A WAVELET COLLOCATION METHOD FOR 2ND-ORDER ODES WITH VARIABLE INITIAL AND BOUNDARY CONDITIONS

ALEXEY IZMAILOV *

Key words. wavelets, numerical solution of ODEs, collocation methods

AMS subject classifications. 65M70, 65T60

1. Introduction. Second-order ODEs arise in a variety of mathematical modelling problems including plate deflection theory, the Lane-Emden equation for the gravitational potential of polytropic fluids in stellar media, the Keller-Miksis equation for describing the phenomenon of sonoluminesence in bubble cavitation, and many other engineering applications. As exact solutions can rarely be derived, a variety of numerical methods have been developed for these problems over the past several decades. The inherent structural complexity (such as equation stiffness) of these models is further increased by interest in variable initial and boundary value conditions (IVPs and BVPs, respectively) which complicate the construction and implementation of proposed algorithms.

Recently, wavelet-based methods for differential equations have gained popularity. Haar wavelets in particular have many useful properties such as orthogonality, compact support, and ease of integration. Compact support of the Haar wavelet basis permits a direct inclusion of different types of boundary conditions in numerical schemes. Finally, although the Haar wavelet basis is not differentiable, explicit integration approaches are simple to derive and evaluate.

The objective is to construct a simple collocation method using the Haar basis functions for the numerical solution of second-order ODEs of the form

$$y''(x) = \phi(x, y, y') \tag{1.1}$$

where ϕ is a function with bounded derivative on the domain $[0, 1]$. The method works for different kinds of IVP and BVP conditions but the following derivation is for IVPs specified by $y(0) = \alpha_1, \quad y'(0) = \beta_1$. The discussion is largely motivated by [3], though we extend that discussion and provide more challenging examples wherein the following procedure outperforms standard library ODE solvers.

2. Methodology. Recall that from the multi-resolution analysis of $L_2(\mathbb{R})$ obtained from a wavelet basis, any function $f \in L_2(\mathbb{R})$ produces a sequence of subspaces $\cdots \subset V_{-1} \subset V_0 \subset \cdots$ which can be used to approximate general functions by projections onto these spaces.

2.1. Haar Wavelets and Collocation Points. The Haar wavelet family for $x \in [0, 1)$ is defined by

$$h_i(x) = \begin{cases} 1 & \text{for } x \in [\alpha, \beta), \\ -1 & \text{for } x \in [\beta, \gamma), \\ 0 & \text{elsewhere}, \end{cases} \tag{2.1}$$

where

$$\alpha = \frac{k}{m}, \quad \beta = \frac{k + 0.5}{m}, \quad \gamma = \frac{k + 1}{m}. \tag{2.2}$$

*Brown University Department of Applied Mathematics

In the above, J indicates the maximum level of resolution, $m = 2^j, j \in \mathbb{Z}_{J+1}$ indicates the level of the wavelet and $k \in \mathbb{Z}_m$ is the translation parameter. The index $i = m + k + 1$ and in the minimal case of $m = 1, k = 0, i = 2$ while in the maximal case, $i = 2M = 2^{J+1}$. For $i = 1$, the function $h_1(x)$ is the usual Haar scaling function Further,

$$x_j = \frac{j - 0.5}{2M}, \quad j = 1, 2, \ldots, 2M \tag{2.3}$$

denote *collocation points*, which form the discretization.

For an equation of the form 1.1, we follow Lepik [2] and assume that

$$y''(x) = \sum_{i=1}^{2M} a_i h_i(x) \tag{2.4}$$

where $2M = 2^{J+1}$. Equation 2.4 is integrated twice with integration limits determined by boundary conditions. Hence, the solution $y(x)$ as well as derivatives $y'(x), y''(x)$ is expressed in terms of Haar functions and their integrals, which can be obtained explicitly. As the discretization is applied using the collocation points, each expansion results in a $2M \times 2M$ system, from which Haar coefficients a_i are computed. In practice, the resulting system is often singular, which motivates either the construction of preconditioners such as in [1], or the use of the expensive Moore pseudo-inverse, which considerably impacts this method's performance. We opt for the latter.

