Projections of Functions: Consider the domain $[-1, 1]$ and the following three functions. The first is a smooth initial condition given by

$$u_0(x) = \sin(2\pi x), \tag{1}$$

and the second is given by

$$u_0(x) = \begin{cases} 1, & x \in [-0.5, 0.5] \\ 0, & \text{else} \end{cases}. \tag{2}$$

Finally, the third function that we consider is

$$u_0(x) = \begin{cases} \frac{1}{6}\left(G(x,\beta,z-\delta) + G(x,\beta,z+\delta)\right) + \frac{2}{3}G(x,\beta,z) & x \in [-0.8,-0.6], \\ 1, & x \in [-0.4,-0.2], \\ 1 - |10(x-0.1)|, & x \in [0, 0.2], \\ \frac{1}{6}\left(F(x,\alpha,c-\delta) + F(x,\alpha,c+\delta) + 4F(x,\alpha,c)\right)) & x \in [-0.8,-0.6], \\ 0, & \text{for all other } x. \end{cases} \tag{3}$$

where

$$G(x,\beta,z) = e^{-\beta(x-z)^2},$$

and

$$F(x,\alpha,a) = \sqrt{\max(1-\alpha^2(x-a)^2, 0)}.$$

For this third function, $c = 0.5,\ z = -0.7,\ \delta = 0.005,\ \alpha = 10$, and $\beta = \frac{\log 2}{36\delta^2}$.
In this exercise, we consider representing the DG approximation in a multi-wavelet basis. There is more than one way to calculate the multiwavelet basis. The first method involves directly projecting onto the multiwavelets. The second method involves taking a difference in approximaations at two different resolutions.

(a) Note that the multi-wavelet basis on $(0,\ 1)$ is given by

p	m	$\Psi_m(\xi)$
0	0	$\sqrt{\frac{1}{2}}$
1	0	$\sqrt{\frac{3}{2}}(2\xi - 1)$
	1	$\sqrt{\frac{1}{2}}(3\xi - 2)$
2	0	$\frac{1}{3}\sqrt{\frac{1}{2}}(30\xi^2 - 24\xi + 1)$
	1	$\frac{1}{2}\sqrt{\frac{3}{2}}(15\xi^2 - 16\xi + 3)$
	2	$\frac{1}{3}\sqrt{\frac{5}{2}}(12\xi^2 - 15\xi + 4)$
3	0	$\sqrt{\frac{15}{34}}(28\xi^3 - 30\xi^2 + 4\xi + 1)$
	1	$\sqrt{\frac{1}{42}}(210\xi^3 - 300\xi^2 + 105\xi - 4)$
	2	$\frac{1}{2}\sqrt{\frac{35}{34}}(64\xi^3 - 105\xi^2 + 48\xi - 5)$
	3	$\frac{1}{2}\sqrt{\frac{5}{42}}(105\xi^3 - 192\xi^2 + 105\xi - 16)$

and is extended to $(-1,\ 0)$ using the relation $\Psi_m(\xi) = (-1)^{m+p+1}\Psi_m(-\xi)$. Project the *approximation* using 16 elements onto this basis.

To define an approximation for each of the $u_0(x)$ as detailed in the Week 1 Worksheet, we will project onto an orthonormal piecewise Legendre basis. The Legendre polynomials are provided by the following recurrence relation

$$P^{(0)}(\xi_j) = 1, \quad P^{(1)}(\xi_j) = \xi_j,$$

$$P^{(m+1)}(\xi_j) = \frac{2m+1}{m+1}\xi_j P^{(m)}(\xi_j) - \frac{m}{m+1}P^{(m-1)}(\xi_j), \quad m = 1, 2, \ldots, p,$$

with $P^{(m)}(\pm 1) = (\pm 1)^m$. The orthonormal Legendre polynomials are defined as

$$\varphi^{(m)}(\xi_j) = \sqrt{m + \frac{1}{2}}P^{(m)}(\xi_j). \tag{4}$$

We will use a local transformation from $[x_{j-1/2}, x_{j+1/2}] \to [-1, 1]$, which is given by

$$\xi_j = \frac{2}{\Delta x_j}(x - x_j),\ j = 1, \ldots, N.$$

Given a uniform mesh, $\Delta x_j = \Delta x$, and thus

$$\xi_j = \frac{2}{\Delta x}(x - x_j),\ j = 1, \ldots, N. \tag{5}$$

We can represent the approximation, $u_h(x, t)$, on element I_j as

$$u_h(x, t) = \sum_{\ell=0}^{p} u_j^{(\ell)}(t)\varphi^{(\ell)}(\xi_j) \quad x \in I_j.$$

Specifically, we are using the approximation

$$V_h^p = \{\varphi^{(m)}(\xi_j)|_{I_j},\ m = 0, \ldots, p,\ j = 1, \ldots, m\}.$$

We can determine the approximation coefficients, $u_j^{(\ell)}(0) = u_j^{(\ell)}$, through the projection

$$(u_h(x, 0), \varphi^{(m)}(\xi))_{I_j} = (u_0(x), \varphi^{(m)}(\xi))_{I_j},\ m = 0, \ldots, p,\ j = 1, \ldots N,$$

where $(f(x), g(x))_{I_j} = \int_{I_j} f(x)g(x)dx$. For the left-hand side of the equation above, we

can use orthogonality of the orthonormal Legendre basis to result in

$$\begin{aligned}
(u_h(x,0),\varphi^{(m)}(\xi))_{I_j} &= \left(\sum_{\ell=0}^{p} u_j^{(\ell)}(0)\varphi^{(\ell)}(\xi_j),\varphi^{(m)}(\xi)\right)_{I_j}, \\
&= \sum_{\ell=0}^{p}\left(u_j^{(\ell)}\varphi^{(\ell)}(\xi_j),\varphi^{(m)}(\xi)\right)_{I_j}, \\
&= \sum_{\ell=0}^{p} u_j^{(\ell)}\left(\varphi^{(\ell)}(\xi_j),\varphi^{(m)}(\xi)\right)_{I_j}, \\
&= u_j^{(m)}\left(\varphi^{(m)}(\xi_j),\varphi^{(m)}(\xi)\right)_{I_j}, \\
&= u_j^{(m)}\int_{I_j}\varphi^{(m)}(\xi_j)\varphi^{(m)}(\xi)dx, \\
&= u_j^{(m)}\frac{\Delta x}{2}\int_{-1}^{1}\varphi^{(m)}(\xi_j)\varphi^{(m)}(\xi)d\xi, \\
&= u_j^{(m)}\frac{\Delta x}{2}.
\end{aligned}$$

We hence have

$$\begin{aligned}
(u_h(x,0),\varphi^{(m)}(\xi))_{I_j} &= (u_0(x),\varphi^{(m)}(\xi))_{I_j}, \\
u_j^{(m)}\frac{\Delta x}{2} &= \int_{I_j} u_0(x)\varphi^{(m)}(\xi)dx, \\
u_j^{(m)}\frac{\Delta x}{2} &= \frac{\Delta x}{2}\int_{-1}^{1} u_0(x)\varphi^{(m)}(\xi)d\xi,
\end{aligned}$$

and thus

$$u_j^{(m)} = \int_{-1}^{1} u_0(x)\varphi^{(m)}(\xi)d\xi, \quad m=0,\ldots,p,\ j=1,\ldots,N. \tag{6}$$

This equation provides the $N(p+1)$ coefficients, or "degrees of freedom", for the basis functions. The integral on the right-hand side may be computed using the Gauss-Legendre quadrature rule,

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i),$$

where x_i is the Gauss-Legendre node and represents the i^{th} root of $P_n(x)$, the n^{th} order Legendre polynomial. The corresponding weight w_i is found using the following formula

$$w_i = \frac{2}{(1-x_i^2)[P_n'(x_i)]^2},$$

as provided in [1]. The implementation of the above to calculate the basis coefficients and the approximations was done as part of Exercise 1 and saved in *OrthoLegendreBasis-Projections.py*. That file along with all other code is available at `https://github.com/sterrab/DGMethods.git`.

The difference between two resolutions, m and $m-1$, with respective approximations $u_h^{(m)}$ and $u_h^{(m-1)}$, is represented by a multi-wavelet basis, Ψ. (Here, resolution $m-1$ has N elements in the mesh and m has $2N$ elements.) Namely,

$$u_h^{(m)}(x,t) = u_h^{(m-1)} + \sum_{j=0}^{2^{m-1}-1} \sum_{l=0}^{p} d_j^{(m-1,l)} \Psi^{(m-1,l)}(x), \tag{7}$$

where $d_j^{(m-1,l)}$ represent the coefficients or weights of the multi-wavelet basis. For Method 1, the objective is to determine these coefficients using Alpert's multi-wavelet basis, Ψ, and reconstruct the multi-resolution difference, $u_h^{(m)}(x,t) - u_h^{(m-1)}$. The Alpert multi-wavelet basis, defined on $(0,1)$, is included in the Table above. The basis can be extended to $(-1,0)$ using the relation $\Psi_m(\xi) = (-1)^{m+p+1}\Psi_m(-\xi)$. Given that the multi-wavelets live in a coarse resolution, we have the following orthogonality property, $(\Psi^{(m-1,l)}, u_h^{(m-1,l)}) = 0$. By projecting Eq. 7 onto an arbitrary k multi-wavelet basis function, $\Psi^{(m-1,k)}$, we can derive the coefficients as shown below:

$$\sum_{j=0}^{2^{m-1}-1} \sum_{l=0}^{p} d_j^{(m-1,l)} \Psi^{(m-1,l)}(x) = u_h^{(m)}(x,t) - u_h^{(m-1)},$$

$$\sum_{j=0}^{2^{m-1}-1} \sum_{l=0}^{p} d_j^{(m-1,l)} \Psi^{(m-1,l)}(x) = \sum_{j=0}^{2^m-1} u_h^{(m)}(x,t)|_{I_j^{(m)}} - \sum_{j=0}^{2^{m-1}-1} u_h^{(m-1)}|_{I_j^{(m-1)}},$$

$$\sum_{j=0}^{2^{m-1}-1} \left(\sum_{l=0}^{p} d_j^{(m-1,l)} \Psi^{(m-1,l)}(x), \Psi^{(m-1,k)}(x) \right) = \sum_{j=0}^{2^m-1} \left(u_h^{(m)}(x,t)|_{I_j^{(m)}}, \Psi^{(m-1,k)}(x) \right) - \sum_{j=0}^{2^{m-1}-1} \left(u_h^{(m-1)}|_{I_j^{(m-1)}}, \Psi^{(m-1,k)}(x) \right),$$

