

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	1
2	Governing Equations	2
3	Numerical Scheme	2
4	High-order accurate modified equation schemes	3
4.1	Efficient evaluation of the modified equation schemes	4
4.2	Hierarchical tensor product modified equation schemes on Cartesian grids.	5
4.3	Hierarchical tensor product modified equation schemes on Curvilinear grids.	7
4.4	Factored Modified Equation (FAME) Schemes.	10
5	Upwind dissipation	13
6	Numerical results	15
6.1	Plane Wave	15
6.2	Time-periodic solution in a box (showing forcing)	17
6.3	Gaussian Plane Wave	19
6.4	Disk eigenmodes	19
6.5	Annulus eigenmodes	20

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)

¹This work was performed under DOE contracts from the ASCR Applied Math Program.

²Research supported by the National Science Foundation under grant DMS-1519934.

³Research 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 $u = u(\mathbf{x}, t)$,

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

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

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

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

$$D_{+t} D_{-t} U_1^n = c^2 \Delta_{4h} U_1^n + \frac{\Delta t^2}{12} (c^2 \Delta_{2h})^2 U_1^n + f(\mathbf{x}_i, t^n) + \frac{\Delta t^2}{12} \left(c^2 \Delta_{2h} f(\mathbf{x}_i, t^n) + \partial_t^2 f(\mathbf{x}_i, t^n) \right)$$

where Δ_{ph} is a p-th order accurate approximation to Δ .

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 $D_{+t}D_{-t}u$,

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

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

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

$$D_{+t}D_{-t}u_j = c^2 \Delta_h u_j + 2\frac{\Delta t^2}{4!}(c^2 \Delta)_h^2 u_j + 2\frac{\Delta t^4}{6!}(c^2 \Delta)_h^3 u_j + \dots \quad (4)$$

$$= \sum_{m=1}^{p/2} 2\frac{\Delta t^{2(m-1)}}{(2m)!}(c^2 \Delta)_h^m u_j \quad (5)$$

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

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

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 $D_{+x}D_{-x}$,

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

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

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

$$\partial_x e^{ikx} = ik e^{ikx}, \quad (9)$$

$$\partial_x^2 e^{ikx} = -k^2 e^{ikx}, \quad (10)$$

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

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

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);
```

	β_0	β_1	β_2	β_3	β_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)

$$\partial_x^2 u \approx D_{+x}D_{-x}u_j - \frac{\Delta x^2}{12}(D_{+x}D_{-x})^2 u_j + \frac{\Delta x^4}{90}(D_{+x}D_{-x})^3 u_j, \quad (13)$$

$$\partial_x^4 u \approx (D_{+x}D_{-x})^2 u_j - \frac{\Delta x^2}{6}(D_{+x}D_{-x})^3 u_j, \quad (14)$$

$$\partial_x^6 u \approx (D_{+x}D_{-x})^3 u_j \quad (15)$$

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 $(p + 2)(p/2)$.

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

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

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

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

$$v_j^{(1)} = D_{+x}D_{-x}u_j, \quad (18)$$

$$v_j^{(2)} = D_{+x}D_{-x}v_j^{(1)} \quad (= (D_{+x}D_{-x})^2 u_j), \quad (19)$$

$$v_j^{(3)} = D_{+x}D_{-x}v_j^{(2)}, \quad (= (D_{+x}D_{-x})^3 u_j), \quad (20)$$

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

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

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

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

and the CPP is

$$\text{CPP}[SA, 1D] = p+1; \quad (23)$$

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 p th-order accurate approximation to $\partial_x^2 \partial_y^2 u$

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

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

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

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 $(p+1)^3$

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

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

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

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

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

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 $3(p+1)$,

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

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

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

