Unsteady Solutions of Non-Linear Differential Equations Using Walsh Function Series

Peter A Gnoffo*
NASA Langley Research Center, Hampton, VA 23681-2199

Walsh functions form an orthonormal basis set consisting of square waves. The discontinuous nature of square waves make the system well suited for representing functions with discontinuities. The product of any two Walsh functions is another Walsh function – a feature that can radically change an algorithm for solving non-linear partial differential equations (PDEs). The solution algorithm of non-linear differential equations using Walsh function series is unique in that integrals and derivatives may be computed using simple matrix multiplication of series representations of functions. Solutions to PDEs are derived as functions of wave component amplitude. Three sample problems are presented to illustrate the Walsh function series approach to solving unsteady PDEs. These include an advection equation, a Burgers equation, and a Riemann problem. The sample problems demonstrate the use of the Walsh function solution algorithms, exploiting Fast Walsh Transforms in multi-dimensions ($O(N\log(N))$). Details of a Fast Walsh Reciprocal, defined here for the first time, enable inversion of a Walsh Symmetric Matrix in $O(N\log(N))$ operations. Walsh functions have been derived using a fractal recursion algorithm and these fractal patterns are observed in the progression of pairs of wave number amplitudes in the solutions. These patterns are most easily observed in a remapping defined as a fractal fingerprint (FFP). A prolongation of existing solutions to the next highest order exploits these patterns. The algorithms presented here are considered a work in progress that provide new alternatives and new insights into the solution of non-linear PDEs.

I. Introduction

The solution of partial differential equations (PDEs) with Walsh functions offers new opportunities to simulate many challenging problems in mathematical physics. The approach was developed to better simulate hypersonic flows with shocks on unstructured grids. Fundamental derivations of Walsh functions, their products, reciprocals, derivatives, and integrals have been documented.[1] Examples of shocked flows in nozzles were used to demonstrate associated algorithms.

While this algorithm holds great promise, there are coding infrastructure challenges that make it difficult to further explore and develop numerical simulation tools using Walsh functions. Some unique strengths of Walsh functions are the capabilities to systematically account for non-linear interactions of component waves and to explicitly propagate boundary conditions across the solution domain. However, the implementation of these capabilities presents tedious programming challenges. Multiplication and division of two Walsh functions are fundamentally different from currently supported operations in any programming language. Keeping track of Jacobians required for the solution of systems of PDEs can become especially tedious in this environment. Fortunately, all of this tedious detail can be supported in the background through use of a module that supports operator overloading and other commonly used functions and routines in the solution of partial differential equations. This support greatly simplifies the learning curve for new users wanting to explore the use of Walsh functions in solving PDEs. A FORTRAN module for supporting Walsh function simulations has been documented.[2]

The purpose of this paper is to document algorithm advances associated with the solution of unsteady PDEs, with emphasis on the Riemann problem. A longer term goal of this work is to evaluate a multi-dimensional Riemann solver with small basis set to replace existing flux reconstruction algorithms on tetrahedral grids. In addition, two supporting algorithms not previously documented are provided within:

*Senior Research Engineer, Aerothermodynamics Branch; AIAA Fellow

- The Fast Walsh Transform (FWT) in multi-dimensions is key to efficient problem solving using a large number (N) of terms in the series. An element of any solution algorithm requires the transformation from wave number representation to discrete value representation and back. The FWT implements this transformation in order $N\log(N)$ operations (if $N = 2^p$).
- The Fast Walsh Reciprocal (FWR) provides the inverse of a Walsh symmetric matrix in order $N\log(N)$ operations. Any problems involving the evaluation of the reciprocal of a function requires evaluation of a Walsh symmetric matrix inverse.

This paper is organized as follows. Section II provides a review of Walsh function fundamentals. This review is based on a more comprehensive article[1] published in February, 2014. Section III presents algorithms for filtering high frequency noise and wave component sequencing (equivalent to grid sequencing in a finite-difference context). Section IV defines the three unsteady test problems and documents how FORTRAN module WALSH_TOOLS[2] is used to obtain their solutions. Both an implicit and explicit formulation are described. The implicit formulation uses an exact Jacobian of the Walsh series components. It converges very well but becomes intractably large for more than 2^{13} degrees of freedom in a sub-domain. The explicit formulation utilizes an approximate Jacobian with respect to the full series representation on a sub-domain. Critical algorithms affecting solution times are described in the Appendices. The Fast Walsh Transform (FWT) in multi-dimensions is described in Sec.VII (Appendix A). The Fast Walsh Reciprocal (FWR) is described in Sec.VIII (Appendix B).

II. Walsh Function Fundamentals

A. Walsh Function Derivation

Let $g_n(x)$ be a basis function over the domain $x_a \le x \le x_b$ with n segments. The orthonormal basis function requirements[3] that

$$\int_{x_a}^{x_b} g_n(x)g_m(x)dx = \delta_{nm} \tag{1}$$

enable a recursive algorithm to define $g_n(x)$ based on the following constraints. Constrain $|g_n(x)|$ equal to a constant across the entire domain. Let index $p \ge 0$ define a segment size dx_p.

$$dx_p = (x_b - x_a)/2^p \tag{2}$$

An additional constraint on segment lengths within $g_n(x)$ requires that at most two segment lengths (dx_p and $2dx_p$) are allowed to create n segments spanning the domain. These new constraints allow basis functions to be grouped according to the smallest segment size. A basis function $g_n(x)$ belongs to group p if

$$2^{p-1} < n \le 2^p \tag{3}$$

As an example, consider the evolution of the first four groups of $g_n(x)$ corresponding to $0 \le p \le 3$ on the domain with $x_a = 0$ and $x_b = 1$. For $p = 0$ the only element of the group according to Eq. 3 is $n = 1$. A single segment with length given by Eq. 2 spans the domain. Therefore, $g_1(x) = 1$ (Fig. 1) is the first basis function, belonging to group $p = 0$ and satisfying Eq. 1. (While $g_1(x) = -1$ also satisfies Eq. 1, it is convenient to adopt a convention where the sign of the function on the first segment is positive.)

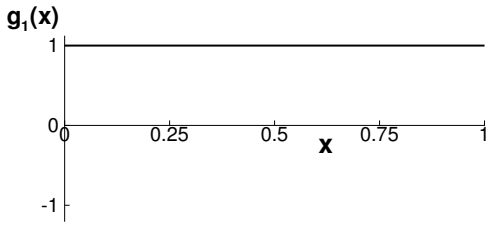


Figure 1. Basis function in group p=0.

For $p = 1$, the only element of the group according to Eq. 3 is $n = 2$. Two segments with length equal to $1/2$ span the domain. Therefore,

$$g_2(x) = \begin{cases} 1 & \text{for} \quad 0 \le x < \frac{1}{2} \\ 0 & \text{for} \quad x = \frac{1}{2} \\ -1 & \text{for} \quad \frac{1}{2} < x \le 1 \end{cases}$$

where function $g_2(x)$ (Fig. 2) is the second basis function, belonging to group $p = 1$ and satisfying Eq. 1. The value of the function at interior segment boundaries is set to 0, the average of the function on either side of the boundary. (The vertical line through interior segment boundaries in Fig. 2 provides strong visual impact of the discontinuity and is not meant to imply all values between –1 and 1 are included.)

The group defined by $p = 2$ is the first to have more than one function, $g_3(x)$ and $g_4(x)$. The allowed segment sizes in this group are 1/4 and 1/2. Each function is initialized with the smallest segment at the beginning and end of the domain. For $n = 3$, this initialization leaves a single segment of length 1/2 available to span the interior of the domain. For $n = 4$, the initialization leaves two segments of length 1/4 to span the interior of the domain. The corresponding functions are shown in Fig. 3. By inspection, it is clear that these first four basis functions all satisfy Eq. 1.

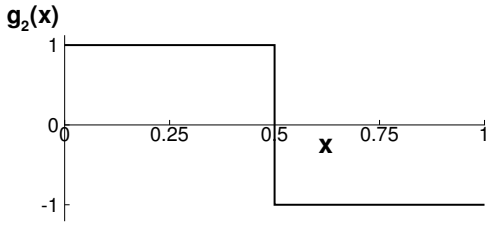
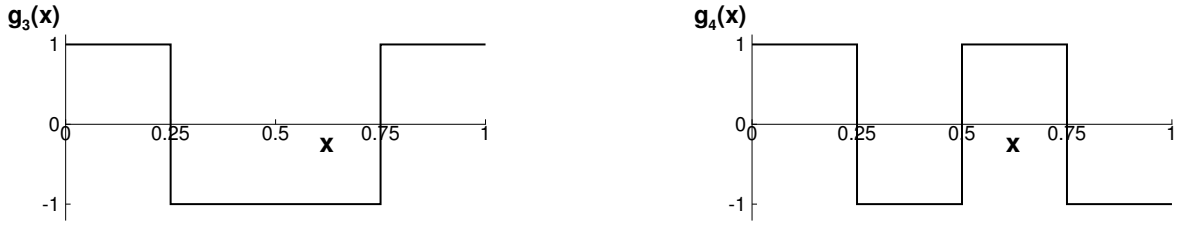


Figure 2. Basis function in group p=1.

$$g_3(x) = \begin{cases} 1 & \text{for} \quad 0 \le x < \frac{1}{4} \\ 0 & \text{for} \quad x = \frac{1}{4} \\ -1 & \text{for} \quad \frac{1}{4} < x < \frac{3}{4} \\ 0 & \text{for} \quad x = \frac{3}{4} \\ 1 & \text{for} \quad \frac{3}{4} \le x < 1 \end{cases} \qquad g_4(x) = \begin{cases} 1 & \text{for} \quad 0 \le x < \frac{1}{4} \\ 0 & \text{for} \quad x = \frac{1}{4} \\ -1 & \text{for} \quad \frac{1}{4} < x < \frac{1}{2} \\ 0 & \text{for} \quad x = \frac{1}{2} \\ 1 & \text{for} \quad \frac{1}{2} < x < \frac{3}{4} \\ 0 & \text{for} \quad x = \frac{3}{4} \\ -1 & \text{for} \quad \frac{3}{4} < x \le 1 \end{cases}$$

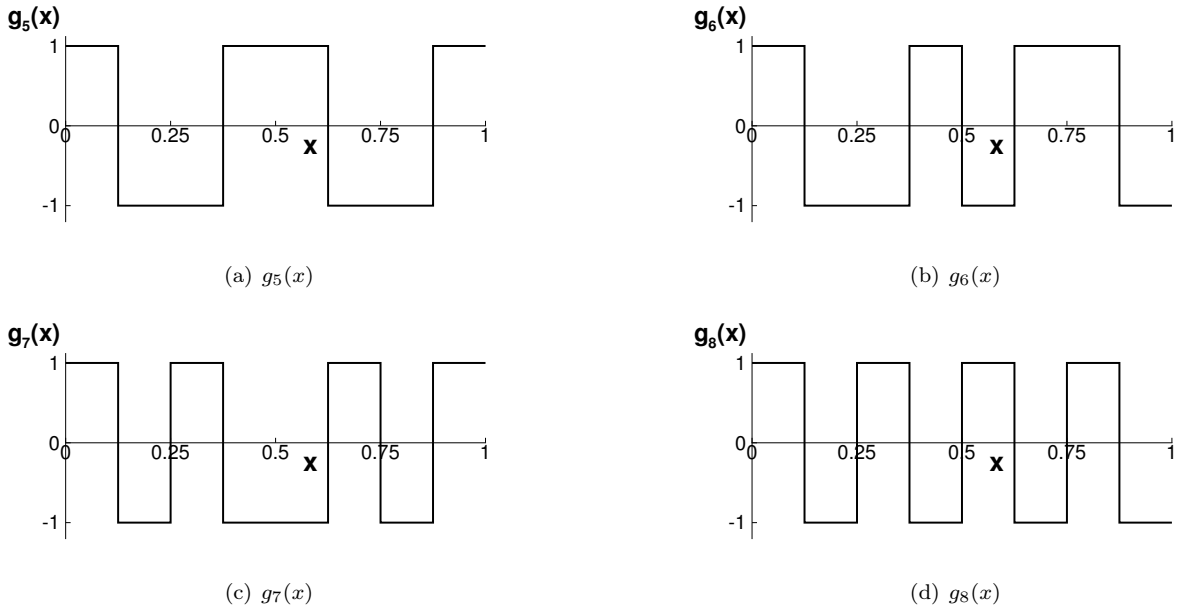


(a) $g_3(x)$ (b) $g_4(x)$

Figure 3. Basis functions in group p=2.

The group defined by $p = 3$ includes four functions, $g_5(x)$ through $g_8(x)$. The allowed segment sizes in this group are 1/8 and 1/4. Each function is again initialized with the smallest segment at the beginning and end of the domain. This is the first group where the distribution of segments required to satisfy Eq. 1 for all combinations of functions in groups 0 – 3 becomes non-trivial. Still, the number of permutations of segment size distribution is small and a solution, evident by inspection, is shown in Fig. 4.

These results suggest the existence of an algorithm that would allow the definition of basis functions for $p >> 3$ and, correspondingly $n >> 8$. Such an algorithm was first identified by Walsh[4] and the basis functions are used extensively in signal and image processing.[5,6] A new derivation applies a fractal-like bisection algorithm focused on the distribution of segment sizes that yields the following definition for basis



(a) $g_5(x)$ (b) $g_6(x)$

(c) $g_7(x)$ (d) $g_8(x)$

Figure 4. Basis functions in group p=3.

functions $g_n(x)$ on the domain $x_a \le x \le x_b$:

$$g_n(x) = \begin{cases} (-1)^{(m+1)}(x_b - x_a)^{-1/2} & \text{for} \quad x_{m-1} < x < x_m & 0 < m \le n \\ 0 & \text{for} \quad x = x_m & 0 < m < n \\ (x_b - x_a)^{-1/2} & \text{for} \quad x = x_a & \\ (-1)^{(n+1)}(x_b - x_a)^{-1/2} & \text{for} \quad x = x_b & \end{cases} \tag{4}$$

where

$$x_m \quad = \quad x_a + \sum_{i=1}^{m} \hat{g}_n(i) dx_p \tag{5}$$

and i, m, and n are positive integers. In Eq. 5, $\hat{g}_n(i)$ is a vector of length n in which the value of element i signifies the length of the i^{th} segment in $g_n(x)$. Thus, it defines a dimensionless distribution such that

$$\hat{g}_n(i) = \begin{cases} 1 & \text{if} \quad x_i - x_{i-1} = dx_p \\ 2 & \text{if} \quad x_i - x_{i-1} = 2dx_p \end{cases} \quad \text{where} \quad 1 \le i \le n \tag{6}$$

and dx_p was defined previously in Eq. 2.

The dimensionless distribution function $\hat{g}_n(i)$ is defined as a function of group number p. The sum of the segment lengths must equal the domain length, consequently it follows that

$$\sum_{i=1}^{n} \hat{g}_n(i) = (x_b - x_a)/dx_p = 2^p \tag{7}$$

Figure 5 presents the sequence of the first 32 $\hat{g}_n$ segment length distribution vectors in a manner that highlights the symmetries inherent in the defining algorithm. (Details of the folding algorithm to obtain this result are provided in the original reference.[1]) Note that the two columns on the right side of the figure include the index n of the basis function and its group number p. Recall that the minimum segment length in a basis function dx_p decreases by a factor of 2 for each increment in group number p. Compare the

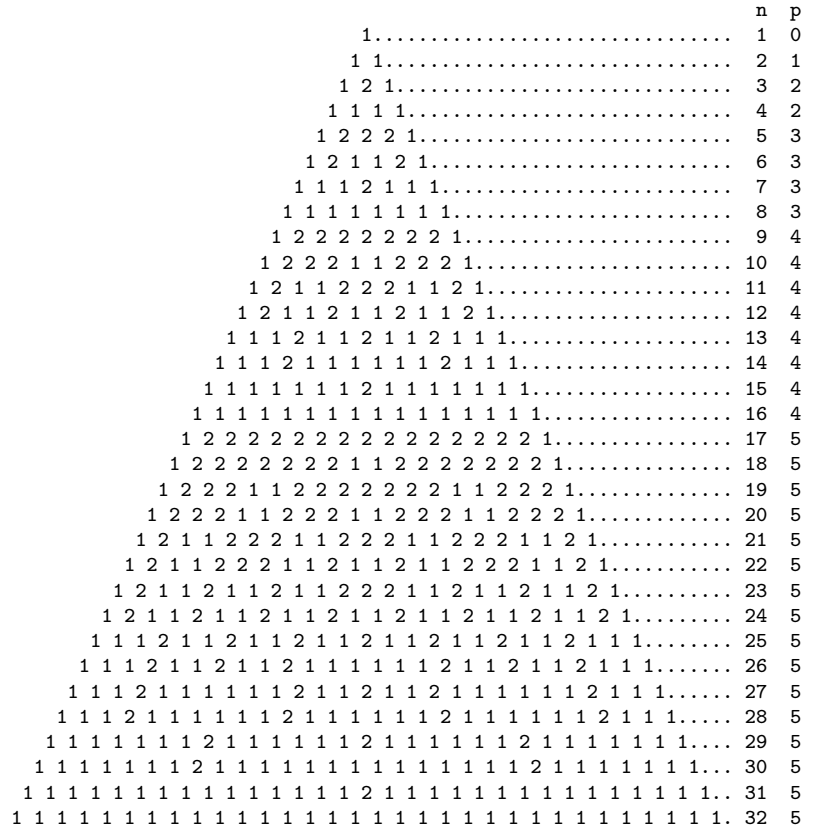


Figure 5. The distribution vector $\hat{g}_n$ for level p ranging from 0 to 5 and $1 \leq n \leq 2^5$.

distribution of segments in Figs. 1 – 4 to the top eight rows of Fig. 5 to confirm understanding of the relation between the basis function $g_n(x)$ and the corresponding segment distribution vector $\hat{g}_n(i)$. Figure 6 shows the segments used to partition the interval $x_a \leq x \leq x_b$ for the first 64 basis functions $g_n(x)$. The symbols on each line indicate segment boundaries. The crease lines for the first three bisecting folds are indicated along the bottom of the figure. A more careful examination of Fig. 6 reveals that the folding algorithm used to define the distribution of segments propagates patterns through ever smaller bisections of the domain and larger values of n. For example, take a sheet of paper to cover the right side of Fig. 6 from bisection 1. The pattern that remains for even (2m) or odd (2m-1) values of n on the bisected figure is exactly that of the whole figure. The only difference between these same two even and odd basis functions on the full domain is that the even function had two segments of length dx_p on either side of the crease (bisection 1) and the odd function had a single segment of length $2dx_p$ spanning the crease. Take another sheet of paper and cover the left side of the figure from bisection 2. The pattern that remains for every fourth value of n is exactly that of the whole. These patterns have been exploited[1] to derive a new recursion formula for $g_n(x)$ that lead to proofs for closure under multiplication and explain the relationships of specific pairs of wave number amplitudes to be discussed in a subsequent section.

B. Product of Two Walsh Functions

A multiplication table of basis functions is developed in Table 1. The table is constructed based on a unit domain where $x_b - x_a = 1$. The appropriate scale factor for the product in the most general case is $(x_b - x_a)^{-1/2}$ derived from the definition of $g_n(x)$ in Eq. 4. This scale factor is inserted in the upper right corner of the table. The first row of results is simply a recognition that a constant value across the entire domain ($g_1(x)$) multiplying any other function ($g_n(x)$) returns the rescaled value of $g_n(x)$. The lower diagonal of the table indicates that the square of $g_n(x)$ on every segment is a constant across the domain (except at a finite number of points where $g_n(x) = 0$ on segment boundaries), returning a rescaled value of $g_1(x)$.

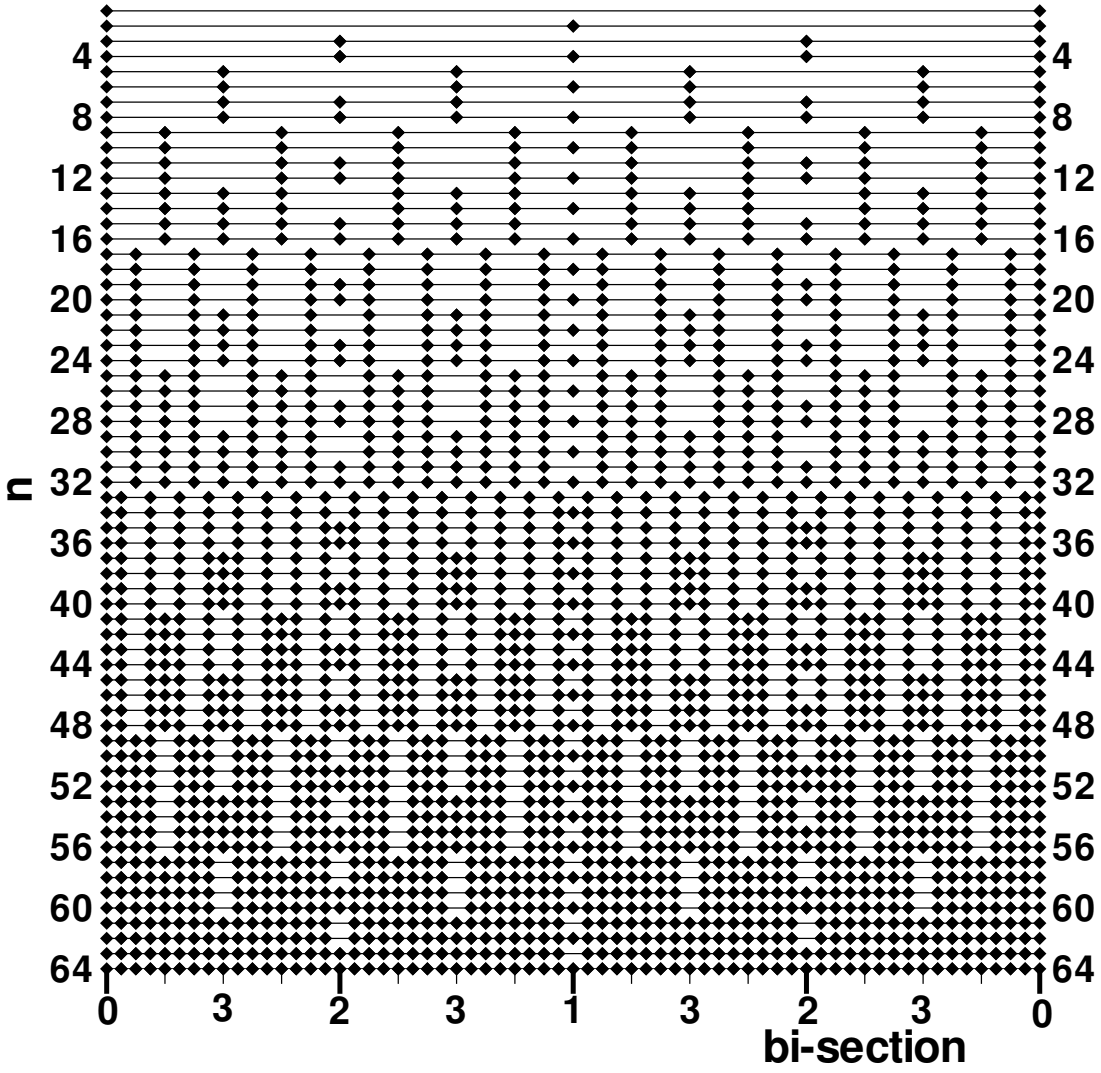


Figure 6. Segments spanning domain for the first 64 basis functions $g_n(x)$.

Every other element of Table 1 can be defined by recursion. It is not generally required to perform more than one bisection and one union to fill out the table.[1]

C. Series Expansions of Functions

A function $f(x)$ may be represented by an infinite series

$$\sum_{n=1}^{\infty} A_n g_n(x) \tag{8}$$

Table 1. A multiplication table of basis functions.

