Applied Mathematical Sciences, Vol. 6, 2012, no. 127, 6311 - 6326

Implementation of Wavelet Solutions to Second Order Differential Equations with Maple

Adefemi Sunmonu

Department of Mathematics and Computer Science
The City University of New York
York College, NY 11451, USA
asunmonu@york.cuny.edu

Abstract

The Haar Wavelet method is used to obtain numerical solutions of one dimensional differential equation with varied conditions prescribed at the boundary: initial, boundary, and mixed boundary conditions. A new approach is introduced to handle mixed boundary conditions that avoid computing the derivatives of the function at the boundaries. The accuracy and simplicity of the wavelet method are demonstrated using differential equations for which exact or series solutions are known.

Mathematics Subject Classification: 65T60, 42C40

Keywords: Haar wavelet, Operational matrix, Dirichlet condition, Boundary condition, Mixed boundary condition, Maple

1 Introduction

Wavelet theory is relatively new and an emerging area in mathematical research. In recent years, wavelets have been applied in different fields of science, engineering, and other areas needing numerical approximations. Different types of wavelets and approximating functions have been used in numerical solution of initial and boundary value problems. Chen and Hsiao [1] popularized the Haar wavelets and exposed some of its advantages. Other researchers like Lepik [3] - [5] have applied Haar wavelets in solving differential equations, nonlinear integro-differential equations and partial differential equations. Researchers who have made notable contributions in this area are too numerous to list here; refer to Phang [2], Lepik [5] and the references cited there for more work in this area.

While many kinds of wavelets are now being employed in various scientific endeavors, the Haar wavelets are the simplest orthonormal wavelet with compact support. Haar wavelets reduce the differential equations to the problem of solving a system of algebraic equations, thus greatly simplifying the problem. It also has the capability of producing very good estimate of the solution even with a few collocation points.

The main motivation of this research is to present a simple and accurate implementation of the Haar wavelets method for the numerical solution of

$$y'' + \alpha_1(t)y' + \alpha_2(t)y = f(t) \tag{1}$$

using the Maple symbolic algebra system. This problem will be solved subject to the following three different initial and boundary conditions:

$$y(0) = y_o,\ y'(0) = y_1, \tag{2}$$

$$y(0) = \beta_o,\ y(1) = \beta_1, \tag{3}$$

$$y(0) = A,\ y'(1) = B,\ \text{and} \tag{4}$$

$$y'(0) = A,\ y(1) = B. \tag{5}$$

The organization of this paper is as follows. Section 2 describes components of the Haar wavelets and their implementation in Maple. In Section 3 the numerical method using Haar wavelets is described for the numerical solution of initial value problems and section 4 discusses the application of the method to boundary value problems. Numerical results for both initial and boundary value problems are given in section 5. Conclusion is given in section 6.

2 Haar Wavelets Method

The Haar wavelet family for $x \in [0, 1)$ is defined as

$$h_i(x) = \begin{cases} 1 & \text{for } x \in [a, b) \\ -1 & \text{for } x \in [b, c) \end{cases} \tag{6}$$

where

$$a = \frac{k}{m}, \qquad b = \frac{k + 0.5}{m}, \qquad c = \frac{k + 1}{m}.$$

In this definition integer $m = 2^j$, $j = 0, 1, \cdots, J$, indicates the level of the wavelet and the integer $k = 0, 1, ..., m - 1$ is the translation parameter. J is the maximum level of resolution. The index i in (6) is computed using the formula $i = m + k + 1$. The minimal value of i is $i = 2$ which is obtained

when $m = 1, k = 0$. The maximal value of i is $i = 2M = 2^{J+1}$. For $i = 1$, the function $h_1(x)$ is defined as

$$h_1(x) = \begin{cases} 1 & \text{for } x \in [0, 1) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Implementation of this function is shown below.

