# CgWave: Composite Grid Wave Equation Solver. User Guide and Reference Manual

W. D. Henshaw[a,2,*]

[a]*Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, USA*

**Abstract**

Here is the user guide and reference manual for CgWave, a composite grid wave equation solver based on Overture. CgWave is meant to be a simple example of an efficient Overture based PDE solver that also runs in parallel. CgWave is also used by CgWaveHoltz to compute solutions to the time-harmonic wave equation (i.e. Helmholtz equation) using Daniel Appelö's WaveHoltz approach.

*Keywords:* Wave equation; overset grids.

## Contents



## 1. Introduction

**Note:** This is a work in progress. Note all features are implemented. yet.

*Department of Mathematical Sciences, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180, USA.
*Email address:* henshw@rpi.edu (W. D. Henshaw)

[1]This work was performed under DOE contracts from the ASCR Applied Math Program.
[2]Research supported by the National Science Foundation under grant DMS-1519934.
[3]Research supported by a U.S. Presidential Early Career Award for Scientists and Engineers.

Here is the user guide and reference manual for CgWave, a composite grid wave equation solver based on Overture. CgWave solves the wave equation in second-order form using overset grids. CgWave solves problems in two and three space dimensions.

Here is a citation [1].

**Things to do:**

1. Add 4th order compatibility conditions.
2. Sixth and eight-order accurate modified equation schemes.
3. Sixth and eight-order accurate BCs.
4. Finish 3D version of CgWave.
5. Finish upwind-dissipation for implicit time-stepping

## 2. Governing Equations

CgWave solves the initial boundary-value problem for the wave equation in second-order form for $u = u(\mathbf{x}, t)$,

$$\partial_t^2 u = c^2 \Delta u + f(\mathbf{x}, t), \qquad \text{for } \mathbf{x} \in \Omega,\ t > 0, \tag{1a}$$

$$Bu = g \qquad \text{for } \mathbf{x} \in \partial\Omega,\ t > 0, \tag{1b}$$

$$u(x, 0) = u_0(\mathbf{x}), \quad \partial_t u(\mathbf{x}, 0) = u_1(\mathbf{x}), \text{for } \mathbf{x} \in \partial\Omega. \tag{1c}$$

Here $Bu = g$ denotes some boundary conditions. Boundary conditions include

- Dirichlet
- Neumann
- Mixed
- Even symmetry
- Radiation boundaries (far field) – to-do.

## 3. Numerical Scheme

The equations 1 are advanced using a modified equation approach [2]. Second and fourth-order accurate schemes are currently available.

At fourth-order accuracy the scheme takes the form

$$D_{+t}D_{-t}U^n_{\mathbf{i}} = c^2\Delta_{4h}U^n_{\mathbf{i}} + \frac{\Delta t^2}{12}(c^2\Delta_{2h})^2 U^n_{\mathbf{i}} + f(\mathbf{x}_{\mathbf{i}}, t^n) + \frac{\Delta t^2}{12}\Big(c^2\Delta_{2h} f(\mathbf{x}_{\mathbf{i}}, t^n) + \partial_t^2 f(\mathbf{x}_{\mathbf{i}}, t^n)\Big)$$

where $\Delta_{ph}$ is a p-th order accurate approximation to $\Delta$.

## 4. High-order accurate modified equation schemes

In this section we consider efficient approaches to implement high-order accurate modified equation schemes on Cartesian and curvilinear grids.

The modified equation schemes are based on the Taylor series expansion $D_{+t}D_{-t}u$,

$$D_{+t}D_{-t}u = \partial_t^2 u + 2\frac{\Delta t^2}{4!}\partial_t^4 u + 2\frac{\Delta t^4}{6!}\partial_t^6 u + \dots \tag{2}$$

$$= \sum_{m=1}^{\infty} 2\frac{\Delta t^{2(m-1)}}{(2m)!}\partial_t^{2m} u \tag{3}$$

Using $\partial_t^2 u = c^2\Delta u$ and discretizing in space leads to the modified equation scheme

$$D_{+t}D_{-t}u_{\mathbf{j}} = c^2\Delta_h u_{\mathbf{j}} + 2\frac{\Delta t^2}{4!}(c^2\Delta)_h^2 u_{\mathbf{j}} + 2\frac{\Delta t^4}{6!}(c^2\Delta)_h^3 u_{\mathbf{j}} + \dots \tag{4}$$

$$= \sum_{m=1}^{p/2} 2\frac{\Delta t^{2(m-1)}}{(2m)!}(c^2\Delta)_h^m u_{\mathbf{j}} \tag{5}$$

There are $p/2$ terms on the right-hand-side, for example, an eighth-order accurate scheme is

$$D_{+t}D_{-t}u_{\mathbf{j}} = c^2\Delta_{h,8}u_{\mathbf{j}} + \frac{\Delta t^2}{12}(c^2\Delta)_{h,6}^2 u_{\mathbf{j}} + \frac{\Delta t^4}{360}(c^2\Delta)_{h,4}^3 u_{\mathbf{j}} + \frac{\Delta t^6}{20,160}(c^2\Delta)_{h,2}^4 u_{\mathbf{j}} \tag{6}$$

where, for example, $(c^2\Delta)_{h,4}^3$ denotes a fourth-order accurate approximation to $(c^2\Delta)^3$.

High-order accurate central difference approximations to spatial derivatives of different orders can be written in the form of the following expansions in powers of $D_{+x}D_{-x}$,

$$\partial_x^{2m+1}u = D_{0x}(D_{+x}D_{-x})^m \left[\sum_{\mu=0}^{\infty}\beta_\mu^{(2m+1)}(-\Delta x^2 D_{+x}D_{-x})^\mu\right] u_{\mathbf{j}}, \quad m = 0,1,2,3,\dots \tag{7}$$

$$\partial_x^{2m}u = (D_{+x}D_{-x})^m \left[\sum_{\mu=0}^{\infty}\beta_\mu^{(2m)}(-\Delta x^2 D_{+x}D_{-x})^\mu\right] u_{\mathbf{j}}, \qquad m = 1,2,3,\dots \tag{8}$$

The coefficients $\beta_\mu^{(\nu)}$ can be found by Taylor series. A nice trick is to substitute $u = e^{ikx}$ and $u_{\mathbf{j}} = e^{ikj\Delta x}$ into the expansions and equate powers of $\xi = k\Delta x$. To do this we use the Fourier symbols

$$\partial_x e^{ikx} = ik\ e^{ikx}, \tag{9}$$

$$\partial_x^2 e^{ikx} = -k^2\ e^{ikx}, \tag{10}$$

$$D_{0x}e^{ikj\Delta x} = \frac{i\sin(\xi)}{\Delta x}\ e^{ikj\Delta x}, \tag{11}$$

$$D_{+x}D_{-x}e^{ikj\Delta x} = -\frac{\sin^2(\xi/2)}{(\Delta x/2)^2}\ e^{ikj\Delta x}, \tag{12}$$

The maple code `cgWave/doc/maple/highOrderDiff.maple` was used to generate the coefficients given in Table 1. Here is some sample code to find the coefficients in the expansion for $\partial_x^2 u$,

```
S := series( 4*(sin(z/2))^2, z=0, 24):
S1 := series( sin(z), z=0, 24):
# u.xx = D+D-( I - b1*dx^2*D+D- + b2*(dx^2*D+D-)^2 -b3*(dx^2*D+D-)^2 + ...)
printf("---------------- uxx ----------------\n");
f2 := 1 - (S/z^2)*(1 + b1*S + b2*S^2 + b3*S^3+  b4*S^4 + b5*S^5  ):
f2s := series( f2,z=0,24 ):

b1:=solve(coeff(f2s,z^2 )=0,b1);
b2:=solve(coeff(f2s,z^4 )=0,b2);
b3:=solve(coeff(f2s,z^6 )=0,b3);
b4:=solve(coeff(f2s,z^8 )=0,b4);
b5:=solve(coeff(f2s,z^10)=0,b5);
```

| | $\beta_0$ | $\beta_1$ | $\beta_2$ | $\beta_3$ | $\beta_4$ |
|---|---|---|---|---|---|
| $\partial_x u$ | 1 | $\frac{1}{6}$ | $\frac{1}{30}$ | $\frac{1}{140}$ | $\frac{1}{630}$ |
| $\partial_x^2 u$ | 1 | $\frac{1}{12}$ | $\frac{1}{90}$ | $\frac{1}{560}$ | $\frac{1}{3150}$ |
| $\partial_x^3 u$ | 1 | $\frac{1}{4}$ | $\frac{7}{120}$ | $\frac{41}{3024}$ | $\frac{479}{151200}$ |
| $\partial_x^4 u$ | 1 | $\frac{1}{6}$ | $\frac{7}{240}$ | $\frac{41}{7560}$ | $\frac{479}{453600}$ |
| $\partial_x^5 u$ | 1 | $\frac{1}{3}$ | $\frac{13}{144}$ | $\frac{139}{6048}$ | $\frac{37}{6480}$ |
| $\partial_x^6 u$ | 1 | $\frac{1}{4}$ | $\frac{13}{240}$ | $\frac{139}{12096}$ | $\frac{37}{15120}$ |

Table 1: Coefficients in central difference approximations in the $D_{+x}D_{-x}$ expansions for different derivatives. Coefficients were generated by cgWave/doc/maple/highOrderDiff.maple.

