Algorithms and Codes for Wave Propagation Problems

Henrik Holst
KTH Royal Institute of Technology
School of Computer Science and Communication
Department of Numerical Analysis
SE-100 44 Stockholm

November 14, 2011

Abstract

This technical report is a summary of selected numerical methods for multiscale wave propagation problems. The main topic is the discussion of finite difference schemes, kernels for computing the mean value of oscillatory functions and how to compute coefficients in an effective equation for long time wave propagation.

1 Introduction

In this technical report we will describe some of the numerical codes written in Matlab [10], Sage [12] and Maple [9] while working on multiscale wave propagation problems, [2, 5, 6, 4]. The intention is not to give a complete description of all codes that we have used, but to provide students and new researchers in the field with a few "tried and tested" ideas to start out with. This report also acts as a forward pointer and appendix to the papers mentioned above, co-authored with Prof. Björn Engquist and Prof. Olof Runborg.

The common denominator for our work has been the wave equation, written in what we call "flux form",

$$\begin{cases} u_{tt} - \nabla \cdot f = 0, & \Omega \times [0,T], \\ u(x,0) = u_0(x), \quad u_t(x,0) = u_1(x), & \forall x \in \Omega, \end{cases} \tag{1}$$

with Ω-periodic boundary conditions. We assume that $\Omega \subset \mathbb{R}^d$ is a d-dimensional cube. We will classify (1) into four types depending on which numerical and mathematical method used:

DNS : We assume that u has oscillations of order ε. The oscillations can be due to fast variations in A^ε where $f = A^\varepsilon \nabla u$, or due to high frequency initial data of order $1/\varepsilon$. DNS is an acronym for *direct numerical simulation* borrowed from computational fluid dynamics (CFD).

HOM : The approximate solution to (1) where we approximated an oscillatory flux $f = A^\varepsilon \nabla u$ with $f = \bar{A} \nabla u$. The coefficient $\bar{A}$ is the homogenized

(HOM) coefficient of A^ε with no fast oscillations, [1]. This model does not need as many grid points as the DNS computation since the solution will not be oscillatory on the ε scale. However, this model is only valid for $T = \mathcal{O}(1)$ with respect to ε. We will refer to $T = \mathcal{O}(1)$ as *finite time* problems.

EFF : The approximate solution to (1) for $T = \mathcal{O}(\varepsilon^{-2})$. We will refer to problems with such T as *long time* problems. In this long time problem, in one dimension, it is not enough to approximate $f = A^\varepsilon u_x$ by $f = \bar{A} u_x$. We need to add an additional term to the flux,

$$f = \bar{A} u_x + \varepsilon^2 \beta u_{xxx},$$

which captures the dispersive behavior of the solution, [11, 4, 6, 7]. We will discuss this dispersive wave equation (e.g., effective equation (EFF)) more in Section 2.2.

HMM : A heterogeneous multiscale method (HMM) solution. The equation (1) is discretized on a coarse grid. The evaluation of f on the discrete grid points are based on computations of micro problems over micro boxes of size proportional to ε. In the micro problems the full wave equation is solved, as described in DNS above. The macroscopic fluxes are then evaluated by computing the mean value of $f = A^\varepsilon \nabla u$ inside each micro box. The HMM method accurately captures the coarse scale solution up to $T = \mathcal{O}(\varepsilon^{-2})$, [3].

In [4] we developed a HMM method where left and right going energy where natural choices of coarse scale variables. The two macroscopic variables satisfied a pair of advection equations which we discretized with an upwind scheme. The implementation was done in a similar fashion as for the wave equation where the fluxes are computed by from micro problems.

2 Finite difference methods for the wave equation

The numerical scheme used to discretize (1) is a *method of lines* (MOL) method [8]. The flux f is discretized on uniform grid with M discrete points (excluding related boundary conditions) in lexicographical ordering. In time, we use a three point second order central difference approximation,

$$u_{tt} \approx \frac{u_m^{n+1} - 2u_m^n + u_m^{n-1}}{\Delta t^2}, \qquad \Delta t = \frac{T}{N}.$$

The discrete system takes the form,

$$u_m^{n+1} = 2u_m^n - u_m^{n-1} + \frac{\Delta t^2}{\Delta x} D \cdot f_m^n, \tag{2}$$

where $\frac{1}{\Delta x} D \cdot f_m^0$ is a discrete approximation of $\nabla \cdot f$ at $t = 0$. We will use the notation $\frac{1}{\Delta x} D u_m^n$ for the discrete approximation of ∇u. The flux f lives on a spatially staggered grid, shifted one half grid point in one coordinate direction.

Staggered grids are a common technique in numerical schemes for conservation laws, [8]. In Section 6 we give a detailed description of the finite difference schemes used for one, two and three dimensional problems, as well as a finite difference scheme which has a smaller phase error. That scheme is especially well suited long time problems, $T = \mathcal{O}(\varepsilon^{-2})$, where we want as little numerical dispersion as possible to avoid contaminating the correct dispersive behavior of the solution.

2.1 Initial time step

In the two-step method (2) we need to provide u_m^{-1} (or u_m^1) with another method than the suggested two-step method. We will compute u_m^{-1} with the one-step approximation,

$$\begin{aligned} u_m^{-1} = u(x_m, 0) - \Delta t u_t(x_m, 0) + \frac{(-\Delta t)^2}{2} u_{tt}(x_m, 0) + \mathcal{O}(\Delta t^3) \\ \approx u_0(x_m) - \Delta t u_1(x_m) + \frac{\Delta t^2}{2\Delta x} D \cdot f_m^0. \end{aligned}$$

2.2 Dispersion analysis for one-dimensional problems.

In [6] and [4] we want to find the dispersive solution of the long time wave equation in one dimension, where f in (1) is of the form,

$$f = \bar{A} u_x + \varepsilon^2 \beta u_{xxx}.$$

We will consider two numerical schemes and analyze the phase error related to the exact phase given from f above. For the analysis we will consider f as an exact quantity given from a piecewise third order interpolation polynomial p interpolating u on a uniform grid, as in [6]. We assume that,

$$f = \left(\bar{A}\partial_x + \beta\varepsilon^2 \partial_{xxx}\right) p, \tag{3}$$

where the polynomial $p_{m-1/2}$ is defined in a neighborhood around the staggered grid points $x_{m-1/2}$, the same grid point where $f_{m-1/2}^n$ is defined. The interpolation polynomial $p_{m-1/2}$ is of the form,

$$\begin{aligned} p_{m-1/2}(x) = c_{m1} + c_{m2}(x - x_{m-2}) + c_{m3}(x - x_{m-2})(x - x_{m-1}) \\ + c_{m4}(x - x_{m-2})(x - x_{m-1})(x - x_m), \end{aligned}$$

where the coefficients c_{mn} are given by,

$$\begin{cases} c_{m1} = u_{m-2}, \\ c_{m2} = \dfrac{u_{m-1} - u_{m-2}}{\Delta x}, \\ c_{m3} = \dfrac{u_m - 2u_{m-1} + u_{m-2}}{2\Delta x^2}, \\ c_{m4} = \dfrac{u_{m+1} - 3u_m + 3u_{m-1} - u_{m-2}}{6\Delta x^3}. \end{cases}$$