$\frac{1}{\sqrt{(x_b-x_a)}}$	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	g_{11}	g_{12}	g_{13}	g_{14}	g_{15}	g_{16}
g_1	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	g_{11}	g_{12}	g_{13}	g_{14}	g_{15}	g_{16}
g_2		g_1	g_4	g_3	g_6	g_5	g_8	g_7	g_{10}	g_9	g_{12}	g_{11}	g_{14}	g_{13}	g_{16}	g_{15}
g_3			g_1	g_2	g_7	g_8	g_5	g_6	g_{11}	g_{12}	g_9	g_{10}	g_{15}	g_{16}	g_{13}	g_{14}
g_4				g_1	g_8	g_7	g_6	g_5	g_{12}	g_{11}	g_{10}	g_9	g_{16}	g_{15}	g_{14}	g_{13}
g_5					g_1	g_2	g_3	g_4	g_{13}	g_{14}	g_{15}	g_{16}	g_9	g_{10}	g_{11}	g_{12}
g_6						g_1	g_4	g_3	g_{14}	g_{13}	g_{16}	g_{15}	g_{10}	g_9	g_{12}	g_{11}
g_7							g_1	g_2	g_{15}	g_{16}	g_{13}	g_{14}	g_{11}	g_{12}	g_9	g_{10}
g_8								g_1	g_{16}	g_{15}	g_{14}	g_{13}	g_{12}	g_{11}	g_{10}	g_9
g_9									g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8
g_{10}										g_1	g_4	g_3	g_6	g_5	g_8	g_7
g_{11}											g_1	g_2	g_7	g_8	g_5	g_6
g_{12}												g_1	g_8	g_7	g_6	g_5
g_{13}													g_1	g_2	g_3	g_4
g_{14}														g_1	g_4	g_3
g_{15}															g_1	g_2
g_{16}																g_1

over the interval $x_a \leq x \leq x_b$. The coefficients A_n are computed

$$A_n = \int_{x_a}^{x_b} f(x) g_n(x) dx \tag{9}$$

Because $g_n(x)$ is a constant on any given segment and changes sign across segments, the integral in Eq. 9 may be expressed as

$$A_n = (x_b - x_a)^{-1/2} \sum_{i=1}^{n} (-1)^{(i+1)} \int_{x_i}^{x_{i+1}} f(x) dx \tag{10}$$

where

$$x_{i+1} = x_a + \sum_{j=1}^{i} \hat{g}_n(j) dx_p \tag{11}$$

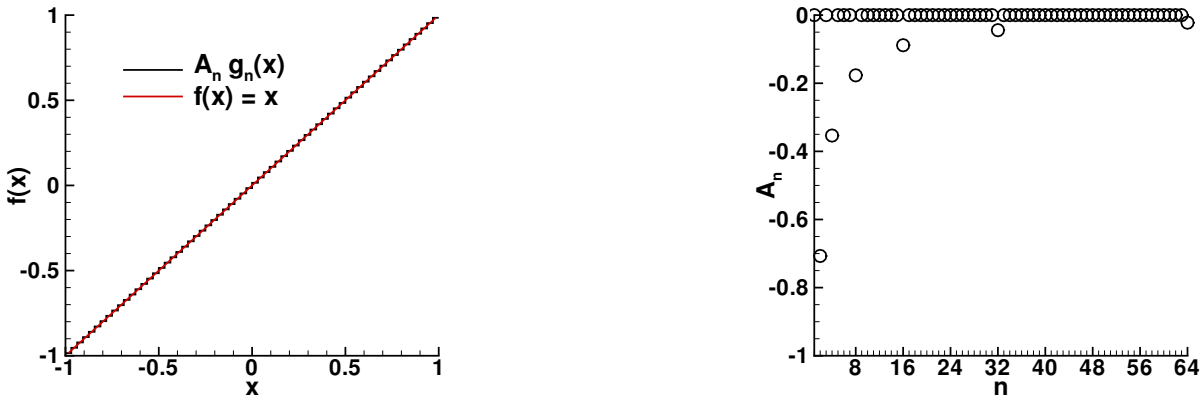
$$x_i = x_{i+1} - \hat{g}_n(i) dx_p \tag{12}$$

Note that for domains comprised of $N = 2^p$ segments the FWT (see Appendix A) can be used to compute the wave components A_n from the integrated average value of f over each segment in order pN operations.

Figure 7 shows the series representation of the function $f(x) = x$ on the interval $-1 \leq x \leq 1$. The basis function series is truncated at $n = 64$ ($p = 6$). Stair-stepping of the segmented representation is evident; nevertheless, the error norm $L_1^6 = 0$ where L_1^p is given by

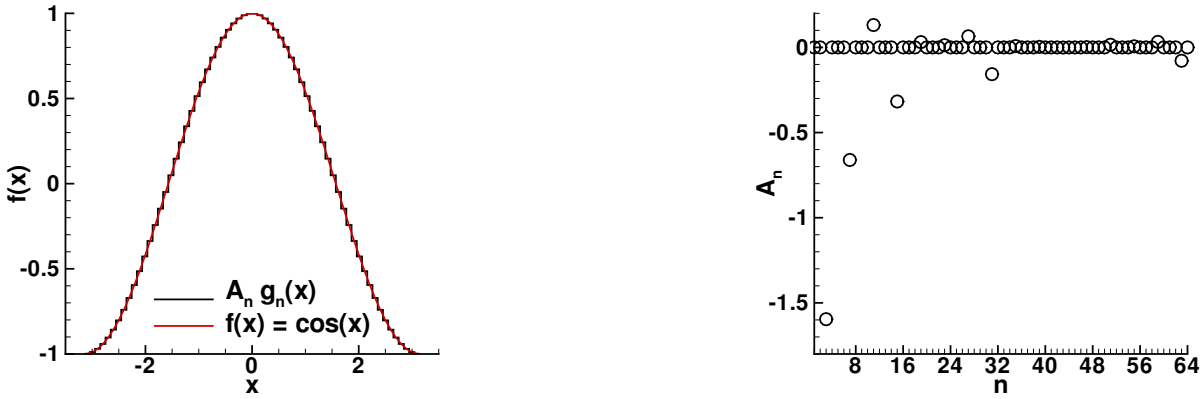
$$L_1^p = (x_b - x_a)^{-1} \sum_{m=1}^{2^p} \left| f(x_{m+1/2}) - \sum_{i=1}^{2^p} A_i g_i(x_{m+1/2}) | \right| dx_p \tag{13}$$

The error norm sums the absolute difference between the function and its series representation at the midpoint of the smallest segment size for each segment. Each contribution is weighted by the ratio of dx_p to the total interval length. A zero norm in this case means that the function and its series representation exactly match at each midpoint of the smallest segment in the series. Note that the only non-zero coefficients in Fig. 7b are A_k where $k = 2, 4, 8, 16,$ and 64. All linear functions $f(x) = C_1 x + C_0$ on all intervals may be represented using coefficients A_k with $k = 2^p$ and $p \geq 0$ with $L_1^p = 0$.



(a) $f(x)$ compared to $\sum_{n=1}^{64} A_n g_n(x)$ (b) Series coefficients

Figure 7. Representation of $f(x) = x$ by series $\sum_{n=1}^{64} A_n g_n(x)$.

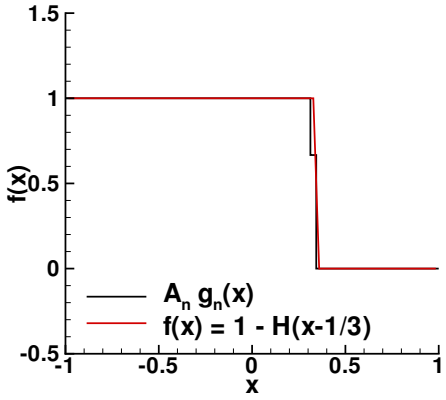


(a) $f(x)$ compared to $\sum_{n=1}^{64} A_n g_n(x)$ (b) Series coefficients

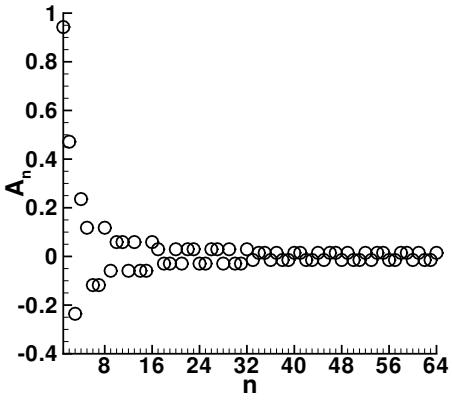
Figure 8. Representation of $f(x) = cos(x)$ by series $\sum_{n=1}^{64} A_n g_n(x)$.

Figure 8 shows the series representation of the function $f(x) = cos(x)$ on the interval $-\pi \leq x \leq \pi$. The error norm for this case is $L_1^6 = 0.000255735$. With successive increases in p (doubling the number of segments) the L_1 norm drops by a factor of 4 ($L_1^7 = 0.00006392036$, $L_1^8 = 0.00001597925$).

Figure 9 shows the series representation of the function $f(x) = 1 - H(x - 1/3)$ on the interval $-1 \leq x \leq 1$. The discontinuity in the Heaviside function is placed at $x = 1/3$ so that it never falls perfectly on a segment boundary. The error norm for this case is $L_1^6 = 0.005208338$. With successive increases in p (doubling number of segments) the L_1 norm drops by a factor of 2 in this case with a discontinuity ($L_1^7 = 0.002604162$, $L_1^8 = 0.001302088$). The stair-stepping across the discontinuity is hardly evident on the full scale using $p = 8$ ($n = 256$ segments), as seen in Fig. 10. There are no overshoots or ringing in the series representation. An important detail here is to accurately approximate the integral where the discontinuity crosses a segment. The variation of the coefficients A_n in Figs. 9(b) and 10(b) show a constant magnitude within a family p and a rapid drop in magnitude as p increases.

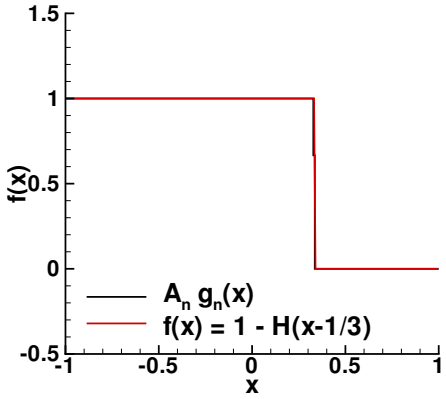


(a) $f(x)$ compared to $\sum_{n=1}^{64} A_n g_n(x)$

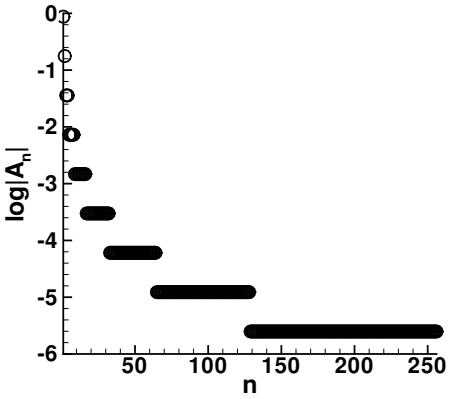


(b) Series coefficients

Figure 9. Representation of $f(x) = 1 - H(x - 1/3)$ by series $\sum_{n=1}^{64} A_n g_n(x)$.



(a) $f(x)$ compared to $\sum_{n=1}^{256} A_n g_n(x)$



(b) Series coefficients

Figure 10. Representation of $f(x) = 1 - H(x - 1/3)$ by series $\sum_{n=1}^{256} A_n g_n(x)$.

D. Series Expansion of Function Reciprocal

If a function $f(x)$ is represented by the truncated series

$$\sum_{n=1}^{2^p} A_n g_n(x) \tag{14}$$

over the interval $x_a \leq x \leq x_b$ and its reciprocal, $r(x) = 1/f(x)$, is represented by the series

$$\sum_{m=1}^{2^p} B_m g_m(x) \tag{15}$$

on the same interval then the coefficients B_m can be computed as a function of coefficients A_n as follows:

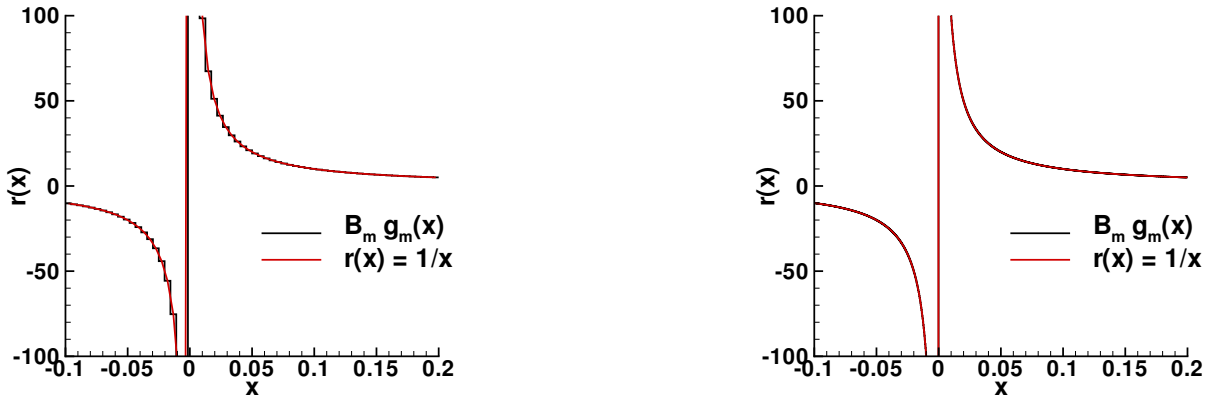
$$B_m = \left[A_{\mathcal{P}(n,m)}\right]^{-1} \delta(n,1)(x_b - x_a) \tag{16}$$

where

$$\left[A_{\mathcal{P}(n,m)}\right] = \begin{bmatrix} A_{\mathcal{P}(1,1)} & A_{\mathcal{P}(1,2)} & A_{\mathcal{P}(1,3)} & \cdots & A_{\mathcal{P}(1,2^p)} \\ A_{\mathcal{P}(2,1)} & A_{\mathcal{P}(2,2)} & A_{\mathcal{P}(2,3)} & \cdots & A_{\mathcal{P}(2,2^p)} \\ A_{\mathcal{P}(3,1)} & A_{\mathcal{P}(3,2)} & A_{\mathcal{P}(3,3)} & \cdots & A_{\mathcal{P}(3,2^p)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{\mathcal{P}(2^p,1)} & A_{\mathcal{P}(2^p,2)} & A_{\mathcal{P}(2^p,3)} & \cdots & A_{\mathcal{P}(2^p,2^p)} \end{bmatrix} \tag{17}$$

and $\mathcal{P}(n,m)$ refers to the multiplication table shown originally in Table 1 with the mapping explicitly shown in Table 4 of Sec.VIII (Appendix B). Note that the inverse of this Walsh symmetric matrix can be computed in $O(N\log(N))$ operations as shown in Appendix B.

As an example, the series expansion for $r(x) = 1/x$ is presented in Fig. 11 over the interval $-0.1 \le x \le 0.2$ as derived from the expansion for $f(x) = x$ using Eq. 16.



(a) $p = 6$, 64 segments

(b) $p = 10$, 1024 segments

Figure 11. Representation of $r(x) = 1/x$ derived from the series expansion for $f(x)$=x.

E. Integrals of Series Expansions

If a function is represented by Eq. 8, its derivative with respect to x is not explicitly represented by the derivative of its component parts in the summation. The basis functions $g_n(x)$ have 0 slope on the interior of each segment and infinite slope at segment boundaries. For this reason, in dealing with any equation of mathematical physics involving derivatives of various orders, the derivative of highest order may be represented by Eq. 8 and subsequent lower-order derivatives of the function are obtained by integration. Thus,

$$\frac{\partial f}{\partial x}(x) = f_x(x) = \sum_{n=1}^{\infty} A_n g_n(x) \tag{18}$$

and

$$f(x) = f(x_a) + \int_{x_a}^{x} \sum_{n=1}^{\infty} A_n g_n(s) ds = f(x_a) + \sum_{n=1}^{\infty} A_n \int_{x_a}^{x} g_n(s) ds \tag{19}$$

An integrating matrix $\chi(n,m)$ is defined such that

$$\int_{x_a}^{x} g_n(s) ds = (x_b - x_a) \sum_{m=1}^{\infty} \chi(n,m) g_m(x) \tag{20}$$

on the interval $x_a \le x \le x_b$. The factor $(x_b - x_a)$ is introduced to keep $\chi(n,m)$ dimensionless. The

coefficients $\chi(n,m)$ are given by

$$\chi(n,m) = (x_b - x_a)^{-3/2} \sum_{i=1}^{m} (-1)^{(i+1)} \int_{x_i}^{x_{i+1}} \left[\int_{x_a}^{x} g_n(s) ds \right] dx \tag{21}$$

where the x_i are defined in Eq. 11. The first 16 rows and columns of the integrating matrix are presented in Eq. 22. It is seen to be sparse; the occurrence of non-zero elements is easily defined so that integrating transforms need not be especially costly, even for large n.

$$\chi(n,m) = \begin{bmatrix}
\frac{1}{2} & \frac{-1}{4} & 0 & \frac{-1}{8} & 0 & 0 & 0 & \frac{-1}{16} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{32} & \cdots \\
\frac{1}{4} & 0 & \frac{-1}{8} & 0 & 0 & 0 & \frac{-1}{16} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{32} & 0 & \cdots \\
0 & \frac{1}{8} & 0 & 0 & 0 & \frac{-1}{16} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{32} & 0 & 0 & \cdots \\
\frac{1}{8} & 0 & 0 & 0 & \frac{-1}{16} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{32} & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & \frac{1}{16} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{32} & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & \frac{1}{16} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{32} & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & \frac{1}{16} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{32} & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
\frac{1}{16} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & 0 & \frac{1}{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & \frac{1}{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & \frac{1}{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & \frac{1}{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & \frac{1}{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
\frac{1}{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{bmatrix} \tag{22}$$

It is now possible to represent $f(x)$ as

$$f(x) = f(x_a) + (x_b - x_a) \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} A_n \chi(n,m) g_m(x) = \sum_{m=1}^{\infty} \tilde{A}_m g_m(x) \tag{23}$$

where

$$\tilde{A}_m = (x_b - x_a) \chi^T(m,n) A_n + \delta_{1m} (x_b - x_a)^{1/2} f(x_a) \tag{24}$$

F. Derivatives of Series Expansions

A differentiation operation — computing the series representation of $\frac{\partial f}{\partial x}$ from the series representation for $f(x)$ — may now be derived from Eq. 24. Differentiation of a function represented by a series truncated to 2^p terms is implemented as follows. Rewrite Eq. 24 using the transpose of the integrating matrix to obtain

$$\tilde{A}_m = (x_b - x_a) \chi^T(m,n) A_n + \delta_{1m} (x_b - x_a)^{1/2} f(x_a) \tag{25}$$

where repeated index n indicates summation. Recall that $\tilde{A}_m$ are the coefficients of the series expansion for $f(x)$ in Eq. 23 and A_n are the coefficients for the series expansion of $f_x(x)$ in Eq. 18. If one truncates the series representations at $n = 2^p$ in Eq. 25, then the inverse of the transpose of the integrating matrix may be calculated. The truncated series coefficients for $f_x(x)$ are now derived from the truncated series coefficients of $f(x)$.

$$A_n = (x_b - x_a)^{-1} [\chi^T(m,n)]^{-1} \left[\tilde{A}_m - \delta_{1m} (x_b - x_a)^{1/2} f(x_a) \right] \tag{26}$$

$$A_n = (x_b - x_a)^{-1} \mathcal{D}^p(n,m) \left[\tilde{A}_m - \delta_{1m} (x_b - x_a)^{1/2} f(x_a) \right] \tag{27}$$

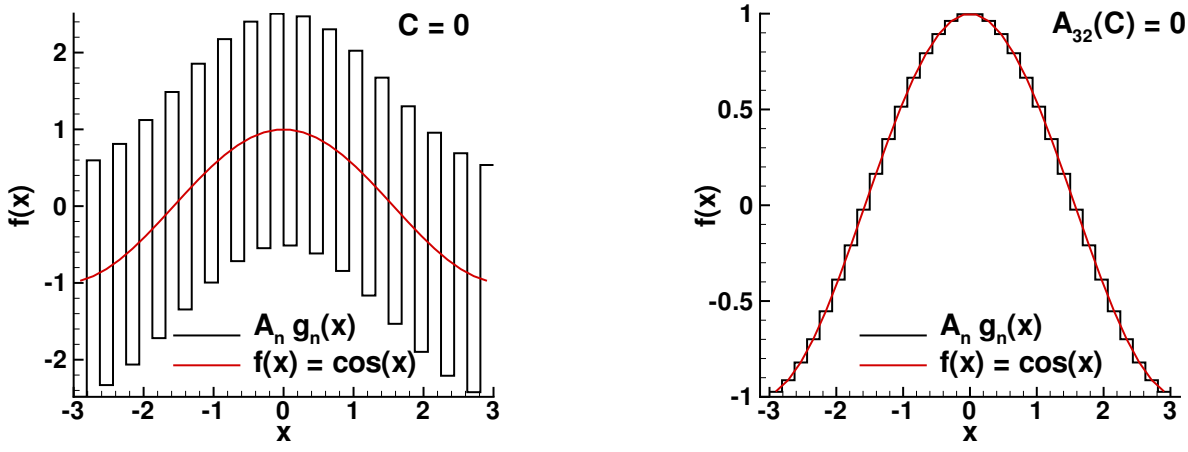
where summation over repeated index m is implied and $\mathcal{D}^p = [\chi^T(m,n)]^{-1}$ with $1 \le m \le 2^p$ and $1 \le n \le 2^p$.

Note that the elements of the integrating matrix χ are not a function of the truncation level of the series but the elements of the differentiating matrix $\mathcal{D}^p$ are a function of the truncation level $n = 2^p$. For example,

$$\mathcal{D}^2 = \begin{bmatrix} 0 & 0 & 0 & -8 \\ 0 & 0 & -8 & 0 \\ 0 & 8 & 0 & -16 \\ 8 & 0 & 16 & 32 \end{bmatrix} \quad \mathcal{D}^3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & -16 \\ 0 & 0 & 0 & 0 & 0 & 0 & -16 & 0 \\ 0 & 0 & 0 & 0 & 0 & -16 & 0 & 0 \\ 0 & 0 & 0 & 0 & -16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16 & 0 & 0 & 0 & -32 \\ 0 & 0 & 16 & 0 & 0 & 0 & -32 & 0 \\ 0 & 16 & 0 & 0 & 0 & 32 & 0 & -64 \\ 16 & 0 & 0 & 0 & 32 & 0 & 64 & 128 \end{bmatrix} \tag{28}$$

$$\mathcal{D}^4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -32 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -32 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -32 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -32 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -32 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -32 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -32 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -64 \\ 0 & 0 & 0 & 0 & 0 & 0 & 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -64 & 0 \\ 0 & 0 & 0 & 0 & 0 & 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -64 & 0 & 0 \\ 0 & 0 & 0 & 0 & 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -64 & 0 & 0 & 0 \\ 0 & 0 & 0 & 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 64 & 0 & 0 & 0 & -128 \\ 0 & 0 & 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 64 & 0 & 0 & 0 & -128 & 0 \\ 0 & 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 64 & 0 & 0 & 0 & 128 & 0 & -256 \\ 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 64 & 0 & 0 & 0 & 128 & 0 & 256 & 512 \end{bmatrix} \tag{29}$$

The sum of elements in any column is 2^{p+1}, the sum of elements in all but the last row is -2^{p+1}, and the sum of elements in the last row is $2^{2p+2} - 2^{p+1}$. The only non-zero element of the first column of the differentiating matrix of any order p is in the last row.



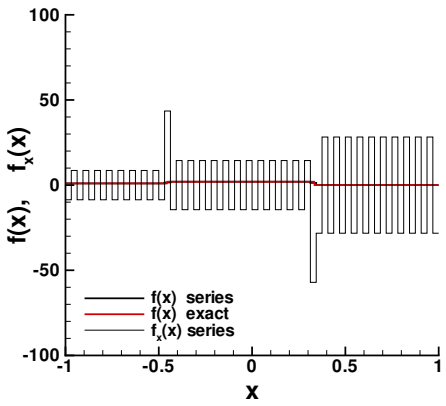
(a) $C = 0$.

(b) $A_{32}(C) = 0$.

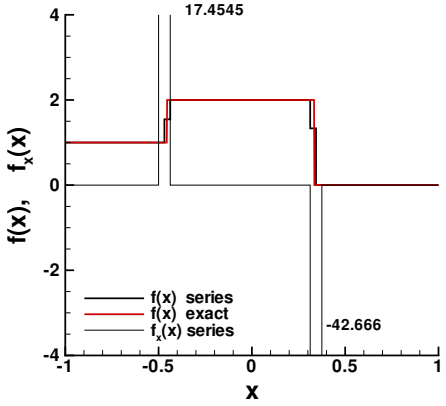
Figure 12. Series representation of $\cos(x)$ computed as the derivative of the series representation of $\sin(x)$ on the interval $-3 \leq x \leq 3$ for $p = 5$.

For smooth functions it is observed that the presence of $f(x_a)$ serves to drive the value of A_{2^p} towards zero in Eq. 27. As an example of this effect, consider again the sample problem of computing the series

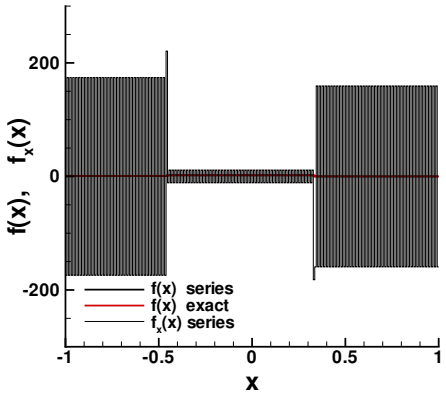
representation of $\cos(x)$ as the derivative of the series representation of $\sin(x)$ using the differentiating matrix. However, in this case change the interval to $-3 \le x \le 3$. If the contribution of $f(x_a)$ is replaced by an arbitrary constant C and C is set to zero, then the differentiated series representation yields the comparison shown in Fig. 12(a) with $L_1^5 = 1.5097$ and $A_{2^p} = -3.698$. If one chooses C such that $A_{2^p} = 0$, then the comparison in Fig. 12(b) is obtained with $L_1^5 = 0.00091067$. If $C = f(x_a) = \sin(-3)$ then $A_{2^p} = -0.0108$ and $L_1^5 = 0.004425$. It should be noted that if the definition of L_1^p in Eq. 13 used the difference of the series representation relative to the function rather than the absolute value of the difference then the representations in Figs. 12(a and b) have identical norms equal to 0.000069322.



