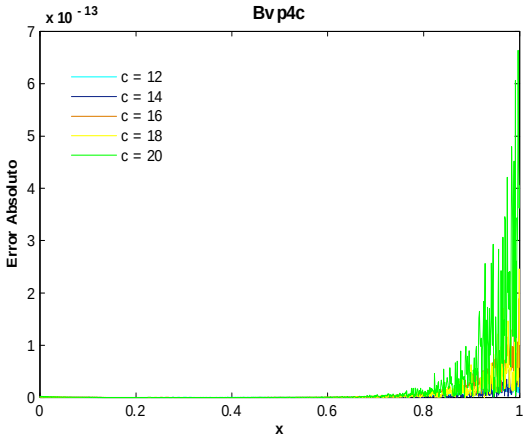


Fig.314. Error Absoluto de la solución Bvp4c para el ejemplo 9 con los valores del parámetro c=12,14,16,18,20 y tolerancias absoluta y relativa de 1e-13.
Cputime = 12.33 s.

El orden del error absoluto utilizando bvp4c y $c = 2, 4, 6, 8, 10$, fue de -5 con tolerancias $AbsTol = 1e-6$, $\mathrm{Re}lTol = 1e-3$ y disminuyó hasta -14 con $AbsTol = 1e-13$, $\mathrm{Re}lTol = 1e-13$, ver figs. 307-310. Para $c = 12, 14, 16, 18, 10$, el orden disminuyó de -5 a -13, considerando los mismos valores de tolerancias anteriores, ver figs. 311-314. El mayor tiempo utilizado, 12.33 s, fue al calcular el error absoluto para $c = 12, 14, 16, 18, 20$ con $AbsTol = RelTol = 1e-13$.

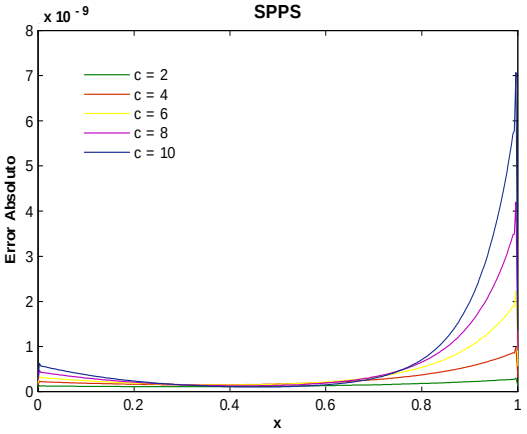


Fig.315. Error Absoluto de la solución SPPS para el ejemplo 9 con los valores del parámetro c=2,4,6,8,10 considerando 20 potencias y 300 puntos.
Cputime = 11.33 s.

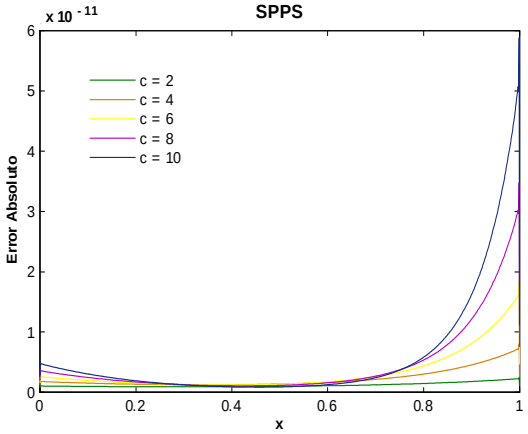


Fig.316. Error Absoluto de la solución SPPS para el ejemplo 9 con los valores del parámetro c=2,4,6,8,10 considerando 20 potencias y 1000 puntos.
Cputime = 29.13 s.

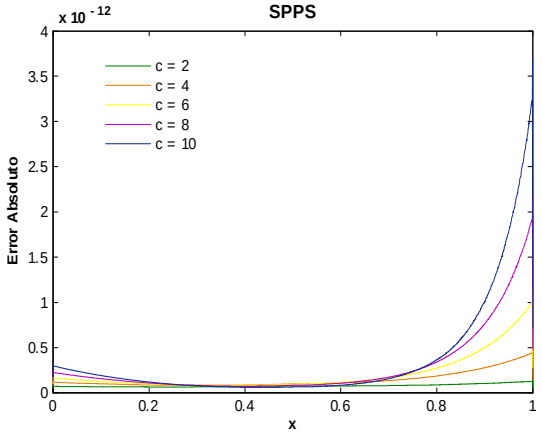


Fig.317. Error Absoluto de la solución SPPS para el ejemplo 9 con los valores del parámetro c=2,4,6,8,10 considerando 20 potencias y 2000 puntos.
Cputime = 54.75 s.

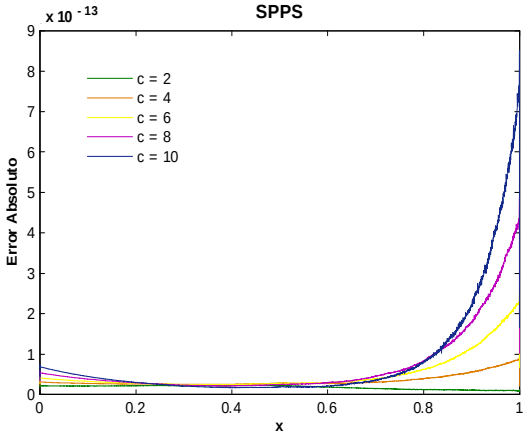


Fig.318. Error Absoluto de la solución SPPS para el ejemplo 9 con los valores del parámetro c=2,4,6,8,10 considerando 20 potencias y 2900 puntos.
Cputime = 77.08 s.

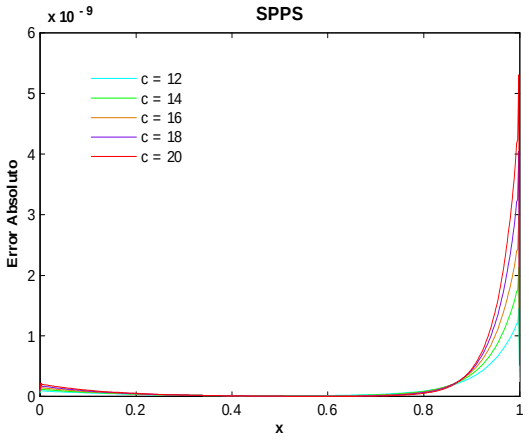


Fig.319. Error Absoluto de la solución SPPS para el ejemplo 9 con los valores del parámetro c=12,14,16,18,20 considerando 25 potencias y 500 puntos.
Cputime = 19.92 s.

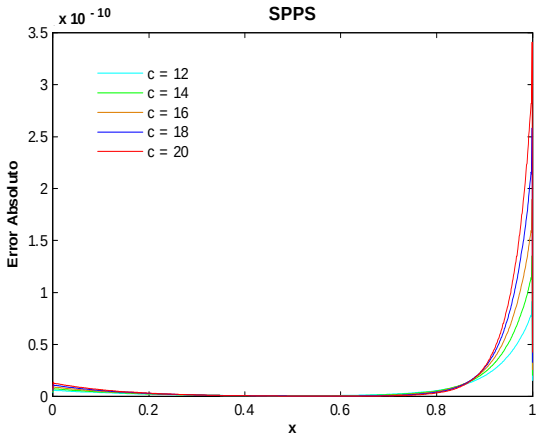


Fig.320. Error Absoluto de la solución SPPS para el ejemplo 9 con los valores del parámetro c=12,14,16,18,20 considerando 25 potencias y 1000 puntos.
Cputime = 35.89 s.

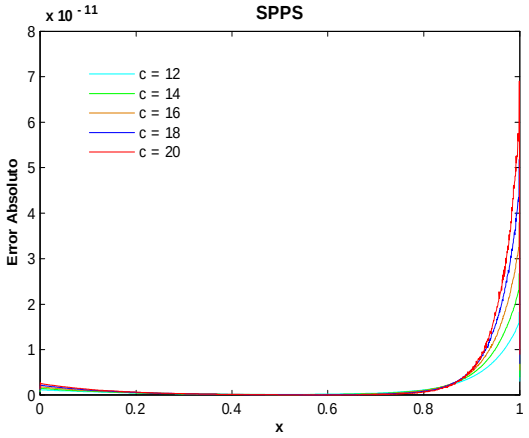


Fig.321. Error Absoluto de la solución SPPS para el ejemplo 9 con los valores del parámetro c=12,14,16,18,20 considerando 25 potencias y 1500 puntos.
Cputime = 51.45 s.

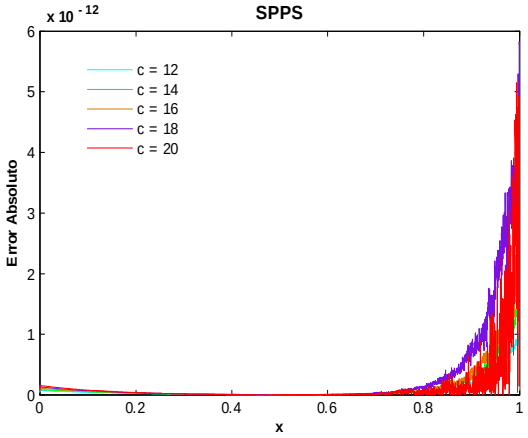


Fig.322. Error Absoluto de la solución SPPS para el ejemplo 9 con los valores del parámetro c=12,14,16,18,20 considerando 25 potencias y 3000 puntos.
Cputime = 99.34s.

En cuanto a SPPS, si $c = 2, 4, 6, 8, 10$ el orden del error mejoró de -9 (fig.315 con 20 potencias y 300 puntos) a -13(fig.318 con 20 potencias y 2900 puntos), para $c = 12, 14, 16, 18, 20$ fue de -9(fig.319 con 25 potencias y 500 puntos) a -12(fig.322 con 25 potencias y 3000 puntos). El tiempo máximo fue de 99.34 s, ver fig. 322.

Para ambos métodos el orden del error empeora al incrementar los valores del parámetro c. Aunque se puede mejorar estos resultados al considerar mayor exactitud: para Bvp4c disminuir los valores de tolerancias y para SPPS aumentar las potencias y los puntos.

Ejemplo 10

Solución Exacta

Considérese el problema con valores en la frontera

$$\begin{aligned} y'' - c^2 y &= 0 \\ y'(a) &= \alpha \\ y'(b) &= \beta \end{aligned} \tag{75}$$

donde $c(\neq 0), \alpha, \beta, a, b \in \mathbb{R}$, $a < b$.
Se sabe que la solución general de la ecuación diferencial en (75) es $y_g = c_1 e^{cx} + c_2 e^{-cx}$ donde c_1 y c_2 son constantes arbitrarias. Entonces se debe cumplir

$$\begin{aligned} y_g'(a) &= c_1 c e^{ca} - c_2 c e^{-ca} = \alpha \\ y_g'(b) &= c_1 c e^{cb} - c_2 c e^{-cb} = \beta. \end{aligned} \tag{76}$$

Por los Teoremas 4 y 5 el problema (75) tiene solución única si y sólo si

$$\Delta = \begin{vmatrix} ce^{ca} & -ce^{-ca} \\ ce^{cb} & -ce^{-cb} \end{vmatrix} = c^2 \left(e^{c(a-b)} - e^{-c(a-b)} \right) = 2c^2 \sinh(cb - ca) \neq 0 \tag{77}$$

Pero nótese que como $c \neq 0$, tenemos $2c^2 \sinh(cb - ca) = 0 \quad \Leftrightarrow \quad \sinh(cb - ca) = 0 \quad \Leftrightarrow \quad c(a - b) = 0 \quad \Leftrightarrow \quad a = b$, de modo que el problema (75) satisface (77).

Resolviendo el sistema (76) mediante la regla de Cramer:

$$c_1 = \frac{\begin{vmatrix} \alpha & -ce^{-ca} \\ \beta & -ce^{-cb} \end{vmatrix}}{\Delta} = \frac{c\left(\beta e^{-ca} - \alpha e^{-cb}\right)}{2c^2 \sinh(cb - ca)} \qquad c_2 = \frac{\begin{vmatrix} ce^{ca} & \alpha \\ ce^{cb} & \beta \end{vmatrix}}{\Delta} = \frac{c\left(\beta e^{ca} - \alpha e^{cb}\right)}{2c^2 \sinh(cb - ca)} \tag{78}$$

De donde se concluye que la solución de (75) es:

$$y_p = \frac{c\left(\beta e^{-ca} - \alpha e^{-cb}\right)}{2c^2 \sinh(cb - ca)} e^{cx} + \frac{c\left(\beta e^{ca} - \alpha e^{cb}\right)}{2c^2 \sinh(cb - ca)} e^{-cx} \tag{79}$$

Las siguientes gráficas muestran algunas soluciones exactas con valores de los parámetros $\alpha = -1$, $\beta = 1$, $a = 0$, $b = 1$, $c = 4, 5, 6, 7, 8, 9, 10, 12, 14$.

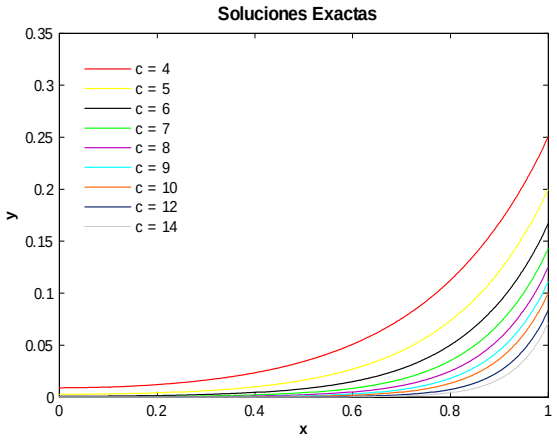


Fig.323

Soluciones Numéricas

A continuación se presentan las graficas obtenidas de los errores absoluto y relativo para los valores $\alpha = -1$, $\beta = 1$, $a = 0$, $b = 1$, $c = 3 - 7, 9 - 13$.

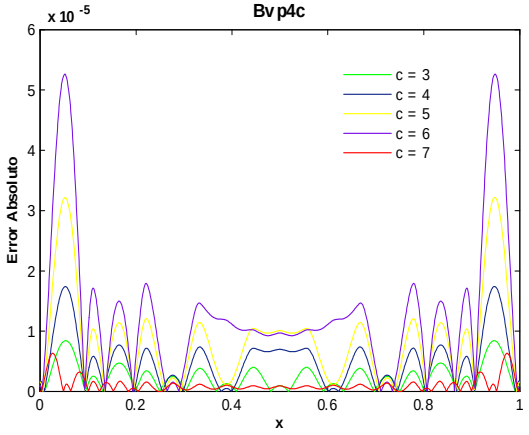


Fig.324. Error Absoluto de la solución Bvp4c para el ejemplo 10 con los valores del parámetro c=3,4,5,6,7 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 0.82 s.

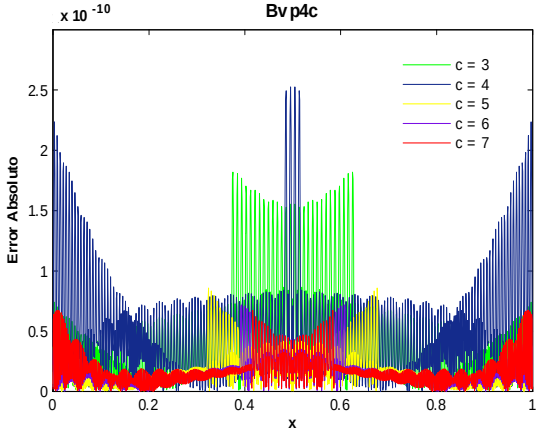


Fig.325. Error Absoluto de la solución Bvp4c para el ejemplo 10 con los valores del parámetro c=3,4,5,6,7 y tolerancias absoluta y relativa de 1e-8.
Cputime = 1.38 s.

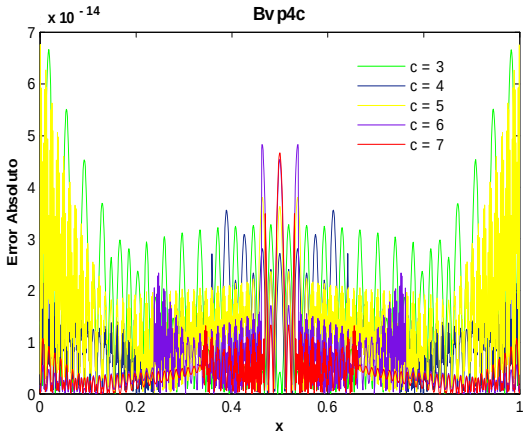


Fig.326. Error Absoluto de la solución Bvp4c para el ejemplo 10 con los valores del parámetro c=3,4,5,6,7 y tolerancias absoluta de y relativa de 1e-11.
Cputime = 4.56 s.

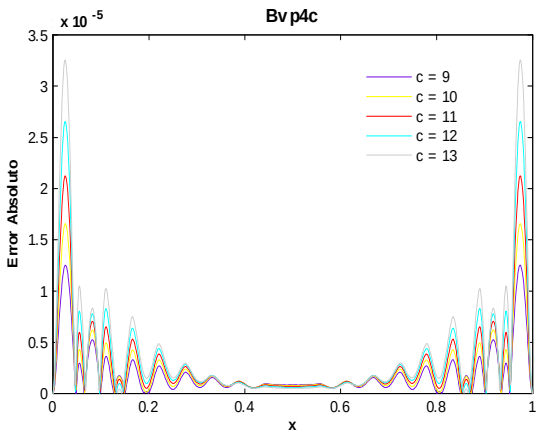


Fig.327. Error Absoluto de la solución Bvp4c para el ejemplo 10 con los valores del parámetro c=9,10,11,12,13 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 0.84 s.