```

1: for i do
2:    $d_{20,i} = D_{+x}D_{-x}u_i$ 
3:    $d_{02,i} = D_{+y}D_{-y}u_i$ 
4: end for
5: for i do
6:    $d_{40,i} = D_{+x}D_{-x}d_{20,i}$ 
7:    $d_{22,i} = D_{+x}D_{-x}d_{02,i}$ 
8:    $d_{04,i} = D_{+y}D_{-y}d_{02,i}$ 
9: end for
10: for i do
11:   // These next variables do not need to be stored:
12:    $d_{60,i} = D_{+x}D_{-x}d_{40,i}$ 
13:    $d_{42,i} = D_{+x}D_{-x}d_{22,i}$ 
14:    $d_{24,i} = D_{+x}D_{-x}d_{04,i}$ 
15:    $d_{06,i} = D_{+y}D_{-y}d_{04,i}$ 
16:
17:    $u_{xx,i} = d_{20,i} - \frac{\Delta x^2}{12}d_{40,i} + \frac{\Delta x^4}{90}d_{60,i}$  ▷ 6th order  $\partial_x^2 u$ 
18:    $u_{yy,i} = d_{02,i} - \frac{\Delta y^2}{12}d_{04,i} + \frac{\Delta y^4}{90}d_{06,i}$ 
19:    $u_{xxx,i} = d_{40,i} - \frac{\Delta x^2}{6}d_{60,i}$  ▷ 4th order  $\partial_x^4 u$ 
20:    $u_{xxyy,i} = d_{22,i} - \frac{\Delta x^2}{12}d_{42,i} - \frac{\Delta y^2}{12}d_{24,i}$ 
21:    $u_{yyyy,i} = d_{04,i} - \frac{\Delta y^2}{6}d_{06,i}$ 
22:    $u_{xxxx,i} = d_{60,i}$  ▷ 2nd order  $\partial_x^6 u$ 
23:    $u_{xxxxyy,i} = d_{42,i}$ 
24:    $u_{xyyyyy,i} = d_{24,i}$ 
25:    $u_{yyyyyy,i} = d_{06,i}$ 
26:
27:    $u_i^{n+1} = 2u_i^n + u_i^{n-1} + (c^2\Delta t^2)(u_{xx,i} + u_{yy,i})$ 
28:    $+ \frac{c^4\Delta t^4}{12}(u_{xxx,i} + 2u_{xxyy,i} + u_{yyyy,i})$ 
29:    $+ \frac{c^6\Delta t^6}{360}(u_{xxxx,i} + 3u_{xxxxyy,i} + 3u_{xyyyyy,i} + u_{yyyyyy,i})$ 
30: 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

```

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