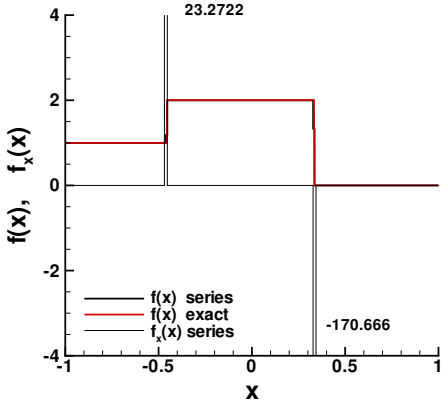
(a) $p = 6$, full approximation



(b) $p = 6$, limited approximation



(c) $p = 8$, full approximation



(d) $p = 8$, limited approximation

Figure 13. Series representation of Eq. 30 and its derivative using Eq. 27.

If the function $f(x)$ has a discontinuity or it is inadequately resolved by its series representation $\tilde{A}_n g_n(x)$, then the representation of its derivative $f_x(x)$ by the series $A_n g_n(x)$ through Eq. 27 is observed to oscillate around the exact solution. This behavior is illustrated in the left side of Fig. 13 where the function has discontinuities at x_1 and x_2.

$$f(x) = \begin{cases} 1 & \text{for} \quad x_a \le x < x_1 \\ 2 & \text{for} \quad x_1 \le x < x_2 \\ 0 & \text{for} \quad x_2 \le x \le x_b \end{cases} \qquad x_1 = -\frac{5}{11} \quad x_2 = \frac{1}{3} \tag{30}$$

In Fig. 13, the original function is shown in red, its series representation is shown as a thick black line, and the derivative of the series representation from Eq. 27 is shown by the thin black line. The derivative of the

original function equals zero everywhere except at the points of discontinuity where

$$\lim_{\epsilon \to 0} \frac{f(x_1 + \epsilon) - f(x_1 - \epsilon)}{2\epsilon} \to \infty \qquad \text{and} \qquad \lim_{\epsilon \to 0} \frac{f(x_2 + \epsilon) - f(x_2 - \epsilon)}{2\epsilon} \to -\infty \tag{31}$$

In all cases, the series representation of the original function is good, consistent with the earlier example of the Heaviside function showing no overshoots or Gibb's phenomena. However, the series representation of the derivative of the function exhibits oscillations of constant magnitude over each piecewise constant segment of the original function that extend all the way to the left and right boundaries. The mean value of the oscillations equals zero over these piecewise constant segments. A representation that averaged this result over a segment length equal to $2dx_p$ provides an excellent result. This averaging is easily accommodated by truncating the highest frequency components of the series from $2^{p-1} < n \leq 2^p$. The right side of Fig. 13 shows this limited series representation of the derivative. The result is implemented by replacing $\mathcal{D}^p$ in Eq. 27 with $\mathcal{D}^p_{lim}$ where

$$\mathcal{D}^p_{lim}(n,m) = \begin{cases} \mathcal{D}^p(n,m) & \text{for} \quad n \leq 2^{p-1} \\ 0 & \text{for} \quad n > 2^{p-1} \end{cases} \tag{32}$$

The "spike" over the discontinuity has width equal to dx_p for the original formulation and is twice as wide in the limited formulation. As p increases the width of the spike decreases and its height grows. Limiting coarsens the resolving ability of the original series by a factor of 2. The limiting is applied uniformly over the domain and will affect both smooth (well resolved) and discontinuous (poorly resolved) functions.

III. Truncation and Prolongation

Truncation (filtering) refers to the reduction or elimination of high-frequency (small-wavelength) structures due to noise or incipient instability from a function represented by a Walsh series. Prolongation (sharpening) refers to the insertion of high-frequency (small-wavelength) structure into a Walsh series to smoothly replace eliminated noise or to serve as the first step of a wave-component-sequencing algorithm (equivalent to a grid-sequencing algorithm in a finite-difference context). Truncation is effective in reducing Gibbs phenomena in discontinuous solutions and in suppressing instabilities in high-order formulations of derivatives.

Consider the baseline distribution $q_0(t_n) = \cos(2\pi t_n)$ with $1 \leq n \leq N$ and $t_n = (n - 0.5)/N$ where $N = 2^{p_\tau}$. Use a random number generator to add noise to the distribution such that $q_f(t_n) = q_0(t_n) + 0.1(rn(t_n) - 0.5)$ where $rn(t_n)$ is a random number in the interval $0 \leq rn(t_n) \leq 1$. Let

$$q_f(t) = \sum_{n=1}^{2^{p_\tau}} A^f_n g_n(t) \tag{33}$$

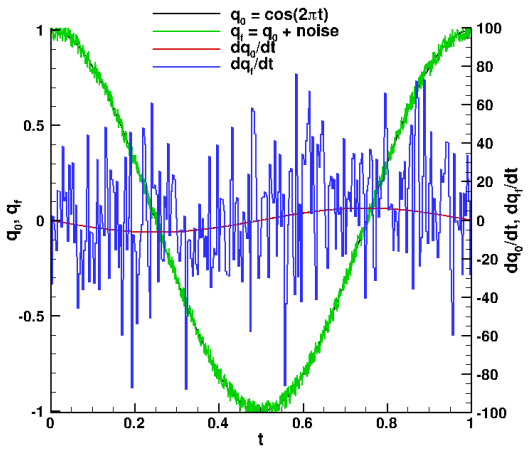
Figure 14(a) shows the noisy, unfiltered distribution in green plotted over the smooth baseline distribution in black. It also shows the time derivative of the noisy, unfiltered solution in blue plotted over the time derivative of the baseline solution in red. A single level of filtering ($n_f = 1$) is implemented by setting the wave components in the highest group number (recall Eq. 7) to zero ($p = 10$ in this example). This action effectively averages adjacent values of $q_f(t_{2i-1})$ and $q_f(t_{2i-1})$ across these time intervals defined by $1 \leq i \leq 2^{10-1}$, but with fewer operations. Two levels of filtering ($n_f = 2$) are implemented by setting the wave components in the two highest group numbers to zero ($p = 10$ and $p = 9$ in this example). In like manner, this action effectively averages four adjacent values of $q_f(t_{4i-1})$, $q_f(t_{4i-2})$, $q_f(t_{4i-3})$, and $q_f(t_{4i})$ across these time intervals defined by $1 \leq i \leq 2^{10-2}$. Each successive level of filtering (n_f) using function **trun** below effectively averages another factor of 2 intervals to form a larger, filtering interval.

$$\mathbf{trun}(q_f(t), n_f) = \sum_{n=1}^{2^{(p_\tau - n_f)}} A^f_n g_n(t) \tag{34}$$

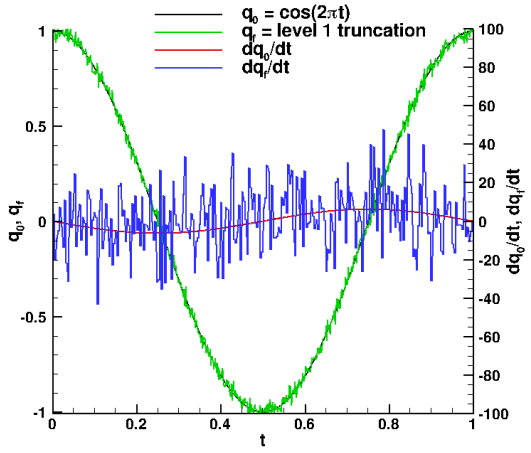
The baseline average value of $q_0(t)$ is accurately approximated at the center of the filtering interval if one assumes (1) the variation of the baseline solution is nearly linear across the filtering interval; and (2) the filtering interval is sufficiently long that noise fluctuations average to zero. In Fig. 14(b) a single level

of filtering creates a filtering interval of length $2^{-(10-1)}$ that shows marginal reduction of fluctuating noise. In Fig. 14(c), five levels of filtering creates a filtering interval of length $2^{-(10-5)}$ that averages noise over 2^5 adjacent segments. Note that $\mathbf{trun}(q_f(t_n), 5)$ (in green) shows a 32 stair-step pattern that is flat over each set of adjacent 2^5 intervals. The baseline function $q_0(t_n)$ (in black) is nearly linear over each of these steps.

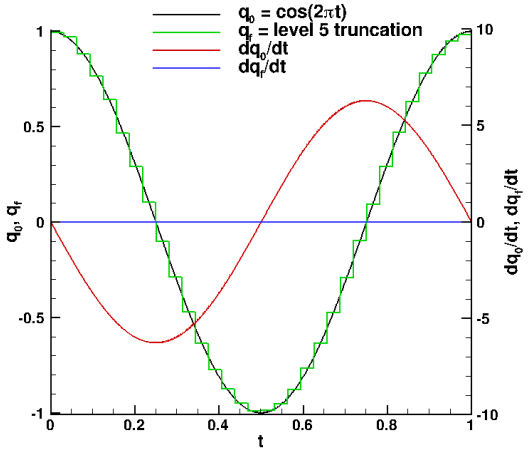
The five-level filtering in Fig. 14(c) has discarded $2^{10} - 2^{10-5}$ wave components (all but 32 out of the original 1024). The time derivative of this piecewise constant distribution from Eq. 27 is zero as shown by the blue line in the figure. (Note that Eq. 27 is not equivalent to a conventional Taylor series representation of a derivative in which a divided difference across the stair riser will always yield a large, non-zero result.) One would like to recover a smooth variation across the original 1024 segments without having to resort to generating a curve fit through the remaining 32 segments.



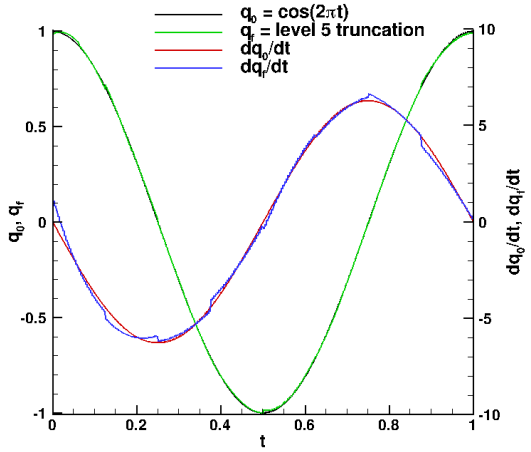
(a) q_f, unfiltered



(b) q_f, filter level 1, with prolongation



(c) q_f, filter level 5, without prolongation



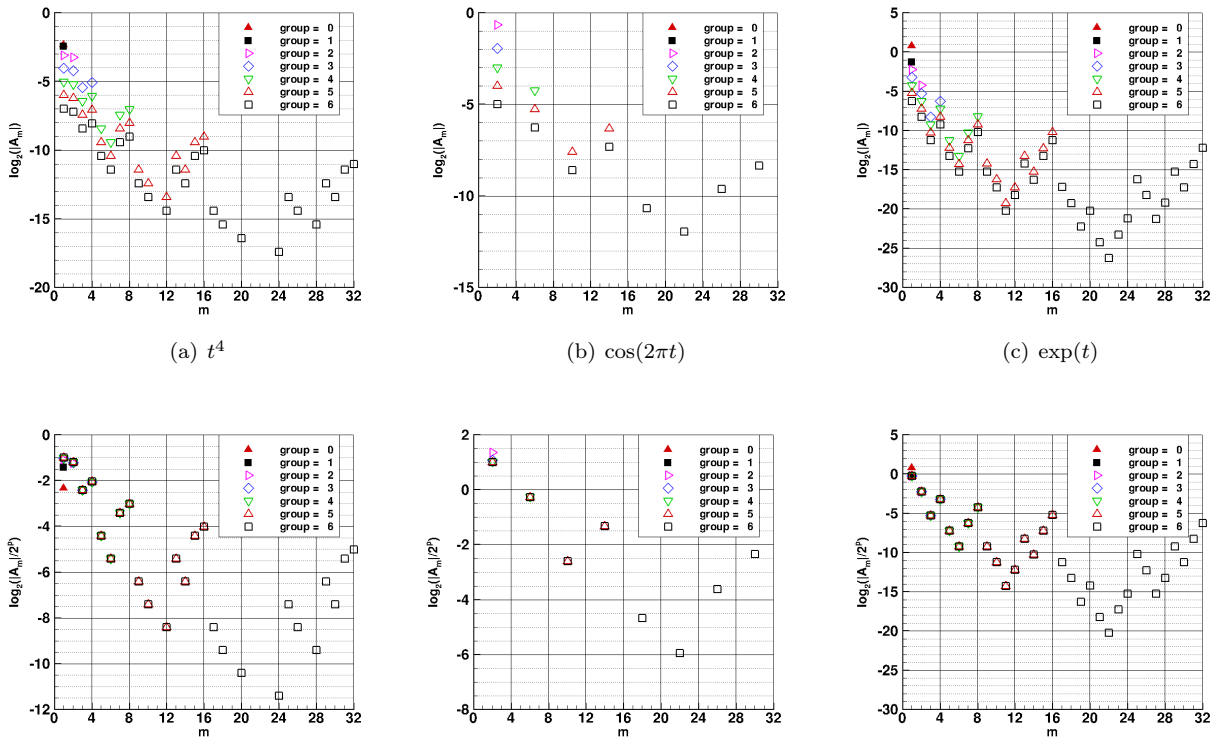
(d) q_f, filter level 5, with prolongation

Figure 14. The use of filtering and prolongation to reduce random noise.

The fractal nature of the Walsh basis functions[1] introduces repeated patterns in the wave component magnitudes that can be exploited for the purpose of prolongation — replacing the truncated, high-wave-number components of the noisy solution with components that yield a smooth solution over the filtering interval. The repeated patterns are suggested in Figs. 7(b), 8(b), 9(b), and 10(b). Knowing to expect patterns makes it easier to find them. To this end, plot the wave components using a mapping that retains group number identity as follows:

$$m = 2^p + 1 - n \quad \text{for} \quad 2^{p-1} < n \leq 2^p \tag{35}$$

Equation 35 plots wave components for each group in reverse sequency order starting from $m = 1$. It enables the superposition of wave components to better reveal underlying patterns as demonstrated in Figs. 15(a–c). (Information regarding the sign of a wave component is lost here in order to focus on component magnitude using logarithms.) The base 2 logarithm of the Walsh series components of three simple functions on the interval $0 \leq t \leq 1$ are plotted as a function of group number using Eq. 35. Elements of each group are plotted with a different symbol and color. These plots are referred to as fractal fingerprints (FFP) of a function over an interval. A one-unit change in these figures between component values in adjacent groups indicates a factor of 2 difference in magnitude. If a function is singularity-free over an interval, then the magnitude of adjacent groups differ by approximately a factor of 2. This trend is highlighted in Figs. 15(d–f) in which the FFP is plotted after dividing the wave components in group p by a factor 2^p.



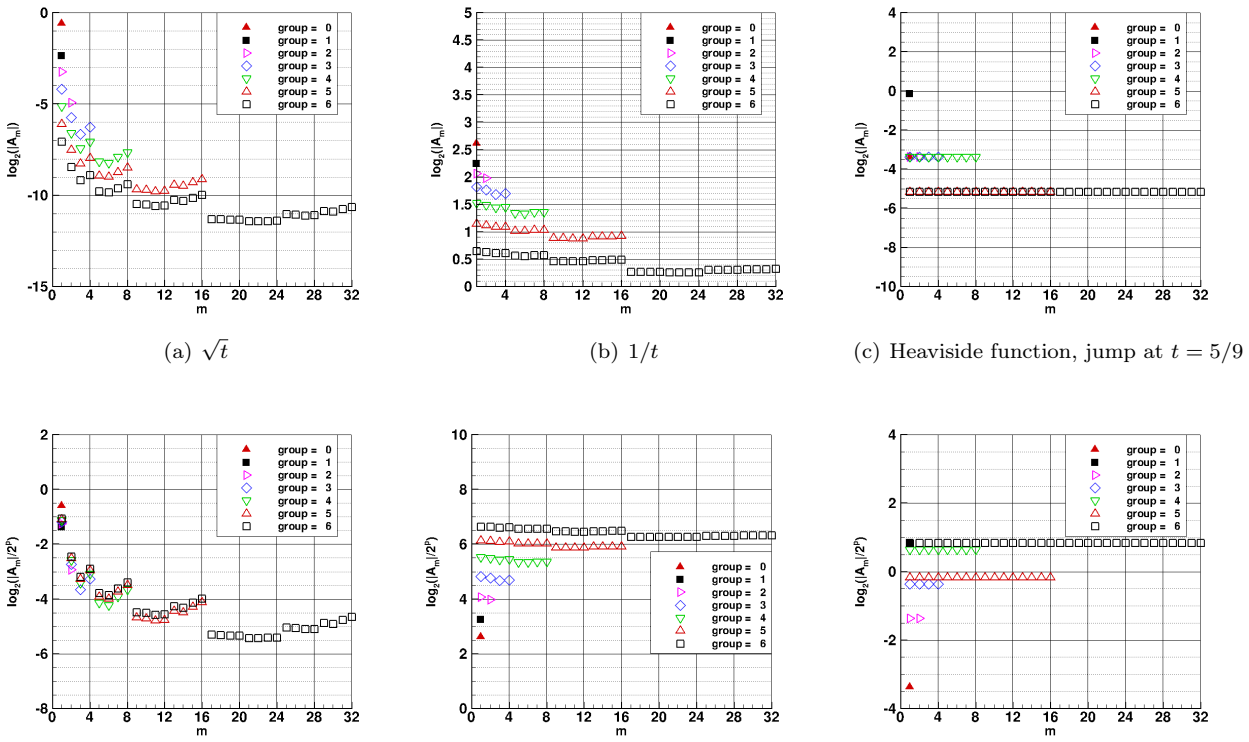
(a) t^4 (b) $\cos(2\pi t)$ (c) $\exp(t)$

(d) t^4, scaled (e) $\cos(2\pi t)$, scaled (f) $\exp(t)$, scaled

Figure 15. The fractal fingerprint (FFP) of Walsh series wave components.

Examples of the FFP for functions that include singularities in the interval $0 \leq t \leq 1$ are presented in Fig. 16. Note that successive groups still show repeating patterns; however, the magnitude of successive groups cannot be described by a single factor of 0.5. If the singularity is moved outside of the domain (for example, $q(t) = 1/(t + 0.1)$), the FFP shows good recovery of the 0.5 scale factor between groups.

The repeated patterns in the FFPs are used as a first approximation to increase the resolution of a Walsh series representation of a function to the next highest group level. It has already been noted that the left half of the highest group number is already well-approximated by simply setting it equal to half the value of the preceding group. It will next be observed that polynomials of degree p exhibit a repeating pattern in their FFP in which the right half of the highest group number can be predicted from the values in the left half. (These repeated patterns for polynomials are consistently observed but not yet proved.) These patterns are exploited to fully construct a good approximation to wave components in group p completely from wave components in group $p - 1$ — a bootstrap approach to prolongation for the purpose of wave component sequencing or smoothing the structure that had been truncated in a filtering operation.



(a) $\sqrt{t}$ (b) $1/t$ (c) Heaviside function, jump at $t = 5/9$

(d) $\sqrt{t}$, scaled (e) $1/t$, scaled (f) Heaviside function, jump at $t = 5/9$, scaled

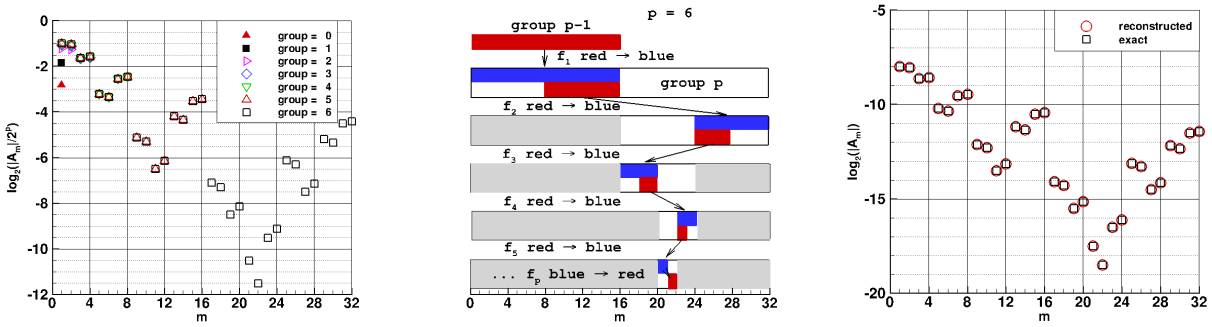
Figure 16. The fractal fingerprint (FFP) of Walsh series wave components for functions with singularities in the domain.

The algorithm to define elements of group p from a Walsh series that terminates at group $p-1$ is illustrated in Fig. 17. In Fig. 17(a), the scaled FFP for $q(t) = t^6$ is used to discern repeated patterns. First, note that the scaled elements of group 6 (black squares) in the left half domain match the scaled elements of group 5 (red delta). Next, note that the distribution in the far right quarter of group 6 (black squares, $25 \leq m \leq 32$) matches the distribution in the second quarter of group 6 ($9 \leq m \leq 16$) but is displaced vertically by one unit (a factor of 2 smaller). Next, observe that the distribution in the first half of this latest group ($25 \leq m \leq 28$) is repeated at the front eighth of the second half ($17 \leq m \leq 20$) but is again displaced vertically by one unit (a factor of 2 smaller). This fractal algorithm of transferring half of the last defined pattern to the opposite end of the remaining undefined domain and reducing the magnitude by factor 2 continues until the entire group is defined.

A schematic of this prolongation algorithm is illustrated in Fig. 17(b). The logical flow goes from the top to the bottom of the figure. Red blocks represent existing patterns. Blue blocks represent target locations where the pattern in red is transferred with magnitude reduced by a factor f_i in step i of the algorithm. Gray blocks represent locations in the new group that have been defined previously. White areas have yet to be filled in. Each step takes half of the latest defined subdomain and transfers it to the opposite end of the undefined domain, reducing magnitude by a factor f_i until the entire group is defined. The reduction factor f_i is currently fixed at 0.5 based on the polynomial template function. Other values may be derived for other template functions, but such an option is beyond the scope of this paper. The current algorithm provides good approximations to the prolongation of singularity-free functions across a domain; errors tend to build on the last reconstruction steps, but these coefficients tend to be orders of magnitude smaller than other coefficients in the Walsh series.

Figure 17(c) confirms that the reconstructed group 6 coefficients (red circles) obtained by prolongation from the group 5 coefficients match the exact group 6 coefficients (black squares) obtained from a Fast Walsh

Transform of $q(t) = t^6$. The sequential application of this prolongation algorithm from group 5 to group 10 was used to recover a smooth baseline solution in Fig. 14(d) (green curve) from the truncated solution in Fig. 14(c). The time derivative of this reconstructed solution (blue curve in Fig. 14(d)) shows some small discontinuities associated with approximate nature of the bootstrap algorithm for general functions.



(a) Scaled FFP for $q(t) = x^6$ including groups $0 \le p \le 6$. The elements in group 6, black squares, are truncated from this series.

(b) Algorithm schematic to bootstrap group p using only elements of group $p-1$.

(c) Comparison of reconstructed elements to original, untruncated elements of group 6.

Figure 17. Overview of prolongation algorithm applied to Walsh series representation of $q(t) = x^6$ starting with 6 groups.

IV. Application Examples

Three test cases provide examples on the use of the Walsh functions to solve partial differential equations. All of the sample problems are time-dependent. Two are non-linear. One involves a system of equations. All solutions utilize the Fast Walsh Transform (FWT) documented in Appendix A to produce the discrete solution values shown in the figures from the Walsh function wave components.

The formulations in Section A are completely executed in Walsh wave component space. Boundary conditions are implicitly included in the formulation of derivatives using Eq. 27. The advantage of this approach is a robust, quadratic convergent algorithm. The Jacobians of every wave component with respect to changes in every other wave component and with respect to boundary conditions are automatically implemented using operator overloading and the chain rule. The disadvantage of this approach is that Jacobian matrices are not sparse and require large amounts of memory. Furthermore, two-point boundary value problems require the introduction of a second-derivative dissipation in order to accommodate two boundary conditions through Eq. 27.

The formulations in Section C replace derivative formulations from Eq. 27 with traditional, finite-difference formulations derived from Taylor series. Finite-difference operations are enabled by a Walsh function **shift** in Eq. 36.

$$\textbf{shift}(F, inc, F_{bc}) = \begin{cases} FWT(F) \rightarrow f \\ FWT(F_{bc}) \rightarrow f_{bc} \\ f_i \rightarrow f_{i+inc} \\ \text{if} \quad inc = \ 1 \quad \text{then} \quad f_{bc} \rightarrow f_1 \\ \text{if} \quad inc = -1 \quad \text{then} \quad f_{bc} \rightarrow f_{N_i} \\ FWT(f) \rightarrow \textbf{shift} \end{cases} \tag{36}$$

This equation uses (1) an FWT from wave components F_α to discrete components f_i; (2) a shift of the discrete indices by ± 1; (3) insertion of the appropriate boundary condition at the first or last index, respectively; and (4) an FWT back to wave components F_α from discrete components f_i. This approach enables explicit

inclusion of boundary conditions and does not require storage of the full Jacobian. More relaxation steps are required per time step than with the implicit formulation but convergence is orders of magnitude faster than the implicit formulation for problems with more than 1024 degrees of freedom.

