Development of a One-Dimensional Adaptive Wavelet Collocation Algorithm for Solving Partial Differential Equations

Brandon Gusto

October 20, 2017

1 Summary of Method

The adaptive wavelet collocation method for differential equations has a number of merits which may make it a viable alternative to traditional finite element, finite volume, or finite difference methods for certain problems. Some important advantages include the following:

- the multi-resolution properties of wavelets allow for natural grid adaptation

Probably the most attractive quality of wavelets for solving partial differential equations (PDEs) or systems of PDEs is for their tremendous ability to compress data. For problems with isolated structures on a large background of relatively homogenous data, the adaptive wavelet algorithm can achieve over ninety percent data compression, while still approximating the solution with reasonable accuracy. For problems such as combustion which have length scales ranging from the size of an engine block, all the way down to the Kolmogorov scale which governs the size of the smallest eddies, and time scales ranging from microseconds for turbulence down to nanoseconds for combustion, the wavelet algorithm is a game changer.

1.1 Wavelets

1.2 Dyadic Grid

In developing an adaptive wavelet collocation algorithm, an appropriate numerical grid should be described. Given the multi-resolution properties of wavelets previously described, a grid with a multi-resolution structure is fitting. This is called a dyadic grid, and it should have as many levels as their are scales in a given problem. It can consist of either uniform or non-uniform spacing of points. In the case of equally spaced grid points, let each grid level $j = 0, \ldots, J$ be computed by

$$x_k^j = 2^{-(j+\delta)}k, \quad \text{for } k = 0, \ldots, 2^{j+\delta}\ , \tag{1}$$

where δ is some integer shifting parameter, allowing one to dictate that the coarsest level of resolution have smaller spacing than 1 as in the case where $\delta = 0$. From here forth we keep in mind that the number of grid points at each level may be shifted at will based on a fixed choice of δ. The grid levels are formally defined by

$$\mathcal{G}^j = \{x_k^j \in \Omega : k \in \mathcal{K}^j\},\ j \in \mathcal{Z}, \tag{2}$$

where $\mathcal{K}^j$ is the integer set representing the spatial locations in the grid at level j. The grids are nested, implying that $\mathcal{G}^j \subset \mathcal{G}^{j+1}$. In other words, the points at level x^j are a perfect subset of the points at level x^{j+1}. This can also be demonstrated by the relation that $x_k^j = x_{2k}^{j+1}$.

1.3 Interpolating Subdivision

The interpolating subdivision scheme is central to the second-generation wavelet collocation approach. The scheme is used to approximate values at odd points x_{2k+1}^{j+1} by using $2N$ nearest points to construct interpolating polynomials of order $2N-1$. Lagrange polynomials are used, and the method can be used with a uniform grid or with nonuniform points such as Chebyshev points. The interpolating scheme is

$$f(x_{2k+1}^{j+1}) = \sum_{l=-N+1}^{N} w_{k,l}^{j} f(x_{k+l}^{j}), \tag{3}$$

where the coefficients $w_{k,l}^{j}$ are general interpolating coefficients. In this algorithm, Lagrange polynomials $L_{k+l}(x_{2k+1}^{j+1})$ are used. The Lagrange polynomial in this notation is given by

$$L_{k+l}(x) = \prod_{\substack{i=k-N+1 \\ i \neq k+l}}^{k+N} \frac{x - x_i}{x_{k+l} - x_i}. \tag{4}$$

The accuracy of such an interpolation is of $\mathcal{O}(2N)$. For example, if $N = 2$, this implies that as the grid spacing between construction points halves, there is a reduction in global error of a factor of 16.

1.4 Forward Wavelet Transform

The existence of a fast wavelet transform is one of the attractive qualities of the method. The transform makes use of the interpolating subdivision algorithm (3) for the calculation of the scaling and detail wavelet coefficients. The forward wavelet transform is given by

$$\begin{aligned} d_k^j &= \frac{1}{2}\left(c_{2k+1}^{j+1} - \sum_l w_{k,l}^{j} c_{2k+2l}^{j+1}\right), \\ c_k^j &= c_{2k}^{j+1}, \end{aligned} \tag{5}$$

and the inverse transform is given by

$$\begin{aligned} c_{2k+1}^{j+1} &= 2d_k^j + \sum_l w_{k,l}^{j} c_{k+l}^{j}, \\ c_{2k}^{j+1} &= c_k^j. \end{aligned} \tag{6}$$

1.5 Construction of Scaling Functions and Wavelets

1.5.1 Scaling Functions

The construction of second-generation interpolating wavelets makes use of the interpolating subdivision algorithm. The scheme is used to interpolate functional values defined at points on level j, to odd points (i.e. x_{2k+1}^{j+1}) at the next higher level of resolution. This scheme is used to construct the scaling and detail wavelet functions. Examples of the scaling and detail functions are