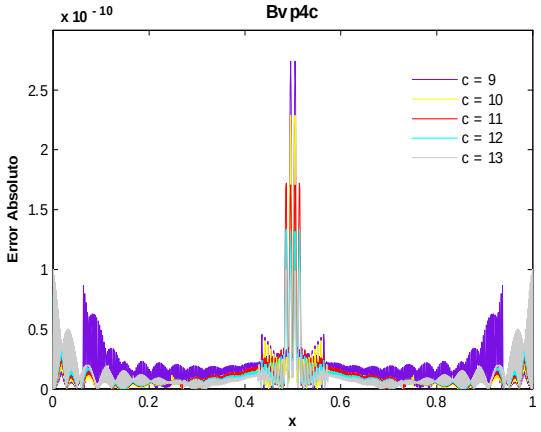


Fig.328. Error Absoluto de la solución Bvp4c para el ejemplo 10 con los valores del parámetro c=9,10,11,12,13 y tolerancias absoluta y relativa de 1e-8.
Cputime = 1.71 s.

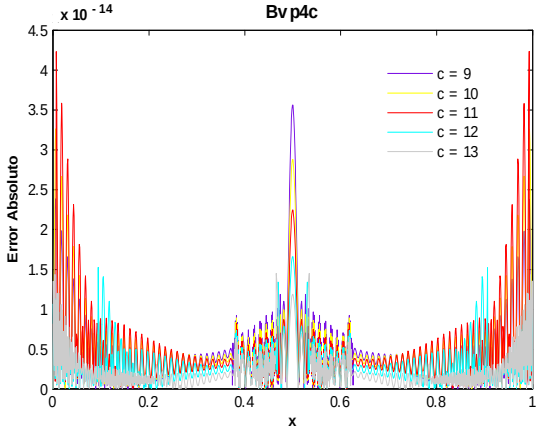


Fig.329. Error Absoluto de la solución Bvp4c para el ejemplo 10 con los valores del parámetro c=9,10,11,12,13 y tolerancias absoluta y relativa de 1e-11.
Cputime = 6.76 s.

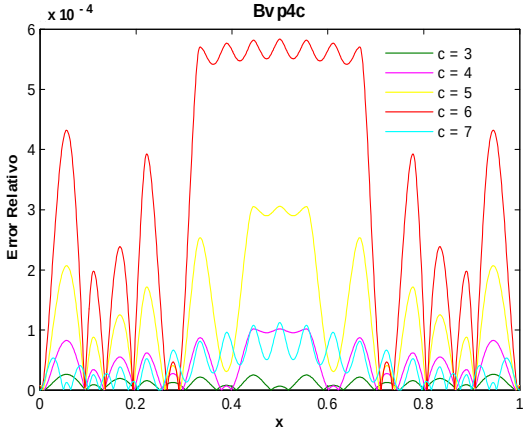


Fig.330. Error Absoluto de la solución Bvp4c para el ejemplo 10 con los valores del parámetro c=3,4,5,6,7 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 0.8 s.

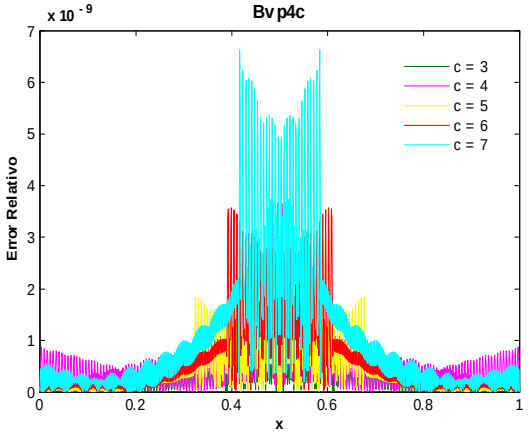


Fig.331. Error Absoluto de la solución Bvp4c para el ejemplo 10 con los valores del parámetro c=3,4,5,6,7 y tolerancias absoluta y relativa de 1e-8.
Cputime = 1.37 s.

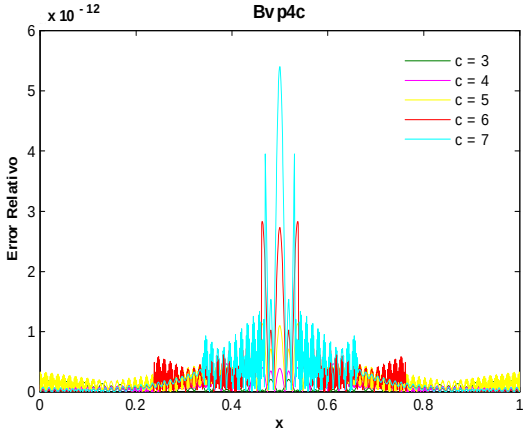


Fig.332. Error Absoluto de la solución Bvp4c para el ejemplo 10 con los valores del parámetro c=3,4,5,6,7 y tolerancias absoluta y relativa de 1e-11.
Cputime = 4.53 s.

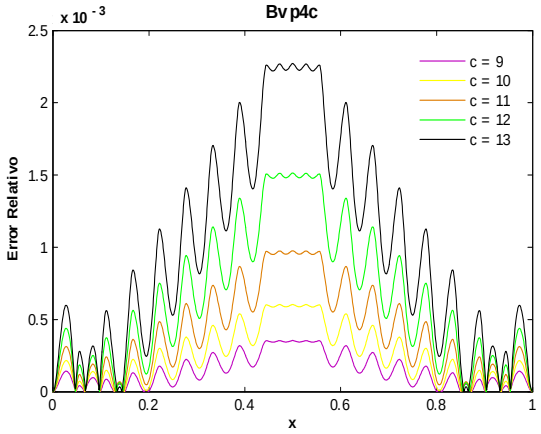


Fig.333. Error Absoluto de la solución Bvp4c para el ejemplo 10 con los valores del parámetro c=9,10,11,12,13 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 0.84 s.

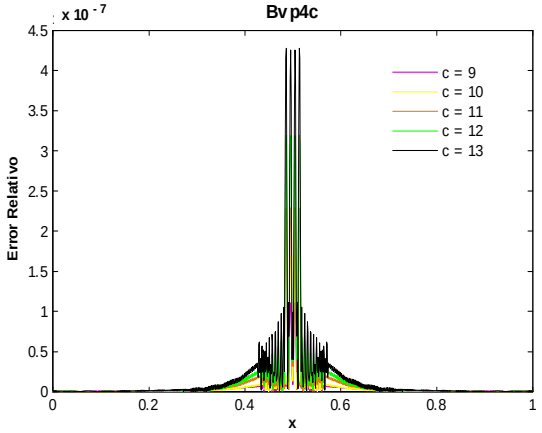


Fig.334. Error Absoluto de la solución Bvp4c para el ejemplo 10 con los valores del parámetro c=9,10,11,12,13 y tolerancias absoluta de y relativa de 1e-8.
Cputime = 1.68 s.

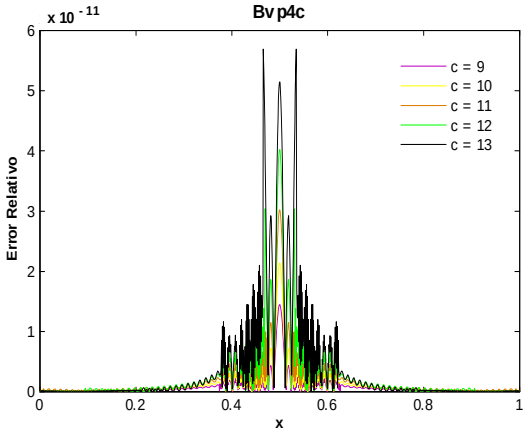


Fig.335. Error Absoluto de la solución Bvp4c para el ejemplo 10 con los valores del parámetro c=9,10,11,12,13 y tolerancias absoluta y relativa de 1e-11.
Cputime = 6.77 s.

El orden del error absoluto para ambos conjuntos de valores de c se comportó de la misma manera, fue de -5 , con $AbsTol = 1e-6, RelTol = 1e-3$, hasta -14, con $AbsTol = RelTol = 1e-11$, ver figs.324-329. Al calcular el error relativo se utilizaron los valores de tolerancias anteriores. Como muestran las figuras 330-332 el orden del error relativo para $c = 3, 4, 5, 6, 7$ fue de -4 hasta -12. Para $c = 9, 10, 11, 12, 13$ fue de -3 a -11 ver figs.333-335. El tiempo máximo fue de 6.77 s, ver fig. 335.

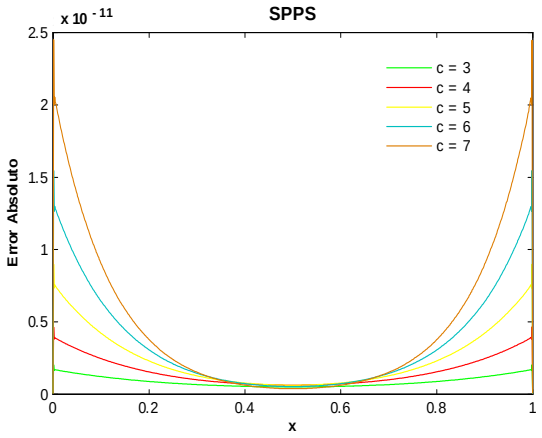


Fig.336. Error Absoluto de la solución SPPS para el ejemplo 10 con los valores del parámetro c=3,4,5,6,7 considerando 20 potencias y 800 puntos.
Cputime = 21.98 s.

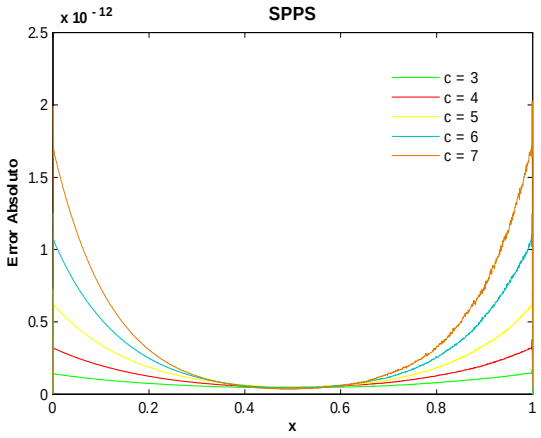


Fig.337. Error Absoluto de la solución SPPS para el ejemplo 10 con los valores del parámetro c=3,4,5,6,7 considerando 20 potencias y 1500 puntos.
Cputime = 39.93 s.

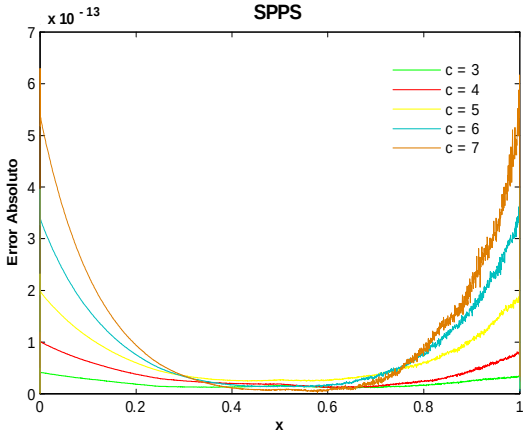


Fig.338. Error Absoluto de la solución SPPS para el ejemplo 10 con los valores del parámetro c=3,4,5,6,7 considerando 20 potencias y 2000 puntos.
Cputime = 52.47 s.

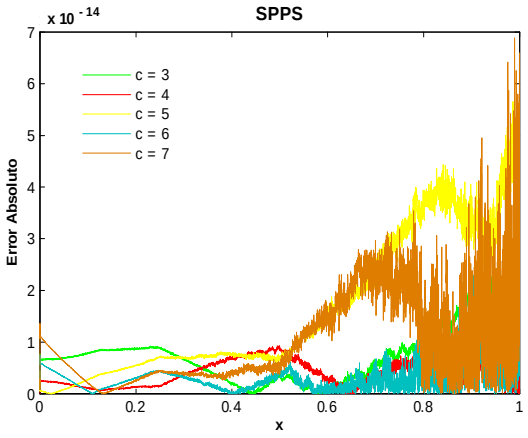


Fig.339. Error Absoluto de la solución SPPS para el ejemplo 10 con los valores del parámetro c=3,4,5,6,7 considerando 20 potencias y 5000 puntos.
Cputime = 127.82 s.

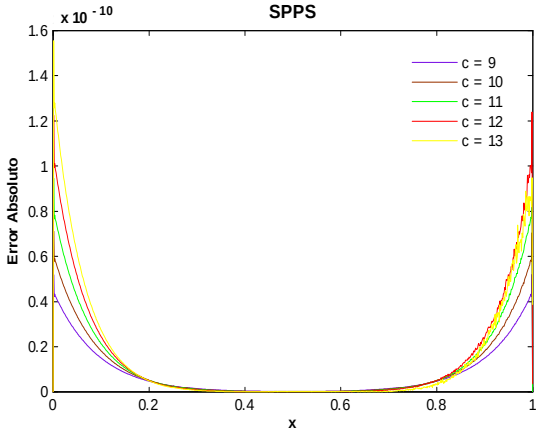


Fig.340. Error Absoluto de la solución SPPS para el ejemplo 10 con los valores del parámetro c=9,10,11,12,13 considerando 25 potencias y 800 puntos.
Cputime = 27.16 s.

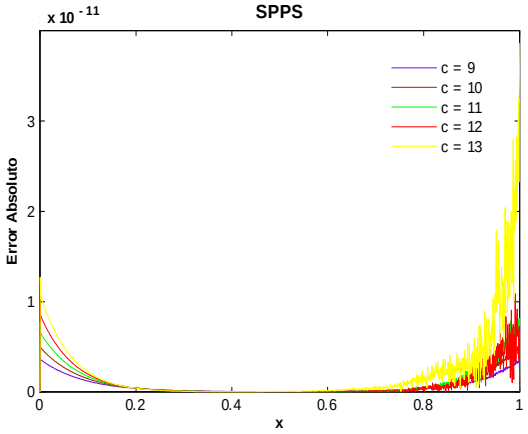


Fig.341. Error Absoluto de la solución SPPS para el ejemplo 10 con los valores del parámetro c=9,10,11,12,13 considerando 25 potencias y 1500 puntos.
Cputime = 49.66 s.

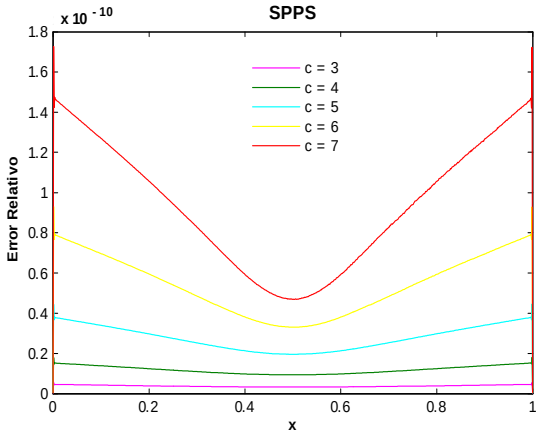


Fig.342. Error Relativo de la solución SPPS para el ejemplo 10 con los valores del parámetro c=3,4,5,6,7 considerando 25 potencias y 800 puntos.
Cputime = 27.44 s.

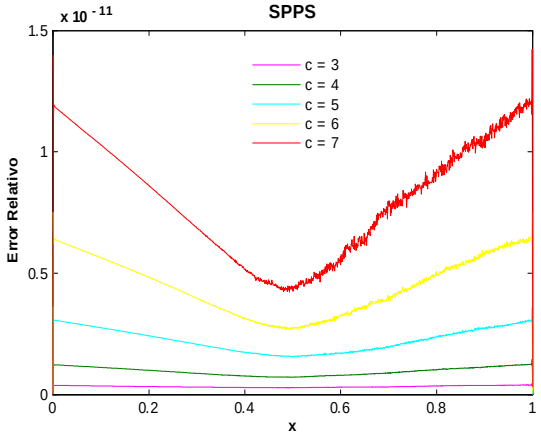


Fig.343. Error Relativo de la solución SPPS para el ejemplo 10 con los valores del parámetro c=3,4,5,6,7 considerando 25 potencias y 1500 puntos.
Cputime = 40.23 s.

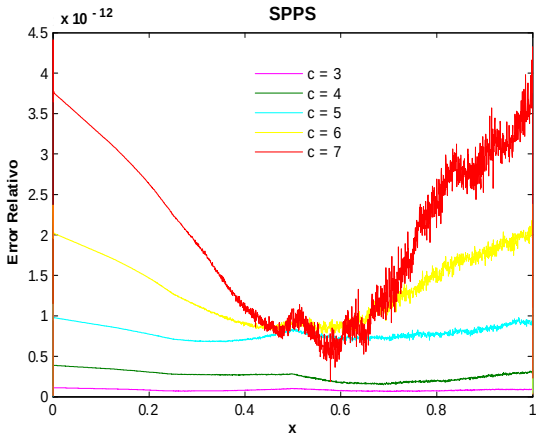


Fig.344. Error Relativo de la solución SPPS para el ejemplo 10 con los valores del parámetro c=3,4,5,6,7 considerando 25 potencias y 2000 puntos. Cputime = 52.47 s.