### *4.1. Efficient evaluation of the modified equation schemes*

Check me...

**Direct approach (DA).** First consider evaluation of the approximations needed for a (p)th-order accurate modified equation scheme in one-dimension, (the case $p = 6$ is given here)

$$\partial_x^2 u \approx D_{+x}D_{-x}u_{\mathbf{j}} - \frac{\Delta x^2}{12}(D_{+x}D_{-x})^2 u_{\mathbf{j}} + \frac{\Delta x^4}{90}(D_{+x}D_{-x})^3 u_{\mathbf{j}}, \tag{13}$$

$$\partial_x^4 u \approx (D_{+x}D_{-x})^2 u_{\mathbf{j}} - \frac{\Delta x^2}{6}(D_{+x}D_{-x})^3 u_{\mathbf{j}}, \tag{14}$$

$$\partial_x^6 u \approx (D_{+x}D_{-x})^3 u_{\mathbf{j}} \tag{15}$$

The stencil width for each of these approximations is $p+1$ and let us say it takes about $p+1$ operations to evaluate the approximation (if we expand out each approximation into a stencil). Do we count just multiplications or additions too? The cost per point (CPP) to evaluate the $(p/2)$ stencils is then proportional to $(p+1)(p/2)$ These approximations are then combined with about $p/2$ operations to give the update for a total CPP of $(p+2)(p/2)$.

$$\text{CPP}[DA, 1D] = (p+2)(p/2). \tag{16}$$

For comparison, the cost-per-point of an FFT for $N$ points is about

$$\text{CPP}[FFT, 1D] = 3\log_2 N. \tag{17}$$

**Hierarchical approach (HA).** Suppose instead we compute a hierarchy of approximations

$$v_{\mathbf{j}}^{(1)} = D_{+x}D_{-x}u_{\mathbf{j}}, \tag{18}$$

$$v_{\mathbf{j}}^{(2)} = D_{+x}D_{-x}v_{\mathbf{j}}^{(1)} \quad (= (D_{+x}D_{-x})^2 u_{\mathbf{j}}), \tag{19}$$

$$v_{\mathbf{j}}^{(3)} = D_{+x}D_{-x}v_{\mathbf{j}}^{(2)}, \quad (= (D_{+x}D_{-x})^3 u_{\mathbf{j}}), \tag{20}$$

with cost per point (CPP) of about $3(p/2)$. These approximations are then combined with about $p/2$ operations to give the update for a total CPP of

$$\text{CPP}[HA, 1D] = 4\,(p/2). \tag{21}$$

Comparing the DA cost of $(p+2)(p/2)$ to the HA cost of $4(p/2)$ shows the HA approach has a speedup of about $(p+2)/4$ in general. For example, for $p=6$ (or $p=8$) the speedup is about $8/4=2$ ($10/4=2.5$).

**Stencil Approach (SA).** In the stencil approach we combine all terms in the time-stepping update. In one-dimension this would be

$$u_{\mathbf{i}}^{n+1} = 2u_{\mathbf{i}}^{n} + u_{\mathbf{i}}^{n-1} + \sum_{m_1=-p/2}^{p/2} c_{m_1}^{(x)} u_{\mathbf{i}+m_1\mathbf{e}_1} \tag{22}$$

and the CPP is

$$\mathrm{CPP}[SA, 1D] = p+1; \tag{23}$$

this may be the fastest, although maybe not that much faster than the HA scheme. In $d$-dimensions, however, the cost is $(p+1)^d$ and then it is not so clear what method is fastest. Further more, for curvilinear grids the coefficients depend on $\mathbf{i}$ and storing these coefficients would require a lot of memory, $M=(p+1)^d$ doubles per grid point (e.g. for $p=6$, $d=3$, $M=7^3=343$ ). This is perhaps worth it in some cases.

**Direct approach (DA) for a mixed derivative.** Now consider evaluating a pth-order accurate approximation to $\partial_x^2\partial_y^2 u$

$$\partial_x^2\partial_y^2 u \approx D_{xx,p} D_{yy,p} u_{\mathbf{j}}, \tag{24}$$

$$D_{xx,p} = D_{+x}D_{-x}u_{\mathbf{j}} - \frac{\Delta x^2}{12}(D_{+x}D_{-x})^2 u_{\mathbf{j}} + \ldots, \tag{25}$$

$$D_{yy,p} = D_{+y}D_{-y}u_{\mathbf{j}} - \frac{\Delta y^2}{12}(D_{+y}D_{-y})^2 u_{\mathbf{j}} + \ldots, \tag{26}$$

The stencil is of size $(p+1)^2$ and let us say it takes about $(p+1)^2$ operations to evaluate the approximation. The DA cost for evaluating $\partial_x^2\partial_y^2\partial_z^2 u$ (arising in $\Delta^3 u$) is $(p+1)^3$

$$\mathrm{CPP}[\mathrm{DA}, \mathrm{XY}, 2\mathrm{D}] = (p+1)^2, \tag{27}$$

$$\mathrm{CPP}[\mathrm{DA}, \mathrm{XYZ}, 2\mathrm{D}] = (p+1)^3. \tag{28}$$

**Tensor product approach (TP).** In the tensor product approach we evaluate in 2 stages,

$$v_{\mathbf{j}}^{(1)} = D_{xx,p} u_{\mathbf{j}}, \tag{29}$$

$$v_{\mathbf{j}}^{(2)} = D_{yy,p} v_{\mathbf{j}}^{(1)} \quad (= (D_{+x}D_{-x})(D_{+y}D_{-y})u_{\mathbf{j}}), \tag{30}$$

The CPP for stage one is $p+1$ and the CPP for stage 2 is also $p+1$ for a total CPP of $2(p+1)$. The TP cost for evaluating $\partial_x^2\partial_y^2\partial_z^2 u$ is just $3(p+1)$,

$$\mathrm{CPP}[\mathrm{TP}, \mathrm{XY}, 2\mathrm{D}] = 2(p+1), \tag{31}$$

$$\mathrm{CPP}[\mathrm{TP}, \mathrm{XYZ}, 2\mathrm{D}] = 3(p+1). \tag{32}$$

Comparing the DA cost of $(p+1)^2$ to the HA cost of $2(p+1)$ shows the TP approach has a speedup of about $(p+1)/2$ in general. For example, for $p=6$ (or $p=8$) the speedup about $7/2=3.5$ (or $9/2=4.5$). The TP cost for evaluating $\partial_x^2\partial_y^2\partial_z^2 u$ is just $3(p+1)$ and here the speedup for the TP scheme is $(p+1)^2/3$.

### *4.2. Hierarchical tensor product modified equation schemes on Cartesian grids.*

The HA and TP approaches can be combined to evaluate the ME approximation in multiple space dimensions. Algorithm 1 gives the sixth-order accurate HA-TP algorithm for the case of a 2D Cartesian grid.