A. Test Case Formulation: Implicit Method

1. Advection

The linear, first-order advection equation is

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \tag{37}$$

where $c = 1$ is the wave speed and $u(x,0) = u_0(x)$ is the initial condition on the domain $0 \leq x \leq 1$. The initial profile moves to the right. A periodic boundary condition is applied so that as the profile exits the right boundary it reemerges from the left. The test is designed to provide an easily evaluated metric to measure how well the initial profile is preserved. The initial profile in the demonstration case (see Fig. 18) is a sawtooth defined by

$$u_0(x) = \begin{cases} 0 & \text{for} \quad 0 \leq x < \frac{1}{4} \\ 1 - 4|x - \frac{1}{2}| & \text{for} \quad \frac{1}{4} \leq x < \frac{3}{4} \\ 1 & \text{for} \quad \frac{3}{4} \leq x \leq 1 \end{cases} \tag{38}$$

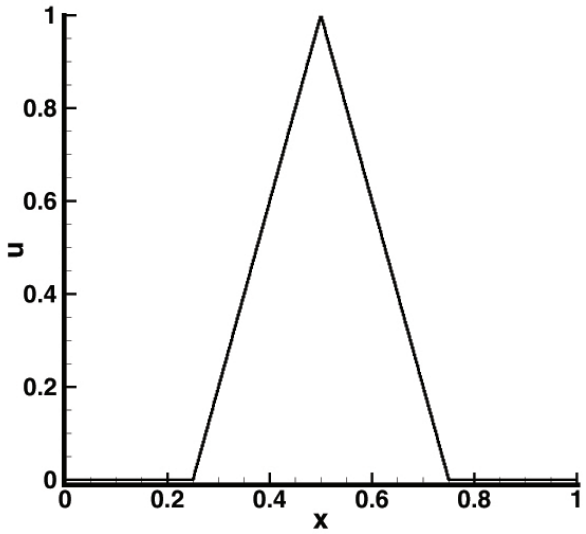


Figure 18. The initial profile for the advection test case, $u_0(x)$.

The basis function series representation of $u(x,t)$ is

$$u(x,t) = \sum_{j=1}^{N_\alpha} \sum_{i=1}^{N_\tau} U(i,j) g_i(t) g_j(x) \tag{39}$$

The indices i and τ indicate a basis function component in time t. The indices j and α indicate a basis function component in space x. The coefficient $U(i,j)$ should never be interpreted as the value of $u(x,t)$ at a specific point in the space-time domain. Rather, it indicates the magnitude of a basis function component of the global solution for $u(x,t)$ across the entire space-time domain. The number of basis function components considered in the x domain is $N_\alpha = 2^{p_\alpha}$. The number of basis function components considered in the t domain

is $N_\tau = 2^{p_\tau}$. It is important to include every member of a family p_α and p_τ in non-linear problems to capture every basis function component that results from a multiplication of two or more functions.

The basis function series representation of $\frac{\partial u(x,t)}{\partial x} = u_x(x,t)$ is

$$u_x(x,t) = \sum_{j=1}^{N_\alpha} \sum_{i=1}^{N_\tau} U_x(i,j) g_i(t) g_j(x) \tag{40}$$

The coefficients $U_x(i,j)$ may be expressed as a function of the coefficients $U(i,j)$ using the integrating matrix from Eq. 24.

$$U(i,\alpha) = \mathcal{L}_x \sum_{j=1}^{N_\alpha} U_x(i,j) \chi_{p_\alpha}(j,\alpha) + \delta(\alpha,1) U_{x0}(i) \tag{41}$$

The factor $\mathcal{L}_x$ is the length of the x domain. It is convenient in the case of a two-point boundary value problem to replace the last term of Eq. 24 with a constant of integration that will be solved implicitly with the other expansion coefficients. The constant of integration $U_{x0}(i)$ is most generally a function of time with basis function component i. The function δ is Kronecker delta. Einstein summation notation is adopted to derive $U_x(i,j)$ as an explicit function of $U(i,j)$ and $U_{x0}(i)$.

$$U_x(i,j) = \mathcal{L}_x^{-1} \left[\mathcal{D}_{p_\alpha}(j,\alpha) U(i,\alpha) - \mathcal{D}_{p_\alpha}(j,1) U_{x0}(i) \right] \tag{42}$$

Here $\mathcal{D}_{p_\alpha}(j,\alpha)$ equals the inverse of $\chi^T_{p_\alpha}(\alpha,j)$.

The basis function series representation of $\frac{\partial u(x,t)}{\partial t} = u_t(x,t)$ is

$$u_t(x,t) = \sum_{j=1}^{N_\alpha} \sum_{i=1}^{N_\tau} U_t(i,j) g_i(t) g_j(x) \tag{43}$$

The coefficients $U_t(i,j)$ are expressed as a function of the coefficients $U(i,j)$ and constant of integration $U_{t0}(j)$ using the same algorithm defined in Eqs. 41–42. Therefore,

$$U_t(i,j) = \mathcal{L}_t^{-1} \left[\mathcal{D}_{p_\tau}(i,\tau) U(\tau,j) - \mathcal{D}_{p_\tau}(i,1) U_{t0}(j) \right] \tag{44}$$

where the factor $\mathcal{L}_t$ is the length of the t domain (the time step). A simple, first-order, backward difference approximation may also be expressed for the case $p_\tau = 0$ and $i = 1$.

$$U_t(1,j) = \mathcal{L}_t^{-1} \left[U(1,j) - U_{t0}(j) \right] \tag{45}$$

A residual equation representation of Eq. 37 for the Walsh function component magnitudes is

$$R(i,j) = U_t(i,j) + c U_x(i,j) \tag{46}$$

The amplitude of wave components $U(i,j)$, the dependent variables, are solved such that each wave component $R(i,j)$ goes to zero. A FORTRAN module, WALSH_TOOLS,[2] simplifies this solution.

Code for Eq. 37 using WALSH_TOOLS is written

```
if(N_tau > 1)then
  resw(1) = intt(q1w(m),fa=q10w(m),diff=.true.)                    &
          + wave_speed*intx(q1w(m),fa=q1aw(m),diff=.true.)
else
  resw(1) = (q1w(m) - q10w(m))/dt                                  &
          + wave_speed*intx(q1w(m),fa=q1aw(m),diff=.true.)
end if
```

All variable names in this example ending in letter `w` are of type `walsh` and include wave components and their Jacobians with respect to the wave components of the dependent variable. The function call `intx(q1w(m),fa=q1aw(m),diff=.true.)` computes $U_x(i,j)$ in Eq. 42. In like manner, the function call `intt(q1w(m),fa=q10w(m),diff=.true.)` computes $U_t(i,j)$ in Eq. 44. For the case with $p_\tau = 0$ (`N_tau = 1`) the expression `(q1w(m) - q10w(m))/dt` computes $U_t(i,j)$ in Eq. 45. The functions `intt` and `intx` implicitly include the differentiating matrices $\mathcal{D}_{p_\tau}(i,\tau)$ and $\mathcal{D}_{p_\alpha}(j,\alpha)$, respectively. Note that for this linear, first-order equation, only a single Walsh function dependent variable `q1aw(m)` is required to satisfy a boundary condition in space, and `q10w(m)` is required to satisfy an initial condition in time. Also note that index `m` refers to a subdomain. The solutions that follow use only one domain.

2. Burgers Equation

The non-linear, second-order Burgers equation is

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial u^2}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2} \tag{47}$$

where the diffusivity $\nu \geq 0$. The initial condition is a linear function $u_0(x) = -x$ on the domain $-1 \leq x \leq 1$ with boundary conditions $u_a(t) = u(-1, t) = 1$ and $u_b(t) = u(1, t) = -1$.

Given these initial and boundary conditions, the solution profile evolves according to the local value of u. At any given time, points with a positive value of u move to the right and points with a negative value of u move to the left with speed u. This motion is dissipated as a function of diffusivity ν. Large values of ν evolve profiles that are nearly linear between the boundary values. Small values of ν evolve profiles with an abrupt, shock-like transition from 1 to –1 at $x = 0$. Examples will be provided in Sec. B2.

If the diffusivity is specified 0, then an artificial diffusion term must be included to maintain a stable computation while accommodating boundary conditions from both the left and right. The artificial dissipation is defined

$$\nu_0 = \frac{1}{2}(dx_p)^2 \left| \frac{\partial u}{\partial x} \right| \tag{48}$$

where dx_p is the smallest segment size used in the Walsh series representation of the solution. The use of a second-order, artificial dissipation here and in the Riemann problem that follows presents a simple solution for communicating information from opposite boundaries. Such communication was not an issue in the linear advection test case in which all waves travel from left to right. It is thought that a characterstic-based formulation of this problem could offer better accuracy as $\nu \to 0$, and the transition from $u = 1$ to $u = -1$ occurs over a length smaller than dx_p. For now, only the artificial dissipation is used to address this issue.

Code for Eq. 47 using WALSH_TOOLS[2] is written

```
if(N_tau > 1)then
  resw(1) = intt(q1w(m),fa=q10w(m),diff=.true.)                    &
          + intx(0.5_dp*q1w(m)**2-tauxw,fa=q1bw(m),diff=.true.)
else
  resw(1) = (q1w(m) - q10w(m))/dt                                  &
          + intx(0.5_dp*q1w(m)**2-tauxw,fa=q1bw(m),diff=.true.)
end if
```

Representation of the time derivative is exactly the same as discussed above for the advection equation. The shear term, `tauxw` $= \nu \partial u/\partial x$, was computed just above this block as

```
uxw = intx(q1w(m),fa=q1aw(m),diff=.true.)
if(nu==0._dp)then
  dx2 = 0.5_dp*dx_eff**2
  tauxw = dx2*absw(uxw)*uxw
else
  tauxw = nu*uxw
end if
```

Note that if $\nu = 0$, then the artificial dissipation is added as defined in Eq. 48. Also note that this second-order equation has two boundary conditions that must be engaged. The Walsh function dependent variables `q1aw(m)` used in the definition of `uxw` $= \partial u/\partial x$ and `q1bw(m)` used in the definition of $\partial(0.5u^2 - \nu \partial u/\partial x)/\partial x$, provide additional degrees of freedom required to satisfy the boundary conditions. The Walsh function dependent variable `q10w(m)` used in the definition of $\partial u/\partial t$ provides the additional degrees of freedom to satisfy the initial condition.

3. Riemann Problem

The Riemann problem in one dimension defines the compressible gas dynamic flow that follows the breaking of a virtual diaphragm separating two constant initial states. The solution typically involves the propagation of a shock, a contact discontinuity, and an expansion fan. A particular instance of the Riemann problem was defined by Sod[7] and is frequently used in the validation of computational fluid dynamics codes.

The one-dimensional, time-dependent, compressible gas dynamic equations are

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0 \tag{49}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial (p + \rho u^2)}{\partial x} = 0 \tag{50}$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial \rho u H}{\partial x} = 0 \tag{51}$$

$$p = (\gamma - 1)\rho e \tag{52}$$

$$e = E - \frac{u^2}{2} \tag{53}$$

$$H = E + \frac{p}{\rho} \tag{54}$$

The need for artificial dissipation was discussed in the previous section and is added in Eqs. 55 – 57.

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}\left[\rho u - \nu_1 \frac{\partial \rho}{\partial x}\right] = 0 \tag{55}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial}{\partial x}\left[p + \rho u^2 - \nu_2 \frac{\partial \rho u}{\partial x}\right] = 0 \tag{56}$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x}\left[\rho u H - \nu_3 \frac{\partial \rho E}{\partial x}\right] = 0 \tag{57}$$

The artificial diffusion coefficients defined in Eqs. 58 – 60 are of order $(dx_p)^2$.

$$\nu_1 = (dx_p)^2 \left|\frac{\partial \rho}{\partial x}\right| + (dx_p)^2 \tag{58}$$

$$\nu_2 = (dx_p)^2 \left|\frac{\partial \rho u}{\partial x}\right| + (dx_p)^2 \tag{59}$$

$$\nu_3 = (dx_p)^2 \left|\frac{\partial \rho E}{\partial x}\right| + (dx_p)^2 \tag{60}$$

The equations are solved on the domain $-1 \le x \le 1$ with $\gamma = 1.4$. Constant initial conditions on the left ($x < 0$) and right ($x > 0$) for the Sod test case[7] are given by

$$\begin{aligned} p_L &= 1 & p_R &= 0.1 \\ \rho_L &= 1 & \rho_R &= 0.125 \\ u_L &= 0 & u_R &= 0 \end{aligned}$$

Boundary conditions are given by

$$\begin{aligned} \frac{\partial \rho}{\partial x}(-1,t) &= 0 & \frac{\partial \rho}{\partial x}(1,t) &= 0 \\ \rho u(-1,t) &= 0 & \rho u(1,t) &= 0 \\ \frac{\partial \rho E}{\partial x}(-1,t) &= 0 & \frac{\partial \rho E}{\partial x}(1,t) &= 0 \end{aligned}$$

Code for Eqs. 55 – 57 using WALSH_TOOLS[2] is written

```
if(N_tau > 1)then
  resw(1) = intt(q1w(m),fa=q10w(m),diff=.true.)                    &
          + intx(q2w(m)-q1xw,fa=q1aw(m),diff=.true.)
  resw(2) = intt(q2w(m),fa=q20w(m),diff=.true.)                    &
          + intx(pw + rhou2w - q2xw,fa=q2aw(m),diff=.true.)
  resw(3) = intt(q3w(m),fa=q30w(m),diff=.true.)                    &
```

```
          + intx(q2w(m)*htw - q3xw,fa=q3aw(m),diff=.true.)
else
  resw(1) = (q1w(m) - q10w(m))/dt                                          &
          + intx(q2w(m)-q1xw,fa=q1aw(m),diff=.true.)
  resw(2) = (q2w(m) - q20w(m))/dt                                          &
          + intx(pw + rhou2w - q2xw ,fa=q2aw(m),diff=.true.)
  resw(3) = (q3w(m) - q30w(m))/dt                                          &
          + intx(q2w(m)*htw - q3xw,fa=q3aw(m),diff=.true.)
end if
```

The main difference between this code sample and previous ones in this section is that there are three dependent variables and three corresponding residual equations to be solved. The reciprocal of density is required to compute velocity and pressure from the conserved variables. The reciprocal evaluation uses the Fast Walsh Reciprocal documented in Sec. VIII (Appendix B).

B. Test Case Solutions with Implicit Method

1. Advection

The Walsh function solution of the advected profile after one cycle is compared to the exact solution in Fig. 19(a). If a Courant number is defined as the ratio of the distance traveled by a wave in time step `dt` divided by the smallest segment size dx_p, then the Courant number in this case is $c\mathrm{dt}/dx_p = 2.55$. The error norm after one cycle equals 1.66 10^{-3}. Some oscillation is evident at the front foot of the profile. If the case is rerun through Eq. 34 with $n_f = 1$ to smooth oscillations, the error norm after one cycle increases to 1.37 10^{-2}, and the profile is shown in Fig. 19(b). Truncation of wave component contributions for $n > 2^{p_\alpha - 1}$ eliminates the oscillations at the expense of smoothing out the discontinuities at the tip and feet of the profile.

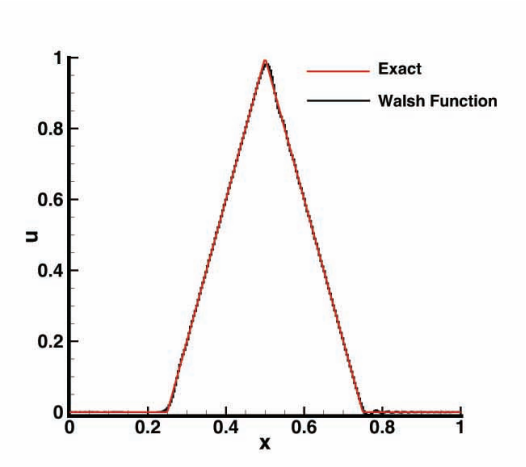
If temporal degrees of freedom are exchanged for spatial degrees of freedom by decreasing p_τ to 0 and increasing p_α to 10, then the Courant number increases to 10.23, and the profile appears in Fig. 19(c).

In stark contrast to the previous case, if spatial resolution is sacrificed for increased temporal revolution by setting $p_\alpha = 6$ and $p_\tau = 6$ and increasing the time step `dt` by a factor of 100, then a complete cycle is computed in a single time step, as shown in Fig. 19(d) with zero error norm for the linear problem. Zero error means that the advected profile exactly crosses the segment midpoints. The Courant number in this case is 63, using the previous definition. A more representative definition of the Courant number is to consider the distance traveled by a wave in the smallest temporal segment size $c(dt)_p$ divided by the smallest segment size in space $(dx)_p$, in which case the Courant number is equal to 1. This result exhibits a resonance in that the computed profile exactly repeats every time step, so that the `l1norm` = 0 in a single relaxation step because the initial condition at the beginning of the time step equals the converged condition at the end of the time step.

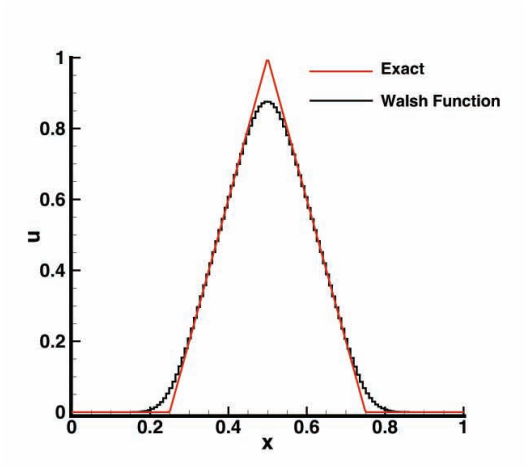
2. Burgers Equation

A steady solution is obtained for $t > 6.6$ based on attaining an `l1norm` $< 10^{-10}$ in a single relaxation step following an advance in time. The steady solution at $t = 10$ is shown in Fig. 20(a). The segmented nature of the underlying Walsh function support is evident in the stair stepping appearance of the solution. The stair stepping is less evident in Fig. 20(b) in which $p_\alpha = 8$ and segment size is a factor of 4 smaller. The error norm for this problem is recorded in Table 2. With each increment in p_α, the number of segments is doubled and the error norm decreases by a factor of approximately 4, indicating second-order accuracy relative to the smallest segment size.

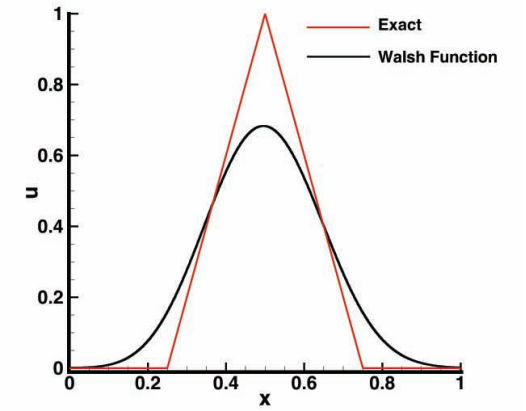
As the shock thickness decreases below the width of $(dx)_p$ with decreasing ν, oscillations may be encountered in the solution. These oscillations are eliminated by truncating the highest family of Walsh function contributions to the solution at the conclusion of a time step. An example is presented in Fig. 21, where the solution without (a) and with (b) truncation is shown when $\nu = 0.001$ and $p_\alpha = 8$. The error norm without truncation is 7.27 10^{-3} and with truncation is 1.94 10^{-3}, indicating roughly a factor of 4 decrease in error. It is a subtle but important point to note that the highest order Walsh functions still contribute to the lower-order non-linear solution retained after truncation through the self-mapping property under multiplication.



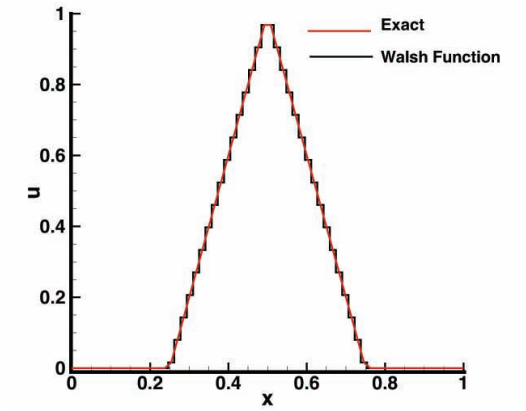
(a) $p_\alpha = 8$, $p_\tau = 2$, truncate = 0, dt = 0.01



(b) $p_\alpha = 8$, $p_\tau = 2$, truncate = 1, dt = 0.01



(c) $p_\alpha = 10$, $p_\tau = 0$, truncate = 0, dt = 0.01

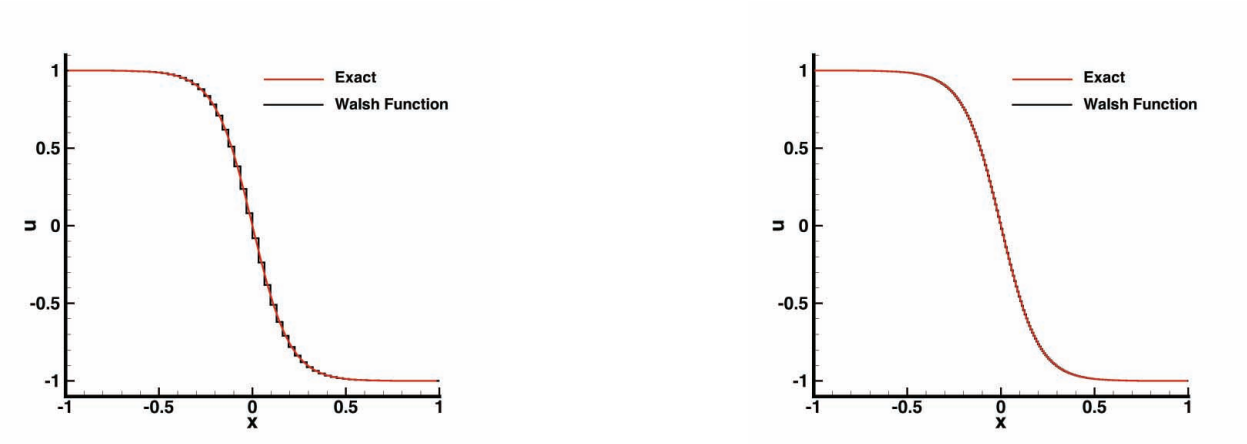


(d) $p_\alpha = 6$, $p_\tau = 6$, truncate = 0, dt = 1.

Figure 19. Advection test problem showing profile after one complete cycle at $t = 1$. Black is the Walsh function series solution. Red is the exact solution.

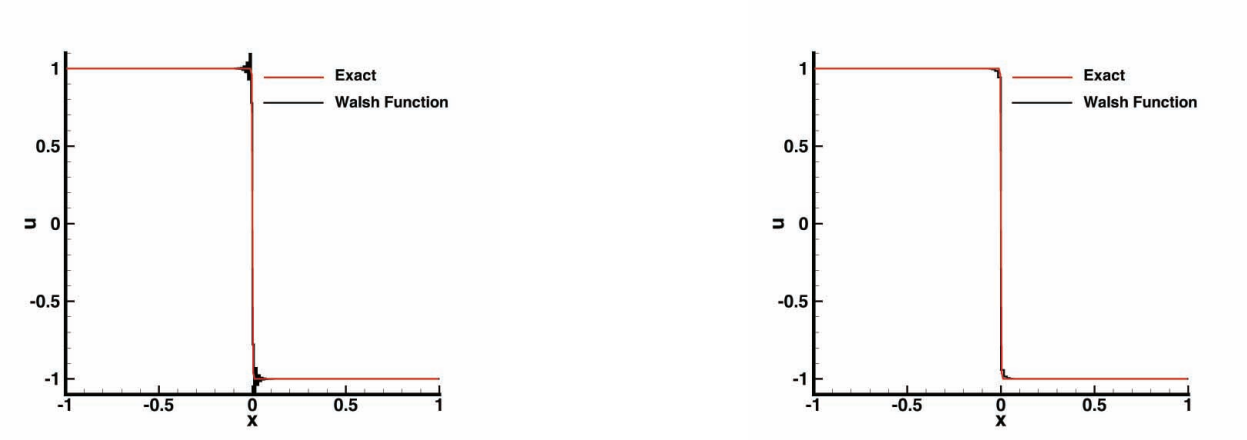
Table 2. Error norms for Burgers equation with $\nu = 0.1$.

p_α	error norm	error norm ratio
3	$1.28\ 10^{-1}$	-
4	$1.33\ 10^{-2}$	9.62
5	$2.60\ 10^{-3}$	5.12
6	$5.89\ 10^{-4}$	4.41
7	$1.40\ 10^{-4}$	4.21
8	$3.45\ 10^{-5}$	4.06
9	$8.92\ 10^{-6}$	3.87



(a) $p_\alpha = 6$ (b) $p_\alpha = 8$

Figure 20. Effect of p_α in case of $\nu = 0.1$, $p_\tau = 0$, and $p_{domain} = 0$ for Burgers equation. Black is the Walsh function series solution. Red is the exact solution.