The first scheme is of the form,

$$u_m^{n+1} = 2u_m^n - u_m^{n-1} + \frac{\Delta t^2}{\Delta x}\left(f_{m+\frac{1}{2}}^n - f_{m-\frac{1}{2}}^n\right), \tag{4}$$

and the second scheme is of the form,

$$u_m^{n+1} = 2u_m^n - u_m^{n-1} + \frac{\Delta t^2}{24\Delta x}\left(-f^n_{m-\frac{3}{2}} + 27f^n_{m+\frac{1}{2}} - 27f^n_{m-\frac{1}{2}} + f^n_{m-\frac{3}{2}}\right). \quad (5)$$

The first scheme is good for finite time problems and the second scheme is suited for long time wave propagation problems. We define the following quantities,

$$\lambda = \sqrt{\bar{A}}\frac{\Delta t}{\Delta x}, \quad \rho = \sqrt{\frac{\beta}{\bar{A}}}\frac{\varepsilon}{\Delta x},$$

and

$$c_1 = \frac{\lambda^2}{24}, \quad c_2 = \frac{\lambda^2}{24^2}, \quad d = 24\rho^2.$$

The first scheme, expanding in terms of u_m^n, is of the form,

$$\begin{aligned} u_m^{n+1} = {} & 2u_m^n - u_m^{n-1} \\ & + c_1\left(-u^n_{m+2} + 28u^n_{m+1} - 54u^n_m + 28u^n_{m-1} - u^n_{m-2}\right) \\ & + c_1 d\left(u^n_{m+2} - 4u^n_{m+1} + 6u^n_m - 4u^n_{m-1} + u^n_{m-2}\right), \end{aligned}$$

and is second order accurate in both time and space. The second scheme is of the form,

$$\begin{aligned} u_m^{n+1} = {} & 2u_m^n - u_m^{n-1} \\ & + c_2\left(u^n_{m+3} - 54u^n_{m+2} + 783u^n_{m+1} - 1460u^n_m\right. \\ & \left.+783u^n_{m-1} - 54u^n_{m-2} + u^n_{m-3}\right) \\ & + c_2 d\left(-u^n_{m+3} + 30u^n_{m+2} - 111u^n_{m+1} + 164u^n_m - 111u^n_{m-1}\right. \\ & \left.+30u^n_{m-2} - u^n_{m-3}\right), \end{aligned}$$

and is also second order in time and space. Both the schemes have the stability condition,

$$|g_i| \le 1, \qquad g_i^2 = (2 + c_i p_i(v))g_i - 1, \qquad i = 1, 2,$$

where g_i is called the amplification factor for scheme i. The polynomial $p_1(v)$ for the first scheme (4) is given by

$$\begin{cases} p_1(v) = Av^2 + Bv + C, \\ A = 4(d-1), \quad B = -8(d-7), \quad C = 4(d-13), \end{cases}$$

and for the second scheme (5), the polynomial $p_2(v)$ is given by

$$\begin{cases} p_2(v) = Av^3 + Bv^2 + Cv + D, \\ A = -8(d-1),\ B = 120\left(d - \dfrac{9}{5}\right),\ C = -216\left(d - \dfrac{65}{9}\right),\ D = 104\,(d-13). \end{cases}$$

In [4] we proved stability for (5) in terms of ε, $\bar{A}$, β, ε, Δt and Δx:

Theorem 1. *The finite difference scheme* (5) *applied to the effective equation* (1) *with 1-periodic boundary conditions and f as in* (3)*, is stable for Δt and Δx such that*

$$\frac{\varepsilon}{\Delta x} \le \sqrt{\frac{7\bar{A}}{24\beta}},$$

and

$$\frac{\Delta t}{\Delta x} \leq \frac{24}{\sqrt{\bar{A}}}\sqrt{h\left(\frac{24\varepsilon^2\beta}{\Delta x^2\bar{A}}\right)},$$

where

$$h(x) = \begin{cases} \dfrac{1}{784-112x}, & 0 \leq x < \dfrac{21}{5}, \\ \dfrac{x^2-2x+1}{128\left(2(x^2-x+1)^{3/2}-2x^3+3x^2+3x-2\right)}, & \dfrac{21}{5} \leq x \leq 7. \end{cases}$$

It is a fact that the schemes (4) and (5) are non-dissipative, see [13, 6], meaning that the amplification factor g satisfies that $|g| = 1$. The major source of concern is therefore the numerical phase error with respect to the exact phase. We will study the phase error for the two schemes above in relation to the exact phase, given below. The solution to (1) with f as in (2.2), is of the form,

$$u(x,t) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} \hat{u}_+(\omega)e^{i\omega(x+t\sqrt{\bar{A}-\beta\varepsilon^2\omega^2})} + \hat{u}_-(\omega)e^{i\omega(x-t\sqrt{\bar{A}-\beta\varepsilon^2\omega^2})}\,\mathrm{d}\omega,$$

where $\hat{u}_+$ and $\hat{u}_-$ are given from the initial data, [13]. To simplify the analysis we assume that the initial data is such that $\hat{u}_+ = 0$ and we have a right going wave. The Fourier transform of u satisfies

$$\hat{u}(\omega, t+\Delta t) = e^{-i\Delta t\omega\sqrt{\bar{A}-\beta\varepsilon^2\omega^2}}\hat{u}(\omega,t).$$

We define the phase speed ϕ for frequency ω as

$$\phi(\omega) = \sqrt{\bar{A}-\beta\varepsilon^2\omega^2},$$

and the numerical phase speed α is defined by, [13]

$$g = |g|e^{-i\Delta t\alpha(\omega)},$$

where g is the amplification factor of the numerical scheme.

We show a dispersion plot for the exact phase and the two numerical phases for the schemes (4) and (5) in Figure 1, where we have used the following parameters:

$$\begin{cases} \varepsilon = 0.01,\ A^\varepsilon(x) = 1.1 + \sin\dfrac{2\pi x}{\varepsilon}, \\ \bar{A} = \sqrt{0.21},\ \beta = 0.01078280318, \\ H = \dfrac{1}{150},\ K = \dfrac{H}{2}. \end{cases}$$

2.3 Matrix formulation

We have chosen to use a matrix formulation for our finite difference solvers. Matlab is very efficient on sparse matrix times vector operations and it is simple to produce discrete difference operators as sparse matrices. In Table 3 we demonstrate the technique described below.

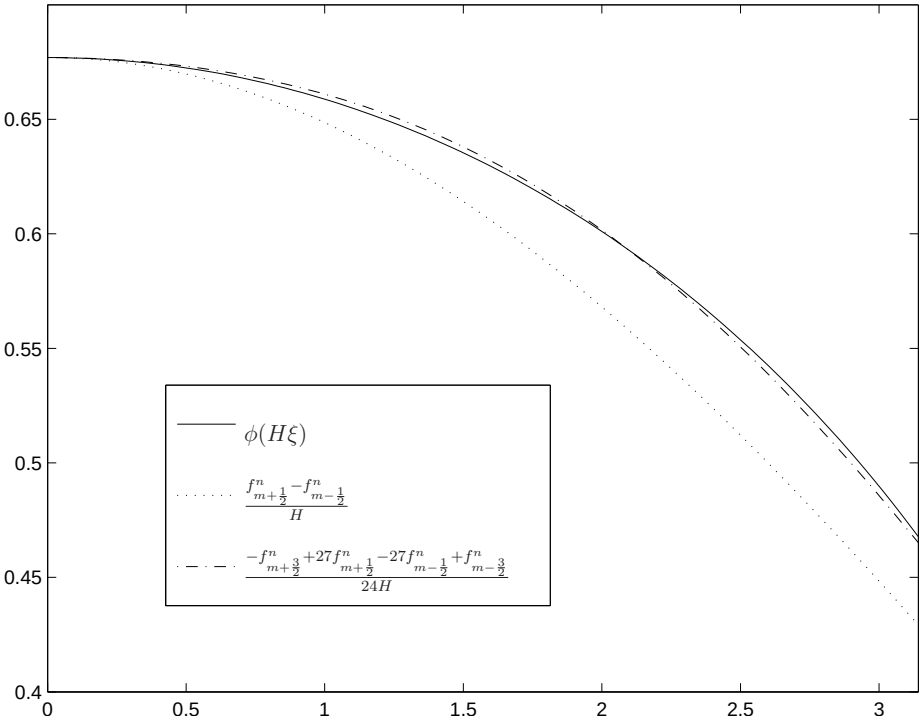

Figure 1: *Dispersion plot over* $0 < H\xi \leq \pi$ *of the exact phase and the two numerical schemes in the example in Section 2.2.*