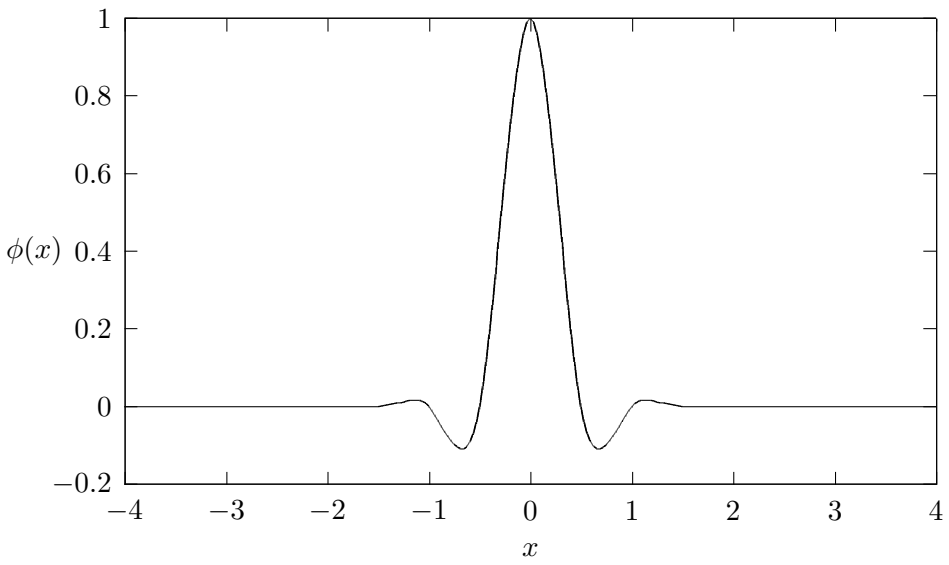

Figure 1: An example of a scaling function, $\phi(x)$, for $N = 3$.

shown in Figure 1 and Figure 2 respectively. To obtain the scaling function $\phi_m^j(x)$, from (5) set $c_k^j = \delta_{k,m}, \forall k \in \mathcal{K}^j$, where $\delta_{k,m}$ is the Kronecker delta function defined by

$$\delta_{k,m} = \begin{cases} 1 & k = m \\ 0 & k \neq m. \end{cases}$$

Then let all $d_l^{j'} = 0, \forall l \in \mathcal{L}^{j'}, \forall j' \geq j$ and perform the inverse transform up to an arbitrarily high level of resolution J.

1.5.2 Wavelets

The wavelet ψ_l^j is computed by setting $d_m^{j'} = \delta_{j',j}\delta_{l,m}, \forall l \in \mathcal{L}^j, \forall j \geq j$, and also $c_k^j, \forall k \in \mathcal{K}^j$. Then perform the inverse wavelet transform up to an arbitrarily high level of resolution J.

1.6 Wavelet Approximation of Functions

The approximation of a function $f(x)$ is done by setting the scaling coefficients at the arbitrary maximum level of resolution J to the function itself. Once the function is sampled this way on $\mathcal{G}^J$, the forward wavelet transform is performed down to the coarsest level of resolution. The function is then represented by

$$f^J(x) = \sum_{k \in \mathcal{K}^0} c_k^0 \phi_k^0(x) + \sum_{j=0}^{J-1} \sum_{l \in \mathcal{L}^j} d_l^j \psi_l^j(x). \tag{7}$$

Often, a large number of wavelet coefficients can be discarded, and the approximation (6) still is adequate. Define some threshold ϵ for the coefficients, then keep only those coefficients which

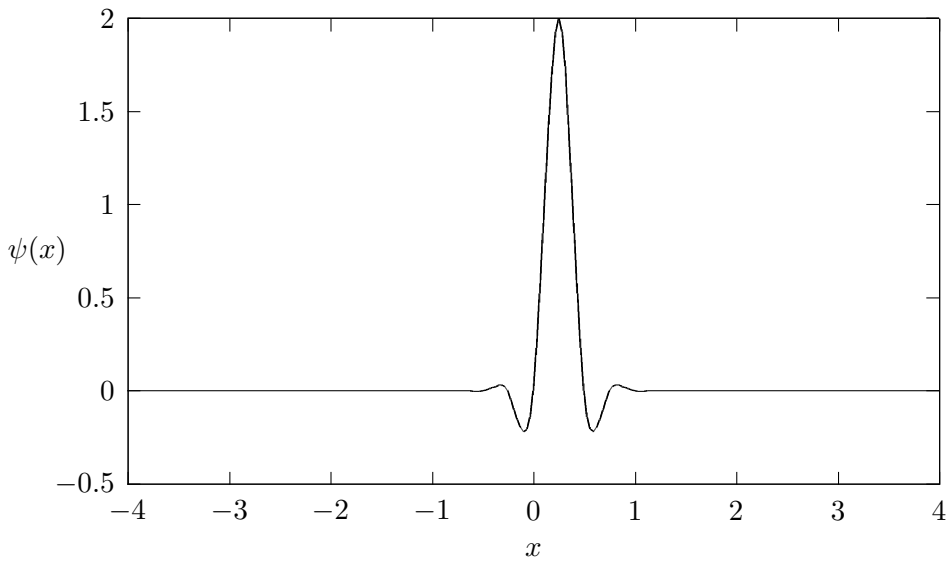

Figure 2: An example of a wavelet, $\psi(x)$, for $N = 3$.