**Algorithm 1** Hierarchical tensor product modified equation scheme - 2D Cartesian Order 6

**for** i **do**
  $d_{20,\mathbf{i}} = D_{+x}D_{-x}u_{\mathbf{i}}$
  $d_{02,\mathbf{i}} = D_{+y}D_{-y}u_{\mathbf{i}}$
**end for**
**for** i **do**
  $d_{40,\mathbf{i}} = D_{+x}D_{-x}d_{20,\mathbf{i}}$
  $d_{22,\mathbf{i}} = D_{+x}D_{-x}d_{02,\mathbf{i}}$
  $d_{04,\mathbf{i}} = D_{+y}D_{-y}d_{02,\mathbf{i}}$
**end for**
**for** i **do**
  // These next variables do not need to be stored:
  $d_{60,\mathbf{i}} = D_{+x}D_{-x}d_{40,\mathbf{i}}$
  $d_{42,\mathbf{i}} = D_{+x}D_{-x}d_{22,\mathbf{i}}$
  $d_{24,\mathbf{i}} = D_{+x}D_{-x}d_{04,\mathbf{i}}$
  $d_{06,\mathbf{i}} = D_{+y}D_{-y}d_{04,\mathbf{i}}$

  $u_{xx,\mathbf{i}} = d_{20,\mathbf{i}} - \frac{\Delta x^2}{12}d_{40,\mathbf{i}} + \frac{\Delta x^4}{90}d_{60,\mathbf{i}}$ ▷ 6th order $\partial_x^2 u$
  $u_{yy,\mathbf{i}} = d_{02,\mathbf{i}} - \frac{\Delta y^2}{12}d_{04,\mathbf{i}} + \frac{\Delta y^4}{90}d_{06,\mathbf{i}}$
  $u_{xxxx,\mathbf{i}} = d_{40,\mathbf{i}} - \frac{\Delta x^2}{6}d_{60,\mathbf{i}}$ ▷ 4th order $\partial_x^4 u$
  $u_{xxyy,\mathbf{i}} = d_{22,\mathbf{i}} - \frac{\Delta x^2}{12}d_{42,\mathbf{i}} - \frac{\Delta y^2}{12}d_{24,\mathbf{i}}$
  $u_{yyyy,\mathbf{i}} = d_{04,\mathbf{i}} - \frac{\Delta y^2}{6}d_{06,\mathbf{i}}$
  $u_{xxxxxx,\mathbf{i}} = d_{60,\mathbf{i}}$ ▷ 2nd order $\partial_x^6 u$
  $u_{xxxxyy,\mathbf{i}} = d_{42,\mathbf{i}}$
  $u_{xxyyyy,\mathbf{i}} = d_{24,\mathbf{i}}$
  $u_{yyyyyy,\mathbf{i}} = d_{06,\mathbf{i}}$

  $u_{\mathbf{i}}^{n+1} = 2u_{\mathbf{i}}^n + u_{\mathbf{i}}^{n-1} + (c^2\Delta t^2)\left(u_{xx,\mathbf{i}} + u_{yy,\mathbf{i}}\right)$
  $\quad + \frac{c^4\Delta t^4}{12}\left(u_{xxxx,\mathbf{i}} + 2u_{xxyy,\mathbf{i}} + u_{yyyy,\mathbf{i}}\right)$
  $\quad + \frac{c^6\Delta t^6}{360}\left(u_{xxxxxx,\mathbf{i}} + 3u_{xxxxyy,\mathbf{i}} + 3u_{xxyyyy,\mathbf{i}} + u_{yyyyyy,\mathbf{i}}\right)$
**end for**

**Note:** Alternatively we could store the $(2p+1)^2$ stencil coefficients in the update (on a Cartesian grid there the stencil is not full and we only need include the non-zero values. This approach does not take advantage of the tensor product terms.

**Algorithm 2** Modified equation scheme with stencil coefficients

**for** i **do**
  $u_{\mathbf{i}}^{n+1} = 2u_{\mathbf{i}}^n + u_{\mathbf{i}}^{n-1} + \sum_{m_1=-p/2}^{p/2}\sum_{m_2=-p/2}^{p/2} c_{m_1,m_2}u_{\mathbf{i}+m_1\mathbf{e}_1+m_2\mathbf{e}_2}$
**end for**

**Algorithm 3** Modified equation scheme with tensor product cross terms

**for** $i_2$ **do**
  **for** $i_1$ **do**
    $S_x = \sum_{m_1=-p/2}^{p/2} c_{m_1}^{(x)} u_{\mathbf{i}+m_1\mathbf{e}_1}$ ▷ Stencil in x
    $S_y = \sum_{m_2=-p/2}^{p/2} c_{m_2}^{(y)} u_{\mathbf{i}+m_2\mathbf{e}_2}$ ▷ Stencil in y
    **for** $m_2 = -p : p$ **do** ▷ Could store values computed for previous $i_2$, then just need $m_2 = p$.
      $j_1 = i_1$, $j_2 = i_2 + m_2$
      $v_{\mathbf{j}} = D_{+x} D_{-x} u_{\mathbf{j}}$
    **end for**
    $S_{xy} = \frac{c^4 \Delta t^4}{12} \left[ D_{+y} D_{-y} v_{\mathbf{i}} - \frac{\Delta x^2}{12} (D_{+y} D_{-y})^2 v_{\mathbf{i}} + \ldots \right]$ ▷ Sample cross term
    $u_{\mathbf{i}}^{n+1} = 2u_{\mathbf{i}}^n + u_{\mathbf{i}}^{n-1} + S_x + S_y + S_{xy}$
  **end for**
**end for**

### *4.3. Hierarchical tensor product modified equation schemes on Curvilinear grids.*

On a curvilinear grid we transform the equations to the parameter space coordinates $\mathbf{r} = (r_1, r_2) = (r, s)$. We assume that we know the mapping metrics,

$$\left[\frac{\partial \mathbf{r}}{\partial \mathbf{x}}\right]_{m,n} = \mathbf{rx}_{m,n} = \frac{\partial r_m}{\partial x_n} \tag{33}$$

but that derivatives of the metrics must be computed.

Using the chain rule we have, for example,

$$\partial_x u = r_x \partial_r u + s_x \partial_s u, \tag{34}$$

$$\partial_x^2 u = (r_x)^2 \partial_r^2 u + r_x s_x \partial_r \partial_s u + (s_x)^2 \partial_s^2 u + (r_x)_x u_r + (s_x)_x u_s. \tag{35}$$

The maple file `cgWave/maple/generateDerivativesByChainRule.maple` can be used to generate expressions for the chain-rule derivatives of $u$ such ([DuDx[nx,ny,nz,d=2:3]])

```
DuDx[2,0,0,2]:=rx^2*urr+2*rx*sx*urs+sx^2*uss+rxx*ur+sxx*us:
DuDx[1,1,0,2]:=rx*ry*urr+(sy*rx+ry*sx)*urs+sx*sy*uss+rxy*ur+sxy*us:
DuDx[0,2,0,2]:=ry^2*urr+2*ry*sy*urs+sy^2*uss+ryy*ur+syy*us:
DuDx[3,0,0,2]:=rx^3*urrr+3*rx^2*sx*urrs+3*rx*sx^2*urss+sx^3*usss+3*rx*rxx*urr
             +(3*rx*sxx+3*rxx*sx)*urs+3*sx*sxx*uss+rxxx*ur+sxxx*us:
DuDx[2,1,0,2]:=rx^2*ry*urrr+(rx^2*sy+2*rx*ry*sx)*urrs+(2*sy*rx*sx+ry*sx^2)*urss+sx^2*sy*usss
              +(2*rxy*rx+rxx*ry)*urr+(2*rx*sxy+rxx*sy+2*rxy*sx+ry*sxx)*urs
              +(2*sx*sxy+sxx*sy)*uss+rxxy*ur+sxxy*us:
DuDx[2,2,0,2]:=rx^2*ry^2*urrrr+(2*ry*sy*rx^2+2*sx*ry^2*rx)*urrrs+(sy^2*rx^2+4*ry*sx*sy*rx+sx^2*ry^2)*urrss
        +(2*sx*sy^2*rx+2*sy*sx^2*ry)*ursss+sx^2*sy^2*ussss+(ryy*rx^2+4*rxy*ry*rx+rxx*ry^2)*urrr
        +(rx^2*syy+(4*rxy*sy+4*sxy*ry+2*ryy*sx)*rx+ry^2*sxx+(2*rxx*sy+4*rxy*sx)*ry)*urrs
        +((2*sx*syy+4*sxy*sy)*rx+(4*sx*sxy+2*sxx*sy)*ry+ryy*sx^2+4*rxy*sy*sx+rxx*sy^2)*urss
        +(sx^2*syy+4*sxy*sy*sx+sxx*sy^2)*usss+(2*rxyy*rx+rxx*ryy+2*rxxy*ry+2*rxy^2)*urr
        +(2*rx*sxyy+rxx*syy+2*rxxy*sy+4*rxy*sxy+2*rxyy*sx+2*ry*sxxy+ryy*sxx)*urs
        +(2*sx*sxyy+sxx*syy+2*sxxy*sy+2*sxy^2)*uss+rxxyy*ur+sxxyy*us:
```