(a) **truncate** $= 0$ (b) **truncate** $= 1$

Figure 21. Effect of truncate in case of $\nu = 0.001$, $p_\alpha = 8$, $p_\tau = 0$, and $p_{domain} = 0$ for Burgers equation. Black is the Walsh function series solution. Red is the exact solution.

3. Riemann Problem

The simulation is run with a time step $dt = 0.002$ with eight sub-domains. The Walsh series representation of each dependent variable includes 128 terms ($p_\alpha = 7$) in each sub-domain. Each sub-domain overlaps its neighbor by dx_p (Eq. 2). The solution for density at $t = 0.42$ is presented in Fig. 22. The expansion moving to the left is evident for $-.5 < x < 0$. The contact discontinuity at $x \approx 0.35$ and the shock at $x \approx 0.7$ move to the right. The constant states between the shock and the contact discontinuity, and between the contact discontinuity and the head of the expansion, are in excellent agreement with the exact solution. The tail of the expansion at $x \approx -0.5$ is slightly rounded. The contact discontinuity appears more dissipated than the shock. The largest differences are thought to be associated with the direction of characteristics approaching the discontinuities.

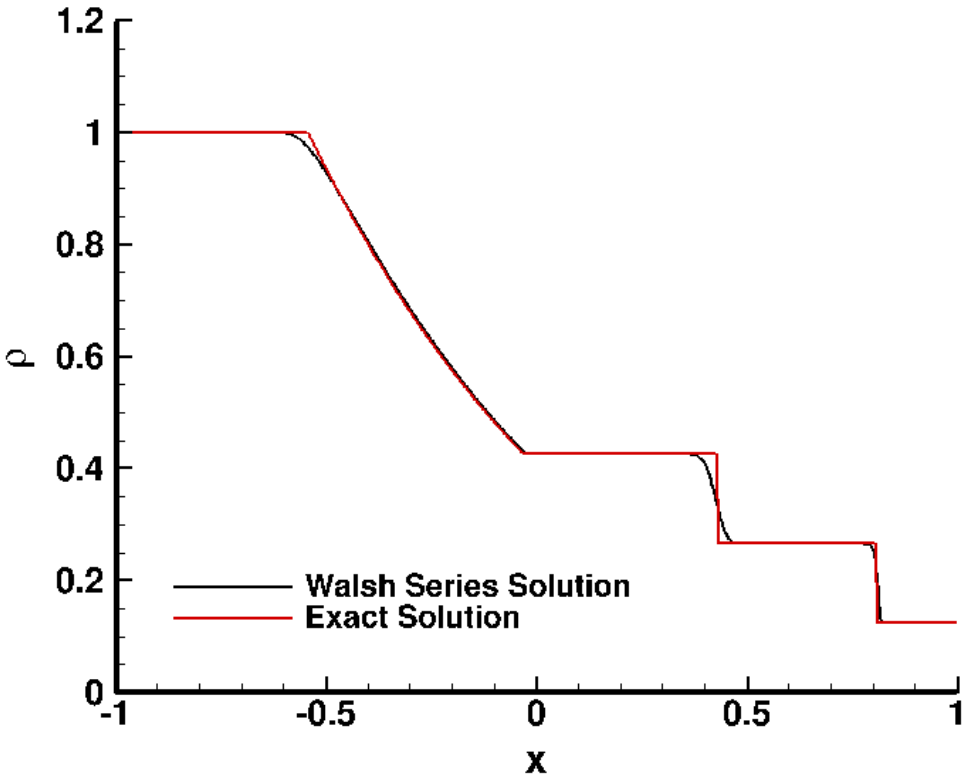


Figure 22. Density profile at $t = 0.42$ for Riemann problem. Black is the Walsh function series solution. Red is the exact solution.

C. Test Case Formulation: Explicit, Upwind Method

The essential difference between the explicit formulations in this section and the implicit formulation defined previously in Sec. A is use of the **shift** function (Eq. 36) to enable explicit treatment of boundary conditions in the formulation of spatial and temporal derivatives – replacing Eq. 27. Additional advantages include the ability to introduce accurate formulations higher than second-order and eliminating the requirement for explicit artificial dissipation for inviscid problems. In essence, numerical tests indicate that the numerical dissipation implicit in the upwind formulation is sufficient to provide good shock capturing.

A generic formulation of the test problems from the previous section follows. Only one space dimension is considered, but extension to multiple space dimensions follows identical patterns.

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x} = 0 \tag{61}$$

$$\frac{\partial \mathbf{q}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{q}}{\partial x} = 0 \tag{62}$$

$$\frac{\partial \mathbf{q}}{\partial t} + \mathbf{R}\mathbf{\Lambda}\mathbf{R}^{-1}\frac{\partial \mathbf{q}}{\partial x} = 0 \tag{63}$$

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{1}{2}\mathbf{R}\left(\mathbf{\Lambda} + |\Lambda|\right)\mathbf{R}^{-1}\frac{\partial \mathbf{q}}{\partial x} + \frac{1}{2}\mathbf{R}\left(\mathbf{\Lambda} - |\Lambda|\right)\mathbf{R}^{-1}\frac{\partial \mathbf{q}}{\partial x} = 0 \tag{64}$$

$$\frac{\partial \mathbf{q}}{\partial t} + \mathbf{A}^{+}\frac{\partial \mathbf{q}}{\partial x} + \mathbf{A}^{-}\frac{\partial \mathbf{q}}{\partial x} = 0 \tag{65}$$

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}^{+}}{\partial x} + \frac{\partial \mathbf{f}^{-}}{\partial x} = 0 \tag{66}$$

Here $\mathbf{A}$ is the Jacobian of $\mathbf{f}$ with respect to $\mathbf{q}$, $\mathbf{R}$ is a right eigenvector matrix of $\mathbf{A}$ and $\mathbf{\Lambda}$ is the diagonal eigenvalue matrix of $\mathbf{A}$. Note that formulations based on Eq. 65 showed poor resolution of jump conditions across discontinuities in approximations higher than 1st-order whereas Eq. 66 was required for conservation.

The differentials $\delta\mathbf{f}^{\pm}_{i+1/2}$ across the interface between segments i and $i+1$ are computed

$$\delta\mathbf{f}^{\pm}_{i\mp1/2} = \pm\mathbf{A}^{\pm}(\tilde{\mathbf{q}}_{i\mp1/2})\,(\mathbf{q}_i - \mathbf{q}_{i\mp1}) \tag{67}$$

$$\delta\mathbf{f}^{+}_{w,-1/2} = \mathbf{A}^{+}(\tilde{\mathbf{q}}_{w,-1/2})\,(\mathbf{q}_w - \mathbf{shift}(\mathbf{q}_w,\quad 1, \mathbf{q}_{w,bc,0})) \tag{68}$$

$$\delta\mathbf{f}^{-}_{w,\ 1/2} = -\mathbf{A}^{-}(\tilde{\mathbf{q}}_{w,\ 1/2})\,(\mathbf{q}_w - \mathbf{shift}(\mathbf{q}_w, -1, \mathbf{q}_{w,bc,N+1})) \tag{69}$$

The differential expressions in Eqs. 68 and 69 are written with subscript w to emphasize that the underlying Walsh series representation is equivalent to the discrete formulation in Eq. 67. The interface average, $\tilde{\mathbf{q}}$, is defined such that Roe's Property U[8] is satisfied, in particular,

$$\mathbf{A}(\tilde{\mathbf{q}}_{i+1/2})\,(\mathbf{q}_{i+1} - \mathbf{q}_i) = (\mathbf{f}_{i+1} - \mathbf{f}_i) \tag{70}$$

The derivative $\frac{\partial\mathbf{f}^{+}}{\partial x}$ will be approximated by a backward difference formula and $\frac{\partial\mathbf{f}^{-}}{\partial x}$ will be approximated by a forward difference formula to inject an accurate zone of dependence and to provide strong diagonal dominance for the explicit, inviscid solver as follows:

$$\left(\frac{\partial\mathbf{f}^{\pm}}{\partial x}\right)_i = \begin{cases} (3\mathbf{f}^{\pm}_i - 4\mathbf{f}^{\pm}_{i\mp1} + \mathbf{f}^{\pm}_{i\mp2})/(2dx) & +O(dx^2) \\ (11\mathbf{f}^{\pm}_i - 18\mathbf{f}^{\pm}_{i\mp1} + 9\mathbf{f}^{\pm}_{i\mp2} - 2\mathbf{f}^{\pm}_{i\mp3})/(6dx) & +O(dx^3) \\ (25\mathbf{f}^{\pm}_i - 48\mathbf{f}^{\pm}_{i\mp1} + 36\mathbf{f}^{\pm}_{i\mp2} - 16\mathbf{f}^{\pm}_{i\mp3} + 3\mathbf{f}^{\pm}_{i\mp4})/(12dx) & +O(dx^4) \end{cases} \tag{71}$$

$$\left(\frac{\partial\mathbf{f}^{\pm}}{\partial x}\right)_i = \begin{cases} (3\delta\mathbf{f}^{\pm}_{i\mp1/2} - \delta\mathbf{f}^{\pm}_{i\mp3/2})/(2dx) & +O(dx^2) \\ (11\delta\mathbf{f}^{\pm}_{i\mp1/2} - 7\delta\mathbf{f}^{\pm}_{i\mp3/2} + 2\delta\mathbf{f}^{\pm}_{i\mp5/2})/(6dx) & +O(dx^3) \\ (25\delta\mathbf{f}^{\pm}_{i\mp1/2} - 23\delta\mathbf{f}^{\pm}_{i\mp3/2} + 13\delta\mathbf{f}^{\pm}_{i\mp5/2} - 3\delta\mathbf{f}^{\pm}_{i\mp7/2})/(12dx) & +O(dx^4) \end{cases} \tag{72}$$

$$\left(\frac{\partial\mathbf{f}^{\pm}}{\partial x}\right)_i = \delta\mathbf{f}^{\pm}_{i\mp1/2}/(dx) + \begin{cases} (\delta\mathbf{f}^{\pm}_{i\mp1/2} - \delta\mathbf{f}^{\pm}_{i\mp3/2})/(2dx) & +O(dx^2) \\ (5\delta\mathbf{f}^{\pm}_{i\mp1/2} - 7\delta\mathbf{f}^{\pm}_{i\mp3/2} + 2\delta\mathbf{f}^{\pm}_{i\mp5/2})/(6dx) & +O(dx^3) \\ (13\delta\mathbf{f}^{\pm}_{i\mp1/2} - 23\delta\mathbf{f}^{\pm}_{i\mp3/2} + 13\delta\mathbf{f}^{\pm}_{i\mp5/2} - 3\delta\mathbf{f}^{\pm}_{i\mp7/2})/(12dx) & +O(dx^4) \end{cases} \tag{73}$$

$$\left(\frac{\partial\mathbf{f}^{\pm}}{\partial x}\right)_w = \delta\mathbf{f}^{\pm}_{w,\mp1/2}/(dx) + \begin{cases} (\delta\mathbf{f}^{\pm}_{w,\mp1/2} - \delta\mathbf{f}^{\pm}_{w,\mp3/2})/(2dx) & +O(dx^2) \\ (5\delta\mathbf{f}^{\pm}_{w,\mp1/2} - 7\delta\mathbf{f}^{\pm}_{w,\mp3/2} + 2\delta\mathbf{f}^{\pm}_{w,\mp5/2})/(6dx) & +O(dx^3) \\ (13\delta\mathbf{f}^{\pm}_{w,\mp1/2} - 23\delta\mathbf{f}^{\pm}_{w,\mp3/2} + 13\delta\mathbf{f}^{\pm}_{w,\mp5/2} - 3\delta\mathbf{f}^{\pm}_{w\mp7/2})/(12dx) & +O(dx^4) \end{cases} \tag{74}$$

where dx is the smallest segment size in the Walsh series representation defined by Eq. 2. The **shift** function (Eq. 36) is used to compute all elements of Eq. 74 and explicitly introduces boundary conditions from the left for $\delta\mathbf{f}^{+}$ and from the right for $\delta\mathbf{f}^{-}$. Again, the subscript w emphasizes that these terms are represented by the Walsh series that span the entire domain (all i). The terms to the right of the brace in Eq. 74 are treated separately as second-, third-, and fourth-order corrections ($\delta\mathbf{f}^{\pm}_{i,o}$, o = 2, 3, 4, respectively) to the baseline, first-order derivative formulation. The magnitude of these corrections tend to be small except near discontinuities in $\mathbf{f}^{\pm}$ where they exhibit a relatively large magnitude oscillation. This behavior is clipped locally in discrete, physical space through $\mathbf{clip}(\delta\mathbf{f}^{\pm}_{i,o})$.

$$\mathbf{clip}(\delta\mathbf{f}^{\pm}_{i,o}) = \begin{cases} \delta\mathbf{f}^{\pm}_{i,o} & \text{for} \quad |\delta\mathbf{f}^{\pm}_{i,o}| \le f_{ref} \\ \max[0, 1 - (1 - \delta\mathbf{f}^{\pm}_{i,o}/f_{ref})^2]\delta\mathbf{f}^{\pm}_{i,o} & \text{for} \quad \delta\mathbf{f}^{\pm}_{i,o} > f_{ref} \\ \max[0, 1 - (-1 - \delta\mathbf{f}^{\pm}_{i,o}/f_{ref})^2]\delta\mathbf{f}^{\pm}_{i,o} & \text{for} \quad \delta\mathbf{f}^{\pm}_{i,o} < -f_{ref} \end{cases} \tag{75}$$

where $f_{ref} = f^0_{ref}\sum_{i=1}^{N_i}|\delta f_{i,o}|/N_i$ has been used herein and f^0_{ref} is a user defined clipping factor relative to the average value over the domain.

Finally, the third- and fourth- order corrections to spatial derivatives are filtered to suppress instabilities using Eq. 34. The temporal derivative $\partial\mathbf{q}/\partial t$ is computed directly from Eq. 74 by substituting $\delta\mathbf{q}_{n-1/2}$ for

$\delta\mathbf{f}^{+}_{i-1/2}$. It is not required to process the high-order corrections for temporal derivatives with the clipping function **clip** or the filtering function **trun**.

The Walsh series representation of $\mathbf{q}_w$ is obtained by driving the L_1 norm of the residual of Eq. 66 to less than 10^{-10} over sub-iteration m for each time step as follows:

$$\mathbf{r}_w = \frac{\partial \mathbf{q}_w}{\partial t} + \frac{\partial \mathbf{f}^{+}_w}{\partial x} + \frac{\partial \mathbf{f}^{-}_w}{\partial x} \tag{76}$$

$$\mathbf{r}^{m+1}_w = \mathbf{r}^{m}_w + \left[\frac{\partial \mathbf{r}_w}{\partial \mathbf{q}_w}\right](\mathbf{q}^{m+1}_w - \mathbf{q}^{m}_w) \tag{77}$$

$$\mathbf{q}^{m+1}_w = \mathbf{q}^{m}_w - \left[\frac{\partial \mathbf{r}_w}{\partial \mathbf{q}_w}\right]^{-1} \mathbf{r}^{m}_w \tag{78}$$

where the Jacobian of the Walsh series representation of the residual with respect to the Walsh series representation of the dependent variable is defined by

$$\left[\frac{\partial \mathbf{r}_w}{\partial \mathbf{q}_w}\right] = \frac{\partial}{\partial \mathbf{q}_w}\left(\frac{\partial \mathbf{q}_w}{\partial t}\right) + \frac{\partial}{\partial \mathbf{q}_w}\left(\frac{\partial \mathbf{f}^{+}_w}{\partial x}\right) + \frac{\partial}{\partial \mathbf{q}_w}\left(\frac{\partial \mathbf{f}^{-}_w}{\partial x}\right) \tag{79}$$

and each component of the Jacobian is approximated by

$$\frac{\partial}{\partial \mathbf{q}_w}\left(\frac{\partial \mathbf{q}_w}{\partial t}\right) \approx \frac{c_\tau}{dt} \tag{80}$$

$$\frac{\partial}{\partial \mathbf{q}_w}\left(\frac{\partial \mathbf{f}^{+}_w}{\partial x}\right) \approx \frac{\partial}{\partial \mathbf{q}_w}\left(\delta\mathbf{f}^{+}_{w,-1/2}/dx\right) \approx \frac{c_\alpha}{dx}\mathbf{A}^{+}_{w,-1/2} \tag{81}$$

$$\frac{\partial}{\partial \mathbf{q}_w}\left(\frac{\partial \mathbf{f}^{-}_w}{\partial x}\right) \approx \frac{\partial}{\partial \mathbf{q}_w}\left(\delta\mathbf{f}^{-}_{w,+1/2}/dx\right) \approx \frac{-c_\alpha}{dx}\mathbf{A}^{-}_{w,+1/2} \tag{82}$$

The relaxation coefficients c_τ and c_α are usually set equal to the leading coefficient of $\delta\mathbf{f}^{\pm}_{w,\mp 1/2}/(dx)$ in Eq. 72. The Jacobian in Eq. 79 is a (3x3) for the one-dimensional Riemann problem where each element includes 2^p components of the Walsh series defining the element. Its inverse is calculated using Cramer's rule for this small system. In contrast, the linear system in the implicit formulation has size (3 (2^p) x 3 (2^p)), neglecting implicit treatment of boundary conditions.

D. Test Case Solutions with Explicit Method

1. Advection

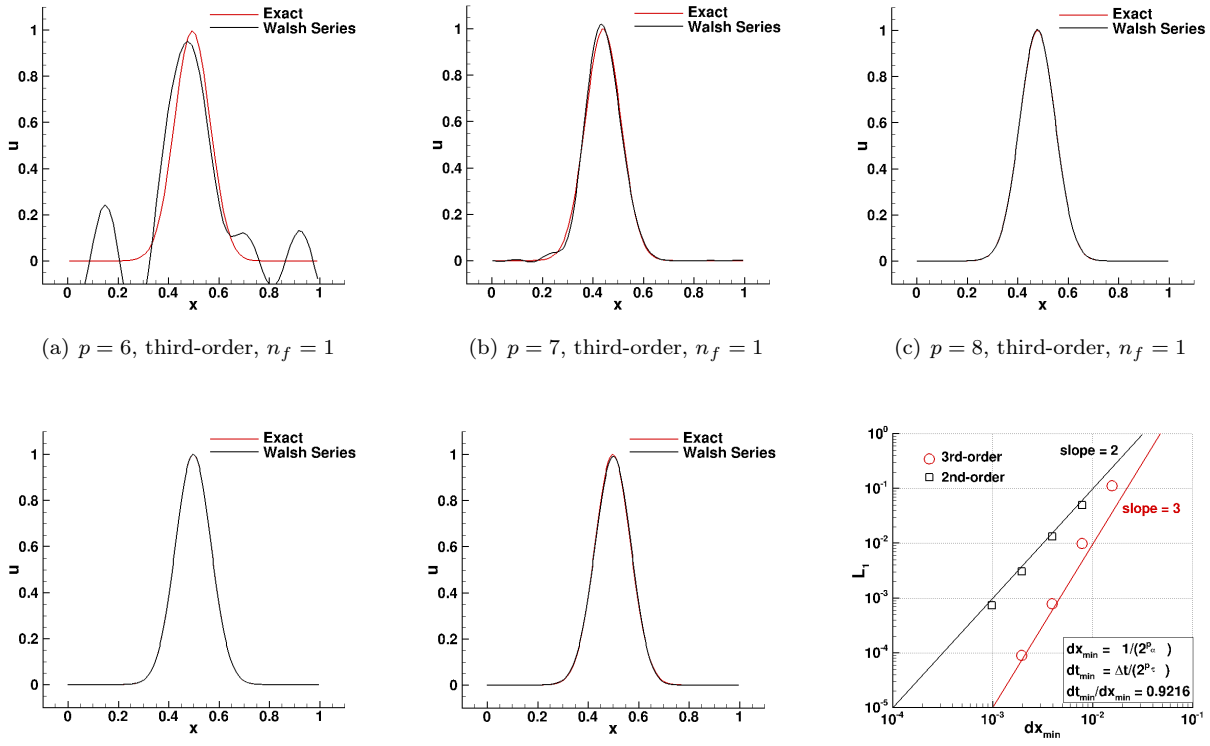
Solutions for the linear, first-order advection equation (Eq. 37) are repeated here with the explicit method. The wave speed c is a constant equal to 1; consequently $\mathbf{A}^{+} = 1$ and $\mathbf{A}^{-} = 0$. In addition to the sawtooth initial condition (Eq. 38) a singularity-free Gaussian initial condition (Eq. 83) is tested to confirm the order of accuracy.

$$u_0(x) = \exp(-10(x - 0.5)^2) \tag{83}$$

Figure 23 summarizes results for the advection of a Gaussian profile. All cases examine the distortion of the initial profile after 20 cycles ($t = 20$) through the domain. A periodic boundary is specified so that the profile exiting the right boundary reemerges through the left boundary. None of the simulations required any clipping operations ($f^0_{ref} = 10000$) because the profile is continuous. The ratio of time step to minimum segment size is fixed at 0.9216. The domain is only one segment in time ($p_\tau = 0$) in these tests.

A progression of solutions with increasing resolution in space and time is shown in Figs. 23(a–d) for the third-order formulation with a single filtering step. An oscillation at the tail of the profile grows large after 20 cycles for the largest segment size ($p = 6$). A single doubling of the number of segments (and halving the time step) still shows a slight oscillation at the tail of the profile after 20 cycles for $p = 7$ in Fig. 23(b). The oscillation is eliminated in subsequent refinements in Figs. 23(c–d). The second-order formulation with $p = 9$ did not require any filtering steps in Fig. 23(e); however, preservation of the initial profile is not quite as good as the corresponding third-order formulation. Solution accuracy is quantified in Fig. 23(f).

Third-order accuracy is confirmed with a slope 3 reduction in the log of the error norm versus the log of the segment size. Second-order accuracy is confirmed with a slope 2 reduction in the log of the error norm versus the log of the segment size. Fourth-order simulations at these conditions required a minimum of two filtering steps to suppress instabilities at the tail of the profile over large times. That filtering precluded a slope 4 reduction in error norm. In fact, the third-order result with a single filtering step was consistently more accurate than the fourth-order formulation with two or more filtering steps.



(a) $p = 6$, third-order, $n_f = 1$ (b) $p = 7$, third-order, $n_f = 1$ (c) $p = 8$, third-order, $n_f = 1$

(d) $p = 9$, third-order, $n_f = 1$ (e) $p = 9$, second-order, $n_f = 0$ (f) L_1 error norm

Figure 23. Advected Gaussian profile after 20 cycles through the domain.

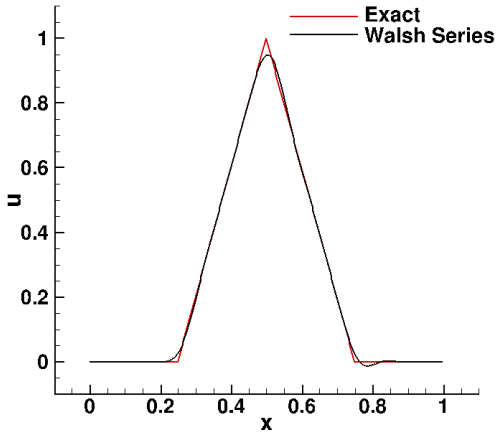
Advection of the sawtooth profile is reexamined in Fig. 24 using the explicit method. Second- and third-order formulations are presented in Figs. 24(a and b) with $p = 9$. Both cases use a CFL $(dt/dx_{min}) = 0.9216$. The third-order formulation requires two filtering steps to suppress oscillations. The CFL is raised to 2.048 in Fig. 24(c) for the third-order formulation, but an extra filtering step was required to suppress oscillations. In Fig. 24(d), it is confirmed that clipping (f^0_{ref}) is an ineffective substitute for filtering to suppress instabilities for this C_0 continuous profile. Though not presented here, additional tests of both second- and third-order formulations only produce slightly better than slope 1 (first-order) error reduction in this problem that is C_1 discontinuous.

2. Burgers Equation with Source Term

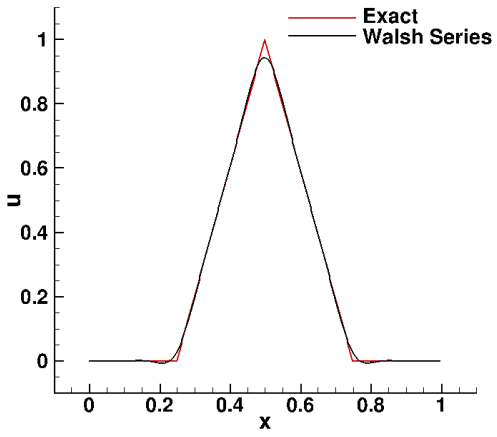
The method of manufactured solutions[9] is used to modify Eq. 47 and verify order-of-accuracy for an unsteady, non-linear problem. A simple change to the initial condition enables an interesting unsteady problem with a moving shock with changing direction. Because the explicit method does not require a second derivative dissipation to close the problem (as with the implicit method), only the inviscid form ($\nu = 0$) in Eq. 47 is considered.