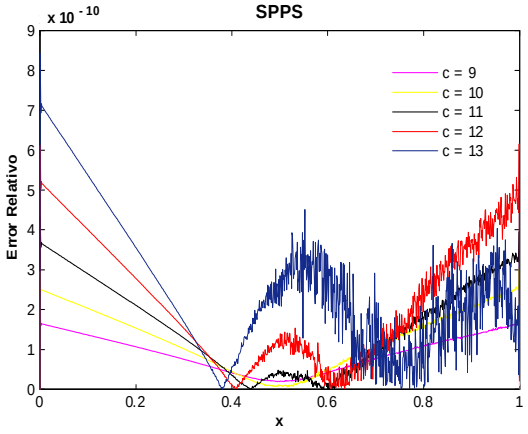


Fig.345. Error Relativo de la solución SPPS para el ejemplo 10 con los valores del parámetro c=9,10,11,12,13 considerando 25 potencias y 1000 puntos. Cputime = 33.8 s.

Las figuras 336-339 nos muestran que al calcular el error absoluto con ayuda del método SPPS para $c = 3, 4, 5, 6, 7$, considerando 20 potencias, se obtiene órdenes de -11, con 800 puntos y de hasta -14 con 5000 puntos. Para $c = 9, 10, 11, 12, 13$ los órdenes que se obtuvieron fueron de -10, con 25 potencias y 800 puntos, y de -11, con 25 potencias y 1500 puntos, ver figs. 340-343. Nótese que al aumentar las potencias y los puntos en el caso anterior, no mejoró el orden. El orden del error relativo para $c = 3, 4, 5, 6, 7$ fue de -10, con 800 puntos y 25 potencias, hasta -12, con 2000 puntos y 25 potencias, ver figs. 342-344. Para $c = 9, 10, 11, 12, 13$ el orden del error relativo obtenido con 1000 puntos y 25 potencias fue de -10, al incrementar las potencias y el número de puntos no mejoró el orden del error, ver fig. 345.

Ejemplo 11

Solución Exacta

Considérese el problema con valores en la frontera

$$\begin{aligned} y'' - k[k\sin^2(cx) + c\cos(cx)]y &= 0 \\ y'(a) &= \alpha \\ y'(b) &= \beta \end{aligned} \tag{80}$$

donde $c, k \neq 0, \alpha, \beta, a, b \in \mathbb{R}$, $a < b$.
De acuerdo a [18] la solución general de la ecuación diferencial en (80) es $y_g = c_1 e^{-\frac{k}{c}\cos(cx)} + c_2 e^{-\frac{k}{c}\cos(cx)} F(x)$ donde c_1 y c_2 son constantes arbitrarias y $F(x) = \int e^{\frac{2k}{c}\cos(cx)} dx$. Por lo que se debe cumplir

$$\begin{aligned} y_g'(a) &= c_1 k\sin(ca)e^{-\frac{k}{c}\cos(ca)} + c_2\left[kF(a)\sin(ca)e^{-\frac{k}{c}\cos(ca)} + e^{\frac{k}{c}\cos(ca)}\right] = \alpha \\ y_g'(b) &= c_1 k\sin(cb)e^{-\frac{k}{c}\cos(cb)} + c_2\left[kF(b)\sin(cb)e^{-\frac{k}{c}\cos(cb)} + e^{\frac{k}{c}\cos(cb)}\right] = \beta. \end{aligned} \tag{81}$$

Por los Teoremas 4 y 5, el problema no homogéneo (80) tiene solución única si y sólo si

$$\begin{aligned} \Delta &= \begin{vmatrix} k\sin(ca)e^{-\frac{k}{c}\cos(ca)} & kF(a)\sin(ca)e^{-\frac{k}{c}\cos(ca)} + e^{\frac{k}{c}\cos(ca)} \\ k\sin(cb)e^{-\frac{k}{c}\cos(cb)} & kF(b)\sin(cb)e^{-\frac{k}{c}\cos(cb)} + e^{\frac{k}{c}\cos(cb)} \end{vmatrix} \\ &= k^2\sin(ca)\sin(cb)e^{-\frac{k}{c}[\cos(ca)+\cos(cb)]}[F(b) - F(a)] \end{aligned} \tag{82}$$

$$+ k\sin(ca)e^{\frac{k}{c}[\cos(cb)-\cos(ca)]} - k\sin(cb)e^{\frac{k}{c}[\cos(ca)-\cos(cb)]} \neq 0$$

Supóngase cierta la condición (82). Resolviendo el sistema (81) mediante la regla de Cramer:

$$\begin{aligned} c_1 &= \frac{\begin{vmatrix} \alpha & kF(a)\sin(ca)e^{-\frac{k}{c}\cos(ca)} + e^{\frac{k}{c}\cos(ca)} \\ \beta & kF(b)\sin(cb)e^{-\frac{k}{c}\cos(cb)} + e^{\frac{k}{c}\cos(cb)} \end{vmatrix}}{\Delta} \\ &= \frac{\alpha\left[kF(b)\sin(cb)e^{-\frac{k}{c}\cos(cb)} + e^{\frac{k}{c}\cos(cb)}\right] - \beta\left[kF(a)\sin(ca)e^{-\frac{k}{c}\cos(ca)} + e^{\frac{k}{c}\cos(ca)}\right]}{\Delta} \\ c_2 &= \frac{\begin{vmatrix} k\sin(ca)e^{-\frac{k}{c}\cos(ca)} & \alpha \\ k\sin(cb)e^{-\frac{k}{c}\cos(cb)} & \beta \end{vmatrix}}{\Delta} = \frac{\beta k\sin(ca)e^{-\frac{k}{c}\cos(ca)} - \alpha k\sin(cb)e^{-\frac{k}{c}\cos(cb)}}{\Delta} \end{aligned} \tag{83}$$

De donde se concluye que la solución de (80) es:

$$y_p = c_1 e^{-\frac{k}{c}\cos(cx)} + c_2 e^{-\frac{k}{c}\cos(cx)} F(x)$$

donde $F(x) = \int e^{\frac{2k}{c}\cos(cx)}dx$ y c_1, c_2 están dadas en (83).

Soluciones Numéricas

Las siguientes figuras representan algunos potenciales $q(x)$ y soluciones exactas del problema (80), para ciertos valores de c y k. Posteriormente se presenta el error absoluto calculado con las soluciones SPPS y Bvp4c considerando los valores de los parámetros: $\alpha = 1, \beta = 1, a = 0, b = 1, k = 4, 17, 29, c = 4, 14, 19, 24, 30, 34, 39, 46, 50, 56, 58, 62, 68, 70, 73$.

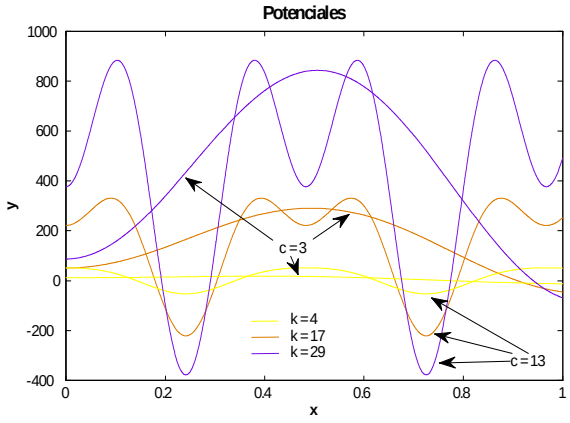


Fig. 346

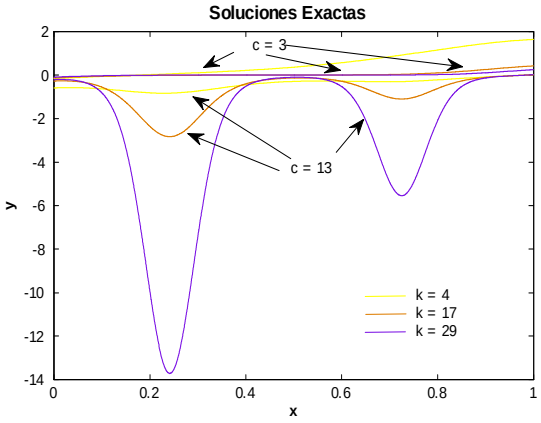


Fig. 347

En este ejemplo no se consideró el error relativo pues la solución exacta tiene zeros en el intervalo $[0, 1]$, de manera que este tipo de error no es significativo.

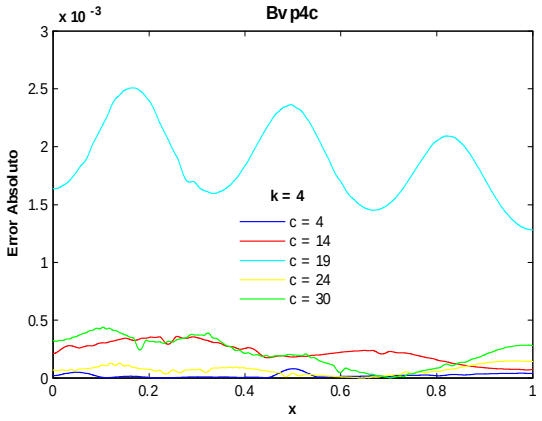


Fig.348. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=4, c=4,14,19,24,30 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 2.60s.

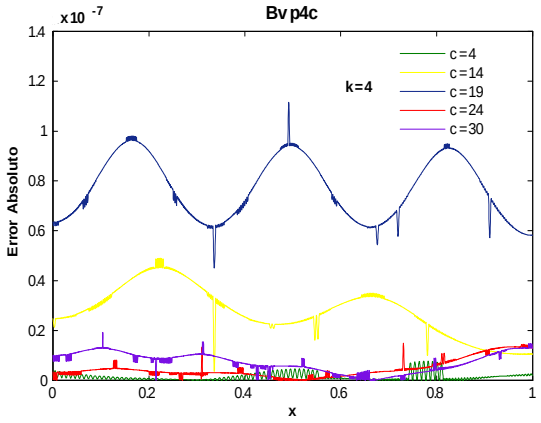


Fig.349. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=4, c=4,14,19,24,30 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 4.44 s.

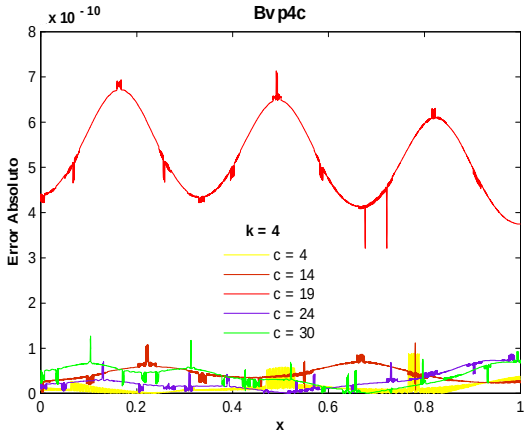


Fig.350. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=4, c=4,14,19,24,30 y tolerancias absoluta de 1e-8 y relativa de 1e-10. Cputime = 8.84 s.

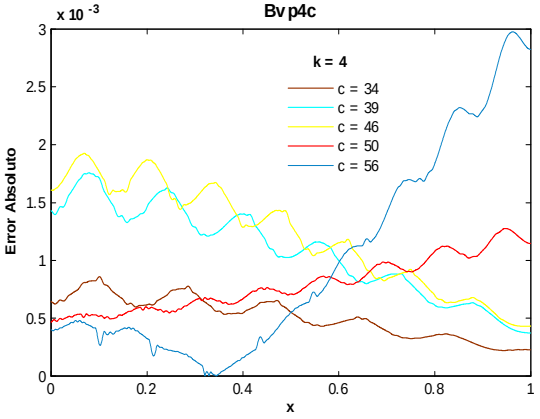


Fig.351. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=4, c=34,39,46,50,56 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 3.34 s

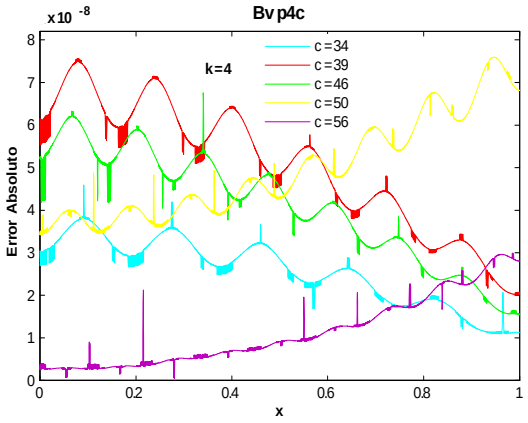


Fig.352. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=4, c=34,39,46,50,56 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 7.78 s.

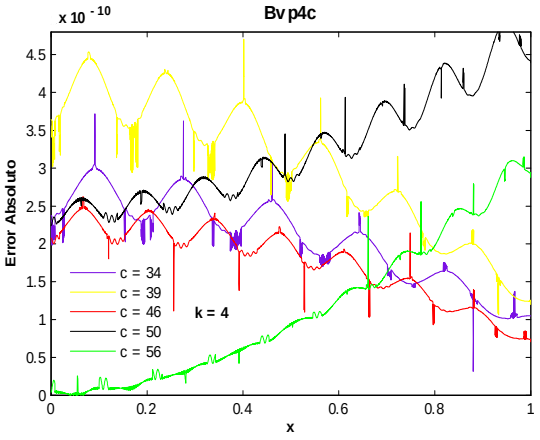


Fig.353. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=4, c=34,39,46,50,56 y tolerancias absoluta de 1e-8 y relativa de 1e-10. Cputime = 16.29 s.

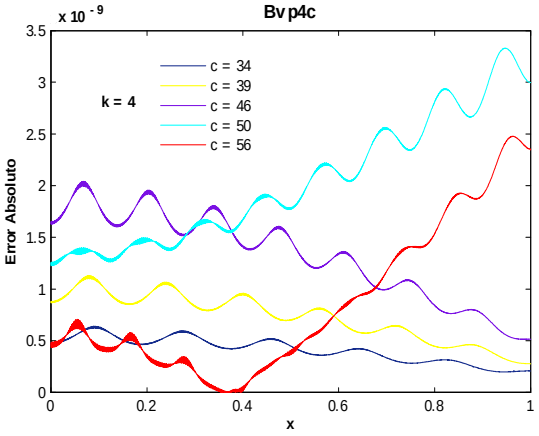


Fig.354. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=4, c=34,39,46,50,56 y tolerancias absoluta de 1e-10 y relativa de 1e-12. Cputime = 9.66 s.

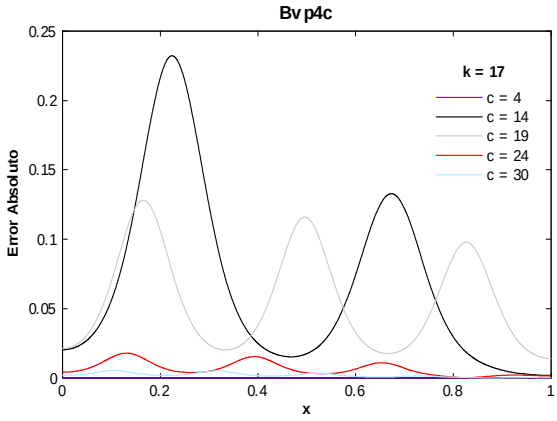


Fig.355. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=17, c=4,14,19,24,30 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 3.35s.

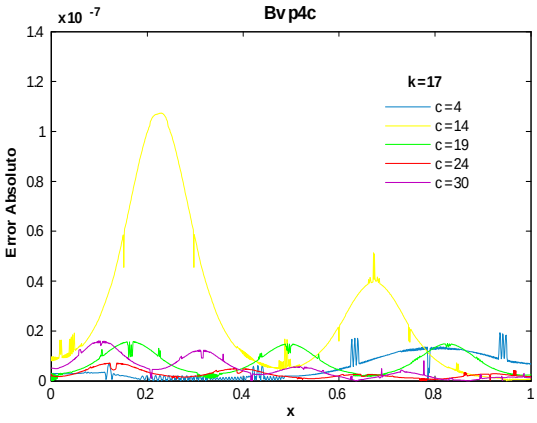


Fig.356. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=17, c=4,14,19,24,30 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 7.26 .

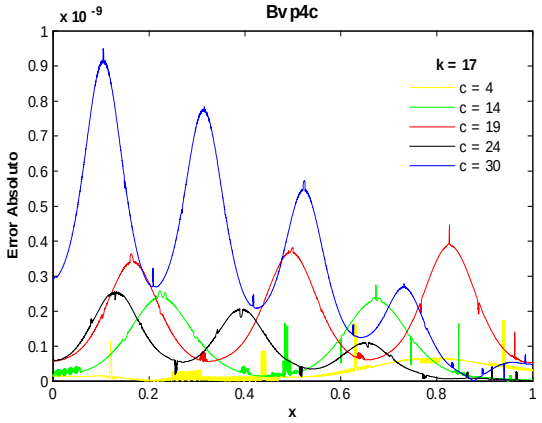


Fig.357. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=17, c=4,17,19,24,30 y tolerancias absoluta de 1e-8 y relativa de 1e-10. Cputime = 15.03s.

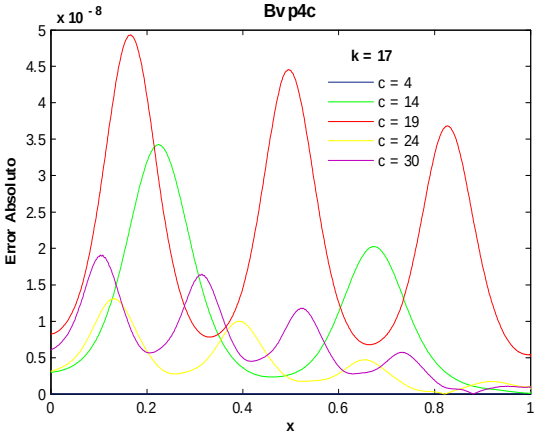


Fig.358. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=17, c=4,17,19,24,30 y tolerancias absoluta de 1e-10 y relativa de 1e-12. Cputime = 9.77 s.