The maple file `cgWave/maple/chainRuleCoefficients.maple` reads these chain rule formulae and generates Fortran code such as the following that can be used to compute spatial derivatives given derivatives of the metrics

```
! ------ Coefficients in expansion for uxxx ------
cuxxx100 = rxxx
cuxxx200 = 3.*rx*rxx
cuxxx300 = rx**3
cuxxx010 = sxxx
cuxxx110 = 3.*rx*sxx+3.*rxx*sx
cuxxx210 = 3.*rx**2.*sx
cuxxx020 = 3.*sx*sxx
cuxxx120 = 3.*rx*sx**2
```

```
cuxxx030 = sx**3

! uxxx = cuxxx100*ur+cuxxx200*urr+cuxxx300*urrr+cuxxx010*us+cuxxx110*urs
!       +cuxxx210*urrs+cuxxx020*uss+cuxxx120*urss+cuxxx030*usss

! ------ Coefficients in expansion for uxxy ------
cuxxy100 = rxxy
cuxxy200 = 2.*rx*rxy+rxx*ry
cuxxy300 = rx**2.*ry
cuxxy010 = sxxy
cuxxy110 = 2.*rx*sxy+rxx*sy+2.*rxy*sx+ry*sxx
cuxxy210 = rx**2.*sy+2.*rx*ry*sx
cuxxy020 = 2.*sx*sxy+sxx*sy
cuxxy120 = 2.*rx*sx*sy+ry*sx**2
cuxxy030 = sx**2.*sy

! uxxy = cuxxy100*ur+cuxxy200*urr+cuxxy300*urrr+cuxxy010*us+cuxxy110*urs
!        +cuxxy210*urrs+cuxxy020*uss+cuxxy120*urss+cuxxy030*usss
```

The Hierarchical Tensor-Product scheme for a curvilinear grid is similar to that for a Cartesian Grid.

1. First compute powers of $D_{+r}D_{-r}$ and $D_{+s}D_{-s}$ applied to both $u_{\mathbf{i}}$ but also to the metrics, $\mathbf{rx}_{\mathbf{i}}$,

$$d_{mn,\mathbf{i}} = (D_{+r}D_{-r})^{m/2}(D_{+s}D_{-s})^{n/2}u_{\mathbf{i}}, \tag{36}$$

$$\mathbf{rx}_{mn,\mathbf{i}} = (D_{+r}D_{-r})^{m/2}(D_{+s}D_{-s})^{n/2}\mathbf{rx}_{\mathbf{i}}, \tag{37}$$

2. Compute appropriate high-order accurate parametric derivatives of both $u$ and the metrics,

$$\partial_r^2 u = d_{2,0,\mathbf{i}} - \frac{\Delta r^2}{12} d_{40,\mathbf{i}} + \ldots, \tag{38}$$

$$\partial_r^2 \mathbf{rx} = \mathbf{rx}_{2,0,\mathbf{i}} - \frac{\Delta r^2}{12} \mathbf{rx}_{40,\mathbf{i}} + \ldots, \tag{39}$$

3. Evaluate the spatial derivatives of $\mathbf{rx}$ using the chain-rule formulae,

$$\partial_x^2 \mathbf{rx} = \text{cuxx20}\ \partial_r^2 \mathbf{rx} + \text{cuxx11}\ \partial_r\partial_s \mathbf{rx} + ... \tag{40}$$

4. Evaluate the spatial derivatives of $u$ using the chain-rule formulae,

$$\partial_x^2 u = \text{cuxx20}\ \partial_r^2 u + \text{cuxx11}\ \partial_r\partial_s u + ... \tag{41}$$

**Storing the coefficients of the ME operator.** On a curvilinear grid the Laplace operator can be written as

$$\Delta u = c_{20}\partial_r^2 u + c_{11}\partial_r\partial_s u + c_{02}\partial_s^2 u + c_{10}\partial_r u + c_{01}\partial_s u \tag{42}$$

Instead of storing the full stencil we could instead store the $2+3=5$ (in 2D) coefficients $c_{mn}$. To evaluate $\Delta_h u_{\mathbf{j}}$ we would need to compute approximations to $\partial_r^2 u$, etc. but we would not need to compute all the derivatives of the metrics.

For the ME scheme we also need to evaluate powers of the Laplacian. $\Delta^2 u$ has the expansion

$$\Delta^2 u = c_{40}\partial_r^4 u + c_{31}\partial_r^3\partial_s u + c_{22}\partial_r^2\partial_s^2 u + \ldots \tag{43}$$

This operator has $2+3+4+5=14$ coefficients in $2D$. The full operator for fourth-order accurate ME scheme could be evaluated as

$$\mathcal{L}_{4h}u = (c\Delta t)^2\Delta u + \frac{(c\Delta t)^4}{12}\Delta^2 u \ = d_{40}\partial_r^4 u + d_{31}\partial_r^3\partial_s u + d_{22}\partial_r^2\partial_s^2 u + \ldots \tag{44}$$

This has 14 coefficients $d_{mn}$ (in 2D). Compare this to storing 25 stencil coefficients in 2D for the fourth-order accurate scheme.

For sixth-order accuracy we need $\Delta^3 u$ which has the expansion

$$\Delta^3 u = c_{60}\partial_r^6 u + c_{51}\partial_r^5\partial_s u + c_{42}\partial_r^4\partial_s^2 u + \dots \tag{45}$$

which contains $2+3+4+5+6+7 = 27$ coefficients. Compare storing 27 coefficients for $\mathcal{L}_{6h}$ to 49 stencil coefficients in 2D for the sixth-order accurate scheme.

At pth order there are $(p+1)(p+2)/2-1$ coefficients in $\mathcal{L}_{ph}$ compared to $(p+1)^2$ stencil coefficients. Thus at eighth order the comparison is 44 for $\mathcal{L}_{8h}$ compared to to 81 stencil coefficients.

*4.4. Factored Modified Equation (FAME) Schemes.*

The Modified Equation Hierarchical Laplacian schemes follow along the lines the algorithm in the LCBC paper.

Here are some preliminary results for a square and nonSquare with $1024^2$ grid points.

Table 3 shows results from the runs/performance script.

| | | | CPU ns/step/pt | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Grid | ord | scheme | pts | total | solve | interior | diss | BC | interp |
| square1024 | 2 | MEHA | 1.1M | 9.8 | 8.6 | 5.8 | 0.0 | 1.8 | 0.0 |
| square1024 | 4 | MEHA | 1.1M | 17.4 | 16.3 | 13.4 | 0.0 | 1.9 | 0.1 |
| square1024 | 6 | MEHA | 1.1M | 37.7 | 36.4 | 32.8 | 0.0 | 2.7 | 0.1 |
| nonSquare1024 | 2 | MEHA | 1.1M | 14.0 | 12.2 | 8.6 | 0.0 | 2.3 | 0.1 |
| nonSquare1024 | 4 | MEHA | 1.1M | 32.1 | 29.8 | 25.2 | 0.0 | 3.3 | 0.1 |
| nonSquare1024 | 6 | MEHA | 1.1M | 81.5 | 77.9 | 61.3 | 0.0 | 15.4 | 0.1 |
| box16 | 2 | MEHA | 4.5M | 31.5 | 23.9 | 6.5 | 0.0 | 11.2 | 0.0 |
| box16 | 4 | MEHA | 4.5M | 61.0 | 52.6 | 31.9 | 0.0 | 14.6 | 0.0 |

Table 2: FAME-V2, compile -O. CPU performance data. ME-HA scheme. Interior = time for updating interior points (no BC's, no upwinding, no interp.). Time in nano-seconds per-step per-grid-point (1 ns = $10^{-9}s$).

| | | | CPU ns/step/pt | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Grid | ord | scheme | pts | total | solve | interior | diss | BC | interp |
| square1024 | 2 | FAMEST2 | 1.1M | 5.8 | 4.6 | 1.9 | 0.0 | 1.8 | 0.0 |
| square1024 | 4 | FAMEST2 | 1.1M | 12.7 | 11.6 | 8.8 | 0.0 | 1.8 | 0.1 |
| square1024 | 6 | FAMEST2 | 1.1M | 31.1 | 29.9 | 26.3 | 0.0 | 2.6 | 0.1 |
| nonSquare1024 | 2 | FAMEST2 | 1.1M | 9.9 | 8.1 | 4.5 | 0.0 | 2.3 | 0.1 |
| nonSquare1024 | 4 | FAMEST2 | 1.1M | 22.5 | 20.2 | 15.7 | 0.0 | 3.2 | 0.1 |
| nonSquare1024 | 6 | FAMEST2 | 1.1M | 63.2 | 59.7 | 43.0 | 0.0 | 15.4 | 0.1 |

Table 3: FAME-STratgey 2, compile -O3. CPU performance data. ME-HA scheme. Interior = time for updating interior points (no BC's, no upwinding, no interp.). Time in nano-seconds per-step per-grid-point (1 ns = $10^{-9}s$).

| | | | CPU ns/step/pt | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Grid | ord | scheme | pts | total | solve | interior | diss | BC | interp |
| square1024 | 2 | FAMEST1 | 1.1M | 5.9 | 4.7 | 1.9 | 0.0 | 1.8 | 0.1 |
| square1024 | 4 | FAMEST1 | 1.1M | 9.7 | 8.7 | 5.9 | 0.0 | 1.7 | 0.1 |
| square1024 | 6 | FAMEST1 | 1.1M | 20.5 | 19.3 | 15.8 | 0.0 | 2.5 | 0.1 |
| box16 | 2 | FAMEST1 | 4.5M | 26.8 | 18.8 | 2.1 | 0.0 | 10.5 | 0.0 |
| box16 | 4 | FAMEST1 | 4.5M | 49.5 | 41.1 | 21.3 | 0.0 | 13.6 | 0.0 |

Table 4: FAME-STratgey 1, compiled -O3. CPU performance data. Time in nano-seconds per-step per-grid-point (1 ns = $10^{-9}s$).