$h_1(\mathbf{t})$

```
h1 := t-> piecewise(0 <= t and t < 1, 1):
```

The Haar wavelet family $h_i(t)$ is implemented using the code

$h_i(\mathbf{t})$

```
hi:=proc(j,k,t)
 local a,b,c,m;  m:=2^(j);
 a := k/m; b := (k+1/2)/m; c := (k+1)/m;
 return piecewise(a <= t and t < b, 1, b <= t and t < c, -1);
end proc:
```

Let us define the collocation points where we will approximate any function as

$$t_n = (n - 0.5)/2M, \text{ for } n = 1, 2, \cdots, 2M. \tag{8}$$

At these collocation points, we discretize the Haar function $h_i(t)$ to obtain a $(2M \times 2M)$ coefficient matrix $\mathbf{H}_{2M\times 2M}$ called Haar Matrix.

To facilitate the computation of $\mathbf{H}$, we define a vector of Haar functions

$$\mathbf{h}_d = [h_0(t) \quad h_1(t) \quad \cdots \quad h_{d-1}(t)]^T \tag{9}$$

where d is the dimension of the vector and T is for transpose. These vectors are now used to compute each column of the Haar matrix as

$$\mathbf{H}_d = [\mathbf{h}_d(t_0) \quad \mathbf{h}_d(t_1) \quad \mathbf{h}_d(t_2) \quad \cdots \quad \mathbf{h}_d(t_{d-1})] \tag{10}$$

where $\frac{i}{d} \le t_i \le \frac{i+1}{d}$.

Equations (8)-(10) are implemented in Maple using the follow code:

$h_d(\mathbf{t})$,$\mathbf{H}$

```
N :=2^J:
hd := Vector(N): H := Matrix(N, N): T := Vector(N):
hd[1] := h1(t):
```

```
for i from 1 to N do T[i] := (i-1/2)/N: end do:

for j from 0 to J-1 do
 m := 2^j:
 for k from 0 to m-1 do
   i := m+k+1:
   hd[i] := hi(j, k, t):
 end do:
end do:
#Now Compute H at the collocation points
for i from 1 to N do
 for j from 1 to N do
   H[i, j] := eval(hd[i], t = T[j]):
 end do:
end do:
```

For verification, print the value of $\mathbf{H}$ for different resolution levels. For instance when $J = 2$ we obtain

$$\mathbf{H}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}.$$

This is in line with the theoretical expectation from [1]. In general, the Haar matrix $\mathbf{H}$ is of the form

$$\mathbf{H}_1 = [1], \qquad \mathbf{H}_d = \begin{bmatrix} \mathbf{H}_{\frac{d}{2}} \otimes [1\ 1] \\ \mathbf{I}_{\frac{d}{2}} \otimes [1\ -1] \end{bmatrix} \tag{11}$$

where $\mathbf{I}_d$ is the identity matrix of dimension d and $\otimes$ is the Kronecker product. Equation (11) allows us to define the Haar matrix at higher resolution recursively using Haar matrices of lower resolutions.

2.1 Integration of Haar Functions

Integration of the Haar functions in equation (6) can be written as

$$p_{i,1}(t) = \int_0^t h_i(x)\, dx \tag{12}$$

and subsequent integration can be recursively defined as

$$p_{i,\nu}(t) = \int_0^t p_{i,\nu-1}(x)\, dx. \tag{13}$$

Using the notations of Chen and Hsiao this integral is written as

$$p_{d,1}(t) = \int_0^t h_d(x)\,dx \approx \mathbf{P}_d \mathbf{H}_d \tag{14}$$

where $\mathbf{P}_d$ is an d-square matrix called an operational matrix of integration whose structure is recursively defined as

$$\mathbf{P}_1 = \left[\frac{1}{2}\right], \qquad \mathbf{P}_d = \begin{bmatrix} \mathbf{P}_{\frac{d}{2}} & -\frac{1}{2d}\mathbf{H}_{\frac{d}{2}} \\ \frac{1}{2d}\mathbf{H}_{\frac{d}{2}}^{-1} & \mathbf{0}_{\frac{d}{2}} \end{bmatrix} \tag{15}$$

where $\mathbf{0}_d$ is a d-dimensional matrix of zeros.

The operational matrix of integration can be generated using the Maple code:

$p_n(\mathbf{t})$

```
pn:=proc(i,n,t)
 if n=1 then return int(hd[i],t) fi:
 return int(pn(i,n-1,t)):
end proc:
```

where the parameter n is the desired number of integrations.