$$\sum_{j=0}^{2^{m-1}-1} \left(\sum_{l=0}^{p} d_j^{(m-1,l)} \Psi^{(m-1,l)}(x), \Psi^{(m-1,k)}(x) \right) = \sum_{j=0}^{2^m-1} \left(\sum_{l=0}^{p} u_j^{(m,l)} \varphi^{(m,l)}(x)|_{I_j^{(m)}}, \Psi^{(m-1,k)}(x) \right),$$

$$\left(\sum_{l=0}^{p} d_j^{(m-1,l)} \Psi^{(m-1,l)}(x)|_{I_j}, \Psi^{(m-1,k)}(x)|_{I_j^{(m-1)}} \right) = \left(\sum_{l=0}^{p} u_j^{(m,l)} \varphi^{(m,l)}(x)|_{I_j}, \Psi^{(m-1,k)}(x)|_{I_j^{(m-1)}} \right),$$

$$\int_{x_{j-1/2}^{(m-1)}}^{x_{j+1/2}^{(m-1)}} \sum_{l=0}^{p} d_j^{(m-1,l)} \Psi^{(m-1,l)}(x) \Psi^{(m-1,k)}(x) dx = \int_{x_{j-1/2}^{(m-1)}}^{x_{j+1/2}^{(m-1)}} \sum_{l=0}^{p} u_j^{(m,l)} \varphi^{(m,l)}(x) \Psi^{(m-1,k)}(x) dx,$$

or

$$\sum_{l=0}^{p} d_j^{(m-1,l)} \int_{x_{j-1/2}^{(m-1)}}^{x_{j+1/2}^{(m-1)}} \Psi^{(m-1,l)}(x)\Psi^{(m-1,k)}(x)dx = \sum_{l=0}^{p} \int_{x_{j-1/2}^{(m-1)}}^{x_{j+1/2}^{(m-1)}} u_j^{(m,l)}\varphi^{(m,l)}(x)\Psi^{(m-1,k)}(x)dx. \tag{8}$$

At this point, we can use the local scaling coordinate, $\xi = \frac{2}{(\Delta x)^{(m-1)}}(x - x_j^{(m-1)})$ or $x = \tilde{\xi} = x_j^{(m-1)} + \frac{(\Delta x)^{(m-1)}}{2}\xi$ to simplify the left-hand side of Eq. 8:

$$\begin{aligned}
\sum_{l=0}^{p} d_j^{(m-1,l)} \int_{x_{j-1/2}^{(m-1)}}^{x_{j+1/2}^{(m-1)}} \Psi^{(m-1,l)}(x)\Psi^{(m-1,k)}(x)dx &= \frac{(\Delta x)^{(m-1)}}{2} \sum_{l=0}^{p} d_j^{(m-1,l)} \int_{-1}^{1} \Psi^{(m-1,l)}(\tilde{\xi})\Psi^{(m-1,k)}(\tilde{\xi})d\xi, \\
&= \frac{(\Delta x)^{(m-1)}}{2} \sum_{l=0}^{p} d_j^{(m-1,l)}\delta_{l,k}, \\
&= \frac{(\Delta x)^{(m-1)}}{2} d_j^{(m-1,k)}.
\end{aligned}$$

As for the right-hand side of Eq. 8, given that two elements of resolution m span an element in $m-1$, we have to divide the integral into two regions and use the correct index change for each element:

$$\begin{aligned}
\sum_{l=0}^{p} u_j^{(m,l)} \int_{x_{j-1/2}^{(m-1)}}^{x_{j+1/2}^{(m-1)}} \varphi^{(m,l)}(x)\Psi^{(m-1,k)}(x)dx &= \sum_{l=0}^{p} u_{2j}^{(m,l)} \int_{x_{2j-1/2}^{(m)}}^{x_{2j+1/2}^{(m)}} \varphi^{(m,l)}(x)\Psi^{(m-1,k)}(x)dx \\
&\quad + \sum_{l=0}^{p} u_{2j+1}^{(m,l)} \int_{x_{2j+1/2}^{(m)}}^{x_{2j+3/2}^{(m)}} \varphi^{(m,l)}(x)\Psi^{(m-1,k)}(x)dx.
\end{aligned}$$

As we will integrate in elements in the m resolution mesh, we not only need to scale $x^{(m)} \mapsto \xi^{(m)}$, we also need to find the correct mapping of $\xi^{(m)} \mapsto \xi^{(m-1)}$. We can find the mapping of element $2j$ in the m resolution to element j in the $m-1$ resolution as follows,

$$\begin{aligned}
\xi^{(m-1)} &= \frac{2}{(\Delta x)^{(m-1)}}(x - x_j^{(m-1)}), \\
&= \frac{1}{(\Delta x)^{(m)}} \left(\frac{(\Delta x)^{(m)}}{2}\xi^{(m)} + x_{2j}^{(m)} - x_j^{(m-1)} \right), \\
&= \frac{1}{2}\xi^{(m)} + \frac{1}{(\Delta x)^{(m)}}(x_{2j}^{(m)} - x_j^{(m-1)}), \\
&= \frac{1}{2}\xi^{(m)} + \frac{1}{(\Delta x)^{(m)}} \left(-\frac{(\Delta x)^{(m)}}{2} \right), \\
&= \frac{1}{2}(\xi^{(m)} - 1).
\end{aligned}$$

To map element $2j+1$ in the m resolution to element j in the $m-1$ resolution, we have

$$\begin{aligned}
\xi^{(m-1)} &= \frac{2}{(\Delta x)^{(m-1)}}(x - x_j^{(m-1)}),\\
&= \frac{1}{(\Delta x)^{(m)}}\left(\frac{(\Delta x)^{(m)}}{2}\xi^{(m)} + x_{2j+1}^{(m)} - x_j^{(m-1)}\right),\\
&= \frac{1}{2}\xi^{(m)} + \frac{1}{(\Delta x)^{(m)}}(x_{2j+1}^{(m)} - x_j^{(m-1)}),\\
&= \frac{1}{2}\xi^{(m)} + \frac{1}{(\Delta x)^{(m)}}\left(+\frac{(\Delta x)^{(m)}}{2}\right),\\
&= \frac{1}{2}(\xi^{(m)} + 1).
\end{aligned}$$

With these mappings, we now have the right-hand side of Eq. 8 as

$$\begin{aligned}
\sum_{l=0}^{p} u_j^{(m,l)} \int_{x_{j-1/2}^{(m-1)}}^{x_{j+1/2}^{(m-1)}} \varphi^{(m,l)}(x)\Psi^{(m-1,k)}(x)dx &= \sum_{l=0}^{p} u_{2j}^{(m,l)} \int_{x_{2j-1/2}^{(m)}}^{x_{2j+1/2}^{(m)}} \varphi^{(m,l)}(x)\Psi^{(m-1,k)}(x)dx\\
&\quad + \sum_{l=0}^{p} u_{2j+1}^{(m,l)} \int_{x_{2j+1/2}^{(m)}}^{x_{2j+3/2}^{(m)}} \varphi^{(m,l)}(x)\Psi^{(m-1,k)}(x)dx,\\
&= \sum_{l=0}^{p} u_{2j}^{(m,l)} \frac{(\Delta x)^{(m)}}{2} \int_{-1}^{1} \varphi^{(m,l)}(\xi^{(m)})\Psi^{(m-1,k)}\left(\frac{1}{2}(\xi^{(m)} - 1)\right) d\\
&\quad + \sum_{l=0}^{p} u_{2j+1}^{(m,l)} \frac{(\Delta x)^{(m)}}{2} \int_{-1}^{1} \varphi^{(m,l)}(\xi^{(m)})\Psi^{(m-1,k)}\left(\frac{1}{2}(\xi^{(m)} - 1)\right)
\end{aligned}$$

Equation 8 simplifies to

$$\begin{aligned}
\frac{(\Delta x)^{(m-1)}}{2} d_j^{(m-1,k)} &= \frac{(\Delta x)^{(m)}}{2} \sum_{l=0}^{p} u_{2j}^{(m,l)} \int_{-1}^{1} \varphi^{(m,l)}(\xi^{(m)})\Psi^{(m-1,k)}\left(\frac{1}{2}(\xi^{(m)} - 1)\right) d\xi\\
&\quad + \frac{(\Delta x)^{(m)}}{2} \sum_{l=0}^{p} u_{2j+1}^{(m,l)} \int_{-1}^{1} \varphi^{(m,l)}(\xi^{(m)})\Psi^{(m-1,k)}\left(\frac{1}{2}(\xi^{(m)} + 1)\right) d\xi,\\
d_j^{(m-1,k)} &= \frac{(\Delta x)^{(m)}}{(\Delta x)^{(m-1)}} \sum_{l=0}^{p} u_{2j}^{(m,l)} \int_{-1}^{1} \varphi^{(m,l)}(\xi^{(m)})\Psi^{(m-1,k)}\left(\frac{1}{2}(\xi^{(m)} - 1)\right) d\xi\\
&\quad + \frac{(\Delta x)^{(m)}}{(\Delta x)^{(m-1)}} \sum_{l=0}^{p} u_{2j+1}^{(m,l)} \int_{-1}^{1} \varphi^{(m,l)}(\xi^{(m)})\Psi^{(m-1,k)}\left(\frac{1}{2}(\xi^{(m)} + 1)\right) d\xi,
\end{aligned}$$

and thus

$$d_j^{(m-1,k)} = \frac{1}{2}\sum_{l=0}^{p} u_{2j}^{(m,l)} \int_{-1}^{1} \varphi^{(m,l)}(\xi^{(m)})\Psi^{(m-1,k)}\left(\frac{1}{2}(\xi^{(m)}-1)\right) d\xi + \frac{1}{2}\sum_{l=0}^{p} u_{2j+1}^{(m,l)} \int_{-1}^{1} \varphi^{(m,l)}(\xi^{(m)})\Psi^{(m-1,k)}\left(\frac{1}{2}(\xi^{(m)}+1)\right) d\xi. \tag{9}$$

The results for the projection method for each of the initial conditions are shown in Fig. 1-3. This along with results in (b) and (c) were computed using the *HW1-Multiwavelets.py* file, with functions from *OrthoLegendreBasis-Projections.py*, *MultiWaveletBasis-Projections.py*, *MultiWavelet-eval.py*, and *InitialConditions.py* files.

(b) Create a function by subtracting the approximation using 8 elements from one using 16 elements. Plot this function at 6 points per element. Note that the multi-wavelet basis is on the coarse grid (8 elements).

The method of differences was implemented by keeping 6 Gauss-Legendre nodes for the coarse resolution $N = 8$ and the corresponding 3 Gauss-Legendre nodes per element (for two elements) in the fine resolution $N = 16$. The difference between the two approximations at those same nodes was taken to determine the multiwavelets on the coarse grid. The approximations of the two resolutions and the corresponding multiwavelets by difference are plotted in Figs. 4-6.

(c) Plot the results of these two ways of forming the multi-wavelet approximation. Plot the error.