Figure 1 displays from results from Table 3. The CPU cycles per-step per-point (which estimates FLOPS per-step per-point) is computed as

$$\text{cycles per-step per-point} = \text{seconds per-step per-point} \times \text{cycles-per-second} \tag{46}$$

and used 2.7 GHz for cycles-per-second for CG6.

Notes for FAME-V2, -O, CG6 CPU: 2.7GHz = $2.7 \times 10^9 Hz$, clock cycle = $.37 \times 10^{-9}$ = .37 nano-seconds, (1 nano second = $10^{-9}$ s)

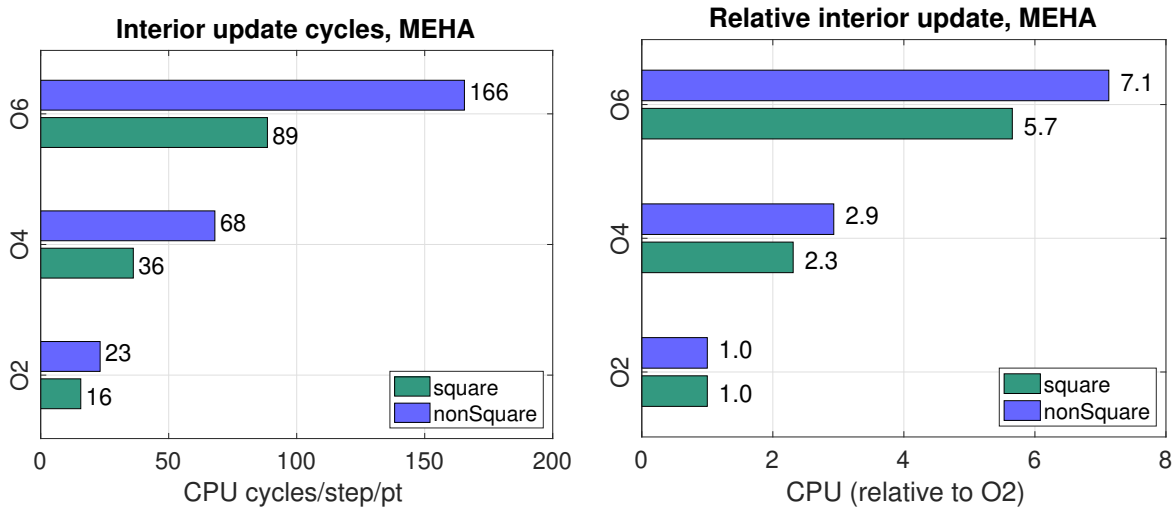


Figure 1: Left: CPU cycles per-step per-point (estimates FLOPS per-step per-point). Right: CPU time relative to order 2 scheme. Data from Table 3.

FD22r (Cartesian) has about 8 adds and 7 mults : 15 FLOPS (16 measured)

FD44r (Cartesian) has about 20 adds and 18 mults : 38 FLOPS (36 measured)

FD22c (curvilinear) has about 16 adds and 10 mults : 26 FLOPS (23 measured)

FD44c (curvilinear) has about 47 adds and 30 mults : 77 FLOPS (68 measured)

FD66c (curvilinear) has about 111 adds and 56 mults : 167 FLOPS (166 measured)

Test fftw - real 2D transform:

```
bench -s r1024x1024
Problem: r1024x1024, setup: 275.59 ms, time: 10.19 ms, ‘‘mflops’’: 5145.1
Time for 2 FFTs = 2*10.2*10^6/(1024^2)  = 19 ns/pt ->  19/.37 = 51 cycles/pt     log2(N)^2 = 100

Problem: r1025x1025, setup: 554.93 ms, time: 27.12 ms, ‘‘mflops’’: 1937.1
Problem: r1025x1027, setup: 942.77 ms, time: 38.50 ms, ‘‘mflops’’: 1367.4
Problem: r1021x1027, setup: 1.34 s, time: 52.31 ms, ‘‘mflops’’: 1002.3
Problem: r1223x1223, setup: 2.38 s, time: 128.60 ms, ‘‘mflops’’: 596.42


bench -s r2048x2048
Problem: r2048x2048, setup: 534.94 ms, time: 59.32 ms, ‘‘mflops’’: 3888.6
Time for 2 FFTs = 2*59.3*10^6/(2048^2) = 28 ns/pt -> 77 cycles/pt
```

Test fftw - real 3D transforms

```
bench -s r256x256x256
Problem: r256x256x256, setup: 2.75 s, time: 251.49 ms, ‘‘mflops’’: 4002.7
Time for 2 FFTs = 2*251.49*10^6/(256^3) = 30 ns/pt

 bench -s r512x512x512
Problem: r512x512x512, setup: 10.96 s, time: 2.61 s, ‘‘mflops’’: 3471
Time for 2 FFTs = 2*2610*10^6/(512^3) = 39 ns/pt
```

*4.5. FAME versus STENCIL.*

Here are preliminary results comparing FAME to the STENCIL approach

| | | | CPU ns/step/pt | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Grid | ord | scheme | pts | total | solve | interior | diss | BC | interp |
| square1024 | 2 | stencil | 1.1M | 3.2 | 2.9 | 1.4 | 0.0 | 1.2 | 0.0 |
| square1024 | 4 | stencil | 1.1M | 4.6 | 4.3 | 2.6 | 0.0 | 1.3 | 0.0 |
| square1024 | 6 | stencil | 1.1M | 10.5 | 10.2 | 8.1 | 0.0 | 1.8 | 0.0 |
| square1024 | 8 | stencil | 1.1M | 23.1 | 18.5 | 15.0 | 0.0 | 3.2 | 0.0 |
| square1024 | 2 | FAME | 1.1M | 3.7 | 3.1 | 1.3 | 0.0 | 1.2 | 0.0 |
| square1024 | 4 | FAME | 1.1M | 6.4 | 5.7 | 4.2 | 0.0 | 0.9 | 0.0 |
| square1024 | 6 | FAME | 1.1M | 13.9 | 13.2 | 10.9 | 0.0 | 1.7 | 0.0 |
| square1024 | 8 | FAME | 1.1M | 56.6 | 47.5 | 43.7 | 0.0 | 3.1 | 0.1 |
| nonSquare1024 | 2 | stencil | 1.1M | 10.3 | 9.6 | 7.5 | 0.0 | 1.6 | 0.0 |
| nonSquare1024 | 4 | stencil | 1.1M | 25.1 | 24.0 | 21.5 | 0.0 | 2.1 | 0.0 |
| nonSquare1024 | 6 | stencil | 1.1M | 61.1 | 59.5 | 51.1 | 0.0 | 7.9 | 0.0 |
| nonSquare1024 | 8 | stencil | 1.1M | 138.8 | 133.2 | 129.3 | 0.0 | 3.4 | 0.0 |
| nonSquare1024 | 2 | FAME | 1.1M | 6.8 | 6.2 | 4.1 | 0.0 | 1.6 | 0.0 |
| nonSquare1024 | 4 | FAME | 1.1M | 25.8 | 25.0 | 18.5 | 0.0 | 6.1 | 0.0 |
| nonSquare1024 | 6 | FAME | 1.1M | 44.0 | 42.8 | 33.7 | 0.0 | 8.7 | 0.0 |
| nonSquare1024 | 8 | FAME | 1.1M | 151.5 | 146.8 | 142.7 | 0.0 | 3.7 | 0.1 |
| box16 | 2 | stencil | 4.2M | 18.4 | 13.9 | 2.4 | 0.0 | 7.9 | 0.0 |
| box16 | 4 | stencil | 4.2M | 24.3 | 19.1 | 7.5 | 0.0 | 8.1 | 0.0 |
| box16 | 6 | stencil | 4.2M | 50.6 | 44.0 | 31.3 | 0.0 | 9.0 | 0.0 |
| box16 | 8 | stencil | 4.2M | 92.6 | 88.3 | 74.5 | 0.0 | 9.4 | 0.0 |
| box16 | 2 | FAME | 4.2M | 18.4 | 13.9 | 2.4 | 0.0 | 7.9 | 0.0 |
| box16 | 4 | FAME | 4.2M | 34.8 | 29.6 | 18.4 | 0.0 | 7.6 | 0.0 |
| box16 | 6 | FAME | 4.2M | 75.3 | 68.7 | 56.1 | 0.0 | 8.9 | 0.0 |
| box16 | 8 | FAME | 4.2M | 184.9 | 180.5 | 166.9 | 0.0 | 9.3 | 0.0 |
| nonBox16 | 2 | stencil | 4.2M | 53.3 | 38.0 | 24.4 | 0.0 | 7.9 | 0.0 |
| nonBox16 | 4 | stencil | 4.2M | 203.2 | 163.8 | 150.0 | 0.0 | 8.0 | 0.0 |
| nonBox16 | 6 | stencil | 4.2M | 792.2 | 693.1 | 678.2 | 0.0 | 8.8 | 0.0 |
| nonBox16 | 2 | FAME | 4.2M | 36.0 | 23.6 | 10.0 | 0.0 | 7.9 | 0.0 |
| nonBox16 | 4 | FAME | 4.2M | 83.5 | 57.3 | 43.8 | 0.0 | 7.7 | 0.0 |
| nonBox16 | 6 | FAME | 4.2M | 290.6 | 231.0 | 216.0 | 0.0 | 8.8 | 0.0 |