2.2. Explicit Integration and Boundary Conditions. For each of the aforementioned cases of IVP and BVPs, we derive the explicit integration forms which incorporate boundary conditions. To do so, we introduce additional notation for integrals of Haar wavelets. Specifically, let

$$p_{i,1}(x) = \int_0^x h_i(s)ds, \quad p_{i,\nu+1}(x) = \int_0^x p_{i,\nu}(s)ds), \quad \nu = 1, 2, \ldots \tag{2.5}$$

which can be evaluated explicitly, with the first two given by

$$p_{i,1}(x) = \begin{cases} x - \alpha & \text{for } x \in [\alpha, \beta) \\ \gamma - x & \text{for } x \in [\beta, \gamma) \\ 0 & \text{else} \end{cases}, \quad p_{i,2}(x) = \begin{cases} \frac{1}{2}(x-\alpha)^2 & \text{for } x \in [\alpha, \beta) \\ \frac{1}{4m^2} - \frac{1}{2}(\gamma - x)^2 & \text{for } x \in [\beta, \gamma) \\ \frac{1}{4m^2} & \text{for } x \in [\gamma, 1) \\ 0 & \text{else.} \end{cases} \tag{2.6}$$

2.3. Case (i), IVPs. For an IVP, integrating the ODE yields

$$y'(x) = \beta_1 + \sum_{i=1}^{2M} a_i p_{i,1}(x) \implies y(x) = \alpha_1 + \beta_1 x + \sum_{i=1}^{2M} a_i p_{i,2}(x) \tag{2.7}$$

by another integration. Substituting these values into the given differential equation yields a system of equations for a_i.

2.4. Overall Algorithm. The following is a summary of the algorithm for obtaining the approximation $y(x_j)$. The focus is on IVPs but the procedure is identical for each of the other initial/boundary data combinations, with the primary difference being the expansions used for approximating functions; see Appendix C and [3] for other forms.

Algorithm 2.1 Wavelet Approximation to ODE IVP

Input Boundary conditions, ODE $y''(x) = \phi(x, y, y')$, level of resolution M
Output Approximations $y(x_j)$ on collocation points
For $j = 1, 2, \ldots, 2M$, set $x_j = \frac{j-0.5}{2M}$
Let $\mathbf{a} = \mathbf{0}$ as initial guess for Newton's method
for $j = 1, 2, \ldots, 2M$ **do**
Apply Newton's method to the system

$$\sum_{i=1}^{2M} a_i h_i(x_j) = \phi\left(x_j, \alpha_1 + \beta_1 x + \sum_{i=1}^{2M} a_i p_{i,2}(x), \beta_1 + \sum_{i=1}^{2M} a_i p_{i,1}(x)\right)$$

with unknowns $a_1, \ldots, a_{2M}$
end for
for $j = 1, 2, \ldots, 2M$ **do**
Set

$$y(x_j) = \alpha_1 + \beta_1 x_j + \sum_{i=1}^{2M} a_i p_{i,2}(x_j).$$

end for
return $(y(x_j))$

The above applies equally well to any of the other boundary conditions derived in Appendix C with the only change being a change in the expansions of terms. Recall that Newton's method has an additional tolerance term ε which is compared against while minimizing the residual. In practice, this value does not change much past $\varepsilon \approx 10^{-6}$ and this value is used in all subsequent examples. Finally, each iteration of Newton's method is computed using a Moore pseudo-inverse as the current method does not include a preconditioner.

2.5. Error Estimate. The above method admits an error bound that ensures the convergence of the Haar wavelet approximation when M is increased as proven in [3].

LEMMA 2.1. *Assume that $y(x) \in L_2(\mathbb{R})$ with bounded first derivative on $(0, 1)$, then the error norm at the Jth level satisfies*

$$||e_J(x)|| \leq C\sqrt{\frac{K}{7}} 2^{-\frac{3}{2}M} \tag{2.8}$$

where $C = \int_0^1 |xh_2(x)|dx$ and K is a positive constant.