```

Algorithm 3 Modified equation scheme with tensor product cross terms

```

1: for  $i_2$  do
2:   for  $i_1$  do
3:      $S_x = \sum_{m_1=-p/2}^{p/2} c_{m_1}^{(x)} u_{\mathbf{i}+m_1\mathbf{e}_1}$  ▷ Stencil in x
4:      $S_y = \sum_{m_2=-p/2}^{p/2} c_{m_2}^{(y)} u_{\mathbf{i}+m_2\mathbf{e}_2}$  ▷ Stencil in y
5:     for  $m_2 = -p : p$  do ▷ Could store values computed for previous  $i_2$ , then just need  $m_2 = p$ .
6:        $j_1 = i_1, j_2 = i_2 + m_2$ 
7:        $v_j = D_{+x}D_{-x}u_j$ 
8:     end for
9:      $S_{xy} = \frac{c^4 \Delta t^4}{12} \left[ D_{+y}D_{-y}v_i - \frac{\Delta x^2}{12} (D_{+y}D_{-y})^2 v_i + \dots \right]$  ▷ Sample cross term
10:     $u_i^{n+1} = 2u_i^n + u_i^{n-1} + S_x + S_y + S_{xy}$ 
11:  end for
12: 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,

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

but that derivatives of the metrics must be computed.

Using the chain rule we have, for example,

$$\partial_x u = r_x \partial_r u + s_x \partial_s u, \quad (34)$$

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

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*urrr+2*rx*sx*urs+sx^2*uss+rx*xur+sxx*us:
DuDx[1,1,0,2]:=rx*ry*urr+(sy*rx+ry*sx)*urs+sx*sy*uss+rx*yur+sxy*us:
DuDx[0,2,0,2]:=ry^2*urrr+2*ry*sy*urs+sy^2*uss+ry*yur+syy*us:
DuDx[3,0,0,2]:=rx^3*urrrr+3*rx^2*sx*urrs+3*rx*sx^2*urss+sx^3*ussss+3*rx*rx*xurrr
+(3*rx*sxx+3*rx*sx)*urs+3*sx*sxx*uss+rx*x*xur+sxx*xus:
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*rx*ry*rx+rx*x*ry)*urr+(2*rx*sxy+rx*x*sy+2*rx*y*sx+ry*sxx)*urs
+(2*sx*sxy+sxx*sy)*uss+rxxy*ur+sxy*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)*urrrs
+(2*sx*sy^2*rx+2*sy*sx^2*ry)*ursss+sx^2*sy^2*ussss+(ryy*rx^2+4*rx*ry*ry*rx+rx*x*ry^2)*urrr
+(rx^2*syy+(4*rx*ry*sy+4*sxy*ry+2*ryy*sx)*rx+ry^2*sxx+(2*rx*x*sy+4*rx*y*sx)*ry)*urrs
+((2*sx*syy+4*sxy*sy)*rx+(4*sx*sxy+2*sxx*sy)*ry+ryy*sx^2+4*rx*ry*sy*sx+rx*x*sy^2)*urss
+(sx^2*syy+4*sxy*sy*sx+sxx*sy^2)*ussss+(2*rxxy*rx+rx*x*ryy+2*rxxy*ry+2*rx*y^2)*urr
+(2*rx*sxy+rx*x*syy+2*rxxy*sy+4*rx*y*sxy+2*rxxy*sx+2*ry*sxy+ryy*sxx)*urs
+(2*sx*sxy+sxx*syy+2*sxx*sy+2*sxy^2)*uss+rxxy*ur+sxy*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.*rx*xsx
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 uxy -----
cuxxy100 = rxy
cuxxy200 = 2.*rx*rxy+rx*ry
cuxxy300 = rx**2.*ry
cuxxy010 = sxy
cuxxy110 = 2.*rx*sxy+rx*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

! uxy = 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_i but also to the metrics, $\mathbf{r}\mathbf{x}_i$,

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

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

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

$$\partial_r^2 u = d_{2,0,i} - \frac{\Delta r^2}{12} d_{40,i} + \dots, \quad (38)$$

$$\partial_r^2 \mathbf{r}\mathbf{x} = \mathbf{r}\mathbf{x}_{2,0,i} - \frac{\Delta r^2}{12} \mathbf{r}\mathbf{x}_{40,i} + \dots, \quad (39)$$

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

$$\partial_x^2 \mathbf{r}\mathbf{x} = \text{cuxx20} \partial_r^2 \mathbf{r}\mathbf{x} + \text{cuxx11} \partial_r \partial_s \mathbf{r}\mathbf{x} + \dots \quad (40)$$

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

$$\partial_x^2 u = \text{cuxx20} \partial_r^2 u + \text{cuxx11} \partial_r \partial_s u + \dots \quad (41)$$

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

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

Instead of storing the full stencil we could instead store the $2 + 3 = 5$ (in 2D) coefficients c_{mn} . To evaluate $\Delta_h u_j$ 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

$$\Delta^2 u = c_{40} \partial_r^4 u + c_{31} \partial_r^3 \partial_s u + c_{22} \partial_r^2 \partial_s^2 u + \dots \quad (43)$$

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

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

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

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

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 p th 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 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.

Grid	ord	scheme	CPU ns/step/pt						
			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 ($1 \text{ ns} = 10^{-9} \text{ s}$).

Grid	ord	scheme	CPU ns/step/pt						
			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-Stratgy 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 ($1 \text{ ns} = 10^{-9} \text{ s}$).

Grid	ord	scheme	CPU ns/step/pt						
			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-Stratgy 1, compiled -O3. CPU performance data. Time in nano-seconds per-step per-grid-point ($1 \text{ ns} = 10^{-9} \text{ s}$).

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

$$\text{cycles per-step per-point} = \text{seconds per-step per-point} \times \text{cycles-per-second} \quad (46)$$

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

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

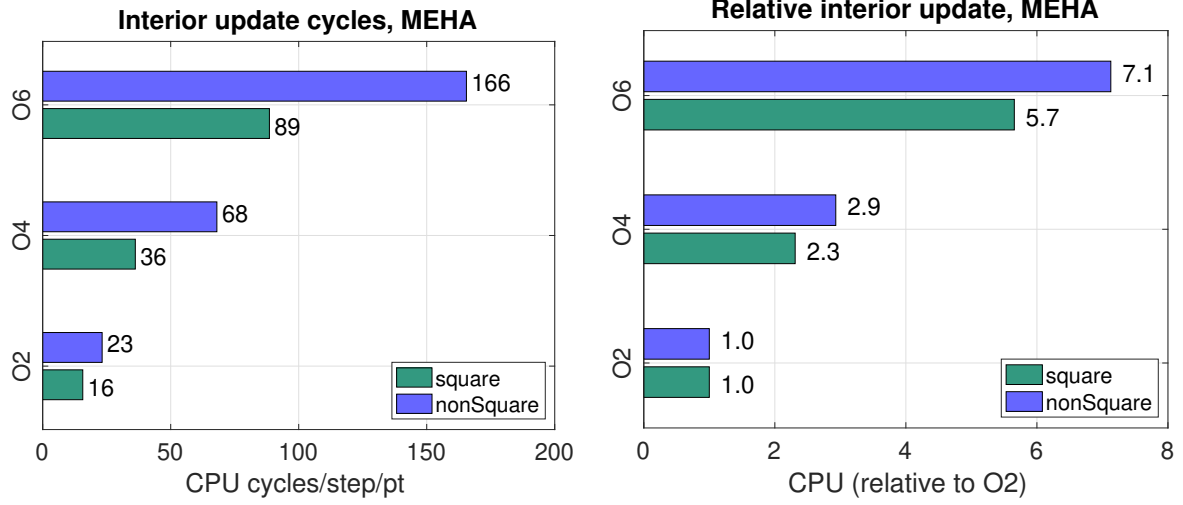


Figure 1: Left: CPU cycles per-step per-point (estimates FLOPS per-step per-point). Right: CPU time relative to order 2 scheme. Data from Table 3.