Finally, the results of the two methods are plotted in Figs. 7-9. For the first initial condition in Fig. 7, there is a discrepancy between the multiwavelets for $p = 0$, which gradually decreases with increasing p value. Given the square wave function in Eq. (2) does not result in non-zero multiwavelets due to to piecewise constant approximation, the two multiwavelets methods are identical to round-off error. As for the complex third initial condition with both multiwavelets shown in Fig. 9, there is quite a bit of discrepancy between the methods.

Solving the Linear Transport Equation: Consider the linear advection equation

$$u_t + au_x = 0, \quad a > 0, \qquad x \in [-1, 1],$$

with the initial condition

$$u(x, 0) = \sin(2\pi x),$$

and periodic boundary conditions,

$$u(-1, t) = u(1, t).$$

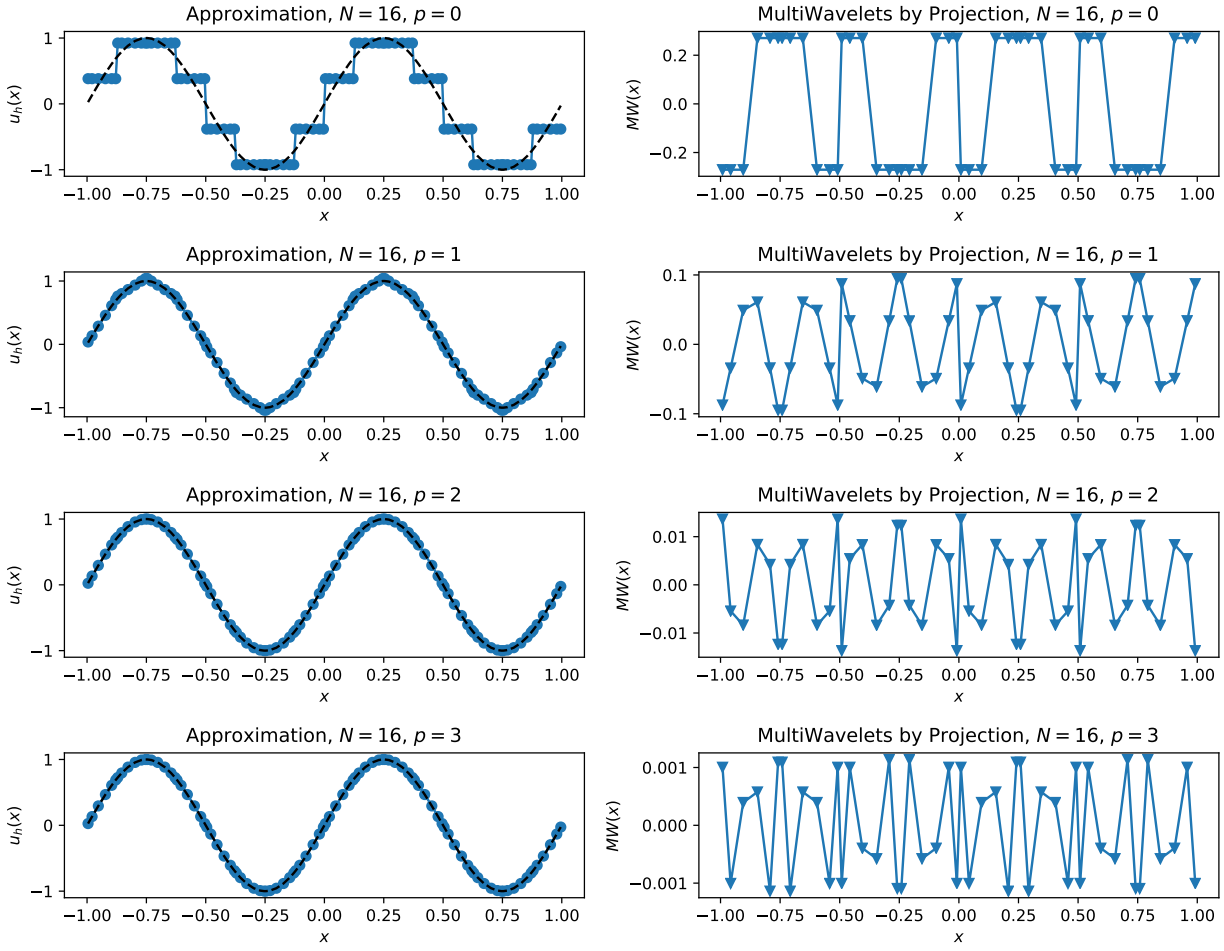

Figure 1: The approximation for $N = 16$ (left, with the dashed black line representing the exact solution) and the multiwavelets by the projection method (right) for the Eq. (1) Initial Condition

.

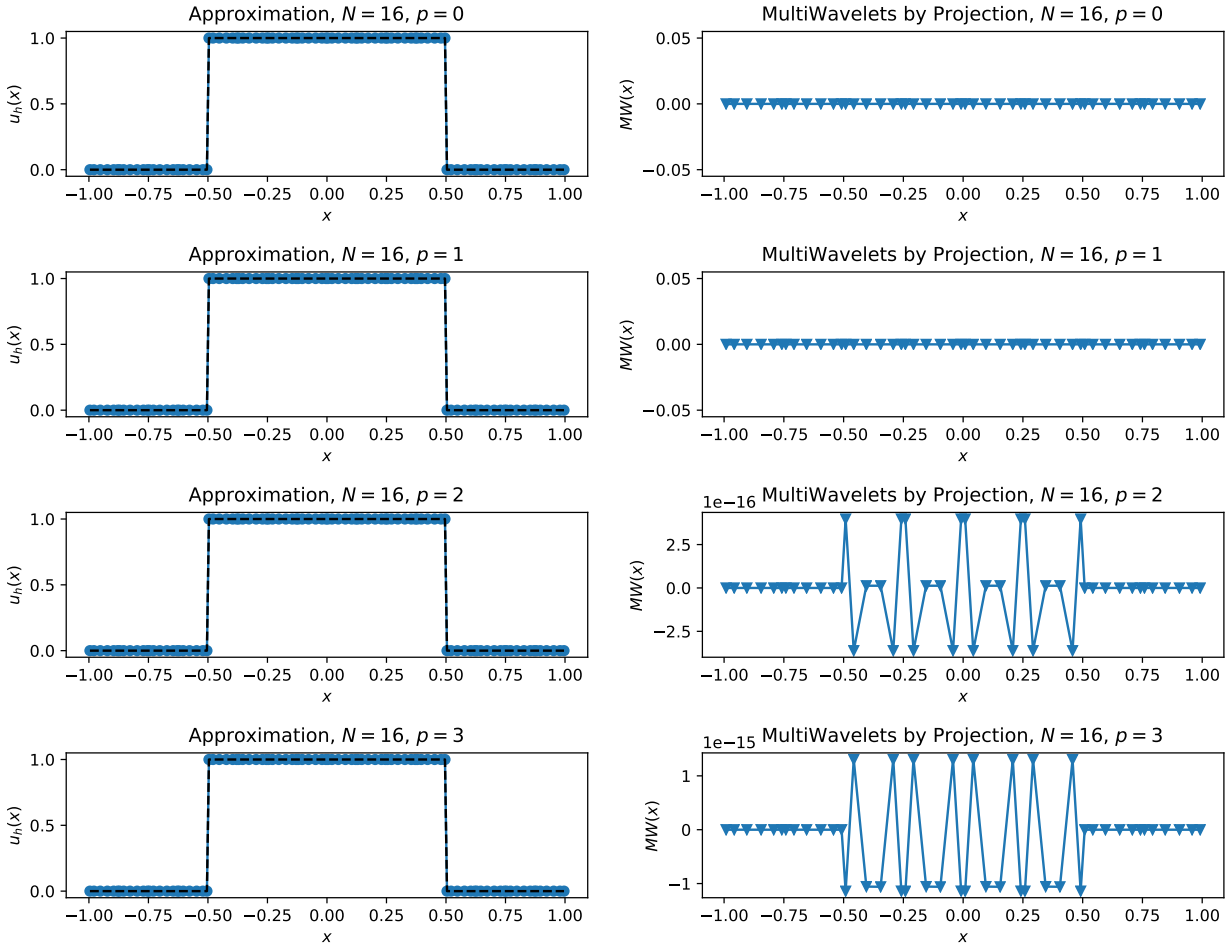

Figure 2: The approximation for $N = 16$ (left, with the dashed black line representing the exact solution) and the multiwavelets by the projection method (right) for the Eq. (2) Initial Condition

.

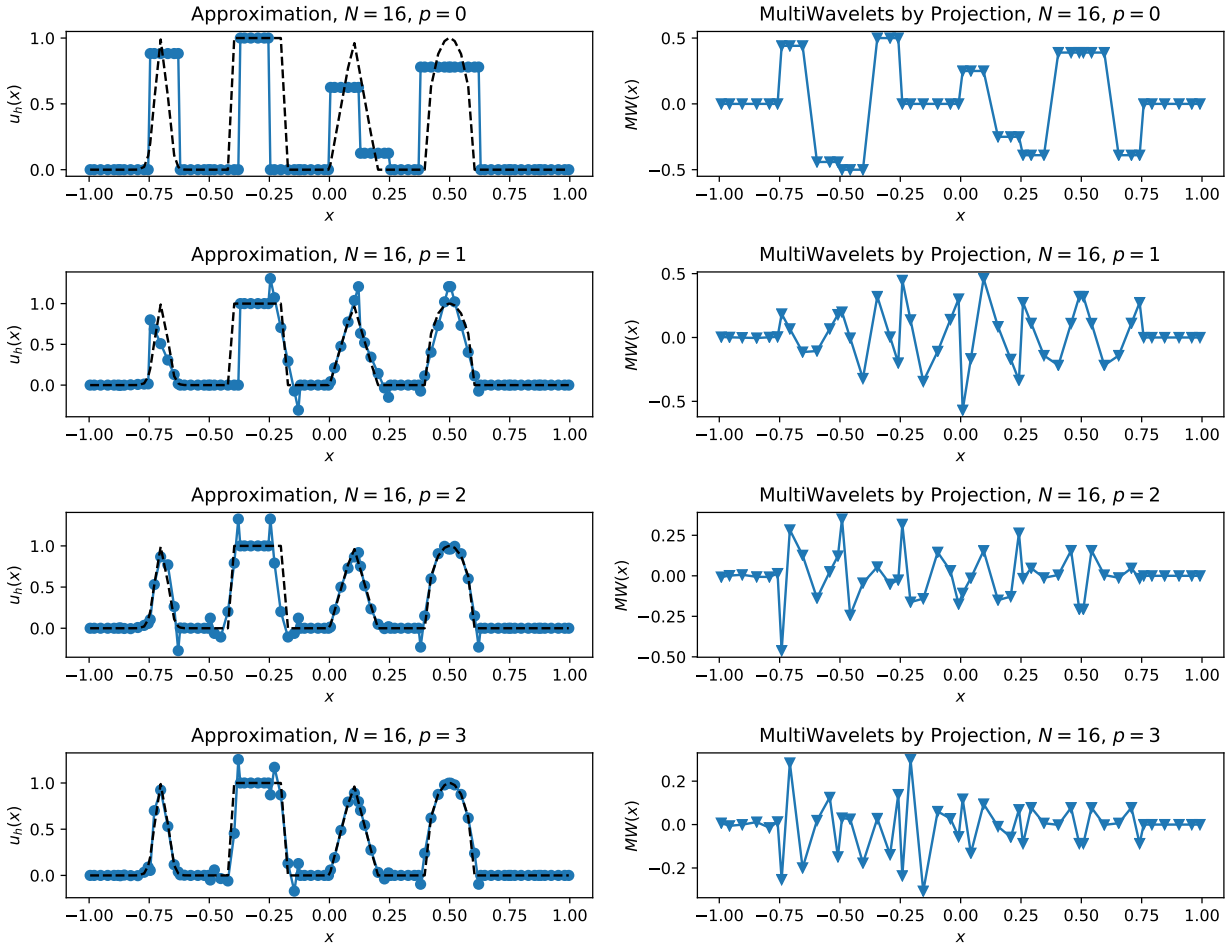

Figure 3: The approximation for $N = 16$ (left, with the dashed black line representing the exact solution) and the multiwavelets by the projection method (right) for the Eq. (3) Initial Condition

.