satisfy $|d_l^j| \geq \epsilon$. The approximation (6) becomes

$$f_{\geq}^{J}(x) = \sum_{k \in \mathcal{K}^0} c_k^0 \phi_k^0(x) + \sum_{j=0}^{J-1} \sum_{\substack{l \in \mathcal{L}^j \\ |d_l^j| \geq \epsilon}} d_l^j \psi_l^j(x). \tag{8}$$

In the figures below are approximations to the function $f(x) = \cos(80\pi x) \exp(-64x^2)$ with an increasing threshold ϵ.

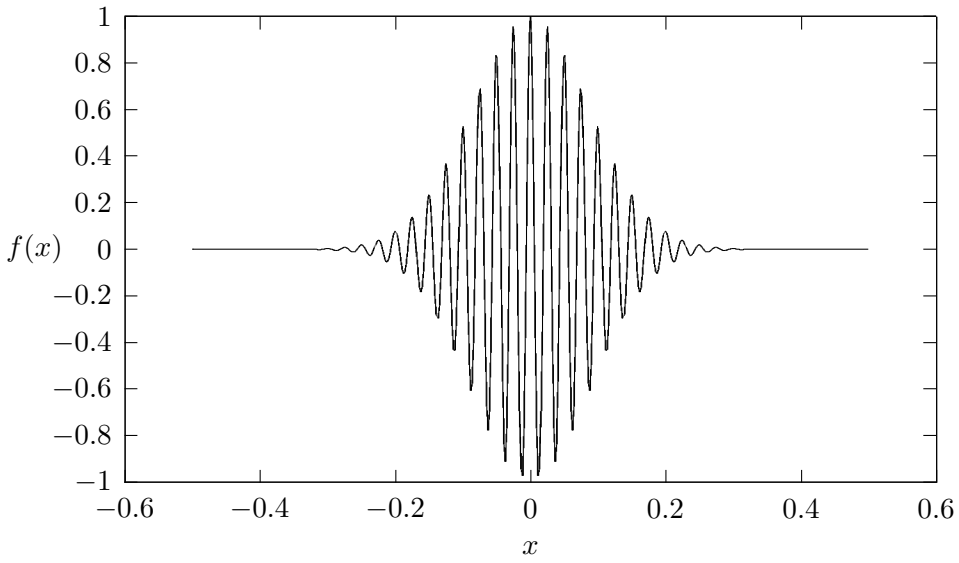

Figure 3: Approximation of $f(x) = \cos(80\pi x)\exp(-64x^2)$, with $N = 3$ and 256 points on $\mathcal{G}^J$. All coefficients are kept.

1.7 Calculation of Spatial Derivatives

The calculation of spatial derivatives involves analyzing the detail coefficients, as they are a measure of how well the given function is approximated by the local interpolant. If a point x_k^j on level j of the dyadic grid, does not have detail coefficients above the threshold at points x_{2k+1}^{j+1} or x_{2k-1}^{j+1}, then the given function's error is bounded by the thresholding parameter ϵ. Thus a stencil consisting of the same points which constructed the underlying polynomial should result in an accurate result at that point, within some constant multiple of the tolerance ϵ.

1.7.1 Differentiating Lagrange Polynomials

The computational complexity of constructing Lagrange interpolating polynomial coefficients is $\mathcal{O}(N)$. Once these terms are known, the weights needed for computing spatial derivatives on the adaptive grid can be computed with only slightly more effort. This is also an $\mathcal{O}(N)$ operation. The analytic formula for the coefficients is given by

$$\frac{d}{dx}L_{k+l}(x) = L_{k+l}(x) \sum_{\substack{i=k-N+1 \\ i\neq k+l}}^{k+N} \frac{1}{x_{k+l} - x_i}. \tag{9}$$

However once this coefficient is calculated for each point in the stencil, it must be multiplied by its functional value, and all points are then summed again in the same way that the Lagrange polynomial is in the first place. Thus the total complexity of computing derivatives this way is $\mathcal{O}(N^2)$, which is not desirable. The order of accuracy for computing one spatial derivative this way is $\mathcal{O}((h^j)^{2N-1})$, where h^j is the local grid spacing. Note that for a stencil consisting of four points ($N = 2$), this method is not only computationally inefficient, but also 1 order of accuracy worse than a comparable four-point centered finite difference scheme, which has $\mathcal{O}((h^j)^{2N})$ accuracy. The

only advantage of this method can be seen when the stencil is non-uniform, in which case the finite difference coefficients would have to be computed either in an implicit sense, or by interpolation, as done here.

1.7.2 Finite Difference Scheme