Note that this code computes the left hand side of (14). To obtain the operational matrix $\mathbf{P}_d$ one must multiply the result of this code with $\mathbf{H}^{-1}$.

3 Function Approximation

Any function $f(t) \in \mathcal{L}^2[0,1)$ can be decomposed as

$$f(t) = \sum_{i=0}^{\infty} a_i h_i(t), \qquad t \in [0,1) \tag{16}$$

where $a_i = \langle f(t), h_i(t)\rangle = 2^j \int_0^1 f(t)h_i(t)\,dt$. $\mathbf{A} = [a_0\, a_1\, \cdots\, a_d]^T$ is a vector of the wavelet coefficients.

For computing finite sum, this infinite sum is terminated at $N = 2^J, \quad J \in \mathbb{N}$ so that $f(t)$ is approximated as

$$f(t) = \sum_{i=0}^{N-1} a_i h_i(t), \qquad t \in [0,1). \tag{17}$$

3.1 Approximating Solutions of Differential Equation

To demonstrate the approximation of solutions to differential equations using Haar wavelet series, consider the general second order differential equation

$$y'' + \alpha_1(t)y' + \alpha_2(t)y = f(t), \qquad y(0) = y_o,\ y'(0) = y_1 \tag{18}$$

where $\alpha_1(t)$ and $\alpha_2(t)$ are assumed to be expandable in Haar series. To begin the approximation, we assume that

$$y''(t) \approx \sum_{i=0}^{N-1} a_i h_i(t) \tag{19}$$

Now integrate (19) with respect to t from 0 to t to obtain

$$y'(t) = \sum_{i=0}^{N-1} a_i p_{i,1}(t) + y'(0). \tag{20}$$

Integrate (20) with respect to t from 0 to t to get the approximation

$$y(t) = \sum_{i=0}^{N-1} a_i p_{i,2}(t) + y'(0)t + y(0). \tag{21}$$

Next substitute these expressions into (18) and simplify to obtain

$$\sum_{i=0}^{N-1} a_i \left[h_i(t) + \alpha_1(t)p_{i,1}(t) + \alpha_2(t)p_{i,2}(t)\right] = f(t) - \alpha_1(t)y'(0) - \alpha_2(t)\left[ty'(0) - y(0)\right] \tag{22}$$

The unknown wavelet coefficients a_i are obtained from solving equation (22) after evaluating t at the collocation points. We achieve this in Maple with the code:

Solve System

```
RHS:= t->f(t)-alpha1(t)*y1-alpha2(t)*(t*y1+y0);
#alpha1(t) and alpha2(t) must be defined as functions of t
R:=Vector(N):
TMP:=Matrix(N,N):
A:=Matrix(N,N):

for i from 1 to N do
 R[i] := evalf(RHS(T[i])):
 tmp := alpha1(t)*pn(i,1,t)+alpha2(t)*pn(i,2,t):
 for j from 1 to N do
```

```
   TMP[i,j]:=eval(tmp, t = T[j]):
 od:
od:

#Compute the wavelet coefficients
A := Transpose(LinearSolve(Transpose(H+TMP), R)):

#Now compute the approximate solution
sol := y1*t+y0+sum('A[m0]*pn(m0,2,t)', m0 = 1 .. N):
#Convert to a function of t for easy comparison with
#exact solution
y:=unapply(sol,t):
```

The three lines highlighted in blue are the only three lines of code to change in other to solve (1) with the other boundary conditions.

3.2 Application to Boundary Value Problems

To apply this solution method to boundary value problem, consider the equation:

$$y''(t) + \alpha_1 y'(t) + \alpha_2 y(t) = f(t), \quad \text{with } y(0) = \beta_o,\ y(1) = \beta_1. \tag{23}$$