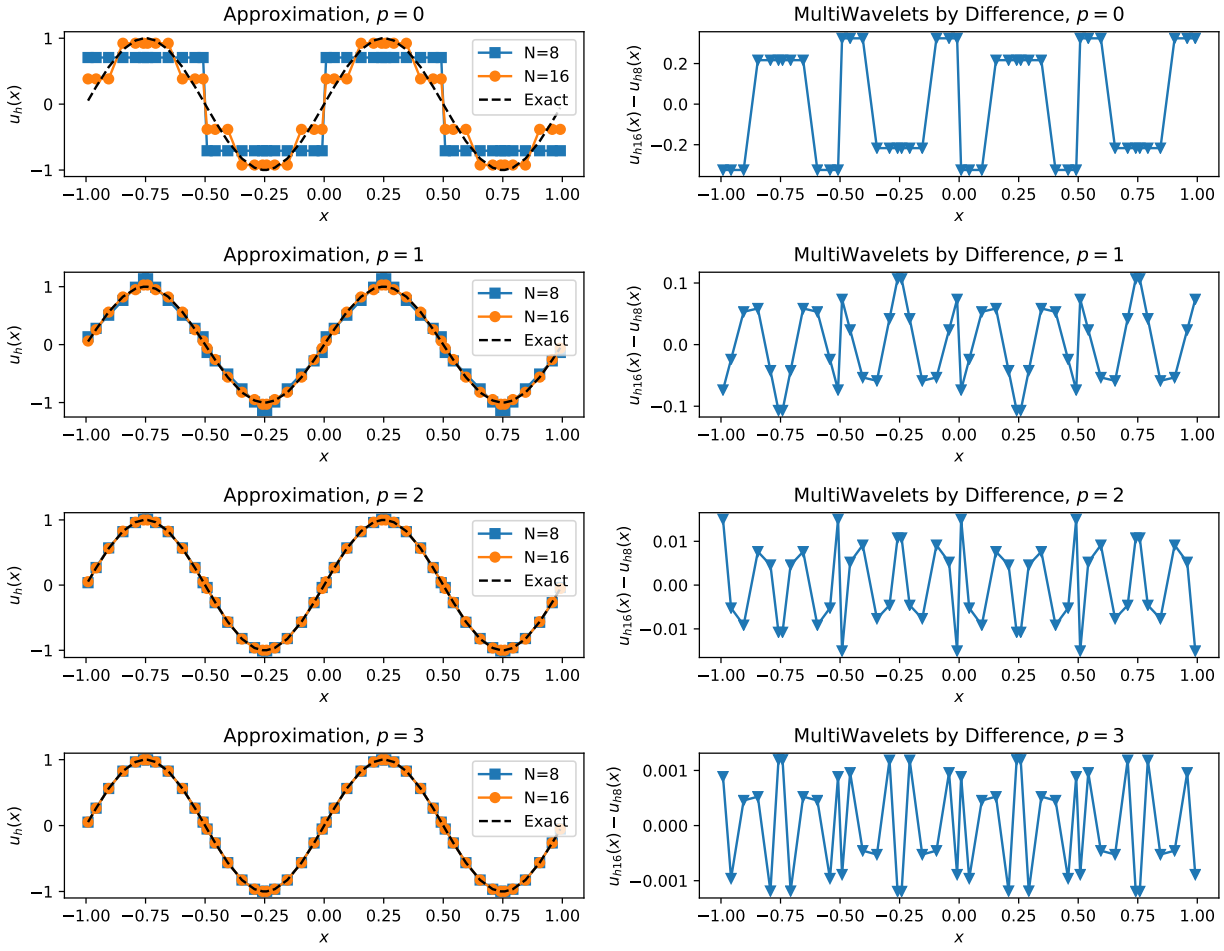

Figure 4: The approximation and exact solution (black dashed line) for $N = 8,\ 16$ (left) and the corresponding multiwavelets due to the difference between resolutions (right) for Eq. (1) Initial Condition.

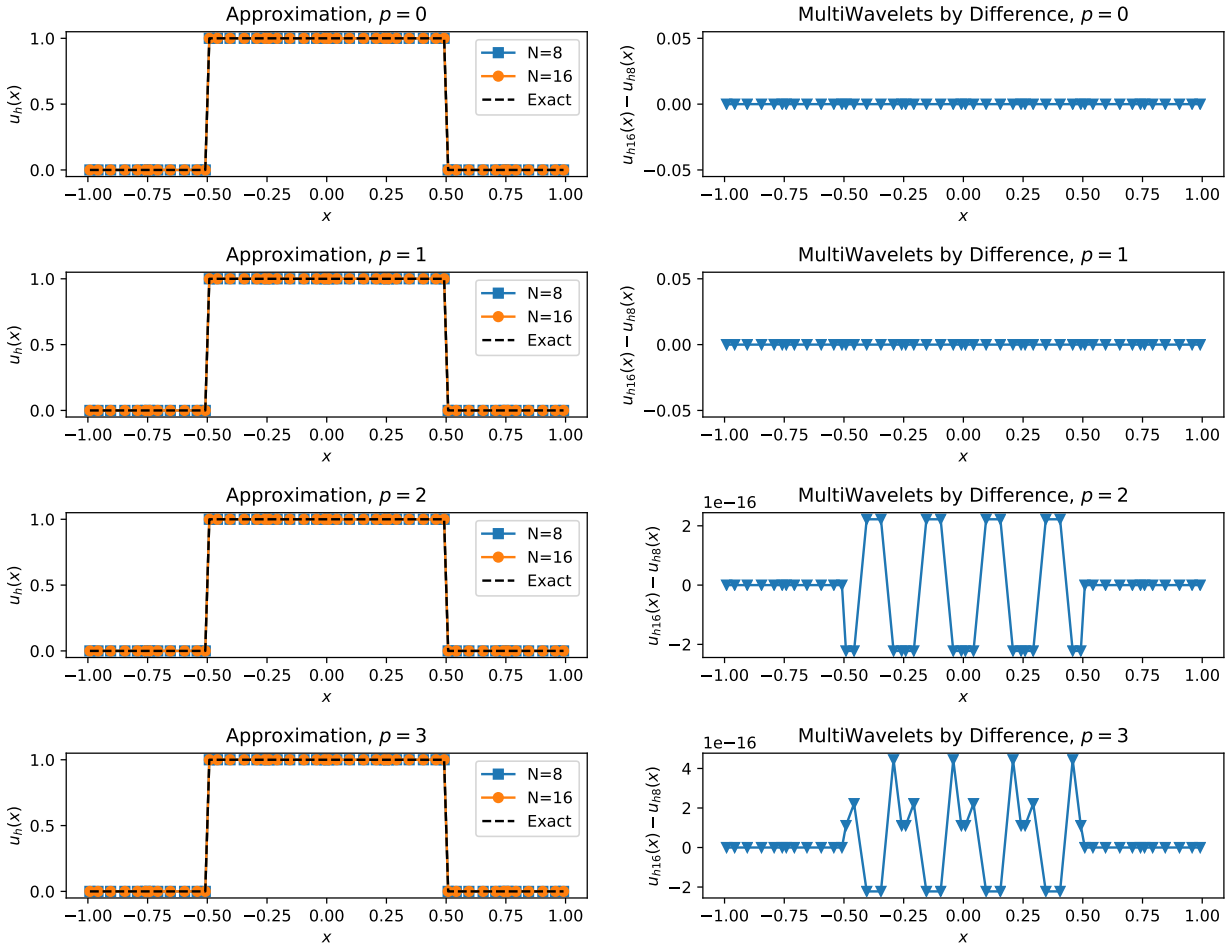

Figure 5: The approximation and exact solution (black dashed line) for $N = 8,\ 16$ (left) and the corresponding multiwavelets due to the difference between resolutions (right) for Eq. (2) Initial Condition.