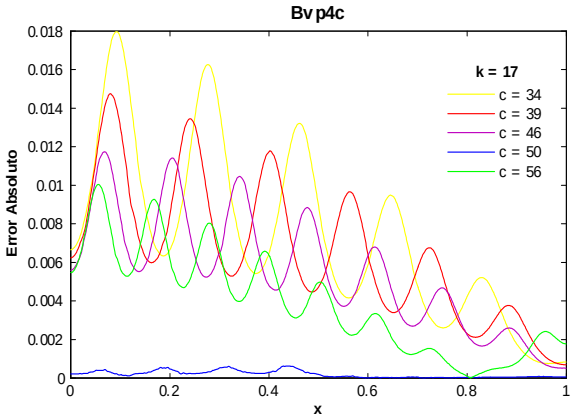


Fig.359. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=17, c=34,39,46,50,56 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 5.3s

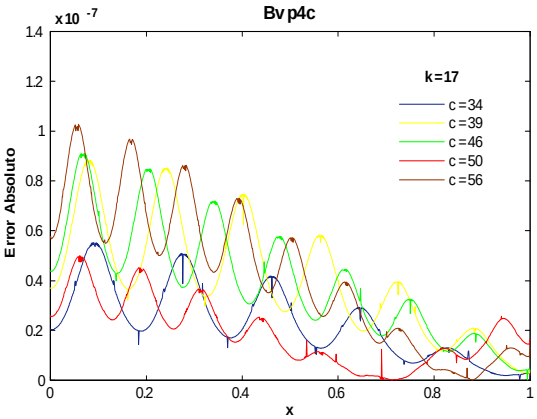


Fig.360. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=17, c=34,39,46,50,56 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 10.54 .

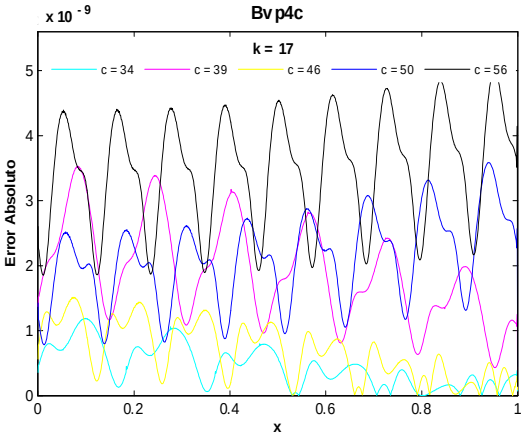


Fig.361. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=17, c=34,39,46,50,56 y tolerancias absoluta de 1e-8 y relativa de 1e-10. Cputime = 15.72 s.

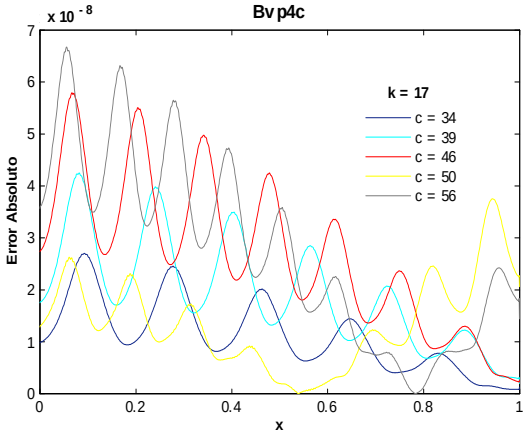
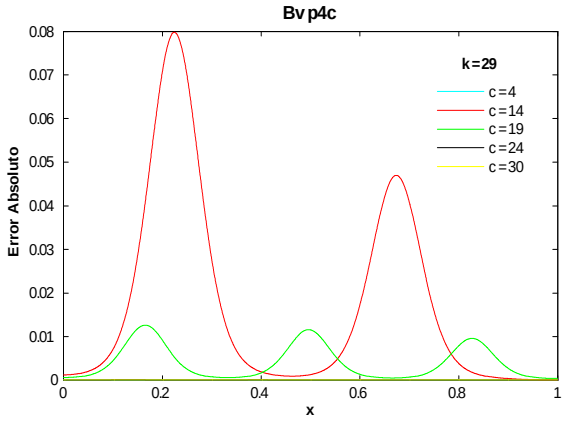


Fig.362. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=17, c=34,39,46,50,56 y tolerancias absoluta de 1e-10 y relativa de 1e-12. Cputime = 9.42 s.



. Fig.363. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=29, c=4,17,19,24,30 y tolerancias absoluta de 1e-6 y relativa de 1e-3. Cputime = 9.63 s.

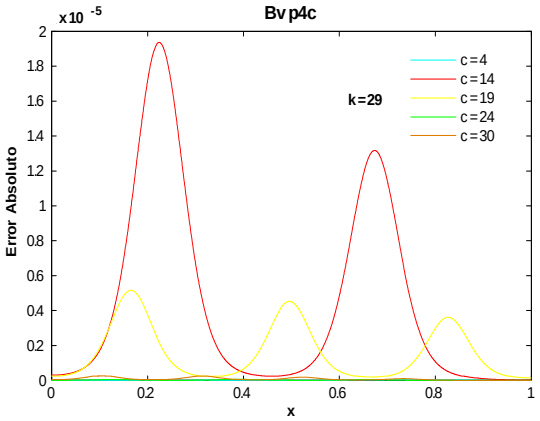


Fig.364. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=29, c=4,17,19,24,30 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 16.74 s.

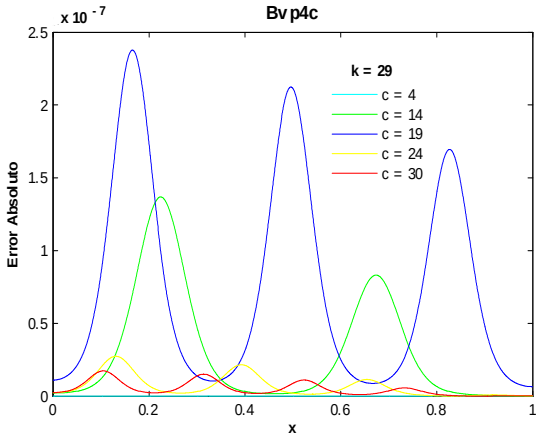


Fig.365. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=29, c=4,17,19,24,30 y tolerancias absoluta de 1e-8 y relativa de 1e-10. Cputime = 21.01 s.

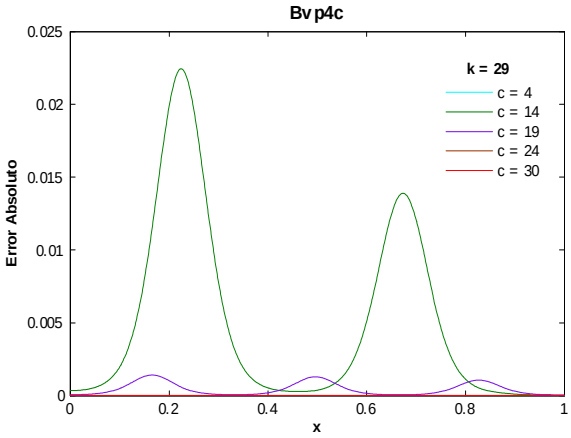


Fig.366. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=29, c=4,17,19,24,30 y tolerancias absoluta de 1e-10 y relativa de 1e-12. Cputime = 15.15 s.

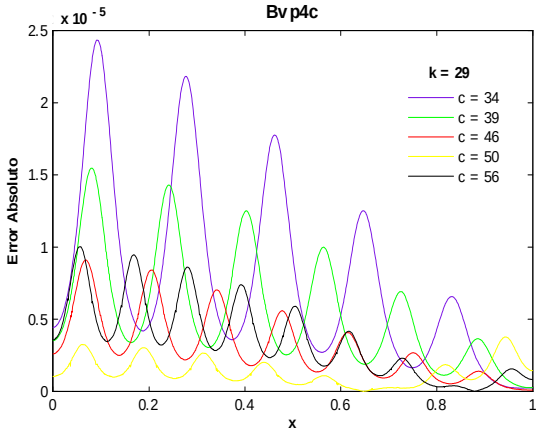


Fig.367. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=29, c=34,39,46,50,56 y tolerancias absoluta y relativa de 1e-6. Cputime = 9.75 s.

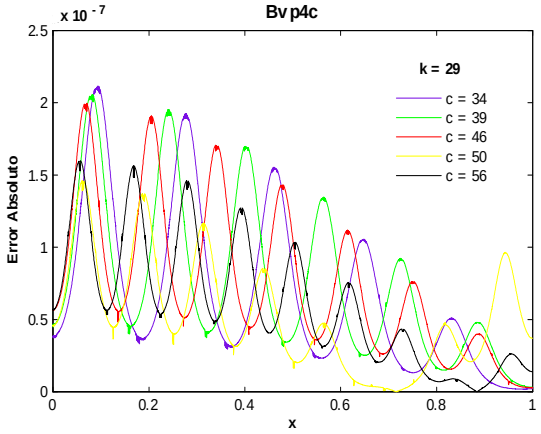


Fig.368. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=29, c=34,39,46,50,56 y tolerancias absoluta de 1e-6 y relativa de 1e-8. Cputime = 15.75 s.

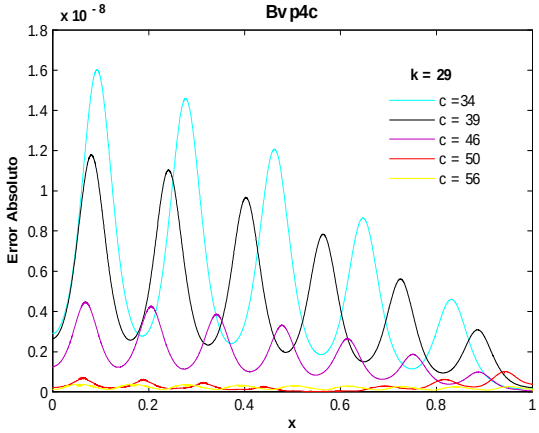


Fig.369. Error Absoluto de la solución Bvp4c para el ejemplo 11 con los valores de los parámetros k=29, c=34,39,46,50,56 y tolerancias absoluta de 1e-10 y relativa de 1e-12. Cputime = 12.44 s.

Las figuras 348-350 muestran que para los valores de $k = 4$ y $c = 19$, el orden del error absoluto obtenido al considerar la tolerancias $AbsTol = 1e-6, RelTol = 1e-3$ fue de -3 y para $AbsTol = 1e-8$, $\mathrm{Re}lTol = 1e-10$ fue de -10. Al tomar valores menores de tolerancias se preservó el orden. Para $c = 50$ los órdenes obtenidos con los valores de tolerancias anteriores fueron -3 y -10, respectivamente, ver figs.351-353. Al considerar $k = 17$ y $c = 14$ los órdenes obtenidos con dichas tolerancias fueron de -1 y -9 resp., ver figs. 355-357. Para $k = 29$ y $c = 34$ el orden obtenido al considerar los valores $AbsTol = RelTol = 1e-6$ fue de -5 y tomando $AbsTol = 1e-10$ y $RelTol = 1e-12$ el orden fue de -8. El mayor tiempo para este ejemplo fue de 21.01 s, al calcular el error absoluto para los valores de $k = 29, c = 4, 14, 19, 24, 30$ con valores de tolerancias $AbsTol = 1e-8$ y $RelTol = 1e-10$.

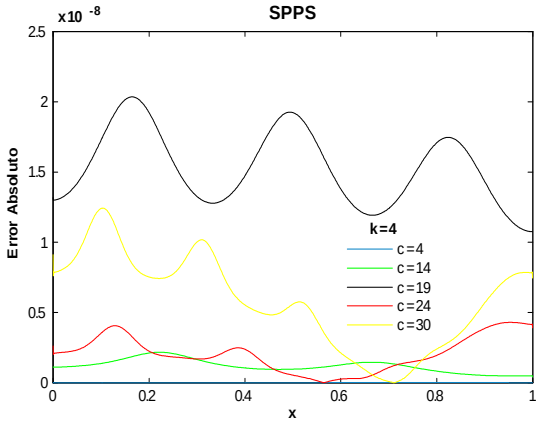


Fig.370. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=4, c=4,14,19,24,30 considerando 15 potencias y 1000 puntos. Cputime = 21.4 s.

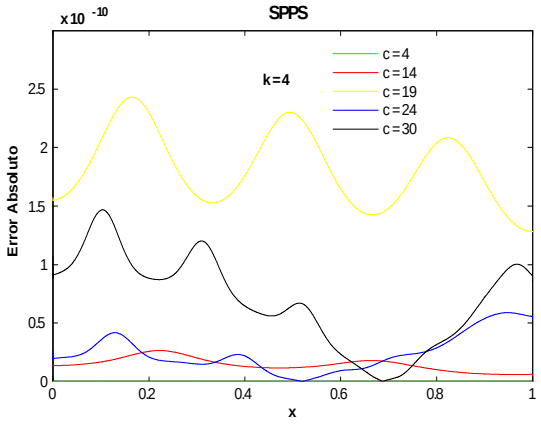


Fig.371. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=4, c=4,14,19,24,30 considerando 15 potencias y 3000 puntos. Cputime = 61.18 s.

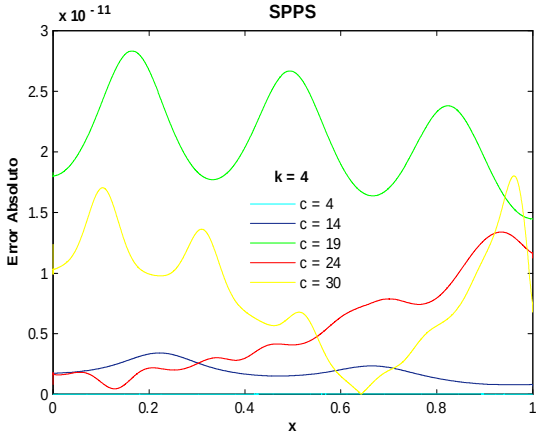


Fig.372. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=4, c=4,14,19,24,30 considerando 15 potencias y 5000 puntos. Cputime = 101.13 s.

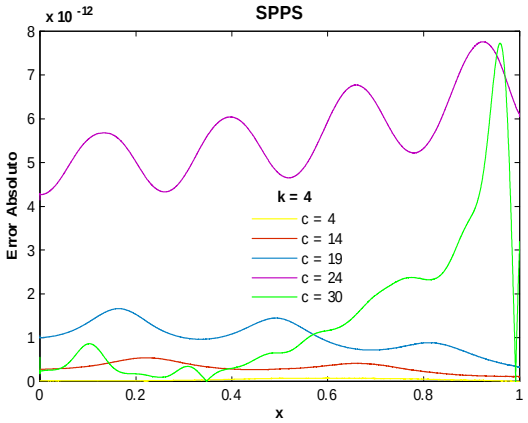


Fig.373. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=4, c=4,14,19,24,30 considerando 15 potencias y 8000 puntos. Cputime = 161.57s.

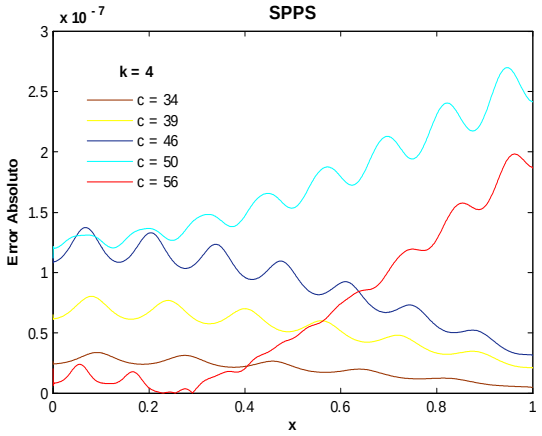


Fig.374. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=4, c=34,39,46,50,56 considerando 30 potencias y 1000 puntos. Cputime = 41.69 s.

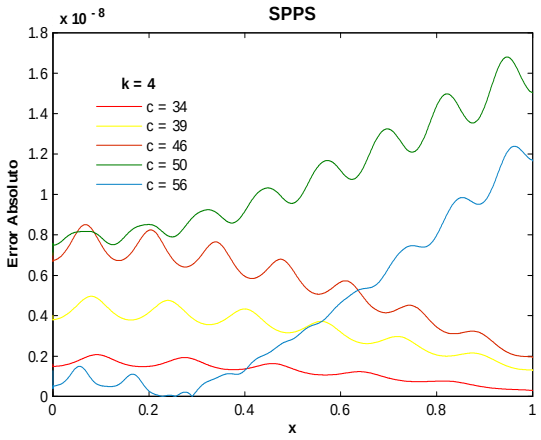


Fig.375. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=4, c=34,39,46,50,56 considerando 30 potencias y 2000 puntos. Cputime = 81.15 s.

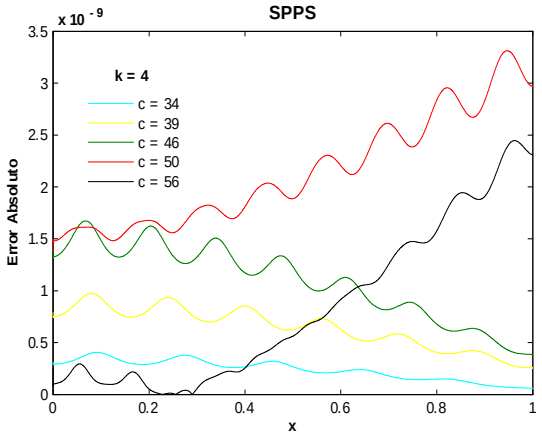


Fig.376. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=4, c=34,39,46,50,56 considerando 30 potencias y 3000 puntos. Cputime = 121.26 s.