Table 5: FAME verus stencil, compiled -O3. CPU performance data. Time in nano-seconds per-step per-grid-point (1 ns = $10^{-9}s$).

## 5. Upwind dissipation

Upwind dissipation for the wave equation in second-order form was first discussed in [3] and later extended to Maxwell's equations in [4]. The simplified version presented here is developed in a paper not yet completed [**?** ].

At the continuous level, the upwind dissipation adds a term proportional to a high spatial derivative of $\partial_t u$, and roughly takes the form in one-dimension as

$$\partial_t^2 u = c^2 \Delta u - \nu_p \frac{c}{\Delta x}(-\Delta x^2 \partial_x^2)^q \, \partial_t u, \tag{47}$$

for some coefficient $\nu_q$, and where $q = p/2 + 1$ is defined in terms of the order of accuracy of the scheme $p$. To avoid a time-step restriction, upwind dissipation is added using a predictor-corrector scheme (UWPC). Let us describe the approach in one space dimension. The predictor consist of the usual modified equation update to determine the predicted value $u_j^p \approx u_j^{n+1}$,

$$u_j^p = 2u_j^2 - u_j^{n-1} + \Delta t^2 \Big( c^2 D_{h,xx} u_j^n + \ldots), \tag{48}$$

$$\text{applyBoundaryConditions}(u^p). \tag{49}$$

The dissipation is added in a corrector step where $\partial_t u^n$ is approximated with $D_{0t}$,

$$u^{n+1} = u^p - \nu_p \, \lambda \, (-\Delta_+ \Delta_-)^q \Big( \frac{u^p - u^{n-1}}{2} \Big), \tag{50}$$

$$\text{applyBoundaryConditions}(u^{n+1}), \tag{51}$$

where $\nu_q$ is the coefficient of the upwind dissipation and $\lambda$ is the CFL parameter,

$$\lambda \stackrel{\text{def}}{=} \frac{c\Delta t}{\Delta x}. \tag{52}$$

The stability condition turns out to be [5] **check me**

$$\lambda < 1, \tag{53}$$

$$\zeta < 2, \tag{54}$$

$$\zeta \stackrel{\text{def}}{=} \nu_p \, \lambda \, (4 \sin^2(\xi/2))^q \tag{55}$$

which implies we need the usual CFL condition, $\lambda < 1$ as well as the restriction on $\nu_p$

$$\nu_p < \frac{2}{\lambda \, 4^q} = \frac{1}{\lambda \, 2^{p+1}} \tag{56}$$

In two-dimensions **check me**

$$\zeta \stackrel{\text{def}}{=} \nu_p \left( (\lambda_x \, (4 \sin^2(\xi_x/2))^q + \lambda_y (4 \sin^2(\xi_y/2))^q \right) \tag{57}$$

and we need $\zeta < 2$ or

$$\nu_p \left( (\lambda_x \, (4 \sin^2(\xi_x/2))^q + \lambda_y (4 \sin^2(\xi_y/2))^q \right) < 2, \tag{58}$$

or

$$\nu_p < \frac{2}{\lambda_x 4^q + \lambda_y 4^q} = \frac{1}{2^{p+1}} \, \frac{1}{\lambda_x + \lambda_y} \tag{59}$$

In two-dimensions the CFL condition is

$$\lambda_x^2 + \lambda_y^2 < 1 \tag{60}$$

which implies $\lambda_x + \lambda_y < \sqrt{2}$. Whence

$$\nu_p < \frac{1}{\sqrt{2}} \frac{1}{2^{p+1}} \tag{61}$$

In three-dimensions

$$\nu_p < \frac{1}{2^{p+1}} \frac{1}{\lambda_x + \lambda_y + \lambda_z} \tag{62}$$

where

$$\lambda_x^2 + \lambda_y^2 + \lambda_z^2 < 1 \tag{63}$$

which implies $\lambda_x + \lambda_y + \lambda + z < \sqrt{3}$ and

$$\nu_p < \frac{1}{\sqrt{3}} \frac{1}{2^{p+1}} \tag{64}$$

**Summary:** In $d$-dimensions we require

$$\nu_p < \frac{1}{\sqrt{d}} \frac{1}{2^{p+1}}. \tag{65}$$

**Note.** The value of $\nu_2 = 1/8$ suggested in [5] seems to be too big in 2D or 3D, instead we seem to need

$$\nu_2 = \frac{1}{\sqrt{d}} \frac{1}{8} \tag{66}$$

in $d$-dimensions which is smaller than $1/8$ for $d = 2, 3$. This conclusion agrees with computations.

For fourth-order, $p = 2$, we require

$$\nu_4 < \frac{1}{\sqrt{d}} \frac{1}{32} \tag{67}$$

The suggested value for $\nu_4 = 5/288 = 1/(57.6)$ Now $32\sqrt{3} \approx 55.456$ and thus this suggested value of $\nu_4 = 5/288$ should work in 2D or 3D. This conclusion also agrees with computations.

**Note:** We could choose $\nu_p$ from the condition $\zeta < 2$ - choose a value slightly less than the largest value allowed by stability.

An alternative scheme which allows a bigger value for $\nu_p$ is to evaluate $D_{0t}U^n$ in a Gauss-Seidel fashion. The predictor sets a preliminary value for $u_j^{n+1}$,

$$u_j^{n+1} = 2u_j^2 - u_j^{n-1} + \Delta t^2 \Big( c^2 D_{h,xx} u_j^n + \ldots), \tag{68}$$

$$\text{applyBoundaryConditions}(u^{n+1}). \tag{69}$$

The corrector adds the upwind dissipation to $u_j^{n+1}$, always using the latest value in the right-hand-side,

$$u_j^{n+1} \leftarrow u_j^{n+1} - \nu_p \, \lambda \, (-\Delta_+ \Delta_-)^q \Big( \frac{u_j^{n+1} - u_j^{n-1}}{2} \Big), \tag{70}$$

$$\text{applyBoundaryConditions}(u^{n+1}), \tag{71}$$

This version is stable for $p = 2$ with $\nu_2 = 1/8$ (found in practice, need to do the analysis). This version has the advantage of not needed storage to hold $(u^p - u^{n-1})/2$.