The two-step schemes we consider, (4) and (5), can be written on matrix form. We have, for all grid points not subject to boundary conditions,

$$\underbrace{\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -\mathcal{F} & 0 & I \end{bmatrix}}_{=B} \begin{bmatrix} u_m^{n+1} \\ u_m^n \\ f_m^{n+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 2I & -I & \frac{\Delta t^2}{\Delta x} D\cdot \\ I & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{=A} \begin{bmatrix} u_m^n \\ u_m^{n-1} \\ f_m^n \end{bmatrix}, \tag{6}$$

and denote $y_m^n = \begin{bmatrix} u_m^n & u_m^{n-1} & f_m^n \end{bmatrix}$. The initial step y_m^0 is given from initial data and B and A are square sparse matrices with the diagonal blocks of size corresponding to the unknowns we are time stepping. The flux operator $\mathcal{F}u_m^n$ should be a discrete flux f, which we assume will be linear in u. For the four different regimes: DNS (also high freq. initial data), HOM, EFF and HMM we have,

$$\begin{cases} \mathcal{F} = \frac{1}{\Delta x} A^\varepsilon D, & \text{(DNS)}, \\ \mathcal{F} = \frac{1}{\Delta x} A D, & \text{(DNS; high freq. initial data)}, \\ \mathcal{F} = \frac{1}{\Delta x} \bar{A} D, & \text{(HOM)}, \\ \mathcal{F} = \frac{1}{\Delta x} \bar{A} D + \frac{\varepsilon^2 \beta}{\Delta x^3} D^{(3)}, & \text{(EFF)}, \\ \mathcal{F} = \frac{1}{\Delta x} \tilde{F}(x_{m-1/2}, x) I_x + \cdots, & \text{(HMM)}, \end{cases}$$

where $\tilde{F}(x_{m-1/2}, Q)$ is the mean value of the solution of the (corrected) HMM micro problem, cf. [4], in the point $x_{m-1/2}$ with initial data Q. The matrix I_x projects the four closest grid points $u_{m-2}, \ldots, u_{m+1}$ around the point $x_{m-1/2}$ on a linear polynomial. Higher order projection operators I_{x^2}, ..., are defined analogously. The matrix A^ε is a $M \times M$ matrix,

$$A_{ij}^\varepsilon = \begin{cases} A^\varepsilon(x_{i-1/2}), & i = j, \\ 0, & \text{otherwise}, \end{cases}$$

and $\bar{A}$ and β are defined analogously. As before, difference operators $\frac{1}{\Delta x}(D\cdot)$ and $\frac{1}{\Delta x} D$ are discrete approximations of $(\nabla\cdot)$ and ∇, taken at a staggered grid in space and periodic boundary conditions. The matrix $\frac{1}{\Delta x^3} D^3 u_m^n$ is finite difference approximation of the third derivative of u. All finite difference matrices are computed with the Matlab code `spfd` in Table 1. In Table 2 we demonstrate a simple use-case of `spfd` and result in Figure 2.

The periodic boundary conditions for u_m^n are implemented with the two matrices,

$$u_{\text{int}}^n = P^\downarrow u^n, \quad u^n = P^\uparrow u_{\text{int}}^n,$$

where u_{int}^n is the interior points of the numerical solution u_m^n. We want u_m^n to hold the solution including the boundary conditions. The projection operators are used to be able to hold a solution over the entire domain Ω to easily plot the entire solution without worrying about applying boundary conditions. Note that we have always used periodic boundary conditions in our computations.

The following of this section will be specific for one dimensional problems. To time step the solution in the usual way we define two projection operators which adds and removes the leftmost value u_0^n. The projection operators $P^\uparrow : \mathbb{R}^M \to \mathbb{R}^{M+1}$ and $P^\downarrow : \mathbb{R}^{M+1} \to \mathbb{R}^M$ are represented by a $(M+1) \times M$ and

$M \times (M+1)$ matrix respectively,

$$P^{\uparrow}_{ij} = \begin{cases} 1, & i = 1 \text{ and } j = M, \\ 1, & i > 1 \text{ and } i - 1 = j, \\ 0, & \text{otherwise}, \end{cases} \quad P^{\downarrow}_{ij} = \begin{cases} 0, & j = 1, \\ 1, & i = j + 1, \\ 0, & \text{otherwise}. \end{cases}$$

We can more easily understand what $P^{\uparrow}$ and $P^{\downarrow}$ does on a pair of grid functions,

$$P^{\uparrow} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_M \end{bmatrix} = \begin{bmatrix} u_M \\ u_1 \\ \vdots \\ u_M \end{bmatrix}, \quad P^{\downarrow} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_M \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_M \end{bmatrix}.$$

2.3.1 Handling nonlinear terms

We consider the Burgers equation, in one dimension,

$$\begin{cases} u_t + \partial_x f = 0, \\ f(u) = \dfrac{1}{2} u^2, \end{cases} \tag{7}$$

with smooth initial data and periodic boundary conditions. If we discretize (7) with upwind scheme,

$$\begin{cases} u_m^{n+1} = u_m^n - \dfrac{\Delta t}{\Delta x} \left(F_m^n - F_{m-1}^n \right), & m = 1, 2, \ldots, M, \\ F_m^{n+1} = \dfrac{1}{2} (u_m^{n+1})^2. \end{cases} \tag{8}$$

We have a nonlinear term $(u_m^{n+1})^2$ appearing in the computation of F_m^{n+1}. We handle this by adding an additional term to the time step recurrence relation (6),

$$\underbrace{\frac{1}{2} D \operatorname{diag}(y^{n+1}) C y^{n+1}}_{\text{new quadratic term}} + B y^{n+1} = A y^n. \tag{9}$$

Now we have in each time step, a nonlinear equation on the form $H(y^{n+1}) = 0$, where $H(x) = \frac{1}{2} D \operatorname{diag}(x) C x + Bx - b$ where $b = Ay^n$. We will use a simple Newton iteration for this job with the start guess y^n. We note that the Jacobian $J(x)$ of H is

$$J(x) = \operatorname{diag}(Dx) C + B,$$

thus, the Newton iteration will take the form,

$$\begin{cases} b = Ay^n, \\ x_0 = y^n, \\ H_k = \dfrac{1}{2} D \operatorname{diag}(x_k) C x_k + B x_k - b \\ J_k = C \operatorname{diag}(D x_k) + B, \\ \Delta_k = (J_k)^{-1} H_k, \\ x_{k+1} = x_k - \Delta_k. \end{cases}$$

We iterate until $\|\Delta_k\|/\|H_k\|$ is smaller than a specified tolerance.