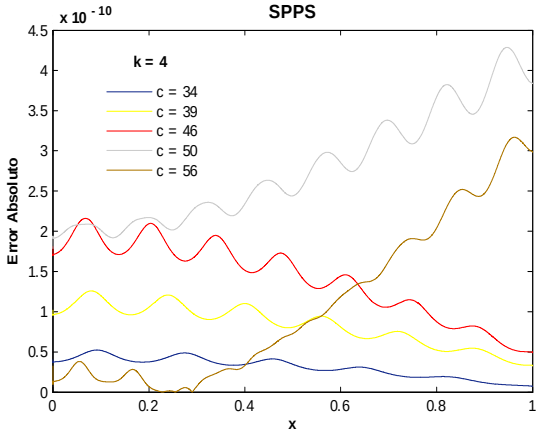


Fig.377. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=4, c=34,39,46,50,56 considerando 30 potencias y 5000 puntos. Cputime = 199.46 s.

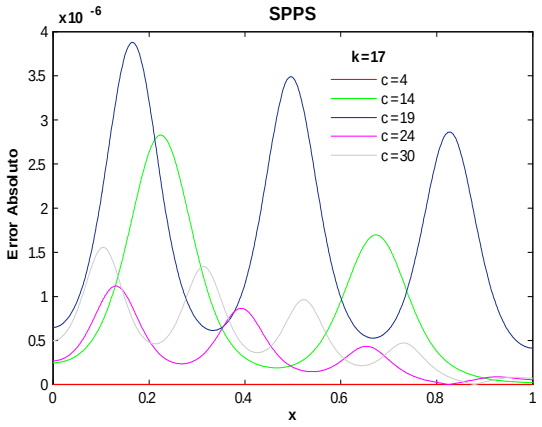


Fig.378. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=17, c=4,14,19,24,30 considerando 30 potencias y 1000 puntos. Cputime = 41.95 s.

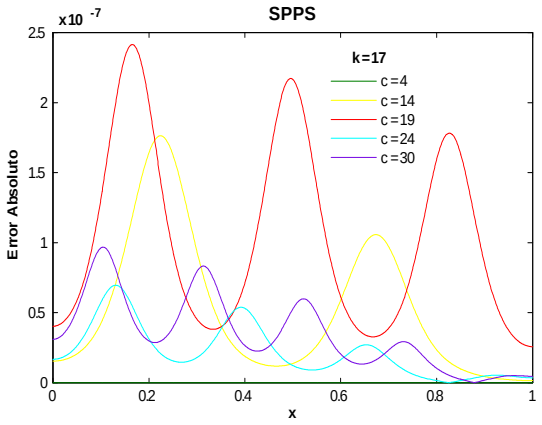


Fig.379. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=17, c=4,14,19,24,30 considerando 30 potencias y 2000 puntos. Cputime = 81.74 s.

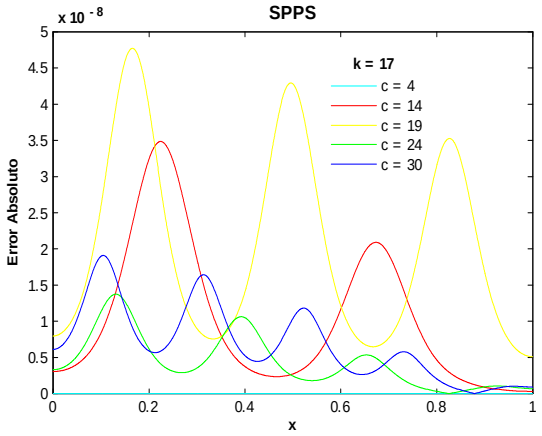


Fig.380. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=17, c=4,14,19,24,30 considerando 30 potencias y 3000 puntos. Cputime = 119.74 s.

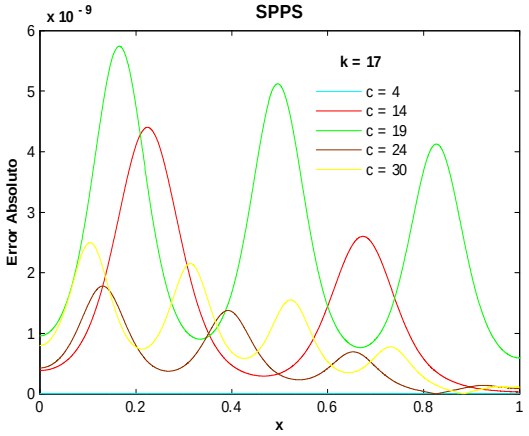


Fig.381. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=17, c=4,14,19,24,30 considerando 30 potencias y 5000 puntos. Cputime = 199.21 s.

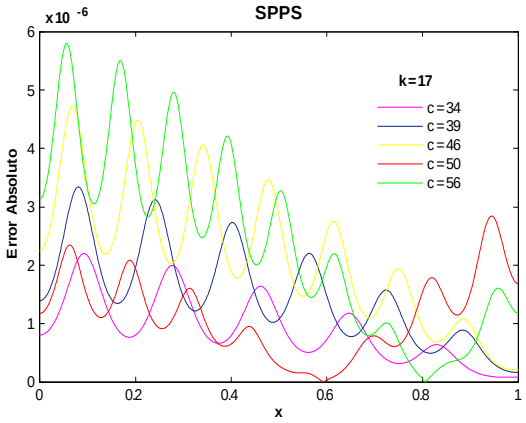


Fig.382. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=17, c=34,39,46,50,56 considerando 30 potencias y 1000 puntos. Cputime = 41.7 s.

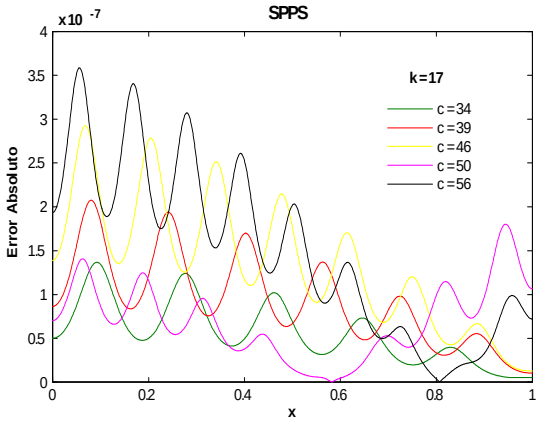


Fig.383. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=17, c=34,39,46,50,56 considerando 30 potencias y 2000 puntos. Cputime = 81.26 s.

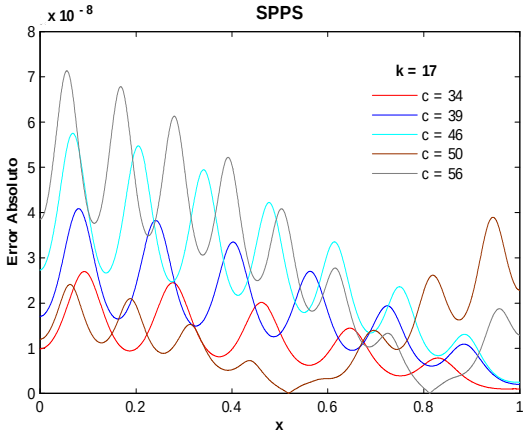


Fig.384. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=17, c=34,39,46,50,56 considerando 30 potencias y 3000 puntos. Cputime = 120.7 s.

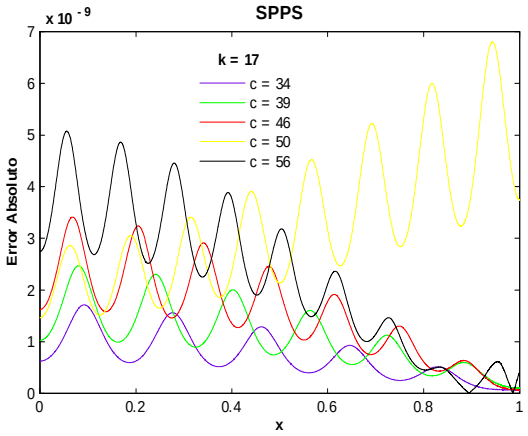


Fig.385. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=17, c=34,39,46,50,56 considerando 30 potencias y 6000 puntos. Cputime = 241.77 s.

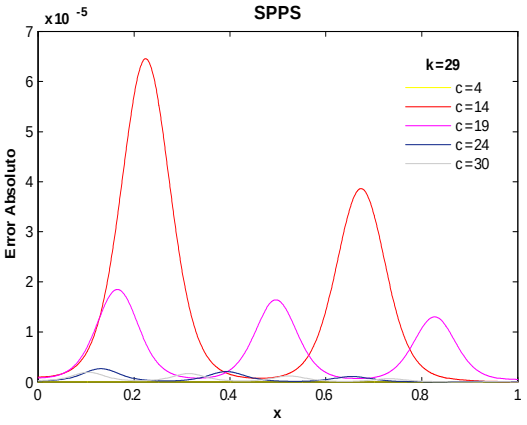


Fig.386. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=29, c=4,14,19,24,30 considerando 40 potencias y 2000 puntos. Cputime = 107.27 s.

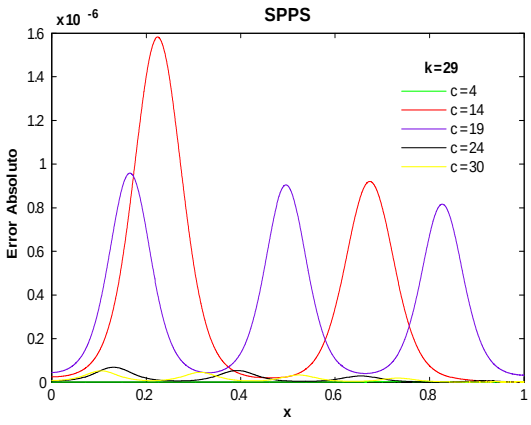


Fig.387. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=29, c=4,14,19,24,30 considerando 40 potencias y 5000 puntos. Cputime = 265.99 s.

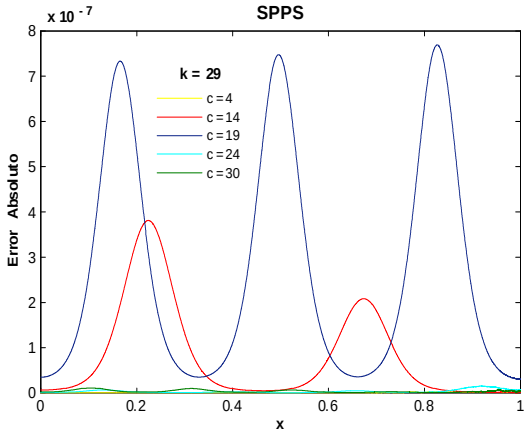


Fig.388. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=29, c=4,14,19,24,30 considerando 40 potencias y 7000 puntos. Cputime = 367.46 s.

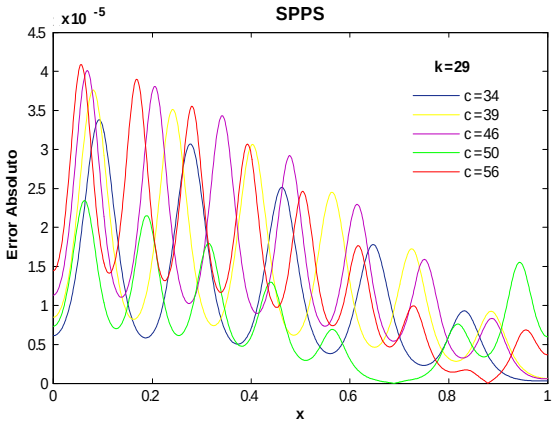


Fig.389. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=29, c=34,39,46,50,56 considerando 40 potencias y 1000 puntos. Cputime = 55.17 s.

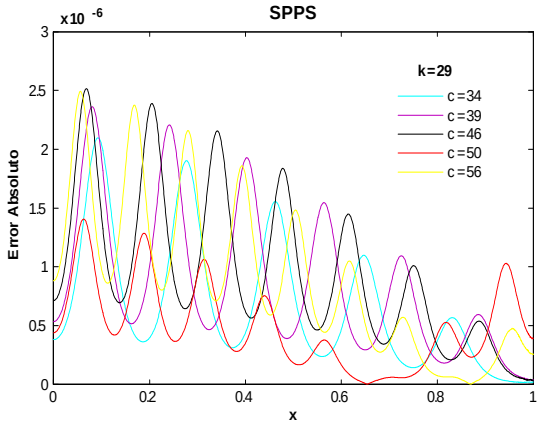


Fig.390. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=29, c=34,39,46,50,56 considerando 40 potencias y 2000 puntos. Cputime = 108.04 s.

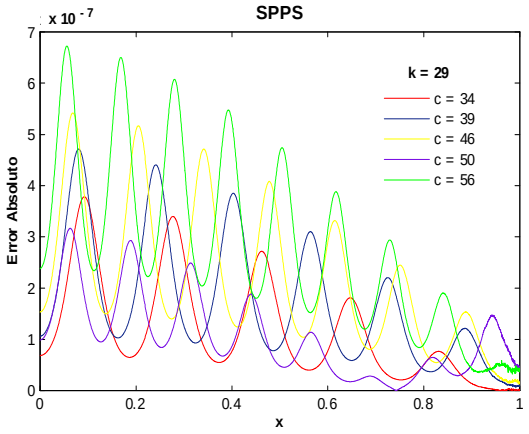


Fig.391. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=29, c=34,39,46,50,56 considerando 40 potencias y 3000 puntos. Cputime = 160.75 s.

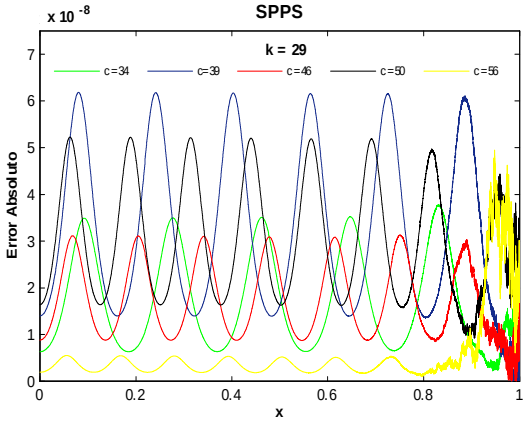


Fig.392. Error Absoluto de la solución SPPS para el ejemplo 11 con los valores de los parámetros k=29, c=34,39,46,50,56 considerando 40 potencias y 15000 puntos. Cputime = 781.99s.

Las figuras anteriores muestran que el orden del error mejora al incrementar la cantidad de puntos. Al aumentar los parámetros c y k el orden empeora, es por eso que es útil considerar más potencias a medida que crecen los parámetros. Para $k = 4$ y $c = 19$ el orden obtenido considerando 15 potencias y 1000 puntos fue de -8, mientras que al tomar 8000 puntos y 15 potencias el orden fue de -12, ver figs. 370-373. Para $k = 4$ y $c = 50$ el orden obtenido con 30 potencias y 1000 puntos fue de -7, mientras que con 5000 puntos y 30 potencias se obtuvo un orden de error de -10, ver figs. 374-377. Al considerar $k = 17$ y $c = 19$ tomando 30 potencias los órdenes obtenidos con 1000 y 5000 puntos fueron de -6 y -9, respectivamente, ver figs.378-381. Tomando a $k = 29$ y $c = 34$ el orden obtenido con 1000 puntos y 40 potencias fue de -5, al considerar 15000 puntos y 40 potencias el orden fue de -8, siendo éste último caso él que tomó más tiempo en calcular la solución y graficar el error, 781.99 s.