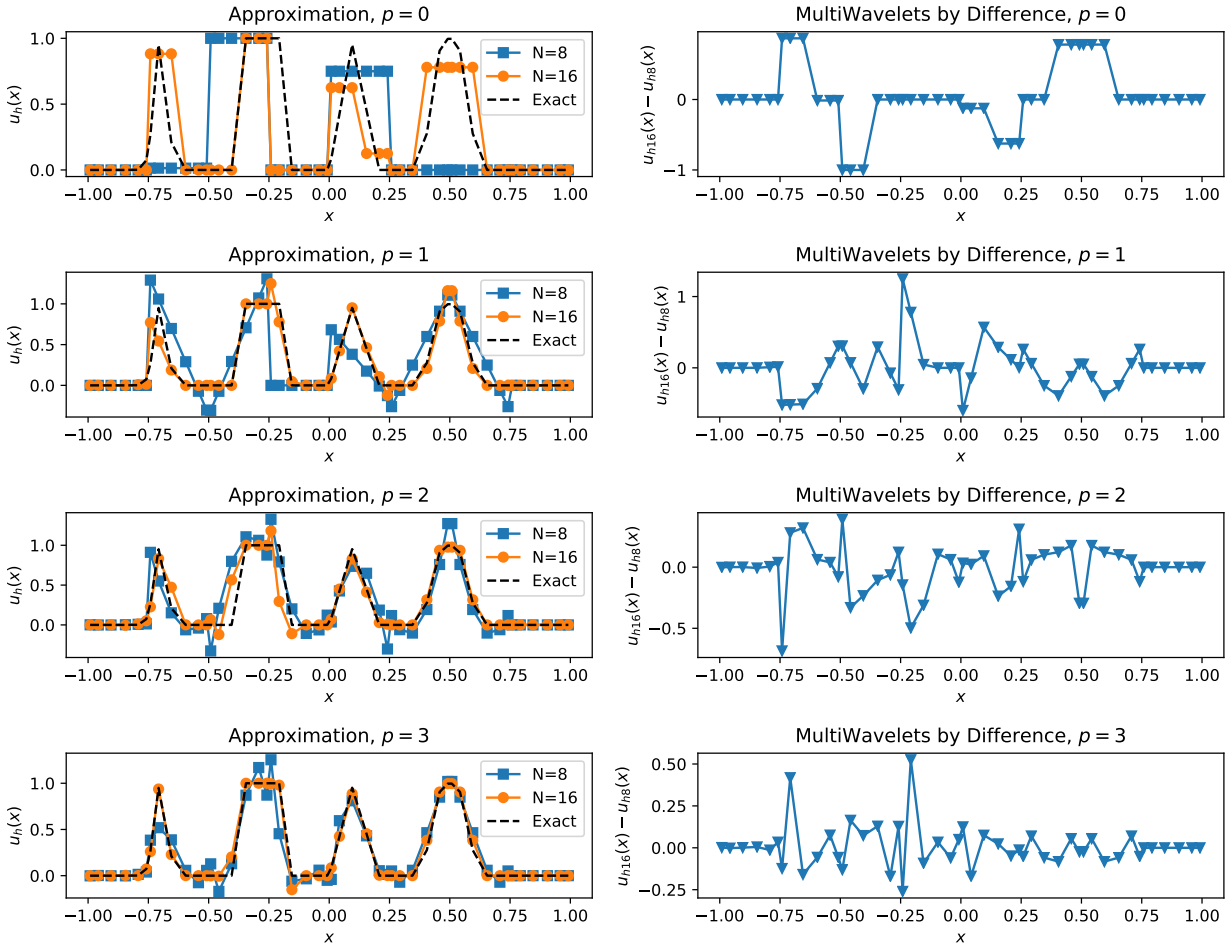

Figure 6: The approximation and exact solution (black dashed line) for $N = 8,\ 16$ (left) and the corresponding multiwavelets due to the difference between resolutions (right) for Eq. (3) Initial Condition.

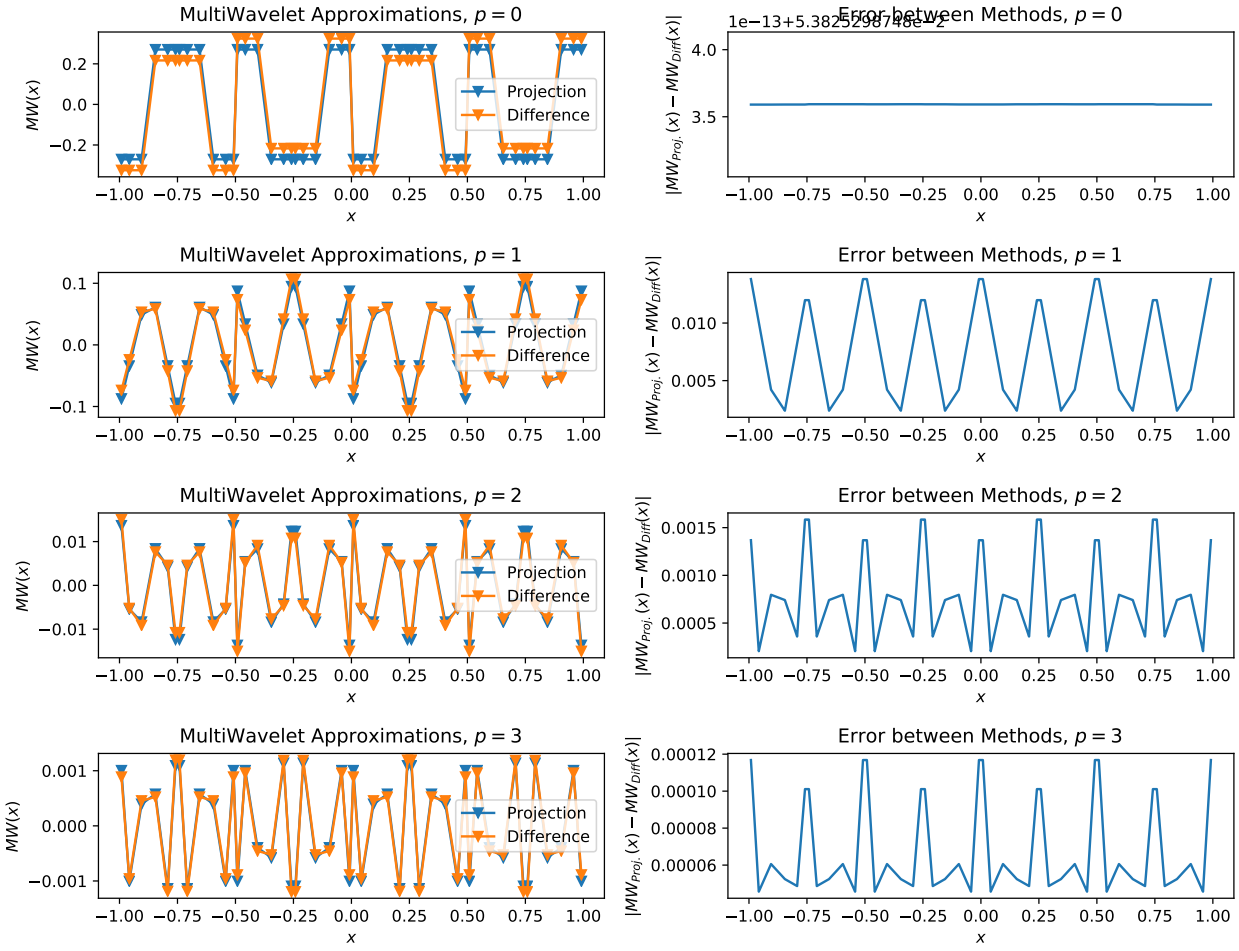

Figure 7: Multiwavelets determined by both the projections and difference methods (left) and the error between the two methods (right) for Eq. (1) Initial Condition.

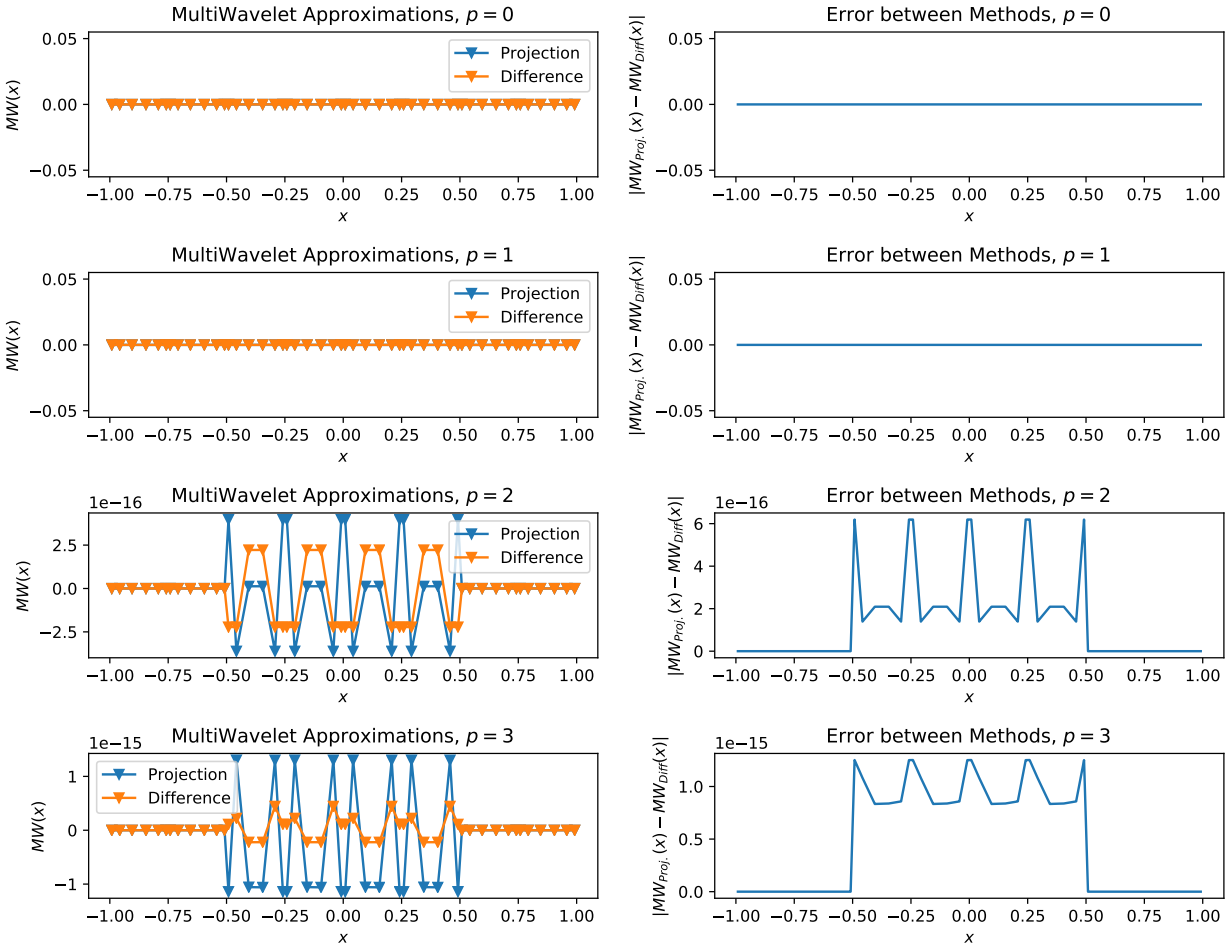

Figure 8: Multiwavelets determined by both the projections and difference methods (left) and the error between the two methods (right) for Eq. (2) Initial Condition.