To approximate the solution to (23) with Haar wavelet, we proceed as in Dirichlet initial condition problem. However, notice that $y'(0)$ is not given, therefore, we have to compute it. To do this substitute $t = 1$ in (21) to get

$$y(1) = \sum_{i=1}^{N-1} a_i p_{2,i}(1) + y'(0) + y(0), \tag{24}$$

or

$$y'(0) = -\sum_{i=1}^{N-1} a_i p_{2,i}(1) - y(0) + y(1). \tag{25}$$

Now substitute this into (21) to obtain the solution

$$\begin{aligned} y(t) &= \sum_{i=0}^{N-1} a_i p_{i,2}(t) + y'(0)t + y(0) && (26)\\ &= \sum_{i=0}^{N-1} a_i p_{i,2}(t) - t\left[\sum_{i=1}^{N-1} a_i p_{2,i}(1) + y(0) - y(1)\right] + y(0) \\ &= \sum_{i=0}^{N-1} a_i \left[p_{i,2}(t) - t p_{2,i}(1)\right] - t(y(0) - y(1)) + y(0). && (27) \end{aligned}$$

To compute the wavelet coefficients, we substitute the expressions (19), (20) and (27) into (23) and rearrange to get

$$\sum_{i=0}^{N-1} a_i \left[h_i(t) + \alpha_1(t)(p_{i,1}(t) - p_{i,2}(1)) + \alpha_2(t)\left[p_{i,2}(t) - tp_{2,i}(1)\right]\right] = R(t) \quad (28)$$

where $R(t) = f(t) - \alpha_1(t)(y(1) - y(0)) - \alpha_2(t)\left[t(y(0) - y(1)) + y(0)\right]$.

This is the equation we now have to solve to obtain the wavelet coefficients.

3.3 Application to Mixed Boundary Condition Problems - Type I

Next, consider (1) with the robin boundary conditions (4). That is, solve

$$y''(t) + \alpha_1 y'(t) + \alpha_2 y(t) = f(t), \quad \text{with } y'(0) = A,\ y(1) = B. \quad (29)$$

Once again, to solve (33), we use results from the previous section. Substituting $t = 1$ in (21) leads to

$$y(0) = -\sum_{i=0}^{N-1} a_i p_{i,2}(1) - y'(0) + y(1). \quad (30)$$

Substitute this into (21) to obtain the approximation

$$y(t) = \sum_{i=0}^{N-1} a_i \left[p_{i,2}(t) - p_{i,2}(1)\right] + y'(0)(t-1) + y(1). \quad (31)$$

Also substitute (30) into (22) to obtain

$$\begin{aligned} &\sum_{i=0}^{N-1} a_i \left[h_i(t) + \alpha_1(t)p_{i,1}(t) + \alpha_2(t)p_{i,2}(t)\right] \\ &= f(t) - \alpha_1(t)y'(0) - \alpha_2(t)\left[ty'(0) - y(0)\right] \end{aligned}$$

$$= f(t) - \alpha_1(t)y'(0) - \alpha_2(t)\left[ty'(0) + \sum_{i=0}^{N-1} a_i p_{i,2}(1) + y'(0) - y(1)\right]$$

So that the equation to solve is given by

$$\begin{aligned} &\sum_{i=0}^{N-1} a_i \left[h_i(t) + \alpha_1(t)p_{i,1}(t) + \alpha_2(t)(p_{i,2}(t) + p_{i,2}(1))\right] = \\ &\quad f(t) - y'(0)\left[\alpha_1(t) + \alpha_2(t)(t+1)\right] + \alpha_2(t)y(1). \end{aligned} \quad (32)$$

3.4 Application to Mixed Boundary Condition Problems - Type II

Finally, we consider (1) with the robin boundary conditions (5). That is, solve

$$y''(t) + \alpha_1 y'(t) + \alpha_2 y(t) = f(t), \quad \text{with } y(0) = A,\ y'(1) = B. \tag{33}$$

Once again, to solve (33), we use results from the previous section. Differentiate (27) to obtain

$$y'(t) = \sum_{i=0}^{N-1} a_i \left[p_{i,1}(t) - p_{2,i}(1)\right] - (y(0) - y(1)). \tag{34}$$

Substitute $t = 1$ into (34) to obtain

$$y'(1) = \sum_{i=0}^{N-1} a_i \left[p_{i,1}(1) - p_{2,i}(1)\right] - (y(0) - y(1)). \tag{35}$$