Ejemplo 12

Solución Exacta

Considérese el problema con valores en la frontera

$$\begin{aligned} y'' + c\left[\frac{1}{2} + \left(\frac{1}{2} - c\right)\tan^2\left(\frac{x}{2}\right)\right] y &= 0 \\ y'(a) &= \alpha \\ y'(b) &= \beta \end{aligned} \tag{84}$$

donde $c \neq 0, \alpha, \beta, a, b \in \mathbb{R}$, $a < b$.
Sabemos que la solución general de la ecuación diferencial en (84) es $y_g = c_1 \cos^{2c}\left(\frac{x}{2}\right) + c_2 F(x)\cos^{2c}\left(\frac{x}{2}\right)$ donde c_1 y c_2 son constantes arbitrarias y $F(x) = \int \frac{dx}{\cos^{4c}\left(\frac{x}{2}\right)}$. Entonces se debe cumplir

$$\begin{aligned} y_g'(a) &= c_1\left[-c\cos^{2c-1}\left(\frac{a}{2}\right)\sin\left(\frac{a}{2}\right)\right] + c_2\left[\frac{1}{\cos^{2c}\left(\frac{a}{2}\right)} - cF(a)\cos^{2c-1}\left(\frac{a}{2}\right)\sin\left(\frac{a}{2}\right)\right] = \alpha \\ y_g'(b) &= c_1\left[-c\cos^{2c-1}\left(\frac{b}{2}\right)\sin\left(\frac{b}{2}\right)\right] + c_2\left[\frac{1}{\cos^{2c}\left(\frac{b}{2}\right)} - cF(b)\cos^{2c-1}\left(\frac{b}{2}\right)\sin\left(\frac{b}{2}\right)\right] = \beta. \end{aligned} \tag{85}$$

Por los Teoremas 4 y 5, el problema no homogéneo (84) tiene solución única si y sólo si

$$\begin{aligned} \Delta &= \begin{vmatrix} -c\cos^{2c-1}\left(\frac{a}{2}\right)\sin\left(\frac{a}{2}\right) & \frac{1}{\cos^{2c}\left(\frac{a}{2}\right)} - cF(a)\cos^{2c-1}\left(\frac{a}{2}\right)\sin\left(\frac{a}{2}\right) \\ -c\cos^{2c-1}\left(\frac{b}{2}\right)\sin\left(\frac{b}{2}\right) & \frac{1}{\cos^{2c}\left(\frac{b}{2}\right)} - cF(b)\cos^{2c-1}\left(\frac{b}{2}\right)\sin\left(\frac{b}{2}\right) \end{vmatrix} \\ &= \frac{c\cos^{2c-1}\left(\frac{b}{2}\right)\sin\left(\frac{b}{2}\right)}{\cos^{2c}\left(\frac{a}{2}\right)} - \frac{c\cos^{2c-1}\left(\frac{a}{2}\right)\sin\left(\frac{a}{2}\right)}{\cos^{2c}\left(\frac{b}{2}\right)} \\ &\quad + c^2\cos^{2c-1}\left(\frac{a}{2}\right)\cos^{2c-1}\left(\frac{b}{2}\right)\sin\left(\frac{a}{2}\right)\sin\left(\frac{b}{2}\right)[F(b) - F(a)] \neq 0 \end{aligned} \tag{86}$$

Supongamos se cumple (86). Resolviendo el sistema (85) mediante la regla de Cramer:

$$\begin{aligned} c_1 &= \frac{\begin{vmatrix} \alpha & \frac{1}{\cos^{2c}\left(\frac{a}{2}\right)} - cF(a)\cos^{2c-1}\left(\frac{a}{2}\right)\sin\left(\frac{a}{2}\right) \\ \beta & \frac{1}{\cos^{2c}\left(\frac{b}{2}\right)} - cF(b)\cos^{2c-1}\left(\frac{b}{2}\right)\sin\left(\frac{b}{2}\right) \end{vmatrix}}{\Delta} \\ &= \frac{\alpha\left[\frac{1}{\cos^{2c}\left(\frac{b}{2}\right)} - cF(b)\cos^{2c-1}\left(\frac{b}{2}\right)\sin\left(\frac{b}{2}\right)\right] - \beta\left[\frac{1}{\cos^{2c}\left(\frac{a}{2}\right)} - cF(a)\cos^{2c-1}\left(\frac{a}{2}\right)\sin\left(\frac{a}{2}\right)\right]}{\Delta} \\ c_2 &= \frac{\begin{vmatrix} -c\cos^{2c-1}\left(\frac{a}{2}\right)\sin\left(\frac{a}{2}\right) & \alpha \\ -c\cos^{2c-1}\left(\frac{b}{2}\right)\sin\left(\frac{b}{2}\right) & \beta \end{vmatrix}}{\Delta} = \frac{\alpha c\cos^{2c-1}\left(\frac{b}{2}\right)\sin\left(\frac{b}{2}\right) - \beta c\cos^{2c-1}\left(\frac{a}{2}\right)\sin\left(\frac{a}{2}\right)}{\Delta} \end{aligned} \tag{87}$$

De donde se concluye que la solución de (84) es:

$$y_p = c_1\cos^{2c}\left(\frac{x}{2}\right) + c_2 F(x)\cos^{2c}\left(\frac{x}{2}\right)$$

donde $F(x) = \int \frac{dx}{\cos^{4c}\left(\frac{x}{2}\right)}$ y c_1, c_2 están dadas en (87).

Soluciones Numéricas

Las siguientes gráficas presentan algunas soluciones exactas para valores de $c = 3, 5, 8, 11, 13, 19, 21$ y $\alpha = -1, \beta = 1, a = 0, b = 1$.

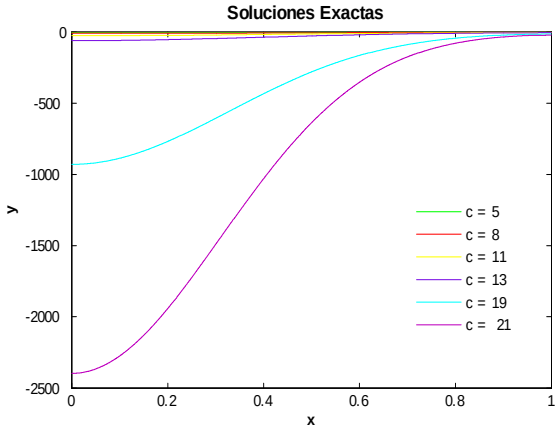


Fig. 393

A continuación se muestran las gráficas de los errores absoluto y relativo para los valores de los parámetros $\alpha = -1, \beta = 1, a = 0, b = 1, c = 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27$. Aquí sólo se presentan los menores órdenes obtenidos, pues calculando ambos errores con Bvp4c y sus valores por defecto se obtuvieron errores muy grandes (del orden de 1e1 y mayores), lo que quiere decir que la solución calculada de esta manera no representa una aproximación a la solución exacta.

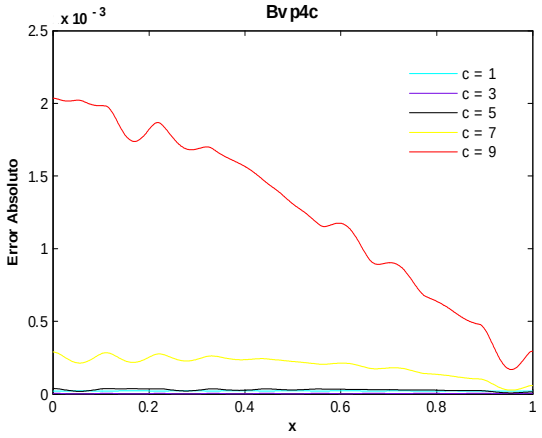


Fig.394. Error Absoluto de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 9.43 s.

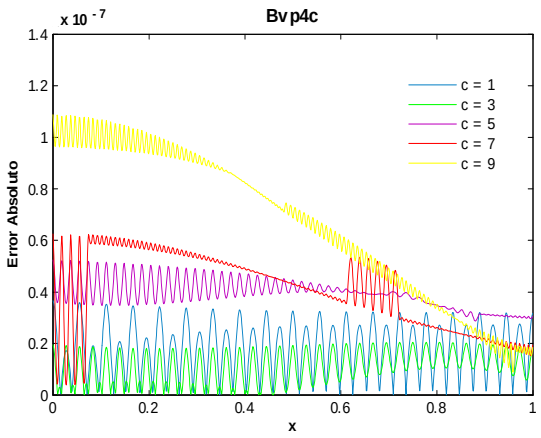


Fig.395. Error Absoluto de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-6 y relativa de 1e-7.
Cputime = 9.81 s.

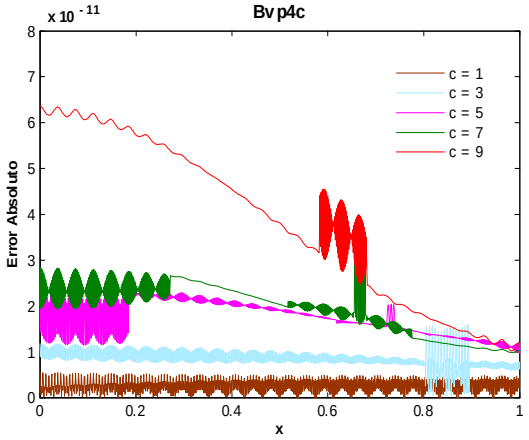


Fig.396. Error Absoluto de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-9 y relativa de 1e-10.
Cputime = 10.72 s.

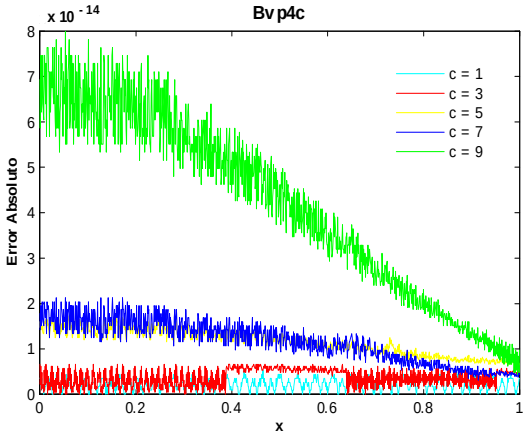


Fig.397. Error Absoluto de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-12 y relativa de 1e-13.
Cputime = 27.61 s.

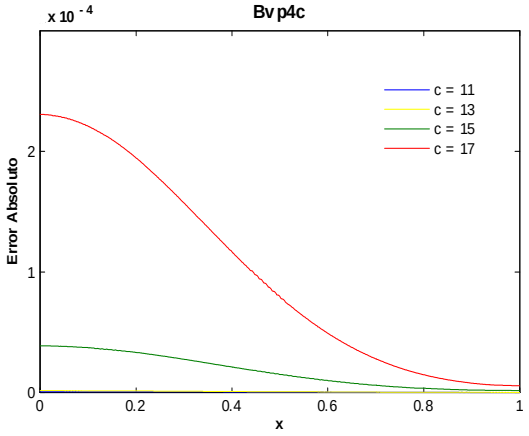


Fig.398. Error Absoluto de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=11,13,15,17 y tolerancias absoluta de 1e-6 y relativa de 1e-7.
Cputime = 58.29 s.

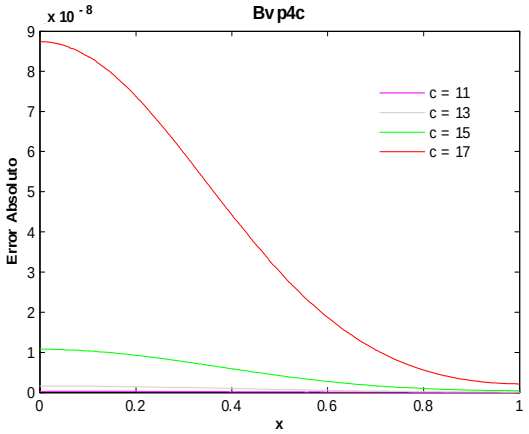


Fig.399. Error Absoluto de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=11,13,15,17 y tolerancias absoluta de 1e-9 y relativa de 1e-10.
Cputime = 61.52 s.

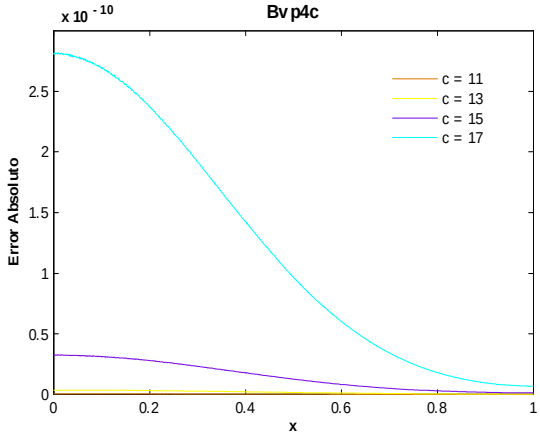


Fig.400. Error Absoluto de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=11,13,15,17 y tolerancias absoluta de 1e-12 y relativa de 1e-13.
Cputime = 79.41 s.

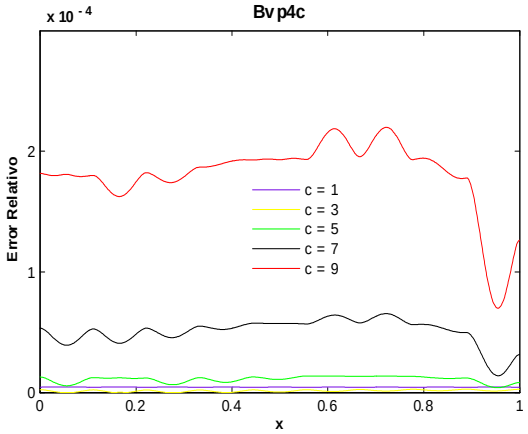


Fig.401. Error Relativo de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 9.45 s

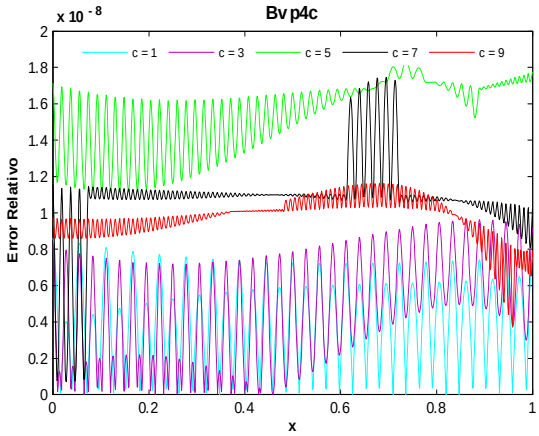


Fig.402. Error Relativo de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-6 y relativa de 1e-7.
Cputime = 9.86 s

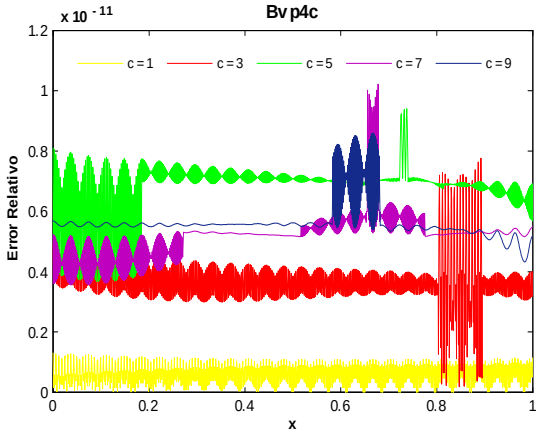


Fig.403. Error Relativo de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-9 y relativa de 1e-10.
Cputime = 10.87 s

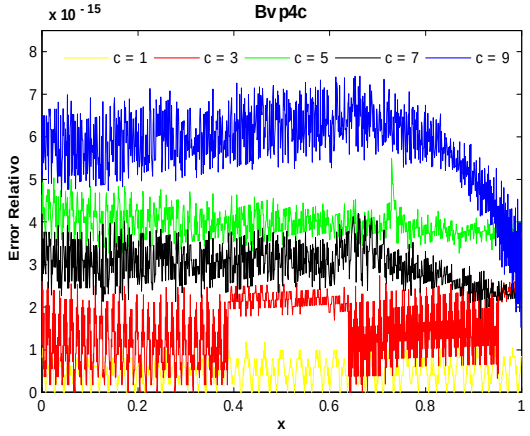


Fig.404. Error Relativo de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=1,3,5,7,9 y tolerancias absoluta de 1e-12 y relativa de 1e-13.
Cputime = 27.05 s

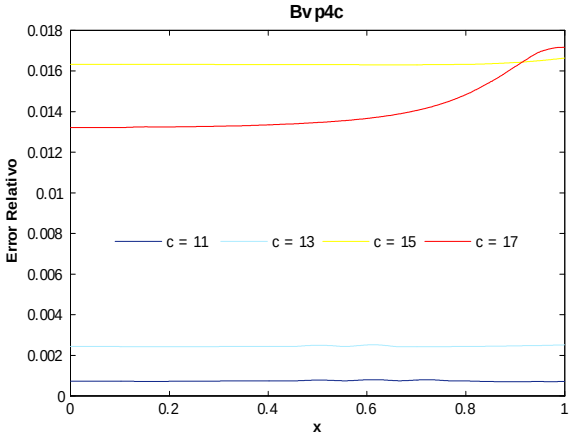


Fig.405. Error Relativo de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=11,13,15,17 y tolerancias absoluta de 1e-6 y relativa de 1e-3.
Cputime = 54.79 s

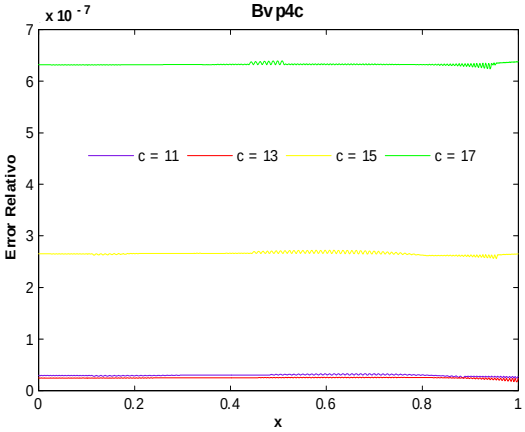


Fig.406. Error Relativo de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=11,13,15,17 y tolerancias absoluta de 1e-6 y relativa de 1e-7.
Cputime = 63.1 s.