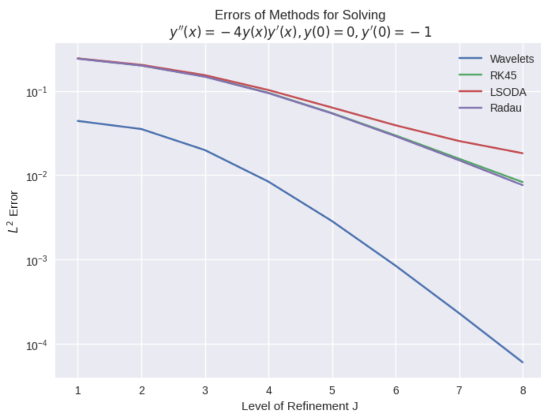
Although this error estimate shows convergence of the method, the error reduction in terms of discretization size h suggests that this method is actually $\mathcal{O}(h)$. This is due to the fact that a linear increase in J corresponds halving the discretization size.

3. Numerical Examples. We provide numerical examples of the above method and compare to standard numerical integrators provided by *SciPy*. Though the focus is on IVPs, comparable results can be obtained for other kinds of problems [3].

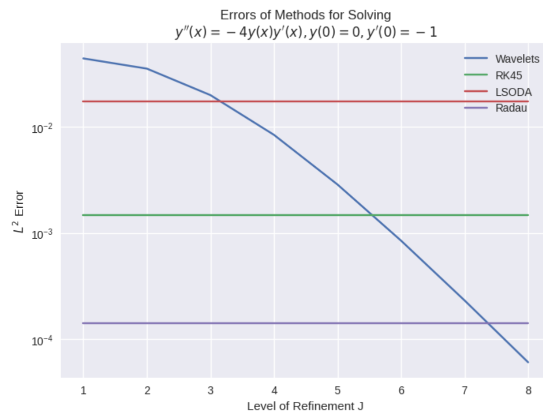
3.1. A Simple Parametric Non-linear IVP. For $n \in \mathbb{R}^+$, consider

$$\begin{cases} y_n''(x) = -ny_n(x)y_n'(x) & \text{on } (0,1) \\ y_n(0) = 0, y_n'(0) = -1 \end{cases} \tag{3.1}$$

with exact solution $y_n(x) = -\sqrt{\frac{2}{n}}\tan\left(\sqrt{\frac{n}{2}}x\right)$. The character of the solution manifold is such that a singularity is encountered for $n \to 5$. As motivated in the talk, the usual integrators follow an incorrect trajectory when approaching this singularity while wavelets can cope with the instability. For a particular case for $n = 4$, the following error plots demonstrate this behavior.



(a) On Collocation Points



(b) Integrator Evaluation Choice

As can be seen in these figures, the above wavelet collocation method has two key properties: iterative improvement as J increases, with a convergence error proportional to that which was previously derived, and secondly, outperforming the *SciPy* integrators. We also evaluate these methods across the entire solution manifold and compute the absolute error to obtain the following results:

The errors for the contour plots were computed across collocation points and the domain has been reduced in size to provide greater detail in the region surrounding the singularity. As can be readily seen, the wavelet method achieves considerably lower accuracy across the entire solution manifold, both in the limiting value of n (as far as could be numerically evaluated) and in less stiff areas.

4. Conclusions. We have demonstrated a simple wavelet collocation method for numerically solving ODEs and presented a few examples on which it outperforms standard ODE solvers, as can be seen in the above examples and Appendix B. Specifically, for stiff systems where classical integrators require very small step sizes, this method performs considerably better. Although this method is simple to derive, the error estimate and expensiveness of the method do not necessarily make it competitive with standard methods. However, following the derivation presented above yields a general method of solution for ODEs with either initial or boundary-value data which can be evaluated explicitly C. In some scenarios this may be useful as dealing with all of these possible cases of boundary data often involves theoretical manipulation of equations and data while the presented method deals with these automatically.

Appendix A. Code Availability. All code for generating the above results is provided in the following repository: https://github.com/alizma/APMA1940Y-FinalProject. The most representative portion of code, which consists of the Newton solver and evaluation of projection coefficients, is provided here:

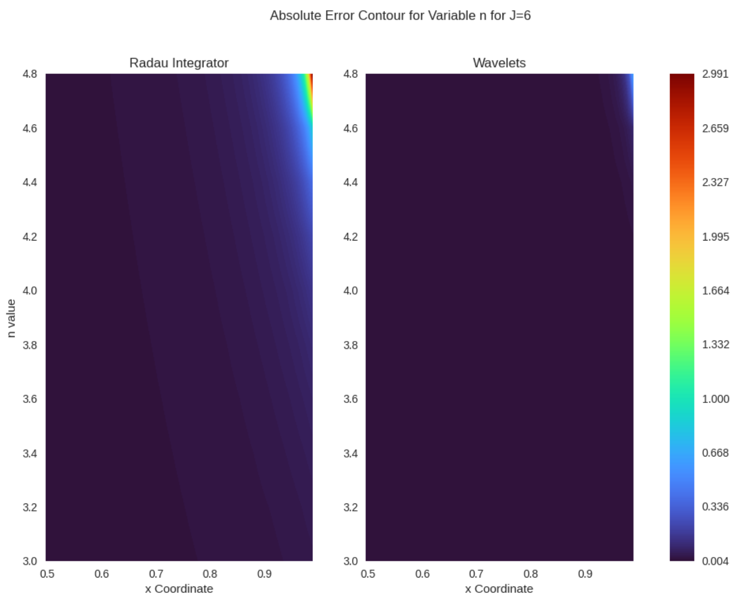


Fig. 2: Absolute Error Contours

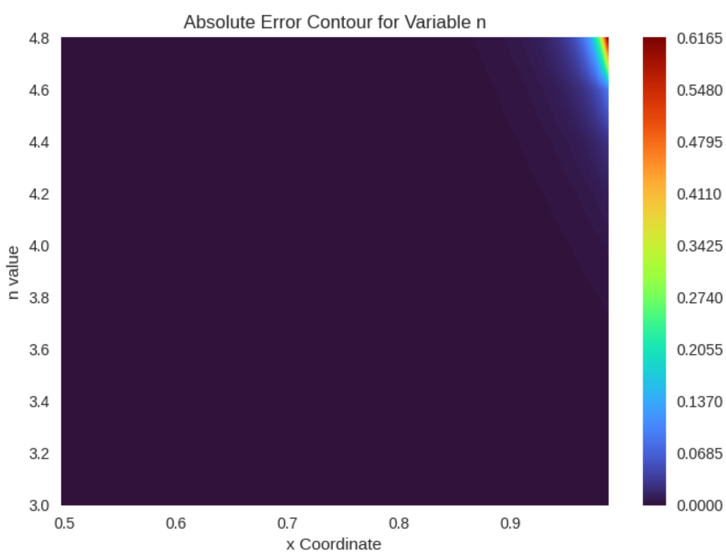


Fig. 3: Absolute Error Wavelets Only

```
def wavelet_solve(J, n):
    N = 2**(J + 1)
    j = np.arange(0, N)
    x = (j - 0.5) / N

```

```
    alpha1 = 0.
    beta1 = -1.
    a1 = beta1 - alpha1

    W = np.zeros((N, N))
    f = np.zeros((N, ))
    a = np.zeros((N, ))

    eps = 1e-6
    gradepstol = 1e-3
    r = np.ones((N, 1))

    iter_idx = 0

    while max(r) > eps:
        H = np.zeros((N, ))
        P1 = np.zeros((N, ))
        P2 = np.zeros((N, ))

        for i in range(N):
            H += a[i] * haar(x, i+1, J)
            P1 += a[i] * pi1(x, i+1, J)
            P2 += a[i] * pi2(x, i+1, J)

        f = n * (alpha1 + beta1 * x + P2) * (beta1 + P1) + H

        for k in range(N):
            W[:, k] = n * pi2(x, k+1, J) * (beta1 + P1) +
                        n * (alpha1 + beta1 + x + P2) * pi1(x, k+1, J)
                        + haar(x, k+1, J)


        a_new = np.linalg.pinv(W) @ (W@a - f)
        r = np.abs(a_new - a)
        a = a_new
        iter_idx += 1


    y = np.zeros((N, ))
    S = np.zeros((N, ))

    for i in range(N):
        S += a[i] * pi2(x, i+1, J)

    y = alpha1 + x * beta1 + S

    return y, x
```

Appendix B. Example of A Stiff System.