The matrix formulation of the form (9) for our scheme (8) will be,

$$\frac{1}{2}\underbrace{\begin{bmatrix}0 & 0\\ -I & 0\end{bmatrix}}_{=D}\underbrace{\begin{bmatrix}u^{n+1} & 0\\ 0 & F^{n+1}\end{bmatrix}}_{=\mathrm{diag}(y^{n+1})}\underbrace{\begin{bmatrix}I & 0\\ 0 & 0\end{bmatrix}}_{=C}\begin{bmatrix}u^{n+1}\\ F^{n+1}\end{bmatrix}+\underbrace{\begin{bmatrix}I & 0\\ 0 & I\end{bmatrix}}_{=B}\begin{bmatrix}u^{n+1}\\ F^{n+1}\end{bmatrix}=$$
$$\underbrace{\begin{bmatrix}I & -\frac{\Delta t}{\Delta x}D\\ 0 & 0\end{bmatrix}}_{=A}\begin{bmatrix}u^{n}\\ F^{n}\end{bmatrix}.$$

3 Coefficients for the long time wave equation

The code in this section computes the coefficients for the one-dimensional effective equation for long time wave propagation, [4, 6]. This effective equation is of the form (1),

$$f = \bar{A}u_x + \varepsilon^2\beta u_{xxx}.$$

Suppose that $A^\varepsilon(x) = A(x, x/\varepsilon)$ where $A(x, y)$ is 2π-periodic in y. The coefficient $\bar{A}$ will be the homogenized coefficient, the same as,

$$\bar{A}(x) = \left(\frac{1}{2\pi}\int_0^{2\pi}\frac{\mathrm{d}y}{A(x,y)}\right)^{-1}$$

with x being held fixed. Santosa and Symes gave another explicit nested tripple integral also for β when $A(x, y) = A(y)$. We "extended" their idea by the same idea as for homogenized equation, by freezing the slowly varying variable $x = x_0$ and do the computation as if A depended only on y. We present a Maple code in Table 4 which computes the coefficients $\bar{A}_{m-\frac{1}{2}} = \bar{A}(x_{m-\frac{1}{2}})$ and $\beta_{m-\frac{1}{2}} = \beta(x_{m-\frac{1}{2}})$ for $m = 1, 2, \ldots, M$ for the grid point $x_{m-\frac{1}{2}} = (m - \frac{1}{2})\Delta x$ on a uniform grid $\Delta x = 2\pi/M$. Note that the integral is computed numerically, as Maple is usually unable to give β in an exact symbolic form.

4 Local average kernel

In the multiscale methods we have developed, we often use a kernel to accelerate the convergence of mean value computations of oscillatory functions $f^\varepsilon(x) = f(x, x/\varepsilon)$, where $f(x, y)$ is periodic in y. The convergence rate in the mean value computation,

$$\frac{1}{2\eta}\int_{-\eta}^{\eta} f^\varepsilon(x)\,\mathrm{d}x \to \bar{f}(x), \qquad \frac{\varepsilon}{\eta}\to 0,$$

can be accelerated by using a smooth kernel K from a function space $\mathbb{K}^{p,q}$ and taking

$$\frac{1}{\eta}\int_{-\eta}^{\eta} K\left(x/\eta\right) f^\varepsilon(x)\,\mathrm{d}x \to \bar{f}(x) \qquad \frac{\varepsilon}{\eta}\to 0.$$

The theory of such local averaging kernels are described in [4] and [6], with a short summary here: we say that $K \in \mathbb{K}^{p,q}$ if K has compact support on $[-1, 1]$

and is q times continuously differentiable and the following p moment conditions are fulfilled:

$$\int_{\mathbb{R}} K(x)x^i \,\mathrm{d}x = \delta_i, \quad 0 \leq i < p.$$

We will compute the coefficients $a_{p-1}, a_{p-2}, \ldots, a_0$ in the polynomial factor $P(x)$ of degree $p-1$ in the polynomial kernel K, factorized on the form,

$$K(x) = \begin{cases} (1-x^2)^{q+1}P(x), & -1 \leq x \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

These p conditions gives us an equation system for $a_{p-1}, a_{p-2}, \ldots, a_0$ of the form $Ax = b$, where

$$\begin{cases} A_{ij} = \int_{-1}^{1} x^i(1-x^2)^{q+1}x^{p-1-j} \,\mathrm{d}x, \\ b_i = \delta_i, \\ x_i = a_{p-1-i}, \end{cases} \qquad 0 \leq i, j < p.$$

The elements in A will be rational numbers, thus the equation system $Ax = b$ involves only rationals. We can solve this linear equation system exact in Sage, which features an exact solver for system of equations over rationals. The computed coefficients $a_{p-1}, a_{p-2}, \ldots, a_0$, will have a minimal numerical error on the order of $\sim 10^{-16}$ when loaded into Matlab, which uses 64-bit IEEE floating point as default.

The code in Table 5 show a Sage code which computes a set of Matlab ASCII matrices containing the $P(x)$ polynomial coefficients, stored in Matlab polynomial ordering, cf. `help polyval`.

5 Code listings

In this section we show the Matlab, Sage and Maple codes discussed in the text.

```
function S = spfd(varargin)
% S=SPFD(NX,SI,SC),                 1d sparse FD op.,
% S=SPFD(NX,SI,NY,SJ,SC),           2d sparse FD op.,
% S=SPFD(NX,SI,NY,SJ,NZ,SK,SC)     3d sparse FD op..
%
% A second order scheme for d^2/dx^2 in 1d and M grid points:
% DX=1/M, S=SPFD(M,-1:1,[1 -2 1]/DX^2).
%
switch nargin
case 3
    S = spfd1(varargin{1},varargin{2},varargin{3});
case 5
    S = spfd2(varargin{1},varargin{2},...
              varargin{3},varargin{4},varargin{5});
case 7
    S = spfd3(varargin{1},varargin{2},...
              varargin{3},varargin{4},...
              varargin{5},varargin{6},varargin{7});
otherwise
    error('wrong argument count')
end

function S = spfd1(nx,si,sc)
nsc = length(sc); a = 0; b = 0; N = nsc*nx;
iv = zeros(1,N); jv = zeros(1,N); sv = zeros(1,N);
for i = 1:nx
    a = b + 1; b = b + nsc;
    kij = repmat(i,[1 nsc]);
    ksc = 1 + mod(i+si-1,nx);
    iv(a:b) = kij;  jv(a:b) = ksc;  sv(a:b) = sc;
end
S = sparse(iv,jv,sv);

function S = spfd2(nx,si,ny,sj,sc)
nsc = length(sc); a = 0; b = 0; N = nsc*nx*ny;
iv = zeros(1,N); jv = zeros(1,N); sv = zeros(1,N);
for j = 1:ny, for i = 1:nx
    a = b + 1; b = b + nsc;
    kij = repmat(sub2ind([nx ny],i,j),[1 nsc]);
    kx = 1 + mod(i+si-1,nx);
    ky = 1 + mod(j+sj-1,ny);
    ksc = sub2ind([nx ny],kx,ky);
    iv(a:b) = kij;  jv(a:b) = ksc;  sv(a:b) = sc;
end, end
S = sparse(iv,jv,sv);

function S = spfd3(nx,si,ny,sj,nz,sk,sc)
nsc = length(sc); a = 0; b = 0; N = nsc*nx*ny*nz;
iv = zeros(1,N); jv = zeros(1,N); sv = zeros(1,N);
for k = 1:nz, for j = 1:ny, for i = 1:nx
    a = b + 1; b = b + nsc;
    kij = repmat(sub2ind([nx ny nz],i,j,k),[1 nsc]);
    kx = 1 + mod(i+si-1,nx);
    ky = 1 + mod(j+sj-1,ny);
    kz = 1 + mod(k+sk-1,nz);
    ksc = sub2ind([nx ny nz],kx,ky,kz);
    iv(a:b) = kij;  jv(a:b) = ksc;  sv(a:b) = sc;
end, end, end
S = sparse(iv,jv,sv);
```