Solving for $y(1)$, we obtain

$$y(1) = y'(1) - \sum_{i=0}^{N-1} a_i \left[p_{i,1}(1) - p_{2,i}(1)\right] + y(0) \tag{36}$$

Hence, in light of (25),

$$y'(0) = y'(1) - \sum_{i=0}^{N-1} a_i p_{i,1}(1). \tag{37}$$

Substitute this into (21) to obtain the solution

$$y(t) = \sum_{i=0}^{N-1} a_i \left[p_{i,2}(t) - t p_{i,1}(1)\right] + t y'(1) + y(0). \tag{38}$$

Also substitute (37) into (22) to obtain

$$\begin{aligned}
\sum_{i=0}^{N-1} a_i \left[h_i(t) + \alpha_1(t) p_{i,1}(t) + \alpha_2(t) p_{i,2}(t)\right] &= f(t) - \alpha_1(t) y'(0) - \alpha_2(t) \left[t y'(0) - y(0)\right] \\
&= f(t) - \alpha_1(t) \left[y'(1) - \sum_{i=0}^{N-1} a_i p_{i,1}(1)\right] \\
&- \alpha_2(t) \left(t \left[y'(1) - \sum_{i=0}^{N-1} a_i p_{i,1}(1)\right] - y(0)\right)
\end{aligned}$$

So that the equation to solve is given by

$$\sum_{i=0}^{N-1} a_i \left[h_i(t) + \alpha_1(t)p_{i,1}(t) + \alpha_2(t)p_{i,2}(t) - (\alpha_1(t) + t\alpha_2(t))\, p_{i,1}(1)\right] = f(t) - (\alpha_1(t) + t\alpha_2(t))\, y'(1) + \alpha_2(t)y(0). \tag{39}$$

4 Numerical Examples

To illustrate the application of the solution procedure discussed in this article, we present four problems of varying degree of difficulties. The results show that the method is fast and accurate even in cases when closed form solutions could not be obtained. All computations were done with resolution of J=3 in order to compare our results with others in the literature.

Example 4.1 *Our first example is taken from [2].*
Solve the equation

$$y'' + y = \sin(t) + t\cos(t), \qquad y(0) = 1,\ y'(0) = 1,\ t \in [0, 1]. \tag{40}$$

The exact solution to this problem is $y(t) = \cos(t) + \frac{5}{4}\sin(t) + \frac{1}{4}(t^2\sin(t) - t\cos(t))$.

The code used to setup this problem this problem is

Example 1

```
alpha1:=t->0: alpha2:=t->1: y0:=1: y1:=1:
f:=t->sin(t)+t*cos(t):
```

Example 4.2 *The second problem does not have elementary closed form solution. We solve the equation*

$$y'' - y' + ty = t^2, \qquad y(0) = 1,\ y'(0) = -1,\ t \in [0, 1]. \tag{41}$$

The series solution to this problem is $y(t) = 1 - t - \frac{1}{2}t^2 - \frac{1}{3}t^3 + \frac{1}{12}t^4 + \frac{1}{24}t^5 + O(t^6)$.

The problem is setup using the code

Example 2

```
alpha1:=t->-1: alpha2:=t->t: y0:=1: y1:=-1:
f:=t->t^2:
```

Node(n/32)	Approx	Exact	Error
1	1.030793	1.030767	0.000026
3	1.089637	1.089496	0.000142
5	1.144892	1.144700	0.000193
7	1.196953	1.196643	0.000310
9	1.245953	1.245594	0.000359
11	1.292290	1.291819	0.000472
13	1.336093	1.335577	0.000517
15	1.377736	1.377118	0.000618
17	1.417333	1.416676	0.000657
19	1.455208	1.454467	0.000741
21	1.491454	1.490681	0.000773
23	1.526319	1.525485	0.000834
25	1.559869	1.559012	0.000857
27	1.592255	1.591364	0.000891
29	1.623509	1.622605	0.000904
31	1.653672	1.652763	0.000909