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 = $2 \times 10.2 \times 10^{-6} / (1024^2)$ = 19 ns/pt -> $19 / .37 = 51$ cycles/pt $\log_2(N)^2 = 100$

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 = $2 \times 59.3 \times 10^{-6} / (2048^2)$ = 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 = $2 \times 251.49 \times 10^{-6} / (256^3)$ = 30 ns/pt

bench -s r512x512x512

Problem: r512x512x512, setup: 10.96 s, time: 2.61 s, 'mflops': 3471

Time for 2 FFTs = $2 \times 2610 \times 10^{-6} / (512^3)$ = 39 ns/pt

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

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

for some coefficient ν_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 $u_j^p \approx u_j^{n+1}$,

$$u_j^p = 2u_j^n - u_j^{n-1} + \Delta t^2 \left(c^2 D_{h,xx} u_j^n + \dots \right), \quad (48)$$

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

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

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

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

where ν_q is the coefficient of the upwind dissipation and λ is the CFL parameter,

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

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

$$\lambda < 1, \quad (53)$$

$$\zeta < 2, \quad (54)$$

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

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

$$\nu_p < \frac{2}{\lambda 4^q} = \frac{1}{\lambda 2^{p+1}} \quad (56)$$

In two-dimensions **check me**

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

and we need $\zeta < 2$ or

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

or

$$\nu_p < \frac{2}{\lambda_x 4^q + \lambda_y 4^q} = \frac{1}{2^{p+1}} \frac{1}{\lambda_x + \lambda_y} \quad (59)$$

In two-dimensions the CFL condition is

$$\lambda_x^2 + \lambda_y^2 < 1 \quad (60)$$

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

$$\nu_p < \frac{1}{\sqrt{2}} \frac{1}{2^{p+1}} \quad (61)$$

In three-dimensions

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

where

$$\lambda_x^2 + \lambda_y^2 + \lambda_z^2 < 1 \quad (63)$$

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

$$\nu_p < \frac{1}{\sqrt{3}} \frac{1}{2^{p+1}} \quad (64)$$

Summary: In d -dimensions we require

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

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

$$\nu_2 = \frac{1}{\sqrt{d}} \frac{1}{8} \quad (66)$$

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

$$\nu_4 < \frac{1}{\sqrt{d}} \frac{1}{32} \quad (67)$$

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 ν_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 ν_p is to evaluate $D_{0t}U^n$ in a Gauss-Seidel fashion. The predictor sets a preliminary value for u_j^{n+1} ,

$$u_j^{n+1} = 2u_j^n - u_j^{n-1} + \Delta t^2 \left(c^2 D_{h,xx} u_j^n + \dots \right), \quad (68)$$

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

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

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

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

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 ,

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

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

$$u_j^{n+1} = 2u_j^n - u_j^{n-1} + \Delta t^2 \left(c^2 D_{h,xx} u_j^n + \dots \right), \quad (73)$$

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

6. Numerical results

Here are some numerical results.

6.1. Plane Wave

Here are errors in computing an exact plane wave solution

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

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 5: 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 6: 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 7: 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 8: 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 9: 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 10: 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

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

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 11: 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 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

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 13: 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 14: 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 15: 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 16: 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 17: 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 18: 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).

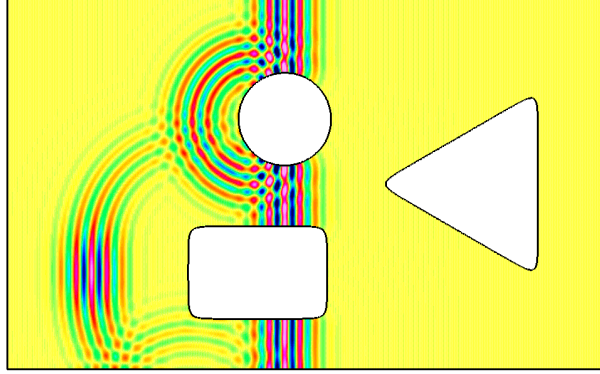


Figure 2: Modulated Gaussian plane wave hitting some shapes, FD44u.

6.4. Disk eigenmodes

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

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

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 19: 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 20: 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 21: 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 22: 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 23: 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).