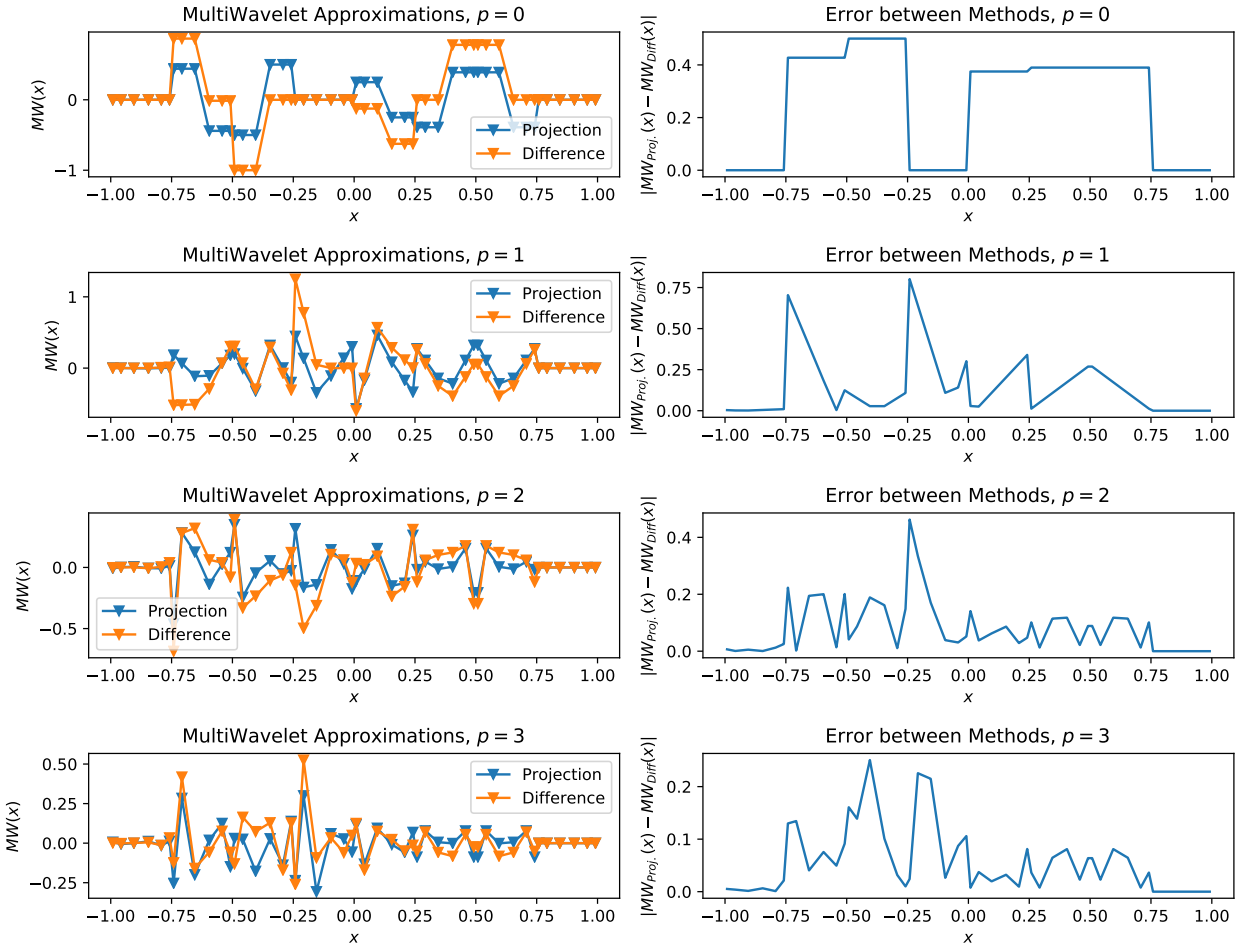

Figure 9: Multiwavelets determined by both the projections and difference methods (left) and the error between the two methods (right) for Eq. (3) Initial Condition.

2. (a) **Theoretical Derivation:** Derive the semi-discrete discontinuous Galerkin scheme using an upwind flux,

$$\hat{u}_{j+1/2} = \begin{cases} u^-_{j+1/2}, & a > 0 \\ u^+_{j+1/2}, & a < 0 \end{cases}.$$

The final form should be the variational formulation in terms of the approximation $u_h(x,t)$ and test function $v_h(x)$. Remember to include the initial and boundary conditions.

To derive the semi-discrete discontinuous Galerkin scheme, we want to multiply the linear advection equation with a test function v and integrate over an element I_j:

$$\begin{aligned} u_t + (au)_x &= 0, \\ v(u_t + (au)_x) &= 0, \\ \int_{I_j} (u_t + (au)_x) v dx &= 0, \\ \int_{I_j} u_t v dx + auv\Big|_{I_j} - \int_{I_j} auv_x dx &= 0, \\ \int_{I_j} u_t v dx &= \int_{I_j} auv_x dx - auv\Big|_{x_{j-1/2}}^{x_{j+1/2}}. \end{aligned}$$

The approximation is multiply defines at the cell endpoints, $x_{j\pm1/2}$. Let's define a numerical flux, $\hat{f} = a\hat{u}$ where given $a > 0$,

$$\hat{f}_{j+1/2} = a\hat{u}_{j+1/2} = au^-_{j+1/2},$$

as we want to pick up the information on the left-most part of the boundary as the wave is moving from left to right. We want to determine $u_h \in V_h^p$ such that

$$\int_{I_j} (u_h)_t v_h dx = \int_{I_j} au_h (v_h)_x dx - \left[a\hat{u}_{j+1/2} v^-_{j+1/2} - a\hat{u}_{j-1/2} v^+_{j-1/2} \right]$$

for all $v_h \in V_h^p(I_j)$. With the upwind flux defined above, we have the semi-discrete weak variantional formulation given as

$$\boxed{\int_{I_j} (u_h)_t v_h dx = a \int_{I_j} u_h (v_h)_x dx - a \left(u^-_{h,j+1/2} v^-_{j+1/2} - u^-_{h,j-1/2} v^+_{j-1/2} \right)},$$

where

$$u_h(x,0) = \sin(2\pi x), \quad u_h(-1,t) = u_h(1,t)$$

to satisfy the periodic initial value problem. The periodic boundary conditions will be enforced by adding ghost cells such that the modes at cell 0 match the nodes at cell N and the modes at cell -1 match the nodes at cell N, where cell $j = 0, \ldots, N-1$ are the N cells in $[-1, 1]$. The solution to the problem is $u(x,t) = \sin(\pi(x - at))$.

(b) **Theoretical derivation for computation:** Use the Legendre basis,

$$V_h^p = \{P^{(m)}(\xi_j),\ m = 0, \ldots, p,\ j = 1, \ldots, m\} \quad \text{with } \xi_j = \frac{2}{\Delta x_j}(x - x_j),$$

to put the above variational formulation into an ODE matrix vector format (including the initial condition). Note that the Legendre polynomials are given by the recurrence relation

$$P^{(0)}(\xi_j) = 1, \quad P^{(1)}(\xi_j) = \xi_j,$$

$$P^{(m+1)}(\xi_j) = \frac{2m+1}{m+1}\xi_j P^{(m)}(\xi_j) - \frac{m}{m+1}P^{(m-1)}(\xi_j), \quad m = 1, 2, \ldots, p,$$

At this point, the components of the matrices should be in terms of integrals of the basis functions.
Given the Legendre basis and the local transformation variable $\xi_j = \frac{2}{\Delta x_j}(x - x_j)$, the approximation and test functions are given as

$$u_h(x,t)\Big|_{I_j} = \sum_{k=0}^{p} u_j^{(k)}(t) P^{(k)}(\xi_j), \quad v_h(x) = P^{(m)}(\xi_j).$$

For the variational form derived in (a), the left-hand side can be simplified as

$$\begin{aligned}
\int_{I_j} (u_h)_t v_h dx &= \int_{I_j} \frac{\partial}{\partial t}\left(\sum_{k=0}^{p} u_j^{(k)}(t) P^{(k)}(\xi_j)\right) P^{(m)}(\xi_j) dx, \\
&= \sum_{k=0}^{p} \frac{d}{dt} u_j^{(k)}(t) \int_{I_j} P^{(k)}(\xi_j) P^{(m)}(\xi_j) dx, \\
&= \frac{\Delta x_j}{2} \sum_{k=0}^{p} \frac{d}{dt} u_j^{(k)}(t) \int_{-1}^{1} P^{(k)}(\xi_j) P^{(m)}(\xi_j) d\xi_j, \\
&= \frac{\Delta x_j}{2} \sum_{k=0}^{p} M(m,k) \frac{d}{dt} u_j{}^{(k)}(t), \\
&= \frac{\Delta x_j}{2} M \frac{d}{dt} \boldsymbol{u_j}(t),
\end{aligned}$$

where M is the $(p+1)\times(p+1)$ mass matrix and $\boldsymbol{u_j} = [u_j^{(0)}\ u_j^{(1)}\dots\ u_j^{(p)}]^T$. Specifically,

$$M = \begin{bmatrix} \int_{-1}^{1} P^{(0)}(\xi_j)P^{(0)}(\xi_j)d\xi_j & \dots & \int_{-1}^{1} P^{(p)}(\xi_j)P^{(0)}(\xi_j)d\xi_j \\ \vdots & \ddots & \vdots \\ \int_{-1}^{1} P^{(0)}(\xi_j)P^{(p)}(\xi_j)d\xi_j & \dots & \int_{-1}^{1} P^{(p)}(\xi_j)P^{(p)}(\xi_j)d\xi_j \end{bmatrix},$$

For the volume integral on the right-hand side of the variational form, it can be simplified in matrix form as follows

$$\begin{aligned} a\int_{I_j} u_h(v_h)_x dx &= a\int_{I_j}\sum_{k=0}^{p} u_j^{(k)}(t)P^{(k)}(\xi_j)\frac{d}{dx}\left(P^{(m)}(\xi_j)\right)dx, \\ &= a\sum_{k=0}^{p} u_j^{(k)}(t)\int_{I_j} P^{(k)}(\xi_j)\left(\frac{d\xi_j}{dx}\frac{d}{d\xi_j}P^{(m)}(\xi_j)\right)dx, \\ &= a\frac{2}{\Delta x_j}\sum_{k=0}^{p} u_j^{(k)}(t)\int_{I_j} P^{(k)}(\xi_j)\left(\frac{d}{d\xi_j}P^{(m)}(\xi_j)\right)dx, \\ &= a\frac{2}{\Delta x_j}\frac{\Delta x_j}{2}\sum_{k=0}^{p} u_j^{(k)}(t)\int_{-1}^{1} P^{(k)}(\xi_j)\left(\frac{d}{d\xi_j}P^{(m)}(\xi_j)\right)d\xi_j, \\ &= a\sum_{k=0}^{p} u_j^{(k)}(t)\int_{-1}^{1} P^{(k)}(\xi_j)\left(\frac{d}{d\xi_j}P^{(m)}(\xi_j)\right)d\xi_j, \\ &= a\sum_{k=0}^{p} D(m,k)u_j^{(k)}(t), \\ &= aD\boldsymbol{u_j}(t), \end{aligned}$$

where D is given as

$$D = \begin{bmatrix} \int_{-1}^{1} P^{(0)}(\xi_j)\left(\frac{d}{d\xi_j}P^{(0)}(\xi_j)\right)d\xi_j & \dots & \int_{-1}^{1} P^{(p)}(\xi_j)\left(\frac{d}{d\xi_j}P^{(0)}(\xi_j)\right)d\xi_j \\ \vdots & \ddots & \vdots \\ \int_{-1}^{1} P^{(0)}(\xi_j)\left(\frac{d}{d\xi_j}P^{(p)}(\xi_j)\right)d\xi_j & \dots & \int_{-1}^{1} P^{(p)}(\xi_j)\left(\frac{d}{d\xi_j}P^{(p)}(\xi_j)\right)d\xi_j \end{bmatrix},$$

where the derivatives of the Legendre polynomials are given as

$$\frac{d}{d\xi_j}P^{(0)}(\xi_j) = \frac{d}{d\xi_j}(1) = 0, \quad \frac{d}{d\xi_j}P^{(1)}(\xi_j) = \frac{d}{d\xi_j}\xi_j = 1,$$

$$\frac{d}{d\xi_j}P^{(m+1)}(\xi_j) = \frac{d}{d\xi_j}\left[\frac{2m+1}{m+1}\xi_j P^{(m)}(\xi_j) - \frac{m}{m+1}P^{(m-1)}(\xi_j)\right], \quad m = 1, 2, \ldots, p,$$
$$= \frac{2m+1}{m+1}P^{(m)}(\xi_j) + \frac{2m+1}{m+1}\xi_j\left(\frac{d}{d\xi_j}P^{(m)}(\xi_j)\right) - \frac{m}{m+1}\left(\frac{d}{d\xi_j}P^{(m-1)}(\xi_j)\right).$$