Table 1: *Matlab routine* `spfd.m` *for one, two and three dimensional periodic problems.*

```
xmin = 0, xmax = 1, nx = 50
si = -1:0, sc = [-1 1]
L = xmax - xmin, dx = L/nx
x = xmin + (1:nx)'*dx;
xm = xmin + (1:2:2*nx)'*(dx/2);
u = sin((2*pi/L)*x);
X2 = spdiags(xm.^2,0,nx,nx);
D = spfd(nx,si,sc/dx);
vx = X2*(D*u);
ux = (2*pi/L)*xm.^2.*cos((2*pi/L)*xm);
abs_err = max(abs(ux-vx))
plot(xm,vx,'x',xm,ux,'o')
grid on, legend('numerical','exact','location','northwest')
```

Table 2: *Simple demonstration of* **spfd.m** *where we compute* $f = x^2 u_x$ *for a smooth function* u *on a uniform grid* $x_{j-1/2}$.

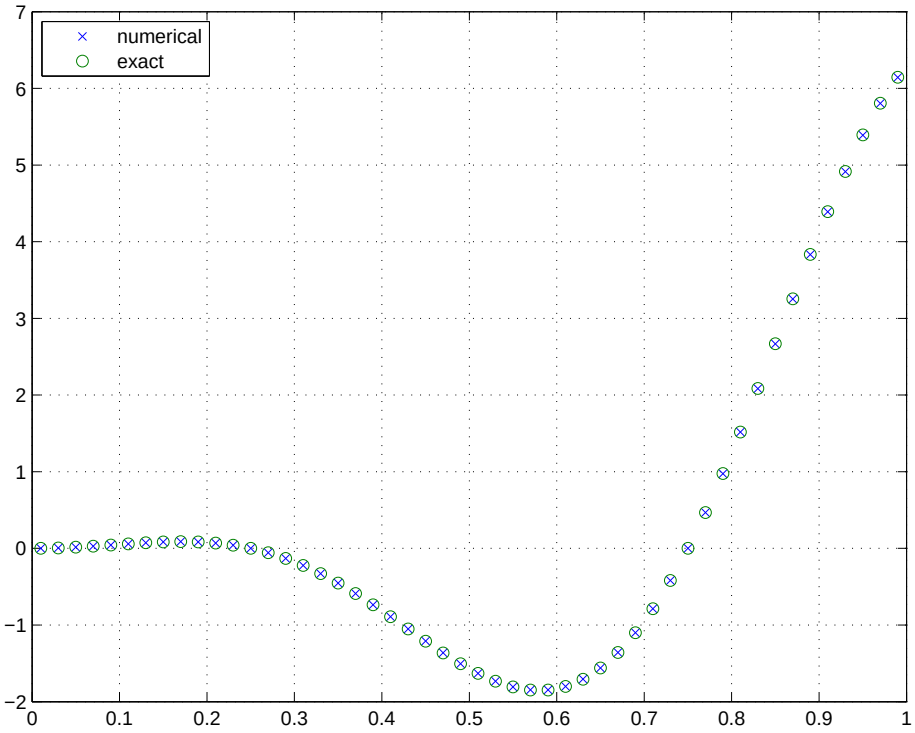

Figure 2: *Result plot from the code in Table 2. The maximum absolute error (*`abs_err`*) is 4.0431e-03.*

```
eps=0.01,c=sqrt(sqrt(.21)),Tmax=1/c
M=1600,N=4000,axis_=[0 1 -.1 1.1],plot_every_nth=10

dx=1/M,dt=Tmax/N,lambda=dt/dx,rho=eps/dx
A_func=@(x)1.1+sin(2*pi*x/eps)
%A_func=@(x)repmat(c^2,size(x));
u0_func=@(x)exp(-100*(mod(x,1)-0.5).^2);
u1_func=@(x)zeros(size(x));
%ue_func=@(x,t)0.5*(u0_func(x+c*t)+u0_func(x-c*t));
x=(0:M)'*dx;xi=x(2:end);xm=(1:2:2*M-1)'*(dx/2);

I=speye(M,M);
Dm=spfd(M,[ 0 -1],[ 1 -1]);
Dp=spfd(M,[ 1  0],[ 1 -1]);
Aop=spdiags(A_func(xm),0,M,M);
Fop=Aop*Dm;
Pdown=[sparse(M,1) speye(M,M)];
Pup=[sparse(1,M-1) 1; speye(M,M)];
p=[1 M+1 2*M+1];q=[M 2*M 3*M];
pe=[1 M+2 2*M+2];qe=[M+1 2*M+2 3*M+1];

u=Pup*u0_func(xi);
f=1/dx*Fop*(Pdown*u);
uold=Pup*(u0_func(xi)-dt*u1_func(xi)+(dt^2/(2*dx))*Dp*f);
y=[u;uold;f];

A=sparse(3*M,3*M);B=speye(3*M,3*M);
Ppre=blkdiag(Pdown,Pdown,I);
Ppost=blkdiag(Pup,Pup,I);
B(p(3):q(3),p(1):q(1))=-(1/dx)*Fop;
A(p(1):q(1),p(1):q(1))=2*I;
A(p(1):q(1),p(2):q(2))=-I;
A(p(1):q(1),p(3):q(3))=(dt^2/dx)*Dp;
A(p(2):q(2),p(1):q(1))=I;

for n=1:N
    y=Ppost*(B\(A*(Ppre*y)));
    if (mod(n,plot_every_nth)==0)||(n==N)
        plot(x,y(pe(1):qe(1)))
        axis(axis_),drawnow
    end
end
```

Table 3: *An illustration of how we setup and use the matrix formulation in Section 2.3.*

```
M := 100;
rho := y -> 1;
mu := y -> 11/10 + sin(y);

dx := 2*Pi/M:
avg := (f::procedure) -> ( 1/(2*Pi)*integrate(f(x),x=0..2*Pi) ):
inv := (f::procedure) -> ( x -> 1/f(x) ):

for m from 1 to M do
    x0 := (m-1/2)*dx:
    c2 := 1/(avg(rho)*avg(inv(mu))):
    Omega1 := sqrt(c2):
    p1 := Integrate(rho(y)*Integrate(inv(mu)(s)*
          Integrate(rho(r),r=0..s),s=0..y),y=0..2*Pi):
    p2 := Integrate(inv(mu)(y)*Integrate(rho(s)*
          Integrate(inv(mu)(r),r=0..s),s=0..y),y=0..2*Pi):
    p3 := Integrate(rho(y)*Integrate(inv(mu)(r),r=0..y),y=0..2*Pi):
    d := 3*Omega1^3/(2*Pi^3) * p1
       + 3*Omega1^5*avg(rho)^2/(2*Pi^3) * p2
       - 1/avg(inv(mu))*(1/Omega1 + 3*Omega1/Pi^2 * p3
                                  - 3*Omega1^3/(4*Pi^4) * p3^2):
    Omega3 := Pi^2 * d / avg(rho):
    alpha[m] := evalf[17](Omega1^2);
    beta[m] := evalf[10](-Omega1*Omega3/(3*2^2*Pi^2));
end do:

f := fopen("alpha_beta.mat", WRITE, TEXT):

for m from 1 to M do
    fprintf(f,"%25.16E%25.16E%25.15E\n",
            (m-1/2)*dx,alpha[m],beta[m]):
end do:

fclose(f):
```