Table 1: Example 1 - Initial Value Problem

Node(n/32)	Approx	Exact	—Error—
1	0.968201	0.968252	0.000051
3	0.901307	0.901588	0.000281
5	0.829915	0.830325	0.000410
7	0.753402	0.754047	0.000645
9	0.671585	0.672378	0.000794
11	0.583986	0.584992	0.001006
13	0.490479	0.491612	0.001133
15	0.390782	0.392021	0.001238
17	0.284842	0.286060	0.001218
19	0.172625	0.173639	0.001014
21	0.054173	0.054738	0.000565
23	-0.070235	-0.070588	0.000353
25	-0.200443	-0.202201	0.001757
27	-0.335793	-0.339880	0.004087
29	-0.475990	-0.483313	0.007323
31	-0.619929	-0.632093	0.012164

Table 2: Example 2 - Initial Value Problem

Example 4.3 *The next example is a second order stationary convection diffusion equation*

$$-y'' + by' = 0, \qquad y(0) = 1,\, y(1) = 0,\ x \in (0,1) \tag{42}$$

with $b > 0$. *The exact solution to this problem is* $y(x) = \frac{e^b - e^{bx}}{e^b - 1}$.

When b is large, the convection term dominates and the solution has a thin layer close to the right boundary point $x = 1$ where the solution drops from one to zero. When centered difference method is applied to this problem, it produces very oscillatory solution. However, Haar approximation method discussed in this paper produces a nice solution even at low resolution.

The code presented in the previous section is modified slightly to accommodate the changes given by (27) and (28). The result is given in the following table.

Node(n/32)	Approx	Exact	Error
1	0.999008	0.998853	0.000155
3	0.996417	0.995943	0.000474
5	0.992925	0.991967	0.000958
7	0.987979	0.986531	0.001448
9	0.981312	0.979102	0.002209
11	0.971869	0.968947	0.002922
13	0.959141	0.955068	0.004073
15	0.941114	0.936096	0.005018
17	0.916814	0.910166	0.006648
19	0.882400	0.874722	0.007677
21	0.836009	0.826277	0.009731
23	0.770309	0.760061	0.010248
25	0.681744	0.669553	0.012191
27	0.556317	0.545845	0.010472
29	0.387239	0.376755	0.010485
31	0.147787	0.145636	0.002151

Table 3: Example 3 with $b = 5$, Boundary value problem

Example 4.4 *Consider the boundary value*

$$y''(x) = e^{4x}, \qquad y(-1) = 0,\, y'(1) = 0,\ x \in (-1,1). \tag{43}$$

The exact solution is $y(x) = \frac{1}{16}e^{4x}$.

To solve this problem the left boundary must first be translated using the change of variable $t = (x+1)/2$. After the translation, the problem becomes

$$y''(t) = 4e^{8t-4}, \qquad y(0) = 0,\ y'(1) = 0,\ t \in (0,1). \tag{44}$$

Replace the three highlighted codes in section 3.1 by those developed in section 3.4. The result produced is shown in Table 4.

Node	Approx	Exact	Error
1	-0.818227	-0.852771	0.034544
3	-2.454327	-2.558010	0.103683
5	-4.089854	-4.262630	0.172776
7	-5.724199	-5.966230	0.242031
9	-7.356987	-7.668149	0.311162
11	-8.986564	-9.367295	0.380732
13	-10.611905	-11.061871	0.449966
15	-12.228518	-12.748911	0.520393
17	-13.833619	-14.423527	0.589908
19	-15.414994	-16.077658	0.662664
21	-16.965074	-17.698016	0.732942
23	-18.450661	-19.262692	0.812031
25	-19.851181	-20.735564	0.884382
27	-21.076391	-22.057075	0.980684
29	-22.070364	-23.129035	1.058671
31	-22.587793	-23.789556	1.201763

Table 4: Example 4 Mixed boundary problem with derivative on right boundary

5 Conclusion

In this paper, we have developed methods for approximating solutions second order differential equations assuming four mostly prescribed initial and boundary value conditions. As expected, the changing the boundary condition necessitated a change in the computer code used to approximate the solution. However, our presentation showed that this can be efficiently achieve by changing few lines of a Maple program. Exact solutions of four test problems were used to illustrate the effectiveness of the wavelet method. It is observed that the wavelet method is quite efficient in approximating the closed form solutions of differential equations.

The method discussed in this paper can be extended easily to higher dimension.