The boundary terms that remain can be written in matrix vector form as

$$-a\left[u^-_{h,j+1/2}v^-_{j+1/2} - u^-_{h,j-1/2}v^+_{j-1/2}\right] = -a\sum_{k=0}^{p}\left[u_j^{(k)}(t)P^{(k)}(1)P^{(m)}(1) - u_{j-1}^{(k)}(t)P^{(k)}(1)P^{(m)}(-1)\right],$$
$$= -aS_1\boldsymbol{u_j}(t) + aS_2\boldsymbol{u_{j-1}}(t),$$

where

$$S_1 = \begin{bmatrix} P^{(0)}(1)P^{(0)}(1) & \ldots & P^{(p)}(1)P^{(0)}(1) \\ \vdots & \ddots & \vdots \\ P^{(0)}(1)P^{(p)}(1) & \ldots & P^{(p)}(1)P^{(p)}(1) \end{bmatrix}$$

and

$$S_2 = \begin{bmatrix} P^{(0)}(1)P^{(0)}(-1) & \ldots & P^{(p)}(1)P^{(0)}(-1) \\ \vdots & \ddots & \vdots \\ P^{(0)}(1)P^{(p)}(-1) & \ldots & P^{(p)}(1)P^{(p)}(-1) \end{bmatrix}.$$

Finally, the variational form found in (a) is written in matrix-vector form as follows:

$$\frac{\Delta x_j}{2}M\frac{d}{dt}\boldsymbol{u_j}(t) = aD\boldsymbol{u_j}(t) - aS_1\boldsymbol{u_j}(t) + aS_2\boldsymbol{u_{j-1}}(t),$$
$$\frac{\Delta x_j}{2}M\frac{d}{dt}\boldsymbol{u_j}(t) = a\left[D\boldsymbol{u_j}(t) - S_1\boldsymbol{u_j}(t) + S_2\boldsymbol{u_{j-1}}(t)\right],$$
$$\frac{\Delta x_j}{2}M\frac{d}{dt}\boldsymbol{u_j}(t) = a\left[(D - S_1)\boldsymbol{u_j}(t) + S_2\boldsymbol{u_{j-1}}(t)\right],$$

which gives us

$$\boxed{\frac{d}{dt}\boldsymbol{u_j}(t) = a\frac{2}{\Delta x_j}M^{-1}\left[(D - S_1)\boldsymbol{u_j}(t) + S_2\boldsymbol{u_{j-1}}(t)\right].}$$

The initial modes will be initiated through the projection

$$(u_h(x,0), P^{(m)}(\xi))_{I_j} = (u_0(x), P^{(m)}(\xi))_{I_j}, \ m = 0, \ldots, p, \ j = 0, \ldots N-1.$$

For the left-hand side of the equation above, we can use orthogonality of the orthonormal Legendre basis to result in

$$\begin{aligned}(u_h(x,0),P^{(m)}(\xi))_{I_j} &= \left(\sum_{\ell=0}^{p} u_j^{(\ell)}(0)P^{(\ell)}(\xi_j),P^{(m)}(\xi)\right)_{I_j},\\ &= \sum_{\ell=0}^{p}\left(u_j^{(\ell)}P^{(\ell)}(\xi_j),P^{(m)}(\xi)\right)_{I_j},\\ &= \sum_{\ell=0}^{p} u_j^{(\ell)}\left(P^{(\ell)}(\xi_j),P^{(m)}(\xi)\right)_{I_j},\\ &= u_j^{(m)}\left(P^{(m)}(\xi_j),P^{(m)}(\xi)\right)_{I_j},\\ &= u_j^{(m)}\frac{\Delta x}{2}\frac{2}{2m+1},\end{aligned}$$

We hence have

$$\begin{aligned}(u_h(x,0),P^{(m)}(\xi))_{I_j} &= (u_0(x),P^{(m)}(\xi))_{I_j},\\ u_j^{(m)}\frac{\Delta x}{2}\frac{2}{2m+1} &= \int_{I_j} u_0(x)P^{(m)}(\xi)dx,\\ u_j^{(m)}\frac{\Delta x}{2}\frac{2}{2m+1} &= \frac{\Delta x}{2}\int_{-1}^{1} u_0(x)P^{(m)}(\xi)d\xi,\end{aligned}$$

and thus

$$u_j^{(m)} = \frac{2m+1}{2}\int_{-1}^{1} u_0(x)P^{(m)}(\xi)d\xi, \quad m=0,\dots,p,\ \ j=0,\dots,N-1.$$

(c) **Theoretical derivation for computation:** Explicitly solve the integrals associated with the components of the matrices for $p=2$ and $p=5$, which should be numerical values. You may do this with the help of Mathematica or Sympy.
The Sympy/Numpy script used to evaluate the integrals of the matrices is given as:

```
import numpy as np
import sympy as sym

## Calculating Matrices for Weak Variational Form for u_t+au_x=0
p=5
xi = sym.symbols('xi')
M = np.zeros((p+1, p+1))
D = np.zeros((p+1,p+1))
S1 = np.zeros((p+1,p+1))
```

```
S2 = np.zeros((p+1,p+1))
for i in range(p+1):
    for j in range(p+1):
        M[i,j] = sym.integrate(sym.legendre(j, ...
            xi)*sym.legendre(i, xi), (xi, -1, 1))
        D[i,j] = sym.integrate(sym.legendre(j, ...
            xi)*sym.diff(sym.legendre(i, xi), xi), (xi, -1, 1))
        S1[i,j] = sym.legendre(j, 1)*sym.legendre(i, 1)
        S2[i,j] = sym.legendre(j, 1)*sym.legendre(i, -1)

M_inverse = np.linalg.inv(M)
print("M, M^{-1}, D-S1, S2 : ")
print(M, M_inverse)
print(D-S1, S2)
```

The matrices for $p = 2$ are

$$M = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2/3 & 0 \\ 0 & 0 & 2/5 \end{bmatrix}, \; M^{-1} = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 3/2 & 0 \\ 0 & 0 & 5/2 \end{bmatrix},$$

$$D - S_1 = \begin{bmatrix} -1 & -1 & -1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \end{bmatrix}, \text{ and } S_2 = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \\ 1 & 1 & 1 \end{bmatrix}.$$

The matrices for $p = 5$ are

$$M = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2/5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2/7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2/9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2/11 \end{bmatrix}, \; M^{-1} = \begin{bmatrix} 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11/2 \end{bmatrix},$$

$$D-S_1 = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}, \text{ and } S_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}.$$

(d) **Theoretical derivation for computation:** Write the procedure for the fully discrete scheme. That is, combine the matrix-vector format in part (b) with the third order strong stability preserving Runge-Kutta scheme (SSPRK3).
From part (b), we have

$$\frac{d}{dt}\boldsymbol{u_j}^{(n)}(t) = a\frac{2}{\Delta x_j} M^{-1}\left[(D - S_1)\boldsymbol{u_j}^{(n)}(t) + S_2\boldsymbol{u_{j-1}}^{(n)}(t)\right],$$

where the time superscript was added such that $\boldsymbol{u_j}^{(n)}(t)$ represents the vector of modes at cell j at time level t^n, with $t^{n+1} = t^n + \Delta t$. With that, we have

$$\frac{d}{dt}\boldsymbol{u_j}^{(n)}(t) = L(\boldsymbol{u_j}^{(n)}(t)), \quad L(\boldsymbol{u_j}^{(n)}(t)) = a\frac{2}{\Delta x_j}M^{-1}\left[(D - S_1)\boldsymbol{u_j}^{(n)}(t) + S_2\boldsymbol{u_{j-1}}^{(n)}(t)\right].$$

The third order strong stability preserving Runge-Kutta scheme (SSPRK3) given as

$$\begin{aligned}
\boldsymbol{u_j}^{(1)} &= \boldsymbol{u_j}^{(n)} + \Delta t L(\boldsymbol{u_j}^{(n)}),\\
\boldsymbol{u_j}^{(2)} &= \frac{1}{4}\left(3\boldsymbol{u_j}^{(n)} + (\boldsymbol{u_j}^{(1)} + \Delta t L(\boldsymbol{u_j}^{(1)}))\right),\\
\boldsymbol{u_j}^{(n+1)} = \boldsymbol{u_j}^{(3)} &= \frac{1}{3}\left(\boldsymbol{u_j}^{(n)} + 2(\boldsymbol{u_j}^{(2)} + \Delta t L(\boldsymbol{u_j}^{(2)}))\right),
\end{aligned}$$

where the boundary conditions for each of the equations are enforced at time levels t^{n+1}, $t^{n+1/2}$, t^{n+1}, respectively. Also, in the above, we stil have the time-dependence of the modes, i.e. $\boldsymbol{u_j}^{(\#)} = \boldsymbol{u_j}^{(\#)}(t)$. With that, we can write the SSPRK3 scheme for our problem as follows with $\lambda = a\Delta t/\Delta x_j$ given $a > 0$:

$$\begin{aligned}
\boldsymbol{u_j}^{(1)} &= \boldsymbol{u_j}^{(n)} + 2\lambda M^{-1}\left[(D - S_1)\boldsymbol{u_j}^{(n)} + S_2\boldsymbol{u_{j-1}}^{(n)}\right],\\
\boldsymbol{u_j}^{(2)} &= \frac{1}{4}\left(3\boldsymbol{u_j}^{(n)} + \left(\boldsymbol{u_j}^{(1)} + 2\lambda M^{-1}\left[(D - S_1)\boldsymbol{u_j}^{(1)} + S_2\boldsymbol{u_{j-1}}^{(1)}\right]\right)\right),\\
\boldsymbol{u_j}^{(n+1)} = \boldsymbol{u_j}^{(3)} &= \frac{1}{3}\left(\boldsymbol{u_j}^{(n)} + 2\left(\boldsymbol{u_j}^{(2)} + 2\lambda M^{-1}\left[(D - S_1)\boldsymbol{u_j}^{(2)} + S_2\boldsymbol{u_{j-1}}^{(2)}\right]\right)\right).
\end{aligned}$$

(e) One contribution to roundoff error comes from inverting the mass matrix,

$$M(m,\ell) = \frac{\Delta x}{2}\int_{-1}^{1} P^{(\ell)}(\xi)P^{(m)}(\xi)d\xi.$$

The Legendre polynomial reduces this error because it is an orthogonal basis. How might we reduce the errors further?
We can use an orthonormal basis such that $M = M^{-1} = I$, resulting in no accumulated roundoff errors.