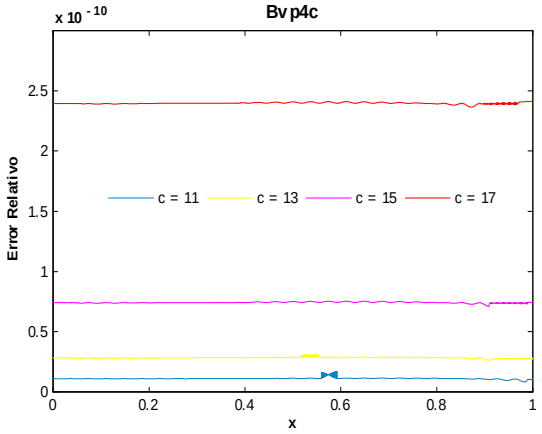


Fig.407. Error Relativo de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=11,13,15,17 y tolerancias absoluta de 1e-9 y relativa de 1e-10.
Cputime = 58.02 s

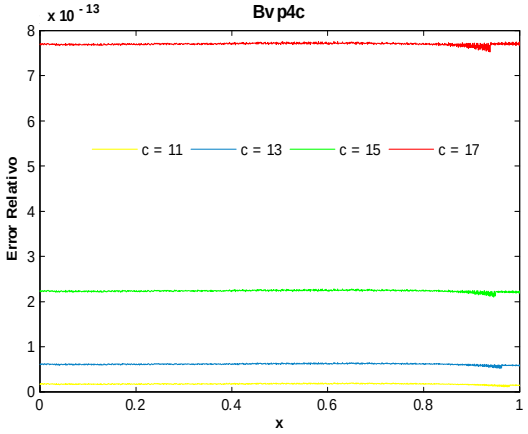


Fig.408. Error Relativo de la solución Bvp4c para el ejemplo 12 con los valores del parámetro c=11,13,15,17 y tolerancias absoluta de 1e-12 y relativa de 1e-13.
Cputime = 75.5 s

Las anteriores figuras muestran que al incrementar el valor del parámetro c el error incrementa. En cuanto al error absoluto, para $c = 9$ (ver fig. 397) el mejor orden de error se obtuvo con los valores de las tolerancias $AbsTol = 1e-12$ y $\mathrm{Re}lTol = 1e-13$ el cuál fue de -14, con valores menores de tolerancias el orden se preserva. El mejor orden obtenido para $c = 17$ (ver fig. 400) fue de -10 con tolerancias $AbsTol = 1e-12$ y $\mathrm{Re}lTol = 1e-13$, al considerar valores menores en las tolerancias el orden incrementa a -9. Para el caso del error relativo el mejor orden obtenido para $c = 9$ (ver fig. 404) fue de -15, con los valores de tolerancias anteriores, con valores menores se obtiene el mismo orden. Con $c = 17$ y los valores de tolerancias antes mencionados (ver fig. 408) el orden fue de -13, considerando valores menores en tolerancias el orden incrementa a -11.

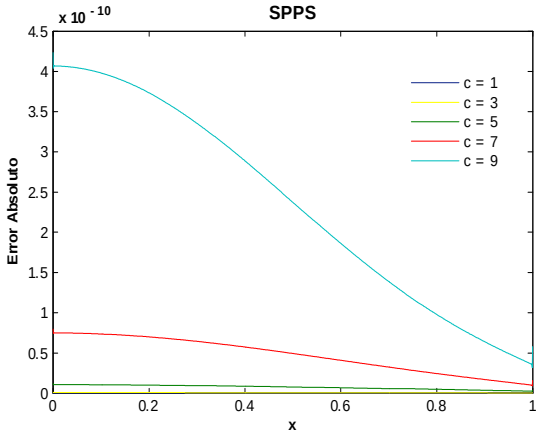


Fig.409. Error Absoluto de la solución SPPS para el ejemplo 12 con los valores de los parámetros c=1,3,5,7,9 considerando 15 potencias y 1000 puntos.
Cputime = 29.68 s.

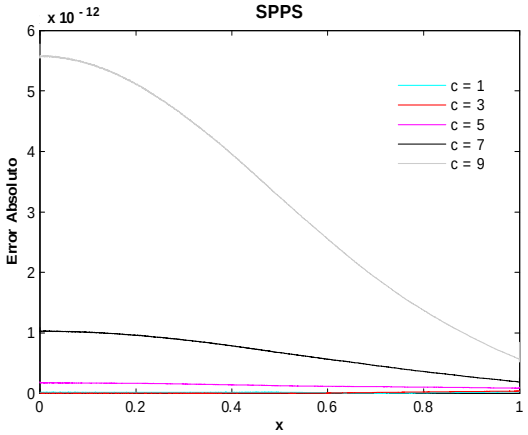


Fig.410. Error Absoluto de la solución SPPS para el ejemplo 12 con los valores de los parámetros c=1,3,5,7,9 considerando 15 potencias y 3000 puntos.
Cputime = 67.93 s.

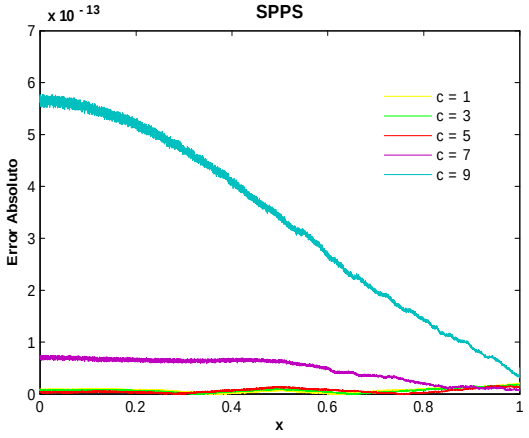


Fig.411. Error Absoluto de la solución SPPS para el ejemplo 12 con los valores de los parámetros c=1,3,5,7,9 considerando 15 potencias y 6000 puntos.
Cputime = 127.92 s.

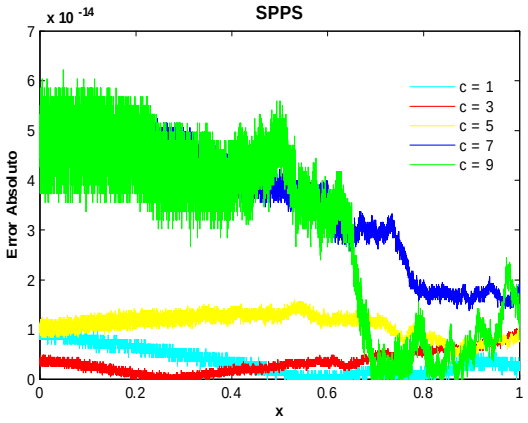


Fig.412. Error Absoluto de la solución SPPS para el ejemplo 12 con los valores de los parámetros c=1,3,5,7,9 considerando 15 potencias y 8180 puntos.
Cputime = 171.37 s.

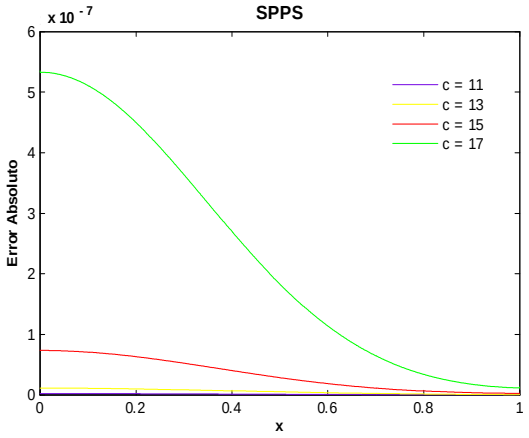


Fig.413. Error Absoluto de la solución SPPS para el ejemplo 12 con los valores de los parámetros c=11,13,15,17 considerando 15 potencias y 1000 puntos.
Cputime = 69.92 s.

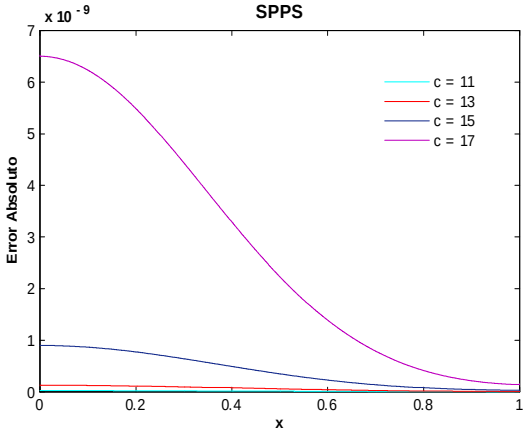


Fig.414. Error Absoluto de la solución SPPS para el ejemplo 12 con los valores del parámetro c=11,13,15,17 considerando 15 potencias y 3000 puntos.
Cputime = 101.85 s.

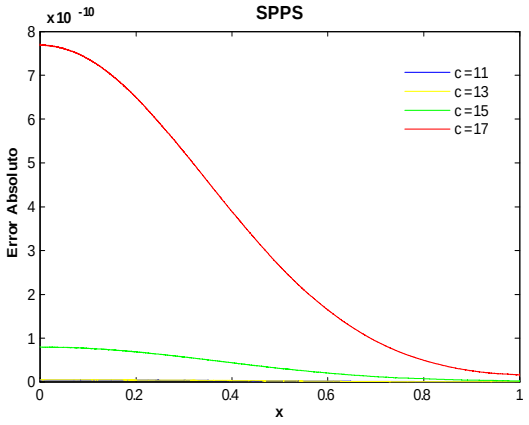


Fig.415. Error Absoluto de la solución SPPS para el ejemplo 12 con los valores del parámetro c=11,13,15,17 considerando 15 potencias y 5000 puntos.
Cputime = 131.2 s.

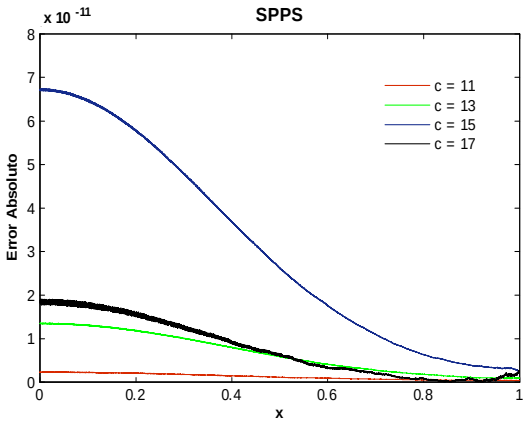


Fig.416. Error Absoluto de la solución SPPS para el ejemplo 12 con los valores del parámetro c=11,13,15,17 considerando 15 potencias y 9300 puntos.
Cputime = 194.99 s.

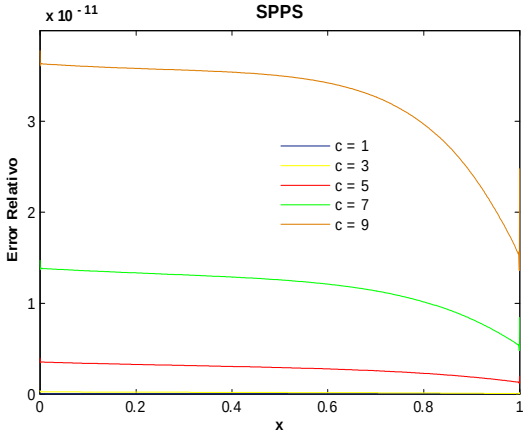


Fig.417. Error Relativo de la solución SPPS para el ejemplo 12 con los valores de parámetroc=1,3,5,7,9 considerando 15 potencias y 1000 puntos.
Cputime = 29.45 s.

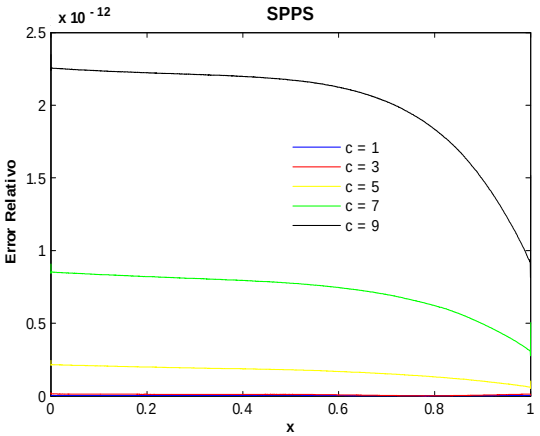


Fig.418. Error Relativo de la solución SPPS para el ejemplo 12 con los valores de parámetroc=1,3,5,7,9 considerando 15 potencias y 2000 puntos.
Cputime = 48.73 s.

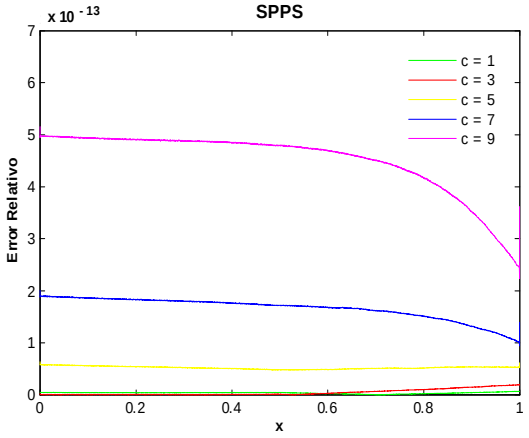


Fig.419. Error Relativo de la solución SPPS para el ejemplo 12 con los valores de parámetroc=1,3,5,7,9 considerando 15 potencias y 3000 puntos.
Cputime = 67.83 s.

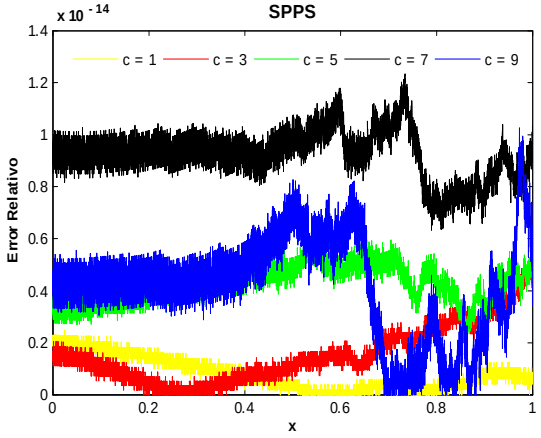


Fig.420. Error Relativo de la solución SPPS para el ejemplo 12 con los valores de parámetroc=1,3,5,7,9 considerando 15 potencias y 8180 puntos.
Cputime = 165.44s.

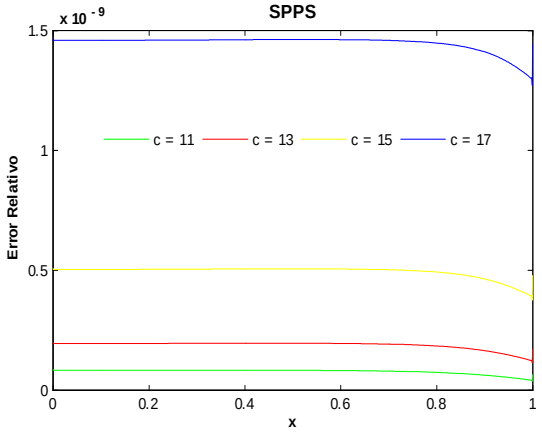


Fig.421. Error Relativo de la solución SPPS para el ejemplo 12 con los valores del parámetros c=11,13,15,17 considerando 15 potencias y 1000 puntos.
Cputime = 70.79 s.

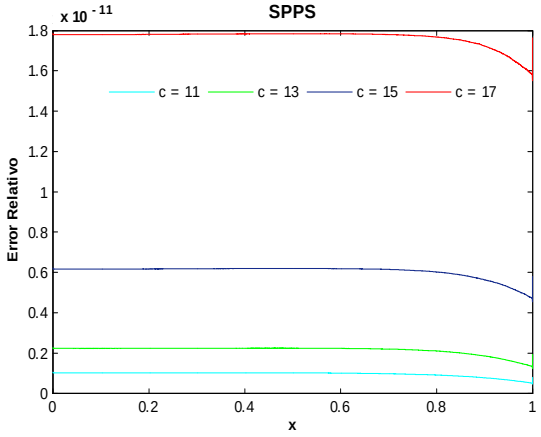


Fig.422. Error Relativo de la solución SPPS para el ejemplo 12 con los valores del parámetros c=11,13,15,17 considerando 15 potencias y 3000 puntos.
Cputime = 100.93 s.

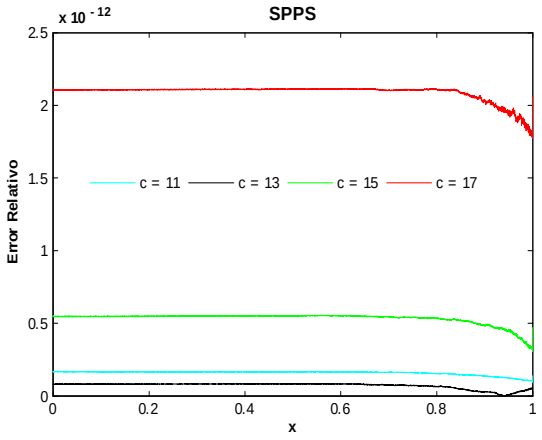


Fig.423. Error Relativo de la solución SPPS para el ejemplo 12 con los valores del parámetros c=11,13,15,17 considerando 15 potencias y 5000 puntos.
Cputime = 136.39 s.

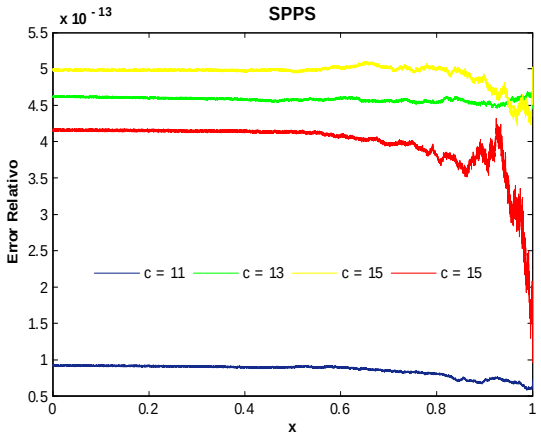


Fig.424. Error Relativo de la solución SPPS para el ejemplo 12 con los valores del parámetros c=11,13,15,17 considerando 15 potencias y 7000 puntos.
Cputime = 162.69 s.