The manufactured solution for this test is

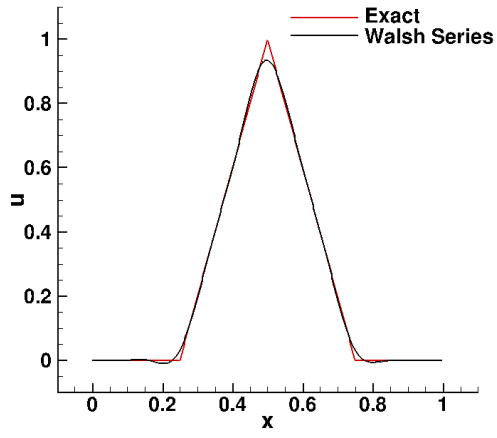
$$u(x,t) = 1 + \sin(x + 2t) \tag{84}$$



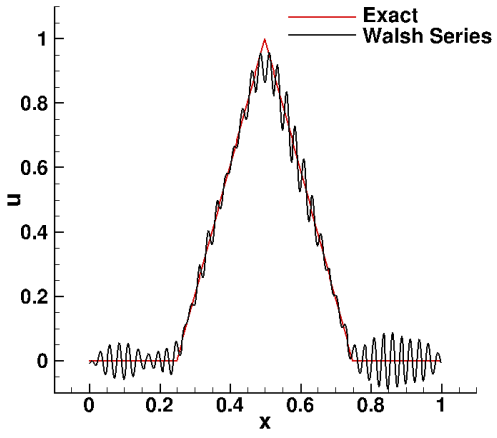
(a) $p = 9$, second-order, $n_f = 0$, $f^0_{ref} = 10^4$, $\Delta t = 0.0018$



(b) $p = 9$, third-order, $n_f = 2$, $f^0_{ref} = 10^4$, $\Delta t = 0.0018$



(c) $p = 9$, third-order, $n_f = 3$, $f^0_{ref} = 10^4$, $\Delta t = 0.004$



(d) $p = 9$, third-order, $n_f = 1$, $f^0_{ref} = 4$, $\Delta t = 0.0018$

Figure 24. Advected Delta profile after 20 cycles through the domain.

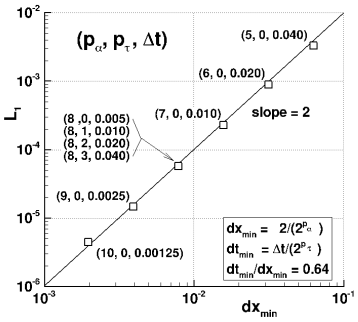
Substitute Eq. 84 into Eq. 47 (with $\nu = 0$) to compute the requisite source term to obtain the manufactured solution.

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial u^2}{\partial x} = s(x,t) \tag{85}$$

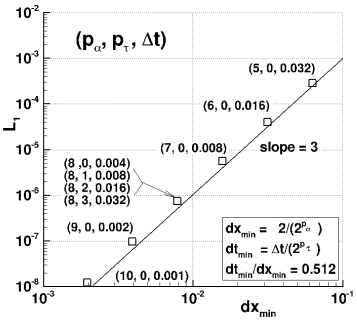
$$\frac{\partial (1+\sin(x+2t))}{\partial t} + \frac{1}{2}\frac{\partial \left[1+\sin(x+2t)\right]^2}{\partial x} = s(x,t) \tag{86}$$

$$2\cos(x+2t) + \left[1+\sin(x+2t)\right]\cos(x+2t) = s(x,t) \tag{87}$$

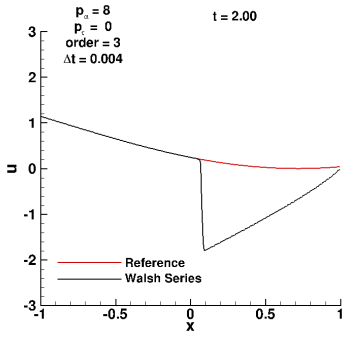
Two initial conditions are tested. In the first, $u(x,0) = 1+\sin(x)$ enables verification that the manufactured solution is recovered by the explicit formulation to the specified order of accuracy. In the second, $u(x,0) = -x$ produces a solution that approaches the manufactured solution through a shocked, intermediate state.



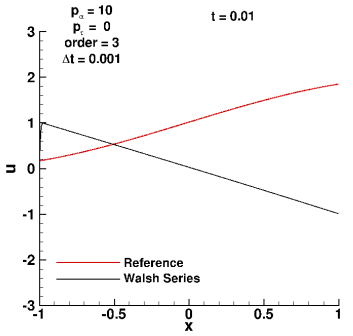
(a) L_1 norm, second-order, exact initial condition



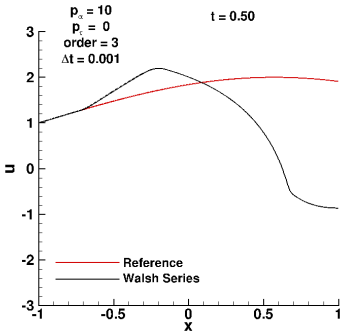
(b) L_1 norm, third-order, exact initial condition



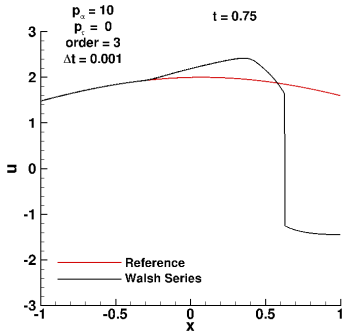
(c) $p = 8$, $t = 2.00$, $u(x, 0) = -x$



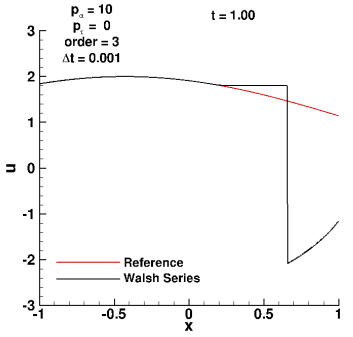
(d) $p = 10$, $t = 0.01$, $u(x, 0) = -x$



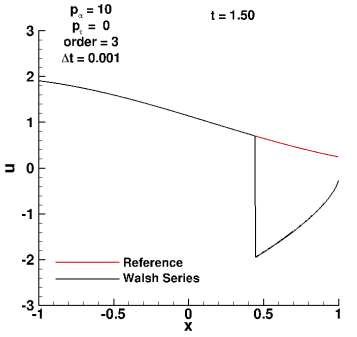
(e) $p = 10$, $t = 0.50$, $u(x, 0) = -x$



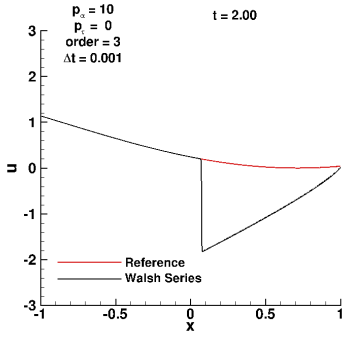
(f) $p = 10$, $t = 0.75$, $u(x, 0) = -x$



(g) $p = 10$, $t = 1.00$, $u(x, 0) = -x$



(h) $p = 10$, $t = 1.50$, $u(x, 0) = -x$



(i) $p = 10$, $t = 2.00$, $u(x, 0) = -x$

Figure 25. Burgers equation with source term from Method of Manufactured Solutions.

In the explicit formulation

$$\tilde{\mathbf{q}}_{w,-1/2} = \frac{1}{2}\left(\mathbf{q}_w + \mathbf{shift}(\mathbf{q}_w, \quad 1, \mathbf{q}_{w,bc,0})\right) \tag{88}$$

$$\tilde{\mathbf{q}}_{w,+1/2} = \frac{1}{2}\left(\mathbf{q}_w + \mathbf{shift}(\mathbf{q}_w, -1, \mathbf{q}_{w,bc,N+1})\right) \tag{89}$$

$$\mathbf{A}^{\pm} = \frac{1}{2}\left(\tilde{\mathbf{q}}_{w,\mp 1/2} \pm |\tilde{\mathbf{q}}_{w,\mp 1/2}|\right) \tag{90}$$

Verification of the convergence rate with the order of accuracy for the first initial condition yielding a continuous solution for all time is shown in Figs. 25(a and b). The L_1 norm at $t = 2$ is recorded with the second-order formulation showing a slope of 2 and the third-order formulation showing a slope of 4.

The third-order results include a single filtering sweep. Fourth-order formulations (not shown) required two filtering sweeps that prevented a slope 4 convergence rate.

Some simulations with $p_\alpha = 8$ were executed with finer resolution in the time domain ($p_\tau = 1, 2, 3$). The time step was increased by a factor of 2^{p_τ} in order that the minimum time step used to compute a CFL was consistent with the other runs. Note that the error norms for all of the $p_\alpha = 8$ cases are consistent.

Figures 25(c–i) present results for the second initial condition that yields a moving shock as the solution evolves in time. The red curve in these figures shows the manufactured solution and the black curve shows the computed solution at the same point in time. All of the computed solutions use the third-order accurate formulation with a single filtering step ($n_f = 1$) and a clipping parameter ($f^0_{ref} = 4$). Fig. 25(c) can be contrasted to Fig. 25(i) to see the sharper jump that is enabled by better spatial resolution. In all cases, there is usually only one segment and never more than two segments in the captured shock. The temporal evolution of the profile is evident in Figs. 25(d–i). The linear, initial profile is still evident at $t = 0.01$ in Fig. 25(d). At $t = 0.5$ (Fig. 25(e)), the computed profile starts to collapse on to the reference profile at the left boundary. The profile is beginning to steepen at $x = 0.6$ because of left moving waves ($u < 0$) associated with the initial condition. At $t = 0.75$ (Fig. 25(f)), the discontinuity is being pushed to the right where the magnitude of u to the left of the jump exceeds the magnitude of u to its right. By $t = 1.0$ (Fig. 25(g)), the shock is just starting to move back to the left as the source term is pulling the magnitude of the left side of the jump condition to lower values. The shock continues to be pulled to the left in Figs. 25(h and i) as the reference solution approaches a minimum value before beginning to rise again. Though not shown here, the shock continues to the left until stopping at $t = 3.08$, $x \approx -0.4$, and then sweeping quickly back to the right. Other reversals of direction occur at $t = 4.56$ and $t = 5.44$ until the computed solution collapses completely on the manufactured solution for $t > 6.2$. It is observed that the clipping and filtering functions maintain a very sharp jump condition without oscillation or overshoots.

3. Riemann Problem

The explicit formulation of the Riemann problem follows for all quantities defined in Eqs. 66–69. All of the operations involve arguments defined by a Walsh series and use operator overloading in their implementation.

$$[\rho, u, H]_{ws,+1} = \mathbf{shift}([\rho, u, H]_w, \quad 1, [\rho, u, H]_{w,bc,0}) \tag{91}$$

$$[\rho, u, H]_{ws,-1} = \mathbf{shift}([\rho, u, H]_w, -1, [\rho, u, H]_{w,bc,N+1}) \tag{92}$$

$$\tilde{\rho}_{w,\mp 1/2} = \sqrt{(\rho_w \rho_{ws,\pm 1})} \tag{93}$$

$$\tilde{\phi}_{w,1,\mp 1/2} = \frac{\rho_w}{\rho_w + \tilde{\rho}_{w,\mp 1/2}} \tag{94}$$

$$\tilde{\phi}_{w,2,\mp 1/2} = \frac{\rho_{ws,\pm 1}}{\rho_{ws,\pm 1} + \tilde{\rho}_{w,\mp 1/2}} \tag{95}$$

$$[u, H]_{w,\mp 1/2} = \tilde{\phi}_{w,1,\mp 1/2} [u, H]_w + \tilde{\phi}_{w,2,\mp 1/2} [u, H]_{ws,\pm 1} \tag{96}$$

$$\alpha_{w,\mp 1/2} = \frac{u^2_{w,\mp 1/2}}{2} \tag{97}$$

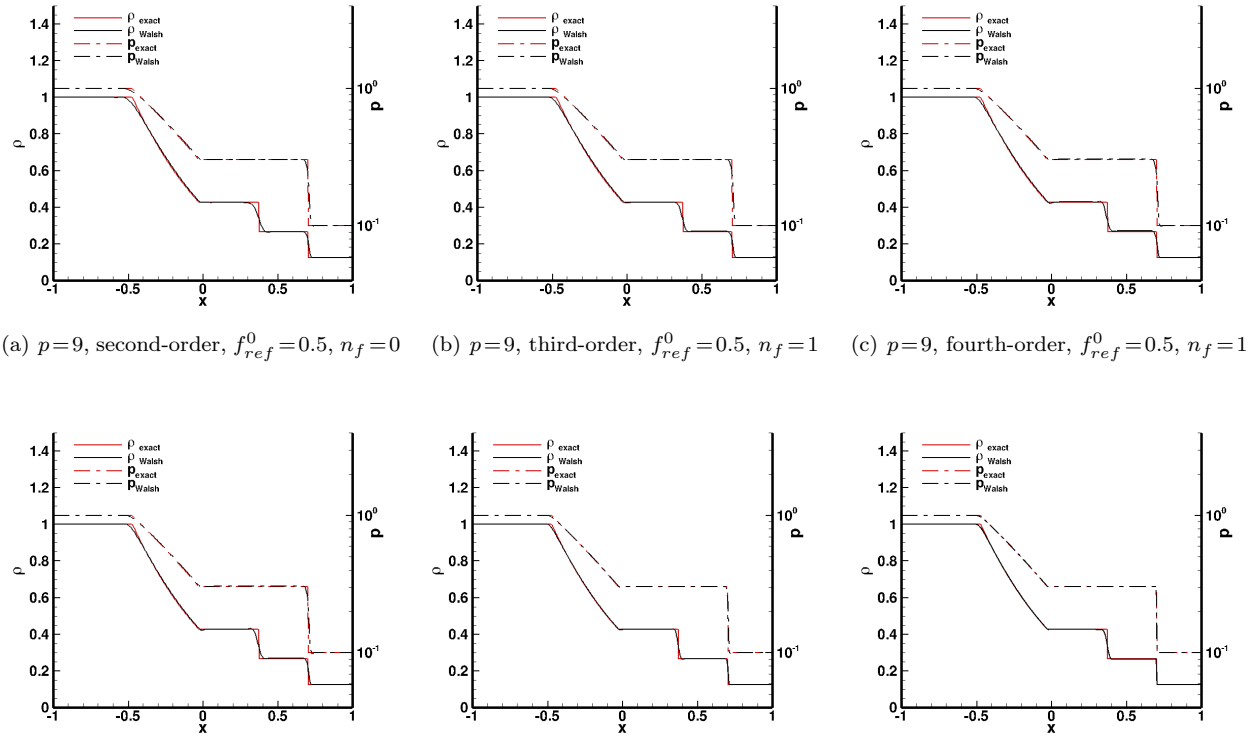
$$c_{w,\mp 1/2} = \sqrt{\beta(H_{w,\mp 1/2} - \alpha_{w,\mp 1/2})} \tag{98}$$

$$\mathbf{R}^{\pm} = \frac{1}{2c^2_{w,\mp 1/2}} \begin{bmatrix} 2 & 1 & 1 \\ 2u & u+c & u-c \\ 2\alpha & H+cu & H-cu \end{bmatrix}_{w,\mp 1/2} \tag{99}$$

$$\Lambda^{\pm} = \begin{bmatrix} u & 0 & 0 \\ 0 & u+c & 0 \\ 0 & 0 & u-c \end{bmatrix}_{w,\mp 1/2} \tag{100}$$

$$\mathbf{R}^{-1,\pm} = \begin{bmatrix} c^2 - \beta\alpha & \beta u & -\beta \\ \beta\alpha - cu & c - \beta u & \beta \\ \beta\alpha + cu & -c - \beta u & \beta \end{bmatrix}_{w,\mp 1/2} \tag{101}$$

The Sod test case was discussed previously in Sec. B3, and its solution by the implicit formulation was presented in Fig. 22. Corresponding results with the explicit formulation are shown in Fig. 26. In this figure, the computed solution in black is compared to the exact solution in red. The simulations show results with $9 \leq p \leq 12$ and second- to fourth-order formulations of the derivatives with time step $dt = 0.001$. The L_1 error norm for these cases is dominated by the solution at the moving shock and contact discontinuity and so error reduction as a function of grid refinement is first-order. In all cases, the shock ($x = 0.74$) is captured with fewer mesh points than the contact discontinuity ($x = 0.38$). The tail of the expansion near $x = 0$ shows a sharp change in slope. The head of the expansion near $x = -0.45$ shows a rounded transition compared to the exact solution. It is difficult to discern any significant differences between second-, third-, and fourth-order accurate simulations where the only non-constant state is in the expansion region ($-0.45 > x > 0$). Increasing resolution has the most significant effect on solution accuracy as seen by comparing Fig. 26(f) to any of Figs. 26(a–e). Still, none of the simulations repeat the exceptionally sharp shock captures (only one interior segment) as achieved with Burgers equation in Figs. 25(f–i). Clipping the high-order correction is effective in eliminating overshoots in all cases. A single level of filtering the high-order corrections is required in the third- and fourth-order simulations to suppress instabilities.



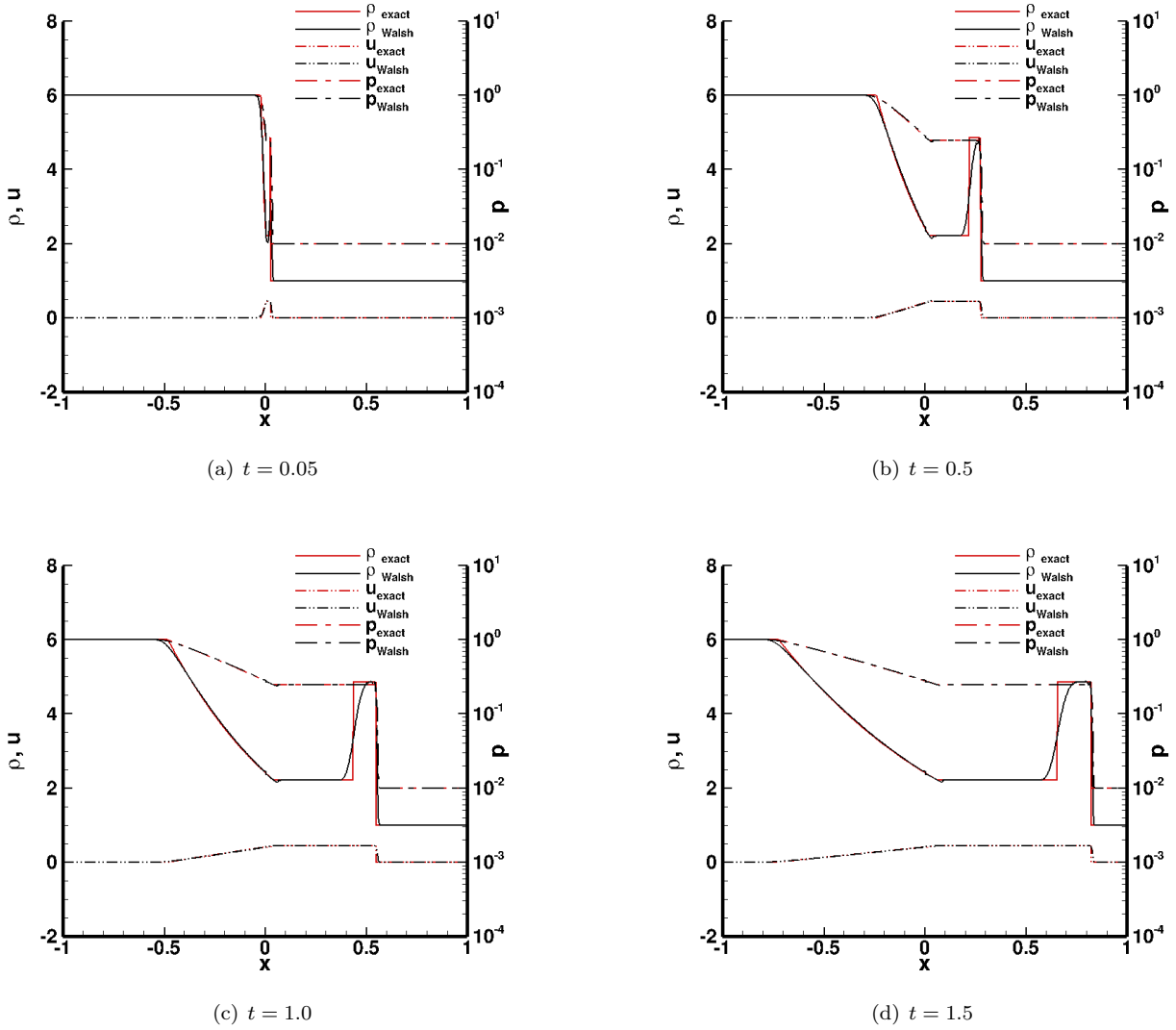
(a) p=9, second-order, f^0_{ref}=0.5, n_f=0 (b) p=9, third-order, f^0_{ref}=0.5, n_f=1 (c) p=9, fourth-order, f^0_{ref}=0.5, n_f=1

(d) p=9, fourth-order, f^0_{ref}=0.5, n_f=2 (e) p=10, fourth-order, f^0_{ref}=0.5, n_f=2 (f) p=12, second-order, f^0_{ref}=0.1, n_f=0

Figure 26. Riemann problem with Sod initial conditions at $t = 0.4$.

The initial conditions across the diaphragm in Fig. 27 correspond to a pressure and density jump corresponding to a Mach number of 8.5. (Tests up to Mach 50 conditions have been generated with equivalent results.) These conditions produce a more challenging simulation in that a very narrow domain between the shock front ($x \approx 0.3$) and the contact discontinuity ($x \approx 0.2$) in Fig. 27(b) creates a slug of high density gas that is more than a factor of 2 denser than in the undisturbed state to its right or the post-expansion state to its left. The domain between the contact discontinuity and the tail of the expansion is slightly supersonic. The expansion from ($-0.3 < x < 0$) goes from subsonic to supersonic conditions. A small expansion shock is evident near $x = 0$ in Figs. 27(b–c). Numerical tests (not shown) indicate the jump is dissipated by setting $|u \pm c|$ to be greater than $c/4$ in Eq. 64 – a simple implementation of the entropy fix.[10] Note that the simulation tracks the motion of all critical points (shock front, contact discontinuity, tail, and head of

the expansion) while running in explicit mode with a CFL = 2.13.



(a) $t = 0.05$

(b) $t = 0.5$

(c) $t = 1.0$

(d) $t = 1.5$

Figure 27. Riemann problem with $p_L = 1$, $\rho_L = 6$, p_R =0.01, and $\rho_R = 1$ with $\Delta t = 0.005$ and $\lambda_{max}\Delta t/dx_{min} = 2.13$. Numerical parameters are $p_\alpha = 10$, third-order, $f^0_{ref} = 0.1$, $n_f = 1$.

Table 3 compares the time to execute 10 time steps for the Riemann problem using the implicit and explicit formulations. Times are based on a compilation using a gfortran compiler with a –O0 option on a 2.93 GHz Intel Core 2 Duo processor. No attempt has been made to tune either the implicit or explicit path.

Table 3. CPU time to advance 10 time steps.

p_α	Implicit time	Explicit time	Iter/step, implicit	Iter/step, explicit
6	12.933	0.485	2	5.2
7	75.421	0.733	2.9	5.2
8	377.712	1.450	3	5.2
9	1691.462	3.438	3	6.1

The average number of relaxation steps per time step to drive the residual below 10^{-10} are presented in the

last two columns. The explicit method requires more relaxation steps per time step to achieve the target residual than the implicit method. However, the approximate Jacobian of the explicit method requires 2^p fewer elements than the exact Jacobian of the implicit method thus enabling orders of magnitude of time-saving for $p > 7$.

V. Summary

Walsh functions form an orthonormal basis set consisting of square waves. The discontinuous nature of square waves make the system well suited for representing functions with discontinuities. A review of Walsh function fundamentals, starting with their derivation and including descriptions of algorithms to form their products, reciprocals, integrals, and derivatives is provided. Details of a Fast Walsh Transform in multi-dimensions is provided, which enables transforms from discrete variable space to wave space and back in order $N \log(N)$ operations. A Fast Walsh Reciprocal is defined that enables the inversion of a Walsh Symmetric Matrix in order $N \log(N)$ operations. This inversion is required in the evaluation of the reciprocal of a function represented by a Walsh series.

Because Walsh functions can be derived using a fractal partitioning of a domain,[1] it is not surprising (perhaps even expected) that Walsh series representations of functions also exhibit recursive, self-similar patterns in their wave components. A remapping of Walsh function components is introduced here that makes it easier to identify these recursive patterns. The resulting figure is defined as a fractal fingerprint (FFP). The FFP of several functions that are singularity-free in a domain reveal a simple factor 2 scaling between components in adjacent groups. The FFP of functions that include singularities in the domain also show recursive patterns but the relation is more complex. These patterns are observed but not proved and more work is required to understand their potential utility in solving partial differential equations (PDEs).