2. **Computational Implementation:** Consider $a > 0$. Use your results from Exercise 2 to implement a DG scheme for the linear advection equation with an upwind flux, where the user inputs the wave speed, a, and polynomial degree, p. Evaluate the solution at six Gauss-Legendre points per element. Below, you will run the code for $a = 1$. Note, the stability limits fro the DG scheme using SSPRK3 are

p	0	1	2	3	4	5
cfl	1.256	0.409	0.209	0.13	0.089	0.066

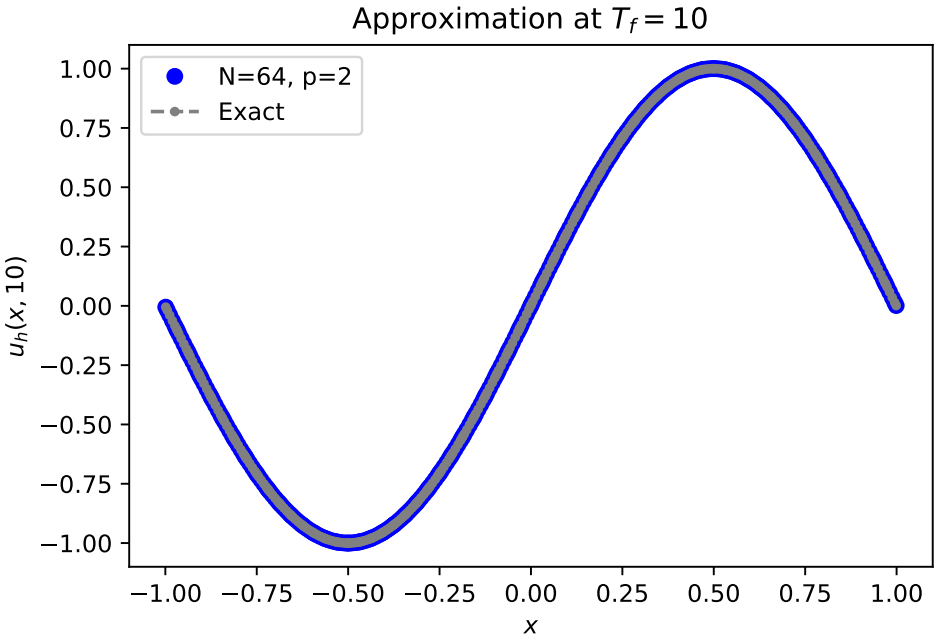

Figure 10: The approximation $u_h(x,T)$ and the exact solution $u(x,T)$ at $T = 10$ for $p = 2,\ N = 64$ and six Gauss-Legendre points per element.

Denote the cfl by λ. For $p+1 \geq 3$, in order to see the correct order of the scheme, we need to ensure that spatial errors dominate. We do this using

$$\lambda_{new} = \lambda_{old} \left(\frac{\Delta x_{new}}{\Delta x_{old}} \right)^{3/(p+1)}.$$

(a) Plot the approximation, $u_h(x,t)$, and exact solution, $u(x,t)$, at the cell center, x_j for $T = 10$ (five periods in time) for $p = 2,\ N = 64$. Compare with plotting at six Gauss-Legendre points per element,

$$x = x_j + \frac{\Delta x}{2}\zeta_n, \quad n = 1, \ldots, 6,$$

where ζ_n is a Gauss-Legendre node. State your observations.
The computation of the SSPRK3 DG scheme was computed using *HW-LinearTransport.py* file, which called on functions in *OrthoLegendreBasis-Projections.py* file. The approximation shown in Fig. 10 along with the exact solution match exactly. This indicates that the $p = 2$ approximation accurately represents the true function at $T_f = 10$.

(b) Plot the semilogy error for the approximation, $u_h(x,t)$, and exact solution, $u(x,t)$, at the cell center, x_j for $T = 10$ (five periods in time) for $p = 2,\ N = 64$. Compare

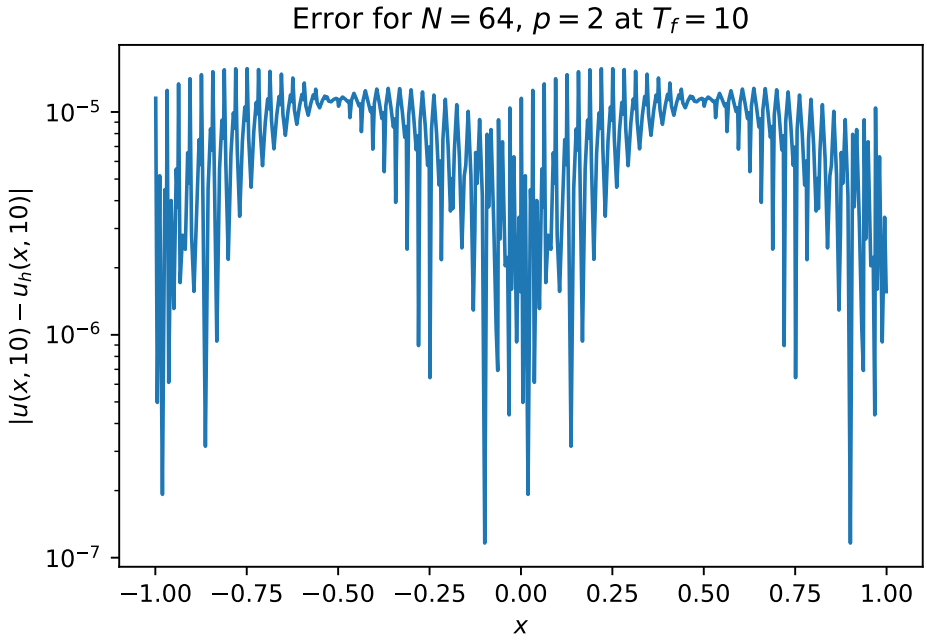

Figure 11: The error of the approximation $u_h(x, T)$ compared to the exact solution $u(x, T)$ at $T = 10$ for $p = 2$, $N = 64$ and six Gauss-Legendre points per element.

with plotting at six Gauss-Legendre points per element. State your observations. *Note: To do this in Matlab, the command is* `semilogy(x,abs(uexact-uapprox))`.

The error on the approximation in Fig. 10 is shown in Fig. 11, where the errors range from $10^{-7} - 10^{-5}$. Additionally, similar to the function and the approximation, the error is roughly periodic in $(-1, 1)$, indicating roughly the same accuracy of the approximation in $(-1, 0)$ as in $(0, 1)$.

(c) Calculate the L^2- and $L^\infty-$ errors along with the order for $p = 1, 2, 3$; $N = 8, 16, 32, 64$. State your observations.
We expect the errors to decrease by $(1/2)^{p+1}$ as we refine the mesh by doubling the number of elements. We also expect the order to be of $p + 1$. Tables 1-2 indicate the L^2- and $L^\infty-$ errors, respectively, for the approximation at $T_f = 10$. For $p = 1$, we do see both the L^2- and $L^\infty-$ errors drop by more than $(1/2)^2 = 1/4$ for each subsequent N value. Additionally, the order does translate to more than 2, given the drop in error relative to a doubling of mesh size, or equivalently halving the grid spacing Δx. In the case of $p = 2$, where we should expect errors to drop by a factor of 8 and the order to be 3, we only see the errors drop by 10 for $N = 16$ compared to $N = 8$. The L^2- errors drop by a factor of less than 8 for subsequent N values while the $L^\infty-$ errors drop by a factor close to 8. As a

	$p=1$		$p=2$		$p=3$	
N	L^2-**Error**	**Order**	L^2-**Error**	**Order**	L^2-**Error**	**Order**
8	1.0486	-	3.6198×10^{-2}	-	6.9331×10^{-3}	-
16	2.1123×10^{-1}	2.3115	2.5376×10^{-3}	3.8344	2.6482×10^{-4}	4.7104
32	4.2211×10^{-2}	2.3231	4.2490×10^{-4}	2.5782	1.1053×10^{-5}	4.5824
64	1.0508×10^{-2}	2.0062	7.5041×10^{-5}	2.5014	5.8452×10^{-7}	4.2411

Table 1: L^2 Error and corresponding Order at $T_f = 10$ for $p = 1, 2, 3$ and $N = 8, 16, 32, 64$.

	$p=1$		$p=2$		$p=3$	
N	$L^\infty-$**Error**	**Order**	$L^\infty-$**Error**	**Order**	$L^\infty-$**Error**	**Order**
8	2.4561×10^{-1}	-	7.6028×10^{-3}	-	1.6382×10^{-3}	-
16	3.8885×10^{-2}	2.6591	7.0683×10^{-4}	3.4271	5.0803×10^{-5}	5.0111
32	6.0407×10^{-3}	2.6864	8.9724×10^{-5}	2.9778	1.8837×10^{-6}	4.7533
64	1.0206×10^{-3}	2.5653	1.1285×10^{-5}	2.9911	8.3239×10^{-8}	4.5001

Table 2: L^∞ Error and corresponding Order at $T_f = 10$ for $p = 1, 2, 3$ and $N = 8, 16, 32, 64$.

consequence, we only see order 3 for the L^2- Error at $N = 16$ and around 2.5 for the other N values. The $L^\infty-$ order is really close to 3 at $N = 32, 64$ and greater than 3 at $N = 16$ given a factor of 10 drop in error. Finally, for $p = 3$, both the L^2- and $L^\infty-$ errors drop by more than a factor of 16 and the orders are thus all greater than 4, given faster convergence than expected. These simulations did run close to 1.5 hours for me, given the very small time-step as the CFL condition was getting smaller and smaller for subsequent mesh sizes.

References

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, New York, tenth printing, 1972 ed., 1964.