Table 4: *Maple code for computing the coefficients in the effective equation. The function ρ is an "undocumented feature". Anything but $\rho \equiv 1$ should be considered untested. In our experiments we have always used $\rho(x) = 1$. Note that the period of ρ and μ is assumed to be 2π-periodic in y, not 1.*

```
# assume K(x) := (1-x^2)^(q+1)*P(x) where
# P(x) = a1*x^(p-1) + a2 x^(p-2) + ... + ap
for p in range(1,20,2):
    for q in range(-1,20,2):
        A = matrix(QQ,p,p)
        rhs = vector(QQ,p)
        for i in range(0,p):
            for j in range(0,p):
                expr = (x^i*(1-x^2)^(q+1)*x^(p-1-j))
                A[i,j] = expr.integrate(x,-1,1)
        rhs[0] = 1
        unknowns = A\rhs
        name = ("olofrkern_p=%i_q=%i.mat" % (p,q))
        f = open(name,"w+")
        for i in range(0,p):
            str = "{0:25.16E}\n".format(float(unknowns[i]))
            f.write(str)
        f.close()
        print "done with: p=",p,",q=",q
```

Table 5: *Sage code to compute the coefficients of* $P(x)$ *in the kernel* $K(x) = (1-x^2)^{q+1}P(x) \in \mathbb{K}^{p,q}$. *The coefficients are stored in Matlab ordering, i.e., coefficients of the polynomial comes in descending powers.*

6 Numerical schemes

Here we give a detailed description of the finite difference methods used for the macro and micro solvers used to solve wave propagation problems. The solvers are designed for one, two and three dimensions in finite time. We also describe a second order accurate scheme used for long time wave propagation problems. This scheme better captures the dispersive effects that are important in such problems. We have chosen to present the schemes in a HMM method context where we have a macro solver with an unknown flux f (computed via micro problems) and micro solver with a known flux $f = A^\varepsilon \nabla u$. The macro solvers can also be used as a solver when f is in fact known. In the case of finite time problems (HOM class) we have $f = \bar{A}u$ and for the effective equation for long time (EFF class) we have $f = \bar{A}u_x + \beta\varepsilon^2 u_{xxx}$.

6.1 1D scheme

The finite difference scheme on the macro level has the form

$$\begin{cases} U_m^{n+1} = 2U_m^n - U_m^{n-1} + K^2 Y_m^n, \\ Y_m^n = \dfrac{1}{H}\left(F^n_{m+\frac{1}{2}} - F^n_{m-\frac{1}{2}}\right), \\ F^n_{m\pm 1/2} = F(x_{m\pm 1/2}, P^n_{m\pm 1/2}), \end{cases}$$

where $P^n_{m-1/2} = \frac{1}{H}\left(U_m^n - U_{m-1}^n\right)$. The micro level scheme defined analogously:

$$\begin{cases} u_m^{n+1} = 2u_m^n - u_m^{n-1} + k^2 y_m^n, \\ y_m^n = \dfrac{1}{h}\left(f^n_{m+1/2} - f^n_{m-1/2}\right), \\ f^n_{m-1/2} = a_{m-\frac{1}{2}} \dfrac{u^n_{m+1} - u^n_m}{h}, \\ f^n_{m+1/2} = a_{m-\frac{1}{2}} \dfrac{u^n_m - u^n_{m-1}}{h}. \end{cases}$$

6.2 2D scheme

A two dimensional problem is discretized with the following schemes: The finite difference scheme on the macro level

$$\begin{cases} U_m^{n+1} = 2U_m^n - U_m^{n-1} + K^2 Y_m^n, \\ Y_m^n = \dfrac{1}{H}\left(F^{(1)}_{m+\frac{1}{2}e_1} - F^{(1)}_{m-\frac{1}{2}e_1}\right) + \dfrac{1}{H}\left(F^{(2)}_{m+\frac{1}{2}e_2} - F^{(2)}_{m-\frac{1}{2}e_2}\right), \\ F^n_{m\pm\frac{1}{2}e_k} = F(x_{m\pm\frac{1}{2}e_k}, P^n_{m\pm\frac{1}{2}e_k}), \end{cases} \tag{10}$$

where $P^n_{m+\frac{1}{2}e_k}$ is given by (see Figure 3)

$$P^n_{m+\frac{1}{2}e_1} = \left[\frac{1}{H}\left(U_{m+e_1} - U_m\right) \quad \frac{1}{2H}\left(\frac{U_{m+e_2}+U_{m+e_2+e_1}}{2} - \frac{U_{m-e_2}+U_{m-e_2+e_1}}{2}\right) \quad \right]^T,$$
$$P^n_{m+\frac{1}{2}e_2} = \left[\frac{1}{2H}\left(\frac{U_{m+e_1}+U_{m+e_1+e_2}}{2} - \frac{U_{m-e_1}+U_{m-e_1+e_2}}{2}\right) \quad \frac{1}{H}\left(U_{m+e_2} - U_m\right)\right]^T.$$

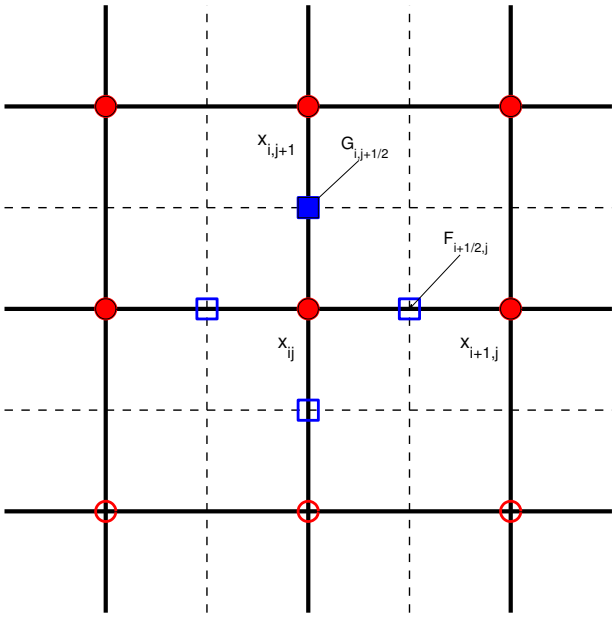

Figure 3: *The numerical scheme* (10) *for* P *in two dimensions. The two components of* f *in two different grid points are given by* $F_{i+1/2,j}$ *and* $G_{i,j+1/2}$*. The* U *points involved in computing* $F^n_{m+\frac{1}{2}e_2} = G_{i,j+1/2}$ *and* $\nabla u \approx P^n_{m+\frac{1}{2}e_2}$ *are indicated by filled circles. Note that the squares are where either* $(A^\varepsilon_{11}\partial_x + A^\varepsilon_{12}\partial_y)u$ *or* $(A^\varepsilon_{21}\partial_x + A^\varepsilon_{22}\partial_y)u$ *are computed, and not the full gradient* $A^\varepsilon \nabla u$*.*