**Note:** The actual upwind scheme implemented in CgWave (and CgMx) uses a slightly modified algorithm that avoids one application of the boundary conditions (applying the BCs and interface conditions in CgMx can sometimes be expensive) . Instead of adding the dissipation at the end of the step to $u^{n+1}$, we add it at the start of the step to $u^n$,

$$u_j^n \leftarrow u_j^n - \nu_p \lambda (-\Delta_+\Delta_-)^q \Big( \frac{u_j^n - u_j^{n-2}}{2} \Big), \tag{72}$$

and do not apply the boundary conditions (this works since formally one application of the dissipation adds a small $\mathcal{O}(h^{p+2})$ correction to $u_j^n$ so the BCs will still be satisfied to the expected order of accuracy). This is followed by the usual update

$$u_j^{n+1} = 2u_j^2 - u_j^{n-1} + \Delta t^2 \Big( c^2 D_{h,xx} u_j^n + \ldots), \tag{73}$$

$$\text{applyBoundaryConditions}(u^{n+1}). \tag{74}$$

## 6. Numerical results

Here are some numerical results.

### *6.1. Plane Wave*

Here are errors in computing an exact plane wave solution

$$u = \sin(2\pi(k_x y + k_y y + k_z z) - \omega t)$$

with $\omega/k = c$ and $k = |\mathbf{k}|$.

**Square - Plane Wave Order 2**

| grid | N | $u$ | r |
|---|---|---|---|
| square8 | 1 | 4.6e-2 | |
| square16 | 2 | 1.3e-2 | 3.66 |
| square32 | 4 | 3.0e-3 | 4.25 |
| square64 | 8 | 7.4e-4 | 4.01 |
| square128 | 16 | 1.9e-4 | 3.94 |
| rate | | 2.00 | |

Table 6: CgWave, planeWave, max norm, order=2, $t$ = .5, cfl=0.9, diss=0, kx=1, ky=0, kz=0, Sun Mar 1 11:12:19 2020

**Square -Plane Wave Order 4**

| grid | N | $u$ | r |
|---|---|---|---|
| square8 | 1 | 1.3e-2 | |
| square16 | 2 | 5.1e-4 | 26.26 |
| square32 | 4 | 2.9e-5 | 17.91 |
| square64 | 8 | 1.4e-6 | 20.35 |
| square128 | 16 | 7.3e-8 | 19.24 |
| square256 | 32 | 4.0e-9 | 18.42 |
| square512 | 64 | 2.3e-10 | 17.30 |
| square1024 | 128 | 1.4e-11 | 16.55 |
| rate | | 4.25 | |

Table 7: CgWave, planeWave, max norm, order=4, $t$ = .5, cfl=0.9, diss=0, kx=1, ky=0, kz=0, Sun Mar 1 11:15:18 2020

**CIC - circle in a channel - Plane Wave Order 2**

| grid | N | $u$ | r |
|---|---|---|---|
| cic2 | 1 | 1.1e-2 | |
| cic4 | 2 | 2.6e-3 | 4.18 |
| cic8 | 4 | 6.2e-4 | 4.20 |
| cic16 | 8 | 1.5e-4 | 4.11 |
| cic32 | 16 | 3.7e-5 | 4.05 |
| rate | | 2.05 | |

Table 8: CgWave, planeWave, max norm, order=2, $t$ = .5, cfl=0.9, diss=0, kx=1, ky=0, kz=0, Sun Mar 1 13:35:54 2020

**CIC - circle in a channel - Plane Wave Order 4**

| grid | N | $u$ | r |
|---|---|---|---|
| cic2 | 1 | 2.7e-4 | |
| cic4 | 2 | 1.3e-5 | 20.61 |
| cic8 | 4 | 5.9e-7 | 22.32 |
| cic16 | 8 | 3.0e-8 | 19.70 |
| cic32 | 16 | 1.7e-9 | 17.33 |
| rate | | 4.33 | |

Table 9: CgWave, planeWave, max norm, order=4, $t$ = .5, cfl=0.9, diss=0, kx=1, ky=0, kz=0, Sun Mar 1 13:36:19 2020

**Box - Plane Wave Order 2**

| grid | N | $u$ | r |
|---|---|---|---|
| box1 | 1 | 5.8e-2 | |
| box2 | 2 | 1.4e-2 | 4.26 |
| box4 | 4 | 2.8e-3 | 4.82 |
| box8 | 8 | 6.3e-4 | 4.49 |
| rate | | 2.18 | |

Table 10: CgWave, planeWave, max norm, order=2, ts=explicit, orderInTime=-1, dtMax0=10000000000, $t$ = .5, cfl=0.9, diss=0, -known=planeWave, kx=1, ky=1, kz=1, Sat Jul 17 06:23:09 2021

**Box - Plane Wave Order 4**

| grid | N | $u$ | r |
|---|---|---|---|
| box1 | 1 | 1.3e-2 | |
| box2 | 2 | 4.6e-4 | 28.38 |
| box4 | 4 | 1.6e-5 | 29.55 |
| box8 | 8 | 6.3e-7 | 24.45 |
| box16 | 16 | 2.8e-8 | 22.64 |
| rate | | 4.71 | |

Table 11: CgWave, planeWave, max norm, order=4, ts=explicit, orderInTime=-1, dtMax0=10000000000, $t$ = .5, cfl=0.9, diss=0, -known=planeWave, kx=1, ky=1, kz=1, Sat Jul 17 06:09:54 2021

*6.2. Time-periodic solution in a box (showing forcing)*

An exact solution to the forced wave equation in a rectangular box $[0,1]^d$ is given by

$$u_e(\mathbf{x},t) \stackrel{\text{def}}{=} \sin(k_x x)\,\sin(k_y y)\,[\sin(k_z z)]\,\cos(\omega t),$$

where the forcing function is

$$f(\mathbf{x},t) = \Big( -\omega^2 + c^2(k_x^2 + k_y^2)\Big) u_e(\mathbf{x},t)$$

**Square - Order 2**

| grid | N | $u$ | r |
|---|---|---|---|
| square8 | 1 | 5.6e-2 | |
| square16 | 2 | 2.1e-3 | 27.37 |
| square32 | 4 | 1.9e-4 | 10.79 |
| square64 | 8 | 1.0e-5 | 18.45 |
| square128 | 16 | 5.6e-7 | 18.26 |
| square256 | 32 | 4.0e-8 | 14.17 |
| square512 | 64 | 2.7e-9 | 14.96 |
| rate | | 4.02 | |

Table 12: CgWave, helmholtz, max norm, order=4, $t = .5$, cfl=0.9, diss=0, kx=2, ky=2, kz=0, Sun Mar 1 13:18:51 2020

**Square - Order 4**

| grid | N | $u$ | r |
|---|---|---|---|
| square8 | 1 | 5.6e-2 | |
| square16 | 2 | 2.1e-3 | 27.37 |
| square32 | 4 | 1.9e-4 | 10.79 |
| square64 | 8 | 1.0e-5 | 18.45 |
| square128 | 16 | 5.6e-7 | 18.26 |
| square256 | 32 | 4.0e-8 | 14.17 |
| square512 | 64 | 2.7e-9 | 14.96 |
| rate | | 4.02 | |

Table 13: CgWave, helmholtz, max norm, order=4, $t = .5$, cfl=0.9, diss=0, kx=2, ky=2, kz=0, Sun Mar 1 13:18:51 2020

**CIC - circle in a channel - Order 2**

| grid | N | $u$ | r |
|---|---|---|---|
| cic2 | 1 | 1.7e-2 | |
| cic4 | 2 | 2.4e-3 | 6.86 |
| cic8 | 4 | 5.8e-4 | 4.17 |
| cic16 | 8 | 1.4e-4 | 4.03 |
| cic32 | 16 | 3.6e-5 | 4.01 |
| rate | | 2.18 | |

Table 14: CgWave, helmholtz, max norm, order=2, $t = .5$, cfl=0.9, diss=0, kx=1, ky=1, kz=0, Sun Mar 1 13:42:02 2020

**CIC - circle in a channel - Order 4**

| grid | N | $u$ | r |
|---|---|---|---|
| cic2 | 1 | 3.3e-4 | |
| cic4 | 2 | 1.9e-5 | 17.74 |
| cic8 | 4 | 7.1e-7 | 26.27 |
| cic16 | 8 | 2.3e-8 | 30.26 |
| cic32 | 16 | 5.9e-10 | 39.80 |
| rate | | 4.78 | |

Table 15: CgWave, helmholtz, max norm, order=4, $t$ = .5, cfl=0.9, diss=0, kx=1, ky=1, kz=0, Sun Mar 1 13:41:12 2020

**Box - Order 2**

| grid | N | $u$ | r |
|---|---|---|---|
| box1 | 1 | 2.8e-3 | |
| box2 | 2 | 6.0e-4 | 4.76 |
| box4 | 4 | 1.4e-4 | 4.29 |
| box8 | 8 | 3.4e-5 | 4.06 |
| rate | | 2.12 | |