Three test problems focusing on unsteady simulations documented herein include linear advection, Burgers equation, and a Riemann problem. The first case provides examples of a profile with discontinuities in slope that are advected across the domain. The other two cases explore shock-capturing and the ability to accurately track a moving discontinuity. Two algorithms are developed to simulate these test problems. The first, a purely implicit algorithm, solves for the Walsh series components individually. While excellent convergence is achieved and the Jacobian of the solution is automatically generated through the use of derived types and operator overloading, the size of the Jacobian rapidly becomes intractable (on a single processor) for more than 2^{13} degrees of freedom. Consequently, a second, explicit algorithm is derived in which the solution is focused on an update to the entire series representation of the function as a single entity – and not on individual wave components as if they were independent variables. Thus, a one-dimensional Riemann problem with 1024 degrees of freedom would require a linear solve of a (3072x3072) system where each element of the system is scalar in the implicit formulation. By contrast, the explicit formulation requires a linear solve of a (3x3) system where each element of the system includes 1024 Walsh series components that define the dependent variables over the entire domain.

This paper presents the initial exploration of algorithms based on Walsh function series to solve unsteady, non-linear PDEs. The use of truncation to suppress instabilities and prolongation to serve as the wave-component counterpart of grid sequencing are demonstrated. Third-order solutions using the explicit formulation are verified using the method of manufactured solutions for a non-linear Burgers equation and comparing to the exact solution for advection of a Gaussian profile. Courant numbers (CFL) up to 2 have yielded accurate results for time dependent problems. Tests involving higher CFL numbers (greater than 5) are stable with filtering of high-order corrections but deform the solution profile, defeating the purpose of a time accurate simulation. A longer term goal to gather metrics comparing the evolving algorithms based on Walsh function series to more traditional approaches will follow from the work presented herein.

VI. Acknowledgements

This work was supported by Revolutionary Computational Aerosciences within the Transformational Tools and Technologies Project. The author thanks summer intern Thomas May from Virginia Tech for preliminary tests using the method of manufactured solutions.

References

[1]Gnoffo, P. A.: Global Series Solutions of Nonlinear Differential Equations with Shocks Using Walsh Functions. *J. Comput. Phys.*, Vol. 258, Feb 2014, pp. 650–688.

[2]Gnoffo, P. A.: A Walsh Function Users' Manual. NASA TM 218536, NASA, Oct. 2014.

[3]Churchill, R. V.: *Fourier Series and Boundary Value Problems*. McGraw-Hill Book Company, 1969.

[4]Walsh, J. L.: A Closed Set of Normal Orthogonal Functions. *American Journal of Mathematics*, Vol. 45, no. 1, Jan. 1923, pp. 5–24.

[5]Tzafestas, S. G.: *Walsh Functions in Signal and Systems Analysis and Design*. Van Nostrand Reinhold Company, 1985.

[6]Beauchamp, K. G.: *Walsh Functions and Their Applications*. Academic Press, 1975.

[7]Sod, G. A.: A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws. *J. Comput. Phys.*, Vol. 27, 1978, pp. 1–31.

[8]Roe, P. L.: Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. *J. Comput. Phys.*, Vol. 43, no. 2, Oct. 1981, pp. 357–372.

[9]Roy, C. J.: Review of code and solution verification procedures for computational simulation. *J. Comput. Phys.*, Vol. 205, no. 1, May 2005, pp. 131–156.

[10]Harten, A.; and Hyman, J. M.: Self Adjusting Grid Methods for One-Dimensional Hyperbolic Conservation Laws. *J. Comput. Phys.*, Vol. 50, 1983, pp. 235–269.

[11]Manz, J. W.: A Sequency-Ordered Fast Walsh Transform. *IEEE Transactions on Audio and Electroacoustics*, Vol. AU-20, no. 3, 1972, pp. 204–205.

[12]Brown, R. D.: A Recursive Algorithm for Sequency-Ordered Fast Walsh Transforms. *IEEE Transactions on Computers*, Vol. C-26, no. 8, 1977, pp. 819–822.

VII. Appendix A: Sequency-Ordered Fast Walsh Transform in Multi-dimensions

Throughout this paper, Walsh functions are represented by g. Subscripts indicate a component of the Walsh series as a function of independent variables (x, y, z, t). This representation is maintained to emphasize the origins of this algorithm as an orthonormal series representation of the solution. However, the mechanics of Fast Walsh Transforms (FWT) from discrete values of a function to its wave/sequency components and back makes explicit evaluation of g unnecessary provided that the number of components is a power of 2 and the discrete functional values are uniformly distributed in computational space. The FWT enables transforms in $\mathrm{O}(N \log(N))$ operations as opposed to $\mathrm{O}(N^2)$ operations as would be required in a conventional evaluation of the series.

Equation 102 presents the expansion of a function f of four independent variables (three space and one time) in terms of a Walsh function series.[1] In this equation, N_α, N_β, N_γ, and N_τ are the total number of Walsh functions used to represent the function in the x, y, z, and t directions, respectively. $F_{\alpha,\beta,\gamma,\tau}$ are the wave number components of the expansion, where α, β, γ, and τ are integers defining the number of segments in each Walsh function component direction.

$$f(x,y,z,t) = \sum_{\tau=1}^{N_\tau} \sum_{\gamma=1}^{N_\gamma} \sum_{\beta=1}^{N_\beta} \sum_{\alpha=1}^{N_\alpha} F_{\alpha,\beta,\gamma,\tau} g_\alpha(x) g_\beta(y) g_\gamma(z) g_\tau(t) \tag{102}$$

Equation 103 defines the domain over which Eq. 102 is valid. This equation also sets up a uniform discretization of space based on the smallest segment size of the Walsh function in each respective direction. Here i, j, k, and n are integers defining the uniform discretization of space in the x, y, z, and t directions, respectively.

$$\begin{array}{llll} x_0 \le x \le x_{N_\alpha} & \Delta x = (x_{N_\alpha} - x_0)/N_\alpha & x_i = x_0 + i\Delta x & 1 \le i, \alpha \le N_\alpha \\ y_0 \le y \le y_{N_\beta} & \Delta y = (y_{N_\beta} - y_0)/N_\beta & y_j = y_0 + j\Delta y & 1 \le j, \beta \le N_\beta \\ z_0 \le z \le z_{N_\gamma} & \Delta z = (z_{N_\gamma} - z_0)/N_\gamma & z_k = z_0 + k\Delta z & 1 \le k, \gamma \le N_\gamma \\ t_0 \le t \le t_{N_\tau} & \Delta t = (t_{N_\tau} - t_0)/N_\tau & t_n = t_0 + n\Delta t & 1 \le n, \tau \le N_\tau \end{array} \tag{103}$$

Equation 104 defines the integral average of $f(x, y, z, t)$ in the discretized element defined by $x_{i-1} \le x \le x_i$, $y_{j-1} \le y \le y_j$, $z_{k-1} \le z \le z_k$, and $t_{n-1} \le t \le t_n$. The representation of the discretized function $f_{i,j,k,n}$

as a Walsh function series in Eq. 105 follows from Eq. 102.

$$f_{i,j,k,n} = \frac{1}{\Delta x \Delta y \Delta z \Delta t} \int_{t_{n-1}}^{t_n} \int_{z_{k-1}}^{z_k} \int_{y_{j-1}}^{y_j} \int_{x_{i-1}}^{x_i} f(x,y,z,t)\ dx\ dy\ dz\ dt \tag{104}$$

$$= \sum_{\tau=1}^{N_\tau} \sum_{\gamma=1}^{N_\gamma} \sum_{\beta=1}^{N_\beta} \sum_{\alpha=1}^{N_\alpha} F_{\alpha,\beta,\gamma,\tau} g_\alpha(x_{i-1/2}) g_\beta(y_{j-1/2}) g_\gamma(z_{k-1/2}) g_\tau(t_{n-1/2}) \tag{105}$$

It is convenient to think of terms like $x_{i-1/2}$ as the point halfway between x_{i-1} and x_i. Because the Walsh function $g_\alpha(x)$ is piecewise constant over a segment, and all segment lengths are integer multiples (2^p) of the smallest segment size, terms like $x_{i-1/2}$ may refer to any point between x_{i-1} and x_i.

Equation 106 defines the Walsh function wave number components $F_{\alpha,\beta,\gamma,\tau}$ as a function of $f(x,y,z,t)$. The definition of $F_{\alpha,\beta,\gamma,\tau}$ as a function of the discretized values $f_{i,j,k,n}$ in Eq. 107 follows from Eqs. 104 and 106.

$$F_{\alpha,\beta,\gamma,\tau} = \int_{t_0}^{t_{N_\tau}} \int_{z_0}^{z_{N_\gamma}} \int_{y_0}^{y_{N_\beta}} \int_{x_0}^{x_{N_\alpha}} f(x,y,z,t) g_\alpha(x) g_\beta(y) g_\gamma(z) g_\tau(t) dx dy dz dt \tag{106}$$

$$= \sum_{n=1}^{N_\tau} \sum_{k=1}^{N_\gamma} \sum_{j=1}^{N_\beta} \sum_{i=1}^{N_\alpha} f_{i,j,k,n} g_\alpha(x_{i-1/2}) g_\beta(y_{j-1/2}) g_\gamma(z_{k-1/2}) g_\tau(t_{n-1/2}) \Delta x \Delta y \Delta z \Delta t \tag{107}$$

If one has all of the elements $F_{\alpha,\beta,\gamma,\tau}$ in Eq. 105, then any single element of $f_{i,j,k,n}$ can be computed in $4N$ multiplications and N additions (assuming Walsh functions are pre-computed) where $N = N_\alpha N_\beta N_\gamma N_\tau$. Consequently, all N elements of $f_{i,j,k,n}$ can be computed in $O(N^2)$ operations. In like manner, if one has all N elements of $f_{i,j,k,n}$ in Eq. 107, then all N elements of $F_{\alpha,\beta,\gamma,\tau}$ can be computed in $O(N^2)$ operations. In the special case where each dimension is an integer power of 2 ($N_\alpha = 2^{p_\alpha}$, $N_\beta = 2^{p_\beta}$, $N_\gamma = 2^{p_\gamma}$, $N_\tau = 2^{p_\tau}$), a fast transform from all N of $F_{\alpha,\beta,\gamma,\tau}$ to $f_{i,j,k,n}$ or back can be implemented in $O(pN)$ operations where integer $p = p_\alpha + p_\beta + p_\gamma + p_\tau$. Closure under multiplication has this same requirement,[1] and so it adds no extra burden in the solution of differential equations.

Note the similarities between Eq. 105 and its inverse transform in Eq. 107. If one executes a change in variable from (i,j,k,n) to $(\alpha,\beta,\gamma,\tau)$ in Eq. 107, then it has the same formulation except for the normalizing factor $\Delta x \Delta y \Delta z \Delta t$ and a transpose of subscripts in the Walsh function factors. Inspection of Fig. 7 of the preceding paper[1] on this topic shows that terms like $g_\alpha(x_{i-1/2})$ are equal to their their transpose, $g_i(x_{\alpha-1/2})$. Indeed, the matrix $g_\alpha(x_{i-1/2})$ is equal to its own inverse within a normalizing factor.

Sequency-ordered (by increasing number of segments) Fast Walsh transforms are well documented for a function of a single variable.[11,12] A simple extension to multi-dimensions is defined here. A recursive algorithm[12] that does not require bit-reversal is modified here to accommodate multi-dimensional functions. This algorithm retains the advantage of producing its own inverse to within a normalizing factor.

A schematic of the algorithm is presented in Fig. 28. The fast transform implemented here works sequentially through the x, y, z, and t dimensions. Scaled discrete functional values of $f_{i,j,k,n}$ (or of Walsh function component factors $F_{\alpha,\beta,\gamma,\tau}$) are stored in a one-dimensional array $f(\iota)$ where

$$\iota = \begin{cases} i + N_\alpha(j - 1 + N_\beta(k - 1 + N_\gamma(n-1))) & \text{for transform from } f_{i,j,k,n} \text{ to } F_{\alpha,\beta,\gamma,\tau} \\ \alpha + N_\alpha(\beta - 1 + N_\beta(\gamma - 1 + N_\gamma(\tau-1))) & \text{for transform from } F_{\alpha,\beta,\gamma,\tau} \text{ to } f_{i,j,k,n} \end{cases} \tag{108}$$

and

$$f(\iota) = \begin{cases} f_{i,j,k,n} \Delta x / \sqrt{x_{N_\alpha} - x_0} & \text{for transform from } f_{i,j,k,n} \text{ to } F_{\alpha,\beta,\gamma,\tau} \\ F_{\alpha,\beta,\gamma,\tau} / \sqrt{x_{N_\alpha} - x_0} & \text{for transform from } F_{\alpha,\beta,\gamma,\tau} \text{ to } f_{i,j,k,n} \end{cases} \tag{109}$$

First, consider the transform from $f_{i,j,k,n}$ to $F_{\alpha,\beta,\gamma,\tau}$ through Loop 1 in the x direction as indicated in Fig. 28. If $p_\alpha = 0$, there is nothing to be done and the algorithm moves on to Loop 2. If $p_\alpha > 0$, then the sums and differences defined in Block 1 are implemented for every i line in the domain with origin at $1 \le j_0 \le N_\beta$, $1 \le k_0 \le N_\gamma$, and $1 \le n_0 \le N_\tau$. The Block 1 algorithm shown schematically in Fig. 28 is

$$F_1(\iota(i_1)) = f(\iota(i_1)) + f(\iota(i_2)) \tag{110}$$

$$F_1(\iota(i_2)) = f(\iota(i_1)) - f(\iota(i_2)) \tag{111}$$

where, for $1 \leq \alpha \leq N_\alpha/2$

$$i_1 = 2\alpha - 1 \tag{112}$$
$$i_2 = i_1 + 1 \tag{113}$$
$$\iota(i) = i + N_\alpha(j_0 - 1 + N_\beta(k_0 - 1 + N_\gamma(n_0 - 1))) \tag{114}$$

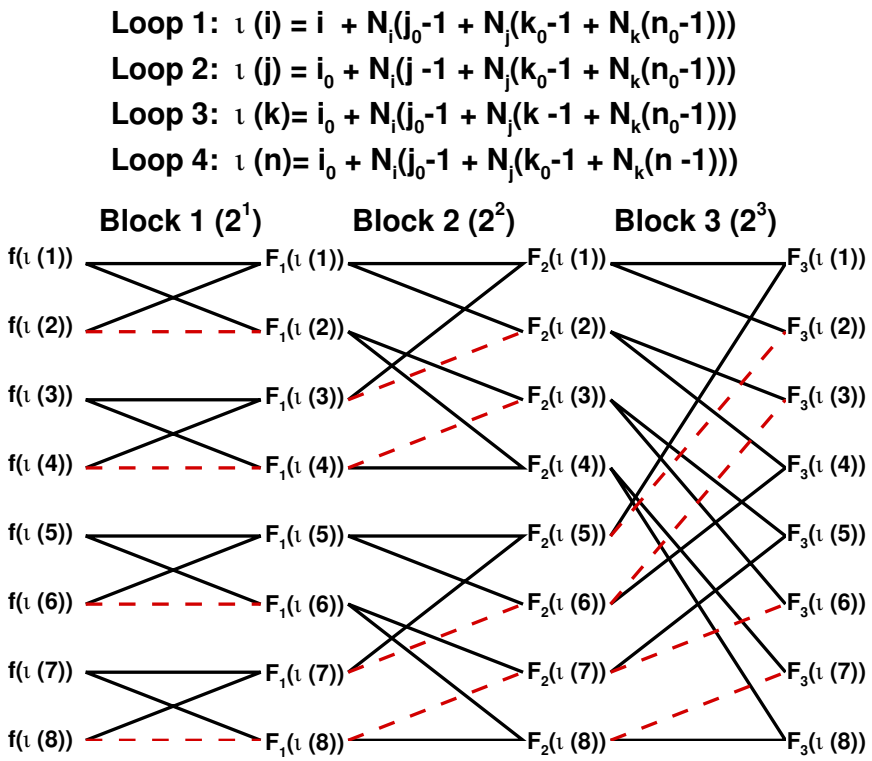


Figure 28. Schematic of sequency-ordered fast-Walsh transform algorithm based on Fig. 2 of Brown.[12] Solid black line indicates addition of quantity at left. Dashed red line indicates subtraction of quantity at left. Each block may be processed by loops 1 through 4 sequentially before advancing to the next block or the transforms may be implemented sequentially in the dimensions corresponding to i, j, k, and n.

If $p_\alpha > 1$, then Block 2 through Block p_α are implemented sequentially in a loop from $p_m = 2$ to $p_m = p_\alpha$. Note that each Block in Fig. 28 is composed of isolated groups of length 2^{p_m} in Block p_m. (The figure only presents an eight element array.) There is no intergroup exchange of information in any block. Note that all of the rays emanating from left to right on the top half of a group are solid black, indicating addition of the quantity on the left to the array location on the right. These top half contributions are implemented by looping over groups with index G of length $N_{p_m} = 2^{p_m}$ where $1 \leq G \leq N_\alpha/N_{p_m}$.

$$F_{p_m}(\iota(i_2)) = F_{p_m-1}(\iota(i_1)) \tag{115}$$
$$F_{p_m}(\iota(i_3)) = F_{p_m-1}(\iota(i_1)) \tag{116}$$

where, for $1 \leq \alpha \leq N_{p_m}/2$

$$i_1 = \alpha + (G-1)N_{p_m} \tag{117}$$
$$i_2 = 2\alpha - 1 + (G-1)N_{p_m} \tag{118}$$
$$i_3 = i_2 + 1 \tag{119}$$

and $\iota(i)$ is defined as it was in Eq. 114. Rays emanating from left to right on the bottom half of the group include one solid black and one dashed red, indicating addition and subtraction, respectively. Starting from

the bottom left in any group and moving up through the group to the midpoint, it is noted that the ray types alternate with respect to the order they are projected to the right. These projections are defined

$$F_{p_m}(\iota(i_2)) = F_{p_m}(\iota(i_2)) + \sigma F_{p_m-1}(\iota(i_1)) \quad (120)$$
$$F_{p_m}(\iota(i_3)) = F_{p_m}(\iota(i_3)) - \sigma F_{p_m-1}(\iota(i_1)) \quad (121)$$

where, for $1 \le \alpha \le N_{p_m}/2$

$$i_1 = N_{p_m} + 1 - \alpha + (G-1)N_{p_m} \quad (122)$$
$$i_2 = N_{p_m} + 2 - 2\alpha + (G-1)N_{p_m} \quad (123)$$
$$i_3 = i_2 - 1 \quad (124)$$
$$\sigma = (-1)^{\alpha+1} \quad (125)$$

and $\iota(i)$ is again defined as it was in Eq. 114.

The results for $F_{p_m}(\iota)$ in Eqs. 120 and 121 are now entered into Loop 2 ($f(\iota)$ in Fig. 28) for the transform in the y direction. Modifications to Eqs. 109 – 125 for subsequent transforms in each direction are self-evident. For example, computations for y transforms are made for every j line in the domain with the origin at $1 \le i_0 \le N_\alpha$, $1 \le k_0 \le N_\gamma$, and $1 \le n_0 \le N_\tau$. The definitions of $\iota(j_1)$, $\iota(j_2)$, and $\iota(j_3)$ replacing $\iota(i_1)$, $\iota(i_2)$, and $\iota(i_3)$ are modified accordingly. The total operation count for a transform across all four directions is $N_{op} = O((p_\alpha + p_\beta + p_\gamma + p_\tau)(N_\alpha N_\beta N_\gamma N_\tau)) = O(\log_2(N)N)$.

VIII. Appendix B: Fast Walsh Symmetric Matrix Inverse

A. Preliminaries

A Walsh reciprocal matrix, W, has rank 2^p and includes 2^p distinct elements. The ordering of these elements in row i and column j is defined using the mapping function $\mathcal{P}(i,j)$ where $\mathcal{P}(i,j)$ describes the self-mapping property of Walsh functions under multiplication. That is $g_i(x)g_j(x) = g_k(x)/\sqrt{L}$ where $1 \le i,j,k \le 2^p$, $k = \mathcal{P}(i,j)$, and L is the length of domain x. Walsh reciprocal matrices are encountered in the solution of differential equations using Walsh functions. For example, the Jacobian of the product of two functions represented by a Walsh series yields a Walsh reciprocal matrix and the inverse of W is required in the evaluation of the reciprocal of a function represented by a Walsh series. It is shown here that W^{-1} can be computed in order $p \times 2^p$ operations. A Walsh reciprocal matrix has rank 2^p where p is an integer greater than 0. It is expressed by

$$\mathbf{W} = \begin{pmatrix} w_{\mathcal{P}(1,1)} & w_{\mathcal{P}(1,2)} & w_{\mathcal{P}(1,3)} & w_{\mathcal{P}(1,4)} & \cdots & w_{\mathcal{P}(1,2^p)} \\ w_{\mathcal{P}(2,1)} & w_{\mathcal{P}(2,2)} & w_{\mathcal{P}(2,3)} & w_{\mathcal{P}(2,4)} & \cdots & w_{\mathcal{P}(2,2^p)} \\ w_{\mathcal{P}(3,1)} & w_{\mathcal{P}(3,2)} & w_{\mathcal{P}(3,3)} & w_{\mathcal{P}(3,4)} & \cdots & w_{\mathcal{P}(3,2^p)} \\ w_{\mathcal{P}(4,1)} & w_{\mathcal{P}(4,2)} & w_{\mathcal{P}(4,3)} & w_{\mathcal{P}(4,4)} & \cdots & w_{\mathcal{P}(4,2^p)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{\mathcal{P}(2^p,1)} & w_{\mathcal{P}(2^p,2)} & w_{\mathcal{P}(2^p,3)} & w_{\mathcal{P}(2^p,4)} & \cdots & w_{\mathcal{P}(2^p,2^p)} \end{pmatrix} \quad (126)$$

where $\mathcal{P}(i,j)$ is a mapping function used to define the product of two Walsh functions.[1] The product of any two Walsh functions on the domain $x_a \le x \le x_b$ is given by $g_i(x)g_j(x) = (x_b - x_a)^{-1/2} g_{\mathcal{P}(i,j)}(x)$. The mapping is closed in the sense that if $1 \le i \le 2^p$ and $1 \le j \le 2^p$, then $1 \le \mathcal{P}(i,j) \le 2^p$. Table 4 provides the mapping for the product of any two of the first 16 Walsh functions.

Closure under multiplication insures that there are at most 2^p distinct elements in the Walsh reciprocal matrix. The ordering is symmetric because multiplication is commutative ($\mathcal{P}(i,j) = \mathcal{P}(j,i)$). Walsh reciprocal matrices are encountered in the solution of differential equations using Walsh functions. It is most obviously encountered in the generation of $1/f(x)$ from the Walsh series representation of $f(x)$. It is also encountered in the evaluation of the Jacobian of the product of two functions represented by Walsh series.

The Walsh function $g_n(x)$ has n distinct segments spanning the domain $x_a \le x \le x_b$ and alternating between positive and negative values $(x_b - x_a)^{-1/2}$. Only two segment lengths are allowed in any function and their distribution can be derived using a fractal algorithm to partition the domain. This fractal derivation manifests itself in a Walsh reciprocal matrix of rank 2^p through the occurrence of smaller Walsh reciprocal

matrices that make up the larger matrix. In Table 4, one observes that the mapping function with rank 16 may be divided into four Walsh reciprocal matrices of rank 8. These rank 8 matrices may be divided into four Walsh reciprocal matrices of rank 4 and so on. Table 4 is partitioned to show the extent of cascading smaller matrices starting in the upper left corner. Other patterns may be observed such as a cascading set starting at the center of the matrix ($8 \leq (i,j) \leq 9$).

These cascading symmetries are highlighted here to suggest that the same algorithmic approaches to deriving a Fast Walsh Transform should work to develop an algorithm to compute a Fast Walsh reciprocal matrix inverse. To this end, Walsh reciprocal matrix inverses are computed for ranks 2, 4, and 8. The patterns revealed in this process are used to develop a general algorithm that has been tested through rank 4096 ($p = 12$).