The micro level scheme is formulated as

$$
\begin{cases}
u^{n+1}_m = 2u^n_m - u^{n-1}_m + k^2 y^n_m, \\
y^n_m = \dfrac{1}{h}\Big(f^{(1)}_{m+\frac{1}{2}e_1} - f^{(1)}_{m-\frac{1}{2}e_1}\Big) + \dfrac{1}{h}\Big(f^{(2)}_{m+\frac{1}{2}e_2} - f^{(2)}_{m-\frac{1}{2}e_2}\Big), \\
f^{(1)}_{m+\frac{1}{2}e_1} = \dfrac{a^{(11)}_{m+\frac{1}{2}e_1}}{h}\Big(u^n_{m+e_1} - u^n_m\Big) + \dfrac{a^{(12)}_{m+\frac{1}{2}e_1}}{2h}\Big(\dfrac{u^n_{m+e_2} + u^n_{m+e_1+e_2}}{2} - \dfrac{u^n_{m-e_2} + u^n_{m+e_1-e_2}}{2}\Big), \\
f^{(1)}_{m-\frac{1}{2}e_1} = \dfrac{a^{(11)}_{m-\frac{1}{2}e_1}}{h}\Big(u^n_{m} - u^n_{m-e_1}\Big) + \dfrac{a^{(12)}_{m-\frac{1}{2}e_1}}{2h}\Big(\dfrac{u^n_{m+e_2} + u^n_{m-e_1+e_2}}{2} - \dfrac{u^n_{m-e_2} + u^n_{m-e_1-e_2}}{2}\Big), \\
f^{(2)}_{m+\frac{1}{2}e_2} = \dfrac{a^{(21)}_{m+\frac{1}{2}e_2}}{2h}\Big(\dfrac{u^n_{m+e_1} + u^n_{m+e_1+e_2}}{2} - \dfrac{u^n_{m-e_1} + u^n_{m-e_1+e_2}}{2}\Big) + \dfrac{a^{(22)}_{m+\frac{1}{2}e_2}}{h}\Big(u^n_{m+e_2} - u^n_m\Big), \\
f^{(2)}_{m-\frac{1}{2}e_2} = \dfrac{a^{(21)}_{m-\frac{1}{2}e_2}}{2h}\Big(\dfrac{u^n_{m+e_1} + u^n_{m+e_1-e_2}}{2} - \dfrac{u^n_{m-e_1} + u^n_{m-e_1-e_2}}{2}\Big) + \dfrac{a^{(22)}_{m-\frac{1}{2}e_2}}{h}\Big(u^n_{m} - u^n_{m-e_2}\Big).
\end{cases}
$$

When approximating $f^{(2)}_{m-\frac{1}{2}e_1}$ we take the average of $u^n_{m\pm e_2}$ and $u^n_{m+e_1\pm e_2}$ to approximate $u(x_{m+\frac{1}{2}e_1\pm e_2}, t_n)$. Then we use those two averages to approximate the y derivate of u at $u(x_{m-\frac{1}{2}e_1}, t_n)$. The scheme is second order in both space and time.

6.3 3D scheme

The macro scheme for the three dimensional problem is on the form

$$\begin{cases} U_m^n = 2U_m^n - U_m^{n-1} + K^2 Y_m^n, \\ Y_m^n = \dfrac{1}{H}\left(F^{(1,n)}_{m+\frac{1}{2}e_1} - F^{(1,n)}_{m-\frac{1}{2}e_1}\right) + \dfrac{1}{H}\left(F^{(2,n)}_{m+\frac{1}{2}e_2} - F^{(2,n)}_{m-\frac{1}{2}e_2}\right) + \dfrac{1}{H}\left(F^{(3,n)}_{m+\frac{1}{2}e_3} - F^{(3,n)}_{m-\frac{1}{2}e_3}\right), \\ F^n_{m\pm\frac{1}{2}e_k} = F(x_{m\pm\frac{1}{2}e_k}, P^n_{m\pm\frac{1}{2}e_k}), \end{cases}$$

where $P^n_{m+\frac{1}{2}e_k}$ is defined as,

$$P^n_{m+\frac{1}{2}e_1} = \begin{bmatrix} \frac{1}{H}\left(U_{m+e_1} - U_m\right) \\ \frac{1}{2H}\left(\frac{U_{m+e_2}+U_{m+e_2+e_1}}{2} - \frac{U_{m-e_2}+U_{m-e_2+e_1}}{2}\right) \\ \frac{1}{2H}\left(\frac{U_{m+e_3}+U_{m+e_3+e_1}}{2} - \frac{U_{m-e_3}+U_{m-e_3+e_1}}{2}\right) \end{bmatrix},$$

$$P^n_{m+\frac{1}{2}e_2} = \begin{bmatrix} \frac{1}{2H}\left(\frac{U_{m+e_1}+U_{m+e_1+e_2}}{2} - \frac{U_{m-e_1}+U_{m-e_1+e_2}}{2}\right) \\ \frac{1}{H}\left(U_{m+e_2} - U_m\right) \\ \frac{1}{2H}\left(\frac{U_{m+e_3}+U_{m+e_3+e_2}}{2} - \frac{U_{m-e_3}+U_{m-e_3+e_2}}{2}\right) \end{bmatrix},$$

$$P^n_{m+\frac{1}{2}e_3} = \begin{bmatrix} \frac{1}{2H}\left(\frac{U_{m+e_1}+U_{m+e_1+e_3}}{2} - \frac{U_{m-e_1}+U_{m-e_1+e_3}}{2}\right) \\ \frac{1}{2H}\left(\frac{U_{m+e_2}+U_{m+e_2+e_3}}{2} - \frac{U_{m-e_2}+U_{m-e_2+e_3}}{2}\right) \\ \frac{1}{H}\left(U_{m+e_3} - U_m\right) \end{bmatrix}.$$

The micro level scheme is a second order accurate scheme defined analogous with the 2D scheme (6.2)