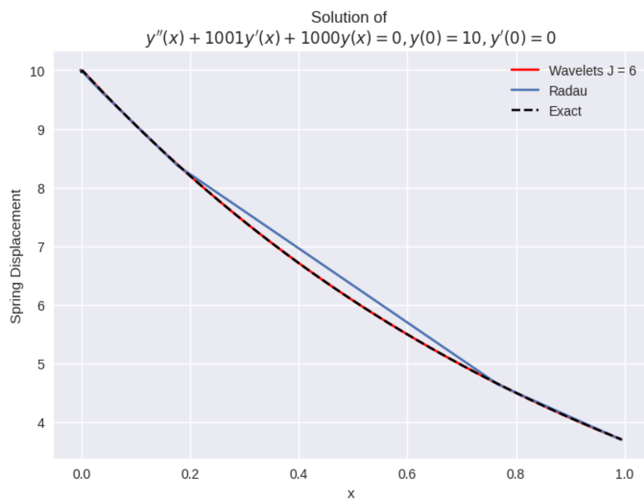
A mass-spring system can be formulated as the IVP

$$y''(x) + 1001y'(x) + 1000y(x) = 0, y(0) = 10, y'(0) = 0 \tag{B.1}$$

has exact solution

$$y(x) = 10\left(\frac{-1}{999}e^{-1000x} + \frac{1000}{999}e^{-x}\right). \tag{B.2}$$

The e^{-1000x} term makes numerical computation very sensitive to step size, meaning integrators must choose a small integration step size. In the following plot, the Radau integrator is unable to cope with the stiffness of the problem as it automatically chooses only 18 points for integration. On the other hand, using the wavelet collocation method at level $J = 6$ leads to a solution with an L^2 error of 10^{-4}:



Appendix C. Other Boundary Conditions.

We provide a few derivations for BVPs and for ODEs with periodic boundary conditions. The other cases enumerated earlier can be derived similarly and are provided in [3]. For certain other boundary data, the following integral is also employed:

$$C_{i,1} = \int_0^1 p_{i,1}(s)ds. \tag{C.1}$$

C.1. Case (iii), BVPs. Integrating the ODE from 0 to x yields

$$y'(x) = y'(0) = +\sum_{i=1}^{2M} a_i p_{i,1}(x) \implies y'(0) = \beta_3 - \alpha_3 - \sum_{i=1}^{2M} a_i C_{i,1}, \tag{C.2}$$

from which the unknown quantity $y'(0)$ can be computed. This also yields the approximation for $y(x)$ as it explicitly depends on this value as well. The approximate solution is then expressed as

(C.3)

$$y'(x) = \beta_3 - \alpha_3 + \sum_{i=1}^{2M} a_i(p_{i,1}(x) - C_{i,1}), \quad y(x) = \alpha_3 + (\beta_3 - \alpha_3)x + \sum_{i=1}^{2M} a_i(p_{i,2}(x) - xC_{i,1}).$$

C.2. Case (vi), Periodic BC. Integrating the ODE and using the boundary condition $y'(0) = y'(1)$,

$$y'(x) = y'(0) + \sum_{i=2}^{2M} a_i p_{i,1}(x) \tag{C.4}$$

while integrating and using the boundary condition $y(0) = y(1)$,

$$y'(0) = -\sum_{i=2}^{2M} a_i C_{i,1} \implies y(x) = y(0) + \sum_{i=2}^{2M} a_i (p_{i,2}(x) - xC_{i,1}). \tag{C.5}$$

REFERENCES

[1] S. Bertoluzza and G. Naldi, *A wavelet collocation method for the numerical solution of partial differential equations*, Applied and Computational Harmonic Analysis, 3 (1996), pp. 1–9, https://doi.org/https://doi.org/10.1006/acha.1996.0001, https://www.sciencedirect.com/science/article/pii/S1063520396900019.

[2] Lepik, *Numerical solution of differential equations using haar wavelets*, Mathematics and Computers in Simulation, 68 (2005), pp. 127–143, https://doi.org/https://doi.org/10.1016/j.matcom.2004.10.005, https://www.sciencedirect.com/science/article/pii/S0378475404002757.

[3] S. ul Islam, I. Aziz, and B. Šarler, *The numerical solution of second-order boundary-value problems by collocation method with the haar wavelets*, Mathematical and Computer Modelling, 52 (2010), pp. 1577–1590, https://doi.org/https://doi.org/10.1016/j.mcm.2010.06.023, https://www.sciencedirect.com/science/article/pii/S0895717710003006.