Table 4. Mapping function $\mathcal{P}(i,j)$ for (1,1) $\leq$ (i,j) $\leq$ (16,16).

i j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	2	1	4	3	6	5	8	7	10	9	12	11	14	13	16	15
3	3	4	1	2	7	8	5	6	11	12	9	10	15	16	13	14
4	4	3	2	1	8	7	6	5	12	11	10	9	16	15	14	13
5	5	6	7	8	1	2	3	4	13	14	15	16	9	10	11	12
6	6	5	8	7	2	1	4	3	14	13	16	15	10	9	12	11
7	7	8	5	6	3	4	1	2	15	16	13	14	11	12	9	10
8	8	7	6	5	4	3	2	1	16	15	14	13	12	11	10	9
9	9	10	11	12	13	14	15	16	1	2	3	4	5	6	7	8
10	10	9	12	11	14	13	16	15	2	1	4	3	6	5	8	7
11	11	12	9	10	15	16	13	14	3	4	1	2	7	8	5	6
12	12	11	10	9	16	15	14	13	4	3	2	1	8	7	6	5
13	13	14	15	16	9	10	11	12	5	6	7	8	1	2	3	4
14	14	13	16	15	10	9	12	11	6	5	8	7	2	1	4	3
15	15	16	13	14	11	12	9	10	7	8	5	6	3	4	1	2
16	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Note that the inverse of a Walsh reciprocal matrix of any rank must also be Walsh reciprocal. This property of the inverse matrix is confirmed by observing that the dot product of the ith row of $\mathbf{W}$ with the kth column of $\mathbf{W}^{-1}$ involves exactly the same terms as the dot product of the kth row of $\mathbf{W}$ with the ith column of $\mathbf{W}^{-1}$. If $m_{\mathcal{P}_{j,k}}$ are the elements $\mathbf{W}^{-1}$ then

$$\sum_{j=1}^{2^p} w_{\mathcal{P}_{i,j}} m_{\mathcal{P}_{j,k}} \equiv \sum_{j=1}^{2^p} w_{\mathcal{P}_{k,j}} m_{\mathcal{P}_{j,i}} \equiv \delta(i,k) \tag{127}$$

Thus, there are only 2^p independent equations available to solve for $m_{\mathcal{P}_{j,k}}$ as a function of $w_{\mathcal{P}_{i,j}}$.

A Walsh reciprocal matrix of rank 2 ($p = 1$) and its inverse is presented in Eq. 128. The tilde ($\tilde{}$) is used to specify a matrix of rank 2.

$$\widetilde{\mathbf{W}} = \begin{pmatrix} \tilde{w}_1 & \tilde{w}_2 \\ \tilde{w}_2 & \tilde{w}_1 \end{pmatrix} \qquad \widetilde{\mathbf{W}}^{-\mathbf{1}} = \begin{pmatrix} \tilde{m}_1 & \tilde{m}_2 \\ \tilde{m}_2 & \tilde{m}_1 \end{pmatrix} \tag{128}$$

Using Eq. 127, the relations required to solve for $\tilde{m}_{\mathcal{P}_{j,k}}$ as a function of $\tilde{w}_{\mathcal{P}_{i,j}}$ are

$$\tilde{w}_1\tilde{m}_1 + \tilde{w}_2\tilde{m}_2 = 1 \tag{129}$$

$$\tilde{w}_1\tilde{m}_2 + \tilde{w}_2\tilde{m}_1 = 0 \tag{130}$$

and their solution is given by

$$\tilde{m}_1 = \frac{\tilde{w}_1}{\tilde{w}_1^2 - \tilde{w}_2^2} \tag{131}$$

$$\tilde{m}_2 = \frac{-\tilde{w}_2}{\tilde{w}_1^2 - \tilde{w}_2^2} \tag{132}$$

The next highest rank Walsh reciprocal matrix has rank 4 ($p = 2$). Both $\widehat{\mathbf{W}}$ and its inverse are presented in Eq. 133. The hat ($\hat{}$) is used to specify a matrix of rank 4.

$$\widehat{\mathbf{W}} = \begin{pmatrix} \hat{w}_1 & \hat{w}_2 & \hat{w}_3 & \hat{w}_4 \\ \hat{w}_2 & \hat{w}_1 & \hat{w}_4 & \hat{w}_3 \\ \hat{w}_3 & \hat{w}_4 & \hat{w}_1 & \hat{w}_2 \\ \hat{w}_4 & \hat{w}_3 & \hat{w}_2 & \hat{w}_1 \end{pmatrix} \quad \widehat{\mathbf{W}}^{-\mathbf{1}} = \begin{pmatrix} \hat{m}_1 & \hat{m}_2 & \hat{m}_3 & \hat{m}_4 \\ \hat{m}_2 & \hat{m}_1 & \hat{m}_4 & \hat{m}_3 \\ \hat{m}_3 & \hat{m}_4 & \hat{m}_1 & \hat{m}_2 \\ \hat{m}_4 & \hat{m}_3 & \hat{m}_2 & \hat{m}_1 \end{pmatrix} \tag{133}$$

Using Eq. 127, the relations required to solve for $\hat{m}_{\mathcal{P}_{j,k}}$ as a function of $\hat{w}_{\mathcal{P}_{i,j}}$ are

$$\hat{w}_1\hat{m}_1 + \hat{w}_2\hat{m}_2 + \hat{w}_3\hat{m}_3 + \hat{w}_4\hat{m}_4 = 1 \tag{134}$$

$$\hat{w}_1\hat{m}_2 + \hat{w}_2\hat{m}_1 + \hat{w}_3\hat{m}_4 + \hat{w}_4\hat{m}_3 = 0 \tag{135}$$

$$\hat{w}_1\hat{m}_3 + \hat{w}_2\hat{m}_4 + \hat{w}_3\hat{m}_1 + \hat{w}_4\hat{m}_2 = 0 \tag{136}$$

$$\hat{w}_1\hat{m}_4 + \hat{w}_2\hat{m}_3 + \hat{w}_3\hat{m}_2 + \hat{w}_4\hat{m}_1 = 0 \tag{137}$$

Note that the sum and difference of Eqs. 134 and 135 yield Eq. 138, and the sum and difference of Eqs. 136 and 137 yield Eq. 139.

$$(\hat{w}_1 \pm \hat{w}_2)(\hat{m}_1 \pm \hat{m}_2) + (\hat{w}_3 \pm \hat{w}_4)(\hat{m}_3 \pm \hat{m}_4) = 1 \tag{138}$$

$$(\hat{w}_1 \pm \hat{w}_2)(\hat{m}_3 \pm \hat{m}_4) + (\hat{w}_3 \pm \hat{w}_4)(\hat{m}_1 \pm \hat{m}_2) = 0 \tag{139}$$

Following the example set by derivation of a Fast Walsh Transform[11] note that a formulation using adjacent sums and differences (Eqs. 140 and 141) simplifies the algorithm.

$$\hat{s}_{1,2} = \hat{w}_1 + \hat{w}_2 \qquad \hat{d}_{1,2} = \hat{w}_1 - \hat{w}_2 \tag{140}$$

$$\hat{s}_{3,4} = \hat{w}_3 + \hat{w}_4 \qquad \hat{d}_{3,4} = \hat{w}_3 - \hat{w}_4 \tag{141}$$

The solution of a (4x4) system is now converted to the solution of two (2x2) systems involving $(\hat{s}_{1,2}, \hat{s}_{3,4})$ and $(\hat{d}_{1,2}, \hat{d}_{3,4})$ using Eqs. 131 and 132. The elements of $\widehat{\mathbf{W}}^{-\mathbf{1}}$ are now presented by taking the appropriate averages of sums and differences of like terms.

$$\hat{m}_1 = \frac{1}{2}\left[\frac{\hat{s}_{1,2}}{(\hat{s}_{1,2}^2 - \hat{s}_{3,4}^2)} + \frac{\hat{d}_{1,2}}{(\hat{d}_{1,2}^2 - \hat{d}_{3,4}^2)}\right] \tag{142}$$

$$\hat{m}_2 = \frac{1}{2}\left[\frac{\hat{s}_{1,2}}{(\hat{s}_{1,2}^2 - \hat{s}_{3,4}^2)} - \frac{\hat{d}_{1,2}}{(\hat{d}_{1,2}^2 - \hat{d}_{3,4}^2)}\right] \tag{143}$$

$$\hat{m}_3 = -\frac{1}{2}\left[\frac{\hat{s}_{3,4}}{(\hat{s}_{1,2}^2 - \hat{s}_{3,4}^2)} + \frac{\hat{d}_{3,4}}{(\hat{d}_{1,2}^2 - \hat{d}_{3,4}^2)}\right] \tag{144}$$

$$\hat{m}_4 = -\frac{1}{2}\left[\frac{\hat{s}_{3,4}}{(\hat{s}_{1,2}^2 - \hat{s}_{3,4}^2)} - \frac{\hat{d}_{3,4}}{(\hat{d}_{1,2}^2 - \hat{d}_{3,4}^2)}\right] \tag{145}$$

The next highest rank Walsh reciprocal matrix has rank 8 ($p = 3$). Both $\mathbf{W}$ and its inverse are presented

in Eq. 146.

$$\mathbf{W} = \begin{pmatrix} w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 \\ w_2 & w_1 & w_4 & w_3 & w_6 & w_5 & w_8 & w_7 \\ w_3 & w_4 & w_1 & w_2 & w_7 & w_8 & w_5 & w_6 \\ w_4 & w_3 & w_2 & w_1 & w_8 & w_7 & w_6 & w_5 \\ w_5 & w_6 & w_7 & w_8 & w_1 & w_2 & w_3 & w_4 \\ w_6 & w_5 & w_8 & w_7 & w_2 & w_1 & w_4 & w_3 \\ w_7 & w_8 & w_5 & w_6 & w_3 & w_4 & w_1 & w_2 \\ w_8 & w_7 & w_6 & w_5 & w_4 & w_3 & w_2 & w_1 \end{pmatrix}$$

$$\mathbf{W^{-1}} = \begin{pmatrix} m_1 & m_2 & m_3 & m_4 & m_5 & m_6 & m_7 & m_8 \\ m_2 & m_1 & m_4 & m_3 & m_6 & m_5 & m_8 & m_7 \\ m_3 & m_4 & m_1 & m_2 & m_7 & m_8 & m_5 & m_6 \\ m_4 & m_3 & m_2 & m_1 & m_8 & m_7 & m_6 & m_5 \\ m_5 & m_6 & m_7 & m_8 & m_1 & m_2 & m_3 & m_4 \\ m_6 & m_5 & m_8 & m_7 & m_2 & m_1 & m_4 & m_3 \\ m_7 & m_8 & m_5 & m_6 & m_3 & m_4 & m_1 & m_2 \\ m_8 & m_7 & m_6 & m_5 & m_4 & m_3 & m_2 & m_1 \end{pmatrix} \tag{146}$$

Using Eq. 127, the relations required to solve for $m_{\mathcal{P}_{j,k}}$ as a function of $w_{\mathcal{P}_{i,j}}$are

$$w_1m_1 + w_2m_2 + w_3m_3 + w_4m_4 + w_5m_5 + w_6m_6 + w_7m_7 + w_8m_8 = 1 \tag{147}$$
$$w_1m_2 + w_2m_1 + w_3m_4 + w_4m_3 + w_5m_6 + w_6m_5 + w_7m_8 + w_8m_7 = 0 \tag{148}$$
$$w_1m_3 + w_2m_4 + w_3m_1 + w_4m_2 + w_5m_7 + w_6m_8 + w_7m_5 + w_8m_6 = 0 \tag{149}$$
$$w_1m_4 + w_2m_3 + w_3m_2 + w_4m_1 + w_5m_8 + w_6m_7 + w_7m_6 + w_8m_5 = 0 \tag{150}$$
$$w_1m_5 + w_2m_6 + w_3m_7 + w_4m_8 + w_5m_1 + w_6m_2 + w_7m_3 + w_8m_4 = 0 \tag{151}$$
$$w_1m_6 + w_2m_5 + w_3m_8 + w_4m_7 + w_5m_2 + w_6m_1 + w_7m_4 + w_8m_3 = 0 \tag{152}$$
$$w_1m_7 + w_2m_8 + w_3m_5 + w_4m_6 + w_5m_3 + w_6m_4 + w_7m_1 + w_8m_2 = 0 \tag{153}$$
$$w_1m_8 + w_2m_7 + w_3m_6 + w_4m_5 + w_5m_4 + w_6m_3 + w_7m_2 + w_8m_1 = 0 \tag{154}$$

Following the example above, note that the sum and difference of Eqs. 147 and 148 yield Eq. 155, the sum and difference of Eqs. 149 and 150 yield Eq. 156, the sum and difference of Eqs. 151 and 152 yield Eq. 157, and the sum and difference of Eqs. 153 and 154 yield Eq. 158.

$$\begin{aligned}(w_1 \pm w_2)(m_1 \pm m_2) + (w_3 \pm w_4)(m_3 \pm m_4) &+ \\ (w_5 \pm w_6)(m_5 \pm m_6) &+ (w_7 \pm w_8)(m_7 \pm m_8) = 1 \end{aligned} \tag{155}$$
$$\begin{aligned}(w_1 \pm w_2)(m_3 \pm m_4) + (w_3 \pm w_4)(m_1 \pm m_2) &+ \\ (w_5 \pm w_6)(m_7 \pm m_8) &+ (w_7 \pm w_8)(m_5 \pm m_6) = 0 \end{aligned} \tag{156}$$
$$\begin{aligned}(w_1 \pm w_2)(m_5 \pm m_6) + (w_3 \pm w_4)(m_7 \pm m_8) &+ \\ (w_5 \pm w_6)(m_1 \pm m_2) &+ (w_7 \pm w_8)(m_3 \pm m_4) = 0 \end{aligned} \tag{157}$$
$$\begin{aligned}(w_1 \pm w_2)(m_7 \pm m_8) + (w_3 \pm w_4)(m_5 \pm m_6) &+ \\ (w_5 \pm w_6)(m_3 \pm m_4) &+ (w_7 \pm w_8)(m_1 \pm m_2) = 0 \end{aligned} \tag{158}$$

Adjacent sums and differences are again formed.

$$s_{1,2} = w_1 + w_2 \qquad d_{1,2} = w_1 - w_2 \tag{159}$$
$$s_{3,4} = w_3 + w_4 \qquad d_{3,4} = w_3 - w_4 \tag{160}$$
$$s_{5,6} = w_5 + w_6 \qquad d_{5,6} = w_5 - w_6 \tag{161}$$
$$s_{7,8} = w_7 + w_8 \qquad d_{7,8} = w_7 - w_8 \tag{162}$$

The solution of an (8x8) system is now converted to the solution of two (4x4) systems. The elements of $\mathbf{W}^{-1}$ are again presented by taking the appropriate averages of sums and differences of like terms.

$$m_1 = \frac{1}{4}\left[\frac{s_{1,2}+s_{3,4}}{(s_{1,2}+s_{3,4})^2-(s_{5,6}+s_{7,8})^2} + \frac{d_{1,2}+d_{3,4}}{(d_{1,2}+d_{3,4})^2-(d_{5,6}+d_{7,8})^2} + \frac{s_{1,2}-s_{3,4}}{(s_{1,2}-s_{3,4})^2-(s_{5,6}-s_{7,8})^2} + \frac{d_{1,2}-d_{3,4}}{(d_{1,2}-d_{3,4})^2-(d_{5,6}-d_{7,8})^2}\right] \tag{163}$$

$$m_2 = \frac{1}{4}\left[\frac{s_{1,2}+s_{3,4}}{(s_{1,2}+s_{3,4})^2-(s_{5,6}+s_{7,8})^2} - \frac{d_{1,2}+d_{3,4}}{(d_{1,2}+d_{3,4})^2-(d_{5,6}+d_{7,8})^2} + \frac{s_{1,2}-s_{3,4}}{(s_{1,2}-s_{3,4})^2-(s_{5,6}-s_{7,8})^2} - \frac{d_{1,2}-d_{3,4}}{(d_{1,2}-d_{3,4})^2-(d_{5,6}-d_{7,8})^2}\right] \tag{164}$$

$$m_3 = \frac{1}{4}\left[\frac{s_{1,2}+s_{3,4}}{(s_{1,2}+s_{3,4})^2-(s_{5,6}+s_{7,8})^2} + \frac{d_{1,2}+d_{3,4}}{(d_{1,2}+d_{3,4})^2-(d_{5,6}+d_{7,8})^2} - \frac{s_{1,2}-s_{3,4}}{(s_{1,2}-s_{3,4})^2-(s_{5,6}-s_{7,8})^2} - \frac{d_{1,2}-d_{3,4}}{(d_{1,2}-d_{3,4})^2-(d_{5,6}-d_{7,8})^2}\right] \tag{165}$$

$$m_4 = \frac{1}{4}\left[\frac{s_{1,2}+s_{3,4}}{(s_{1,2}+s_{3,4})^2-(s_{5,6}+s_{7,8})^2} - \frac{d_{1,2}+d_{3,4}}{(d_{1,2}+d_{3,4})^2-(d_{5,6}+d_{7,8})^2} - \frac{s_{1,2}-s_{3,4}}{(s_{1,2}-s_{3,4})^2-(s_{5,6}-s_{7,8})^2} + \frac{d_{1,2}-d_{3,4}}{(d_{1,2}-d_{3,4})^2-(d_{5,6}-d_{7,8})^2}\right] \tag{166}$$

$$m_5 = -\frac{1}{4}\left[\frac{s_{5,6}+s_{7,8}}{(s_{1,2}+s_{3,4})^2-(s_{5,6}+s_{7,8})^2} + \frac{d_{5,6}+d_{7,8}}{(d_{1,2}+d_{3,4})^2-(d_{5,6}+d_{7,8})^2} + \frac{s_{5,6}-s_{7,8}}{(s_{1,2}-s_{3,4})^2-(s_{5,6}-s_{7,8})^2} + \frac{d_{5,6}-d_{7,8}}{(d_{1,2}-d_{3,4})^2-(d_{5,6}-d_{7,8})^2}\right] \tag{167}$$

$$m_6 = -\frac{1}{4}\left[\frac{s_{5,6}+s_{7,8}}{(s_{1,2}+s_{3,4})^2-(s_{5,6}+s_{7,8})^2} - \frac{d_{5,6}+d_{7,8}}{(d_{1,2}+d_{3,4})^2-(d_{5,6}+d_{7,8})^2} + \frac{s_{5,6}-s_{7,8}}{(s_{1,2}-s_{3,4})^2-(s_{5,6}-s_{7,8})^2} - \frac{d_{5,6}-d_{7,8}}{(d_{1,2}-d_{3,4})^2-(d_{5,6}-d_{7,8})^2}\right] \tag{168}$$

$$m_7 = -\frac{1}{4}\left[\frac{s_{5,6}+s_{7,8}}{(s_{1,2}+s_{3,4})^2-(s_{5,6}+s_{7,8})^2} + \frac{d_{5,6}+d_{7,8}}{(d_{1,2}+d_{3,4})^2-(d_{5,6}+d_{7,8})^2} - \frac{s_{5,6}-s_{7,8}}{(s_{1,2}-s_{3,4})^2-(s_{5,6}-s_{7,8})^2} - \frac{d_{5,6}-d_{7,8}}{(d_{1,2}-d_{3,4})^2-(d_{5,6}-d_{7,8})^2}\right] \tag{169}$$

$$m_8 = -\frac{1}{4}\left[\frac{s_{5,6}+s_{7,8}}{(s_{1,2}+s_{3,4})^2-(s_{5,6}+s_{7,8})^2} - \frac{d_{5,6}+d_{7,8}}{(d_{1,2}+d_{3,4})^2-(d_{5,6}+d_{7,8})^2} - \frac{s_{5,6}-s_{7,8}}{(s_{1,2}-s_{3,4})^2-(s_{5,6}-s_{7,8})^2} + \frac{d_{5,6}-d_{7,8}}{(d_{1,2}-d_{3,4})^2-(d_{5,6}-d_{7,8})^2}\right] \tag{170}$$

A pattern becomes evident as the process continues. The algorithm for defining the Walsh reciprocal matrix inverse for arbitrary rank 2^p with $p > 1$ is now defined. A work vector σ_i is initialized in Eq. 171 with the first row of matrix $\mathbf{W}$. Note that only two levels of storage are required for σ, but it is convenient to define the algorithm below without the necessity of toggling current and previous iterates.

$$\sigma_i^0 = w_i \qquad \text{for} \quad 1 \le i \le 2^p \tag{171}$$

Following the example used first in Eqs. 140 and 141, adjacent sums are gathered in the first half of vector σ and adjacent differences are gathered in the last half of vector σ. A loop through the algorithm forming adjacent sums and differences using entries from the previous loop is implemented $p-1$ times in Eq. 173. Equation 173 involves $(p-1)2^p$ additions.

$$\sigma_i^{n+1} = \sigma_{2i-1}^n + \sigma_{2i}^n \qquad \text{for} \quad 1 \le i \le 2^{p-1} \qquad \text{and} \qquad 0 < n \le p-1 \tag{172}$$

$$\sigma_{i+2^{p-1}}^{n+1} = \sigma_{2i-1}^n - \sigma_{2i}^n \qquad \text{for} \quad 1 \le i \le 2^{p-1} \qquad \text{and} \qquad 0 < n \le p-1 \tag{173}$$

A single loop through Eqs. 174–176 forms elemental ratios involving adjacent pairs as in Eqs. 142–145. These

equations involve an additional 2^{p+1} multiplies and 2^{p-1} additions.

$$\zeta_{2i-1} = (\sigma_{2i-1}^p)^2 - (\sigma_{2i}^p)^2 \tag{174}$$

$$\sigma_{2i-1}^{p+1} = \frac{\sigma_{2i-1}^p}{\zeta_{2i-1}} \qquad \text{for} \quad 1 \le i \le 2^{p-1} \quad \text{and} \quad p > 0 \tag{175}$$

$$\sigma_{2i}^{p+1} = \frac{\sigma_{2i}^p}{\zeta_{2i-1}} \tag{176}$$

Sums and differences of elemental ratios are once again reordered by placing sums of odd numbered pairs in the first quarter of σ, placing differences of odd numbered pairs in the second quarter of σ, placing sums of even numbered pairs in the third quarter of σ, and placing differences of even numbered pairs in the last quarter of σ. This single loop through Eqs. 177–180 involves 2^p more additions.

$$\sigma_i^{p+2} = \sigma_{4i-3}^{p+1} + \sigma_{4i-1}^{p+1} \qquad \text{for} \quad 1 \le i \le 2^{p-2} \quad \text{and} \quad p > 1 \tag{177}$$

$$\sigma_{i+2^{p-2}}^{p+2} = \sigma_{4i-3}^{p+1} - \sigma_{4i-1}^{p+1} \qquad \text{for} \quad 1 \le i \le 2^{p-2} \quad \text{and} \quad p > 1 \tag{178}$$

$$\sigma_{i+2^{p-1}}^{p+2} = \sigma_{4i-2}^{p+1} + \sigma_{4i}^{p+1} \qquad \text{for} \quad 1 \le i \le 2^{p-2} \quad \text{and} \quad p > 1 \tag{179}$$

$$\sigma_{i+2^{p-1}+2^{p-2}}^{p+2} = \sigma_{4i-2}^{p+1} - \sigma_{4i}^{p+1} \qquad \text{for} \quad 1 \le i \le 2^{p-2} \quad \text{and} \quad p > 1 \tag{180}$$

At this point, all of the pairings of elemental ratios are organized to complete a fast evaluation of the inverse matrix. Use Eq. 181 to initialize storage level 1 in vector σ with the result of Eqs. 177–180.

$$\sigma_i^1 = \sigma_i^{p+2} \qquad \text{for} \quad 1 \le i \le 2^p \tag{181}$$

Now execute $p-2$ loops through Eqs. 182–185 to complete the assembly of sums and differences of pairs of elemental ratios as previously observed in Eqs. 163–170. These loops introduce an additional $(p-2)2^p$ additions.

$$\sigma_i^{n+1} = \sigma_{2i-1}^n + \sigma_{2i}^n \qquad \text{for} \quad 1 \le i \le 2^{p-2} \quad \text{and} \quad 1 \le n \le p-2 \tag{182}$$

$$\sigma_{i+2^{p-2}}^{n+1} = \sigma_{2i-1}^n - \sigma_{2i}^n \qquad \text{for} \quad 1 \le i \le 2^{p-2} \quad \text{and} \quad 1 \le n \le p-2 \tag{183}$$

$$\sigma_{i+2^{p-1}}^{n+1} = \sigma_{2i-1+2^{p-1}}^n + \sigma_{2i+2^{p-1}}^n \qquad \text{for} \quad 1 \le i \le 2^{p-2} \quad \text{and} \quad 1 \le n \le p-2 \tag{184}$$

$$\sigma_{i+2^{p-1}+2^{p-2}}^{n+1} = \sigma_{2i-1+2^{p-1}}^n - \sigma_{2i+2^{p-1}}^n \qquad \text{for} \quad 1 \le i \le 2^{p-2} \quad \text{and} \quad 1 \le n \le p-2 \tag{185}$$

Finally, the leading coefficient observed in Eqs. 142–145 and in Eqs. 163–170 is applied in a single loop through Eqs. 186 and 187. This final step to evaluate the elements m_i of $\mathbf{W}^{-1}$ introduces 2^p additional multiplies.

$$m_i = \frac{1}{2^{p-1}} \sigma_i^{p+3} \qquad \text{for} \quad 1 \le i \le 2^{p-1} \tag{186}$$

$$m_{i+2^{p-1}} = \frac{-1}{2^{p-1}} \sigma_{i+2^{p-1}}^{p+3} \qquad \text{for} \quad 1 \le i \le 2^{p-1} \tag{187}$$

The total operation count for computing the inverse of a Walsh reciprocal matrix is

$$N_{op} = (2p - \frac{3}{2})2^p \quad \text{additions} \quad + 2^{p+1} \quad \text{multiplications} \tag{188}$$

The operation count as a function of matrix rank N is

$$N_{op} = O(N \log N) \tag{189}$$

No proof is provided that the algorithm is valid for all p. It has been tested on various problems for $1 < p \le 12$ ($2 < N \le 4096$) and found to yield at least 15 digits of accuracy in the evaluation of Eq. 127.