References

[1] C. F. Chen and C. H. Hsiao; Haar Wavelet Method for Solving Lumped and Distributed-Parameter Systems, *IEE Proceedings of Control Theory & Applications.*, Vol. 144 No. 1(1997), 87-94.

[2] Phang Chang and Phang Piau; Haar Wavelet Matrices Designation in Numerical Solution of Ordinary Differential Equations, *International Journal of Applied Mathematics*, 38:3(2011)

[3] U. Lepik; Haar wavelet method for solving stiff differential equations, *Mathematical Modeling and Analysis*, Vol. 14: 4 (2009), 467 481.

[4] U. Lepik; Application of the Haar wavelet transform to solving integral and differential equations, *Proc. Estonian Acad. Sci. Phys. Math.*, 56:1 (2007), 2846.

[5] U. Lepik; Numerical solution of differential equations using haar wavelets, *Math. Comput. Simul.*, vol. 68, (2005), 127143.

[6] C. H. Hsiao; Numerical solution of stiff differential equations via Haar wavelets, *Int. J. Comp. Math.*, 82:9 (2005), 11171123.

[7] K. Maleknejad and F. Mirzaee; Using rationalized Haar wavelet for solving linear integral equations,*Appl. Math. Comput.*, 160 (2005), 579587.

[8] K. Maleknejad, T. Lotfi, Y. Rostami; Numerical computational method in solving Fredholm integral equations of the second kind by using Coifman wavelet, *Applied Mathematics and Computation*, 186,(2007), 212218.

[9] Fazal-i-Haq, Iltaf Hussain and Arshed Ali; A Haar Wavelets Based Numerical Method for Third-order Boundary and Initial Value Problems, *World Applied Sciences Journal*, 13:10 (2011), 2244-2251.

[10] Al-Said and M.A. Noor; Numerical solutions of third-order system of boundary value problems, *Appl. Math. Comput.*, 190: (2007) 332-383.

[11] J. A. C. Weideman, S. C. Reddy, A MATLAB Differentiation Matrix Suite, *ACM Transactions on Mathematical Software*, Vol. 26, No. 4, (2000), 465519.

Appendix - Sample Code

Example 1

```
restart: with(LinearAlgebra):
h1 := t-> piecewise(0 <= t and t < 1, 1):
hi:=proc(j,k,t)
 local a,b,c,m;  m:=2^(j);
 a := k/m; b := (k+1/2)/m; c := (k+1)/m;
 return piecewise(a <= t and t < b, 1, b <= t and t < c, -1);
end proc:

J:=3:
N :=2^J:
hd := Vector(N): H := Matrix(N, N): T := Vector(N):
hd[1] := h1(t):

for i from 1 to N do T[i] := (i-1/2)/N: end do:

for j from 0 to J-1 do
 m := 2^j:
 for k from 0 to m-1 do
   i := m+k+1:
   hd[i] := hi(j, k, t):
 end do:
end do:
#Now Compute H at the collocation points
for i from 1 to N do
 for j from 1 to N do
   H[i, j] := eval(hd[i], t = T[j]):
 end do:
end do:
pn:=proc(i,n,t)
 if n=1 then return int(hd[i],t) fi:
 return int(pn(i,n-1,t)):
end proc:
RHS:= t->f(t)-alpha1(t)*y1-alpha2(t)*(t*y1+y0);
#alpha1(t) and alpha2(t) must be defined as functions of t
R:=Vector(N):
TMP:=Matrix(N,N):
A:=Matrix(N,N):
```

```
for i from 1 to N do
 R[i] := evalf(RHS(T[i])):
 tmp := alpha1(t)*pn(i,1,t)+alpha2(t)*pn(i,2,t):
 for j from 1 to N do
   TMP[i,j]:=eval(tmp, t = T[j]):
 od:
od:

#Compute the wavelet coefficients
A := Transpose(LinearSolve(Transpose(H+TMP), R)):

#Now compute the approximate solution
sol := y1*t+y0+sum('A[m0]*pn(m0,2,t)', m0 = 1 .. N):
#Convert to a function of t for easy comparison with
#exact solution
y:=unapply(sol,t):
```

Received: February 06, 2012