En este ejemplo el mejor orden de los errores absoluto y relativo calculado con ayuda de la solución SPPS para $c = 1, 3, 5, 7, 9$ fue de -14 considerando 8180 puntos y 15 potencias, ver figs. 412 y 420. Para el error absoluto y $c = 11, 13, 15, 17$ el mejor orden calculado fue de -11 utilizando 15 potencias y 9300 puntos. Para el caso del error relativo y $c = 11, 13, 15, 17$ el mejor orden obtenido fue de -13 con 15 potencias y 7000 puntos, ver figs. 416 y 424.

9. Conclusiones y recomendaciones para trabajos futuros

En el trabajo se implementó, se estudió y se probó numéricamente un método propuesto recientemente para la solución de las ecuaciones diferenciales lineales de segundo orden con coeficientes variables llamado el método SPPS (spectral parameter power series). Ofreciendo una representación analítica de la solución, el método SPPS al mismo tiempo proporciona una poderosa y competitiva herramienta para la solución numérica de problemas de Cauchy, problemas con valores en la frontera así como problemas espectrales relacionados con la ecuación diferencial. Diferentes aspectos teóricos y numéricos del método han sido estudiados en varias publicaciones en los últimos tres años (vea la bibliografía al final de la tesis). La contribución del presente trabajo consiste en la aplicación del método SPPS a los problemas con valores en la frontera y su comparación con las mejores técnicas numéricas existentes incorporadas en el paquete estándar de Matlab llamado bvp4c. El aspecto numérico del método SPPS en aplicación a problemas con valores en la frontera no ha sido estudiado en los trabajos anteriores y se presenta por primera vez en el presente documento. Se realizaron decenas de experimentos numéricos y en general se pudo concluir que el método SPPS representa una técnica 1) relativamente sencilla en el manejo, aprendizaje y programación, 2) precisa, con un buen control de error de cálculo y las estimaciones a-priori disponibles, 3) robusta (el método no se desploma repentinamente debido a pequeños cambios en los parámetros del mismo), 4) rápidamente convergente (con las estimaciones teóricas disponibles de la convergencia), 5) con varias mejorías adicionales disponibles las cuales no se implementaron en el presente trabajo y sin duda permitirían obtener resultados aún mejores.

En general como se demostró en el presente trabajo el método SPPS es comparable en su eficiencia numérica con bvp4c. Como lo prevén las estimaciones arriba mencionadas sin las herramientas complementarias mencionadas en el punto 5) del párrafo anterior el método funciona mejor en intervalos de longitud más corta, aunque la misma observación es válida también en el caso de la rutina bvp4c. El método converge tan rápido que la precisión comparable con la precisión de máquina es alcanzable con relativamente pocos términos. La recomendación general derivada de este trabajo es que vale la pena incluir el método SPPS en los programas de estudio del curso de ecuaciones diferenciales debido a su sencillez y al mismo tiempo a la ausencia en dichos programas de métodos de solución de ecuaciones diferenciales lineales con coeficientes variables. Su estudio teórico en clase puede ser complementado con prácticas numéricas que agregarían mayor valor a la materia de ecuaciones diferenciales. Otra recomendación para trabajos futuros consiste en implementar las técnicas adicionales como la división del intervalo de interés y la implementación del método en los subintervalos respectivos las cuales permitirán aumentar la precisión del método y convertirlo en una de las mejores técnicas numéricas existente.

10. Bibliografía

[1] Agarwal R., O'Regan D., *An Introduction to ordinary Differential Equations*, Springer, 2008.

[2] Bartle Robert G., Sherbert Donald R., *Introduction to Real Analysis*, 3a ed., John Wiley & Sons, 2000.

[3] E. John Osborn, L. Ronald Lipsman, M. Jonathan Rosenberg, R. Brian Hunt, *Differential Equations with Matlab*. Second Edition. A John Wiley & Sons, Inc., Publication, 2005.

[4] Howard P., *Solving ODE in Matlab*, 2007.

[5] Khmelnytskaya V. K., Kravchenko V V, Castillo P R, Oviedo G H, 2009, *Efficient calculation of the reflectance and transmittance of finite inhomogeneous layers*, Pure Appl. Opt.

[6] Khmelnytskaya V. K. , H C Rosu., *An amplitude-phase (Ermakov–Lewis) approach for the Jackiw–Pi model of bilayer graphene*. J of Phys. A: Mathematical and Theoretical, v. 42, No. 4, 2009, 042004.

[7] Khmelnytskaya V. K. , H. C. Rosu ,*Spectral parameter power series representation for Hill's discriminant*, Annals of Physics,2010 ;325(11): 2512-2521(10pp).

[8] Khmelnytskaya V. K. , H.C. Rosu, A. González, *Periodic Sturm–Liouville problems related to two Riccati equations of constant coefficients*. Annals of Physics 2010, v. 33, issue 4, 469-472.

[9] Khmelnytskaya V. K. , T. V. Torchynska ,*Reconstruction of potentials in quantum dots and other small symmetric structures.* Mathematical Methods in the Applied Sciences, 2010, v. 33, issue 4, 469-472.

[10] Kravchenko V V 2008 *Complex Variables and Elliptic Equations* 53 775

[11] Kravchenko V V, Porter M 2010 *Spectral parameter power series for Sturm-Liouville problems*, Math. Meth. Appl. Sci.v. 33, issue 4, 459-468

[12] Kravchenko V.V., *Applied pseudoanalytic function theory*. Birkhäuser, Series: Frontiers in Mathematics, 2009.

[13] *Partial Differential Equations and Boundary-value Problems With Applications*, Pinsky Mark A. , 3a ed., AMS, 2011.

[14] Shampine Lawrence F., Kierzenka Jacek, Reichelt Mark W., *Solving Boundary Value Problems for Ordinary Differential Equations in Matlab with bvp4c*, 2000.

[15] Simmons George F., *Differential Equations with Applications and Historical Notes*, 2a ed., United States, Mc Graw Hill, 1991.

[16] Spivak Michael, *Cálculo Infinitesimal*, 2a ed., Reverté, 1992.

[17] Stanoyevitch Alexander , *Introduction to numerical ordinary and partial differential equations using Matlab*. A John Wiley & Sons, Inc., Publication, 2005.

[18] Polyanin A.D., Zaitsev V.E. , *Handbook of exact solutions for ordinary Differential Equations*, CRC, 2003.

[19] V. S. Vladimirov *Equations of Mathematical Physics* MARCEL DEKKER, INC., New York 1971

11. Apéndice A

A continuación se presenta el código del programa que calcula las soluciones exacta y la obtenida con bvp4c correspondientes al ejemplo 1 en el intervalo $[0, \sqrt{2}]$ para los valores de $c = 9, 10, 11, 12, 13$ y grafica el error absoluto.

```
function Ej1
 tic
 clc                          %Limpia la ventana de comandos
 clear all                     %Limpia el espacio de trabajo y libera memoria del sistema
 global alpha                    %Variable global usada en la función bcbvp1
 global beta                    %Variable global usada en la función bcbvp1
 global c                        %Variable global usada en la función bvp1
 %Datos del problema
 alpha = 0; beta = 1; a = 0; b = sqrt(2); xpts = 1000;
 %Obtiene la malla inicial y las estimaciones inciales
 solinit = bvpinit(linspace(a,b,10),[1,0]);
 %División del intervalo [a,b] en xpts-1 subintervalos
 x = linspace(a,b,xpts);
 %Establece la exactitud y el No de puntos malla
 options = bvpset('AbsTol',1e-12,'RelTol',1e-12,'NMax',5000);

 for c = [9 10 11 12 13]
  %---------------Solución Exacta-------------------
  %Determinante del Sistema
  det = cos(c*a)*sin(c*b)-sin(c*a)*cos(c*b);
  %------------Coeficientes de la solución exacta
  c1 = (alpha*sin(c*b)-beta*sin(c*a))/det;
  c2 = (beta*cos(c*a)-alpha*cos(c*b))/det;
  %Evalúa la solución exacta en cada punto de x
  ExSol = c1*cos(c*x) + c2*sin(c*x);
  %---------------Solución Bvp4c----------------
  %Calcula la solución del problema con bvp4c
  sol = bvp4c(@bvp1,@bcbvp1,solinit,options);
  %Evalúa la solución obtenida por bvp4c en cada punto de x
  y = deval(sol,x);

  %Calcula el error absoluto y relativo
  AbsError = abs(ExSol-y(1,:));
  RelError = AbsError./abs(ExSol);
  %Grafica el Error absoluto
  plot(x,AbsError);
  hold on;
 end
 toc
end
%-------------------------------------------------------------------------------
function dy = bvp1(x,y)
global c
dy = [y(2) -(c^2)*y(1)];        %Sistema de ecuaciones diferenciales de 1er orden
end
%-------------------------------------------------------------------------------
function bc = bcbvp1( ya,yb )
global alpha beta
bc = [ya(1)-alpha yb(1)-beta];    %Condiciones de frontera
```

```
end
```

El siguiente código calcula las soluciones bvp4c y exacta del problema presentado en el ejemplo 11 y grafica el error absoluto para los valores de $k = 29$ y $c = 4, 14, 19, 24, 30$.

```
function Ej11
  tic
  clc                  %Limpia la ventana de comandos
  clear all              %Limpia el espacio de trabajo y libera memoria del sistema
  syms fun t                 %Define a las variables fun y t como simbólicas
  global alpha              %Variable global usada en la función bcbvp2
  global beta              %Variable global usada en la función bcbvp2
  global c k              %Variables globales usadas en la función bvp2
  %Datos del problema con valores en la frontera
  a = 0; b = 1; k = 29; alpha = 1; beta = 1; xpts = 3000;
  %Crea la estimación inicial para el problema
  solinit = bvpinit(linspace(a,b,10),[1,0]);
  %Obtiene un vector x con xpts puntos igualmente distribuidos desde a hasta b
  x = linspace(a,b,xpts);
  %Establece la exactitud y el No de puntos malla
  options = bvpset('AbsTol',1e-6,'RelTol',1e-3,'NMax',5000);
  %Establece el orden del spline
  SplineOrder = 3;
  for c = [4 14 19 24 30]
    %----------------Solución Exacta----------------------------
    %Función integral de la solución exacta, se calcula numéricamente
    I = fnval(fnint(spapi(SplineOrder,x,exp(2*k*cos(c*x)/c))),x);
    y1 = exp(-k*cos(c*t)/c);
    %Determinante del sistema
    det = (k^2)*sin(c*a)*sin(c*b)*exp((-k/c)*(cos(c*a)+cos(c*b)))*(I(length(x))-I(1))
        +k*sin(c*a)*exp((k/c)*(cos(c*b)-cos(c*a)))-k*sin(c*b)*exp((k/c)*(cos(c*a)-cos(c*b)));
    %Coeficientes de la solución exacta
    c1 = (alpha*(k*I(length(x))*sin(c*b)*subs(y1,t,b)+exp((k/c)*cos(c*b)))
          -beta*(k*I(1)*sin(c*a)*subs(y1,t,a)+exp((k/c)*cos(c*a))))/det;
    c2 = (beta*k*sin(c*a)*subs(y1,t,a)-alpha*k*sin(c*b)*subs(y1,t,b))/det;
    %Evalúa la solución exacta en cada valor de x
    ExSol = subs(y1,t,x).*(c1+c2*I);
    %----------------------Solución Bvp4c----------------------
    %Calcula la solución del problema con bvp4c
    sol = bvp4c(@bvp4,@bcbvp4,solinit,options);
    %Evalúa la solución obtenida por bvp4c en cada punto de x
    y = deval(sol,x);
    %Se calculan los errores absoluto y relativo
    AbsError = abs(ExSol-y(1,:));
    RelError = AbsError./abs(ExSol);
    %Grafica el error absoluto
    plot(x,AbsError);
    hold on;
  end
  toc
  end
  %-------------------------------------------------------------------------------------
  function dy = bvp4(x,y)
  global c k
```

```
%Sistema de ecuaciones diferenciales de 1er orden
dy = [y(2) k*(k*sin(c*x)^2+c*cos(c*x))*y(1)];
end
%-------------------------------------------------------------------------------
function bc = bcbvp4(ya,yb)
global alpha beta
%Condiciones de frontera
bc = [ya(2)-alpha yb(2)-beta];
end
```

12. Apéndice B

A continuación se presenta el código que calcula la solución SPPS y la exacta para el ejemplo 2 en el intervalo $[1, 1/2]$ para los valores de $c = 3, 5, 7, 13, 15$.

```
tic
clear all; clc
hold all
format long          %Formato de impresión de 15 dígitos decimales
%Número de potencias a calcular y la cantidad de puntos
N = 30; xpts = 2000;
%Datos del problema con valores en la Frontera
a = 0; b = 1/2; alpha = 1; beta = 3;
%Se divide [a, b/2] en subintervalos de igual longitud, los nodos se almacenan en x1
x1 = linspace(a,b/2,xpts);
%Se divide [b/2, b] en subintervalos de igual longitud, los nodos se almacenan en x1
x2 = linspace(b/2,b,xpts);
%Unión de [a, b/2] y [b/2, b]
usedPoints = [x1 x2(2:xpts)];
%Se establece el orden del spline
SplineOrder = 3;
%Se inicializa el valor de p y q2
p = ones(1,length(usedPoints));
q2 = ones(1,length(usedPoints));
for c = [3 5 7 13 15]
    %-----------Solución Exacta-------------
    %Determinante del sistema
    det = 2*sinh(c*b/2)*(tan(c*b)*sin(c*b/2)+cos(c*b/2))
           +2*cosh(c*b/2)*(tan(c*b)*cos(c*b/2)-sin(c*b/2))
    %------------Solución Exacta-Intervalo-[0,b/2)-------------------
    %Coeficientes de la solución
    d2 = ((-beta/cos(c*b))+alpha*exp(c*b/2)*tan(c*b)*(sin(c*b/2)+cos(c*b/2))
               +alpha*exp(c*b/2)*(cos(c*b/2)-sin(c*b/2)))/det;
    d1 = alpha-d2;
    %Solución exacta
    ExSol1 = d1*exp(c*x1)+d2*exp(-c*x1);
    %------------Solución Exacta-Intervalo-(b/2,b]-------------------
    %Coeficientes de la solución
    k2 = ((2*beta/cos(c*b))*(sin(c*b/2)*sinh(c*b/2) + cos(c*b/2)*cosh(c*b/2))
               +2*alpha*exp(c*b/2)*(sinh(c*b/2)-cosh(c*b/2)))/det;
    k1 = (beta-k2*sin(c*b))/cos(c*b);
    %Solución exacta
    ExSol2 = k1*cos(c*x2) + k2*sin(c*x2);
    %---------Solución Exacta-Intervalo-[a,b]
    ExSol = [ExSol1 ExSol2(2:xpts)];
    %--------------------Solución SPPS-------------------
    %Potencial por intervalos
    h1 = -(c^2)*ones(1,xpts-1);
    h2 = 0;
    h3 = (c^2)*ones(1,xpts-1);
    q = [h1 h2 h3];
    q1 = -q;
    %Se obtine estructura con los valores de las integrales Xtilde y X
    [X0,Xtil0] = PowersXnew(q1,q2./(p),2*N,usedPoints, SplineOrder);
    %Soluciones linealmente independientes
```

```
    y1value = ones(1,length(usedPoints));
    y2value = zeros(1,length(usedPoints));
    %Se evalúa Xtilde y X en el intervalo de interés
    for k = 1:2*N
        if rem(k,2) == 0
            Xtil0Val = fnval(Xtil0(k),usedPoints);
            y1value = y1value+Xtil0Val;
        else
            X0Val = fnval(X0(k),usedPoints);
            y2value = y2value+X0Val;
        end
    end
    %Coeficientes para la solución SPPS
    c1 = alpha;
    c2 = (beta-alpha*y1value(numel(usedPoints)))./y2value(numel(usedPoints));
    %Solución SPPS
    Sol = c1*y1value+c2*y2value;
    %Calcula el Error Absoluto
    AbsError = abs(Sol-ExSol);
    %Calcula el Error Relativo
    RelError = AbsError./abs(ExSol);
    %Grafica el Error Absoluto
    plot(usedPoints,AbsError);
    hold on;
end
toc

function [Xp,Xtilp] = PowersXnew(q1,q2,N1,usedPoint, SplOrder)
        %Se define el primer término X^(0)(x) = 1
        X0p = ones(1,length(usedPoint));
        %Se define el primer término X~^(0)(x) = 1
        Xtil0p = ones(1,length(usedPoint));
        %Se obtienen numéricamente X^(0)(x)y X~^(1)(x)
        Xp(1) = fnint(spapi(SplOrder,usedPoint,(X0p.*q2)));
        Xtilp(1) = fnint(spapi(SplOrder,usedPoint,(Xtil0p.*q1)));
        %Ciclo para calcular la potencias 2 a N de X y X~
        for n = 2:N1
            %Se evalúa X~^(n-1) en los puntos usedPoint
            Xtilpval = fnval(Xtilp(n-1),usedPoint);
            %Se evalúa X^(n-1) en los puntos usedPoint
            Xpval = fnval(Xp(n-1),usedPoint);
            %Potencias para n par
            if rem(n,2) == 0
                %Integral X~^(n-1)
                Xtilp(n) = fnint(spapi(SplOrder,usedPoint,(Xtilpval.*q2)));
                %Integral X^(n-1)
                Xp(n) = fnint(spapi(SplOrder,usedPoint,(Xpval.*q1)));
            else %Potencias para n impar
                %Integral X~^(n-1)
                Xtilp(n) = fnint(spapi(SplOrder,usedPoint,(Xtilpval.*q1)));
                %Integral X^(n-1)
                Xp(n) = fnint(spapi(SplOrder,usedPoint,(Xpval.*q2)));
            end
end
```