Table 16: CgWave, helmholtz, max norm, order=2, ts=explicit, orderInTime=-1, dtMax0=10000000000, $t$ = .7, cfl=0.9, diss=0, , kx=1, ky=1, kz=1, Sat Jul 17 06:49:00 2021

**Box - Order 4**

| grid | N | $u$ | r |
|---|---|---|---|
| box1 | 1 | 1.5e-3 | |
| box2 | 2 | 1.2e-4 | 12.26 |
| box4 | 4 | 6.4e-6 | 19.44 |
| box8 | 8 | 3.3e-7 | 19.45 |
| box16 | 16 | 1.4e-8 | 23.10 |
| rate | | 4.20 | |

Table 17: CgWave, helmholtz, max norm, order=4, ts=explicit, orderInTime=-1, dtMax0=10000000000, $t$ = .7, cfl=0.9, diss=0, , kx=1, ky=1, kz=1, Sat Jul 17 06:37:57 2021

**Non-Box - Order 2**

| grid | N | $u$ | r |
|---|---|---|---|
| nonBox1 | 1 | 6.0e-4 | |
| nonBox2 | 2 | 1.4e-4 | 4.29 |
| nonBox4 | 4 | 3.4e-5 | 4.06 |
| nonBox8 | 8 | 8.5e-6 | 4.02 |
| rate | | 2.04 | |

Table 18: CgWave, helmholtz, max norm, order=2, ts=explicit, orderInTime=-1, dtMax0=10000000000, $t$ = .7, cfl=0.9, diss=0, , kx=1, ky=1, kz=1, Sat Jul 17 06:49:38 2021

**Non-Box - Order 4**

| grid | N | $u$ | r |
|---|---|---|---|
| nonBox1 | 1 | 1.2e-4 | |
| nonBox2 | 2 | 6.4e-6 | 19.44 |
| nonBox4 | 4 | 3.3e-7 | 19.45 |
| nonBox8 | 8 | 1.4e-8 | 23.10 |
| rate | | 4.36 | |

Table 19: CgWave, helmholtz, max norm, order=4, ts=explicit, orderInTime=-1, dtMax0=10000000000, $t$ = .7, cfl=0.9, diss=0, , kx=1, ky=1, kz=1, Sat Jul 17 08:19:54 2021

### *6.3. Gaussian Plane Wave*

Figure 2 shows a modulated Gaussian plane wave hitting some shapes, scheme FD44u (fourth-order upwind).

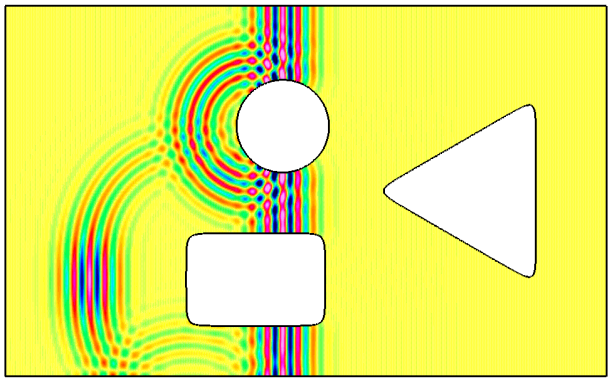

Figure 2: Modulated Gaussian plane wave hitting some shapes, FD44u.

### *6.4. Disk eigenmodes*

The eigenmodes of the wave equation on a disk take the form

$$u = \cos(c\lambda_{m,n}t)\, J_n(\lambda_{m,n}r)\cos(n\theta) \tag{75}$$

**DISK - FD22u - Order 2 - DIRICHLET**

| grid | N | $u$ | r |
|---|---|---|---|
| sic2 | 1 | 6.0e-3 | |
| sic4 | 2 | 1.3e-3 | 4.69 |
| sic8 | 4 | 3.0e-4 | 4.29 |
| sic16 | 8 | 7.0e-5 | 4.20 |
| rate | | 2.13 | |

Table 20: CgWave, diskEig, max norm, order=2, ts=explicit, orderInTime=-1, dtMax0=10000000000, $t$ = .65, cfl=0.9, upwind=1, bcApproach=cbc, -known=diskEig, kx=1, ky=0, kz=0, nBessel=1, mTheta=1, Fri Jan 21 06:38:06 2022

**DISK - FD44u - Order 4 - DIRICHLET - one-sided.** Results using the default one-side BCs.

| grid | N | $u$ | r |
|---|---|---|---|
| sic2 | 1 | 1.1e-4 | |
| sic4 | 2 | 4.6e-6 | 22.84 |
| sic8 | 4 | 2.5e-7 | 18.63 |
| sic16 | 8 | 1.5e-8 | 16.49 |
| rate | | 4.26 | |

Table 21: CgWave, diskEig, max norm, order=4, ts=explicit, orderInTime=-1, dtMax0=10000000000, $t$ = .65, cfl=0.9, upwind=1, bcApproach=oneSided, -known=diskEig, kx=1, ky=0, kz=0, nBessel=1, mTheta=1, Fri Jan 21 06:28:00 2022

**DISK - FD44u - Order 4 - DIRICHLET - CBC.** Here are some initial results using compatibility BCs.

| grid | N | $u$ | r |
|---|---|---|---|
| sic2 | 1 | 8.3e-5 | |
| sic4 | 2 | 4.6e-6 | 17.97 |
| sic8 | 4 | 2.5e-7 | 18.56 |
| sic16 | 8 | 1.5e-8 | 16.56 |
| rate | | 4.15 | |

Table 22: CgWave, diskEig, max norm, order=4, ts=explicit, orderInTime=-1, dtMax0=10000000000, $t$ = .65, cfl=0.9, upwind=1, bcApproach=cbc, -known=diskEig, kx=1, ky=0, kz=0, nBessel=1, mTheta=1, Fri Jan 21 06:23:30 2022

**DISK - FD66u - Order 6 - DIRICHLET - CBC.** Here are some initial results using ORDER=6 compatibility BCs. These CBC's use extrapolation as initial guesses for cross terms. Upwinding needs CFL=.5 for some reason, unstable at the boundary at cfl=.9

| grid | N | $u$ | r |
|---|---|---|---|
| sic2 | 1 | 2.2e-6 | |
| sic4 | 2 | 3.2e-8 | 70.44 |
| sic8 | 4 | 4.6e-10 | 69.28 |
| sic16 | 8 | 7.0e-12 | 65.11 |
| rate | | 6.09 | |

Table 23: CgWave, diskEig, max norm, order=6, ts=explicit, orderInTime=-1, dtMax0=10000000000, $t$ = .65, cfl=.5, upwind=1, bc=, bcApproach=cbc, -known=diskEig, kx=1, ky=0, kz=0, nBessel=1, mTheta=1, -tz=poly, degreeInSpace=2, -degreeInTime=2, Sat Feb 19 13:15:57 2022

### *6.5. Annulus eigenmodes*

**ANNULUS - FD44u - Order 4 - DIRICHLET - CBC.** Here are some initial results using compatibility BCs.

| grid | N | $u$ | r |
|---|---|---|---|
| annulus2 | 1 | 3.6e-4 | |
| annulus4 | 2 | 1.8e-5 | 20.39 |
| annulus8 | 4 | 9.3e-7 | 19.33 |
| annulus16 | 8 | 5.2e-8 | 17.81 |
| rate | | 4.26 | |

Table 24: CgWave, annulusEig, max norm, order=4, ts=explicit, orderInTime=-1, dtMax0=10000000000, $t$ = .65, cfl=0.9, upwind=1, bcApproach=cbc, -known=annulusEig, kx=1, ky=0, kz=0, nBessel=1, mTheta=1, Fri Jan 21 06:27:15 2022

## References


[1] W. D. Henshaw, D. W. Schwendeman, Parallel computation of three-dimensional flows using overlapping grids with adaptive mesh refinement, J. Comput. Phys. 227 (16) (2008) 7469–7502.

[2] W. D. Henshaw, A high-order accurate parallel solver for Maxwell's equations on overlapping grids, SIAM J. Sci. Comput. 28 (5) (2006) 1730–1765.

[3] J. W. Banks, W. D. Henshaw, Upwind schemes for the wave equation in second-order form, J. Comput. Phys. 231 (17) (2012) 5854–5889.

[4] J. Angel, J. W. Banks, W. D. Henshaw, High-order upwind schemes for the wave equation on overlapping grids: Maxwell's equations in second-order form, J. Comput. Phys. 352 (2018) 534–567.

[5] J. Angel, J. W. Banks, W. D. Henshaw, Efficient high-order upwind difference schemes for the second-order wave equation on overlapping grids, Tech. rep., in preparation (2018).