$$
\begin{cases}
u_m^{n+1} = 2u_m^n - u_m^{n-1} + k^2 y_m^n \\
y_m^n = \frac{1}{h}\Big(f^{(1)}_{m+\frac{1}{2}e_1} - f^{(1)}_{m-\frac{1}{2}e_1}\Big) + \frac{1}{h}\Big(f^{(2)}_{m+\frac{1}{2}e_2} - f^{(2)}_{m-\frac{1}{2}e_2}\Big) + \frac{1}{h}\Big(f^{(3)}_{m+\frac{1}{2}e_3} - f^{(3)}_{m-\frac{1}{2}e_3}\Big) \\
f^{(1)}_{m+\frac{1}{2}e_1} = \frac{a^{(11)}_{m+\frac{1}{2}e_1}}{h}\Big(u^n_{m+e_1} - u^n_m\Big) + \frac{a^{(12)}_{m+\frac{1}{2}e_1}}{2h}\Big(\frac{u^n_{m+e_1+e_2}+u^n_{m+e_2}}{2} - \frac{u^n_{m+e_1-e_2}+u^n_{m-e_2}}{2}\Big) \\
\qquad\qquad + \frac{a^{(13)}_{m+\frac{1}{2}e_1}}{2h}\Big(\frac{u^n_{m+e_1+e_3}+u^n_{m+e_3}}{2} - \frac{u^n_{m+e_1-e_3}+u^n_{m+e_3}}{2}\Big) \\
f^{(1)}_{m-\frac{1}{2}e_1} = \frac{a^{(11)}_{m-\frac{1}{2}e_1}}{h}\Big(u^n_m - u^n_{m-e_1}\Big) + \frac{a^{(12)}_{m-\frac{1}{2}e_1}}{2h}\Big(\frac{u^n_{m+e_2}+u^n_{m-e_1+e_2}}{2} - \frac{u^n_{m-e_2}+u^n_{m-e_1-e_2}}{2}\Big) \\
\qquad\qquad + \frac{a^{(13)}_{m-\frac{1}{2}e_1}}{2h}\Big(\frac{u^n_{m+e_3}+u^n_{m-e_1+e_3}}{2} - \frac{u^n_{m-e_3}+u^n_{m-e_1-e_3}}{2}\Big) \\
f^{(2)}_{m+\frac{1}{2}e_2} = \frac{a^{(21)}_{m+\frac{1}{2}e_2}}{2h}\Big(\frac{u^n_{m+e_1+e_2}+u^n_{m+e_1}}{2} - \frac{u^n_{m-e_1+e_2}+u^n_{m-e_1}}{2}\Big) + \frac{a^{(22)}_{m+\frac{1}{2}e_2}}{h}\Big(u^n_{m+e_2} - u^n_m\Big) \\
\qquad\qquad + \frac{a^{(23)}_{m+\frac{1}{2}e_2}}{2h}\Big(\frac{u^n_{m+e_2+e_3}+u^n_{m+e_3}}{2} - \frac{u^n_{m+e_2-e_3}+u^n_{m-e_3}}{2}\Big) \\
f^{(2)}_{m-\frac{1}{2}e_2} = \frac{a^{(21)}_{m-\frac{1}{2}e_2}}{2h}\Big(\frac{u^n_{m+e_1}+u^n_{m+e_1-e_2}}{2} - \frac{u^n_{m-e_1}+u^n_{m-e_1-e_2}}{2}\Big) + \frac{a^{(22)}_{m-\frac{1}{2}e_2}}{h}\Big(u^n_m - u^n_{m-e_2}\Big) \\
\qquad\qquad + \frac{a^{(23)}_{m-\frac{1}{2}e_2}}{2h}\Big(\frac{u^n_{m+e_3}+u^n_{m-e_2+e_3}}{2} - \frac{u^n_{m-e_3}+u^n_{m-e_2-e_3}}{2}\Big) \\
f^{(3)}_{m+\frac{1}{2}e_3} = \frac{a^{(31)}_{m+\frac{1}{2}e_3}}{2h}\Big(\frac{u^n_{m+e_1+e_3}+u^n_{m+e_1}}{2} - \frac{u^n_{m-e_1+e_3}+u^n_{m-e_1}}{2}\Big) \\
\qquad + \frac{a^{(32)}_{m+\frac{1}{2}e_3}}{2h}\Big(\frac{u^n_{m+e_2+e_3}+u^n_{m+e_2}}{2} - \frac{u^n_{m-e_2+e_3}+u^n_{m-e_2}}{2}\Big) + \frac{a^{(33)}_{m+\frac{1}{2}e_3}}{h}\Big(u^n_{m+e_3} - u^n_m\Big) \\
f^{(3)}_{m-\frac{1}{2}e_3} = \frac{a^{(31)}_{m-\frac{1}{2}e_3}}{2h}\Big(\frac{u^n_{m+e_1}+u^n_{m+e_1-e_3}}{2} - \frac{u^n_{m-e_1}+u^n_{m-e_1-e_3}}{2}\Big) \\
\qquad + \frac{a^{(32)}_{m-\frac{1}{2}e_3}}{2h}\Big(\frac{u^n_{m+e_2}+u^n_{m+e_2-e_3}}{2} - \frac{u^n_{m-e_2}+u^n_{m-e_2-e_3}}{2}\Big) + \frac{a^{(33)}_{m-\frac{1}{2}e_3}}{h}\Big(u^n_m - u^n_{m-e_3}\Big)
\end{cases}
$$

6.4 1D scheme for long time

The finite difference scheme on the macro level

$$
\begin{cases}
U_m^{n+1} = 2U_m^n - U_m^{n-1} + K^2 Y_m^n, \\
Y_m^n = \dfrac{1}{H}\left(F^n_{m+\frac{1}{2}} - F^n_{m-\frac{1}{2}}\right), \\
F^n_{m\pm 1/2} = F(x_{m\pm 1/2}, P^n_{m\pm 1/2}, Q^n_{m\pm 1/2}, R^n_{m\pm 1/2}),
\end{cases}
$$

where P, Q and R are defined via U,

$$\begin{cases} P^n_{m-1/2} = \dfrac{-U_{m+1} + 27U_m - 27U_{m-1} + U_{m-2}}{24H} = U_x(x_{m-1/2}, t_n) + \mathcal{O}(H^4), \\ Q^n_{m-1/2} = \dfrac{U_{m+1} - U_m - U_{m-1} + U_{m-2}}{2H^2} = U_{xx}(x_{m-1/2}, t_n) + \mathcal{O}(H^2), \\ R^n_{m-1/2} = \dfrac{U_{m+1} - 3U_m + 3U_{m-1} - U_{m-2}}{H^3} = U_{xxx}(x_{m-1/2}, t_n) + \mathcal{O}(H^2). \end{cases}$$

The micro level scheme has better dispersive properties than the normal two point divergence approximation, cf. Section 2.2,

$$\begin{cases} u^{n+1}_m = 2u^n_m - u^{n-1}_m + k^2 y^n_m, \\ y^n_m = \dfrac{1}{24h}\Big(-f^n_{m-\frac{3}{2}} + 27f^n_{m-\frac{1}{2}} - 27f^n_{m+\frac{1}{2}} + f^n_{m+\frac{3}{2}}\Big), \\ f^n_{m+\frac{3}{2}} = \dfrac{a_{m+\frac{3}{2}}}{24h}\Big(-u^n_{m+3} + 27u_{m+2} - 27u_{m+1} + u_m\Big), \\ f^n_{m+\frac{1}{2}} = \dfrac{a_{m+\frac{1}{2}}}{24h}\Big(-u^n_{m+2} + 27u_{m+1} - 27u_m + u_{m-1}\Big), \\ f^n_{m-\frac{1}{2}} = \dfrac{a_{m-\frac{1}{2}}}{24h}\Big(-u^n_{m+1} + 27u_m - 27u_{m-1} + u_{m-2}\Big), \\ f^n_{m-\frac{3}{2}} = \dfrac{a_{m-\frac{3}{2}}}{24h}\Big(-u^n_m + 27u_{m-1} - 27u_{m-2} + u_{m-3}\Big). \end{cases}$$

References

[1] Doina Cioranescu and Patrizia Donato. *An Introduction to Homogenization*. Number 17 in Oxford Lecture Series in Mathematics and its Applications. Oxford University Press Inc., 1999.

[2] Björn Engquist, Henrik Holst, and Olof Runborg. Multiscale methods for the wave equation. In *Sixth International Congress on Industrial Applied Mathematics (ICIAM07) and GAMM Annual Meeting*, volume 7. Wiley, 2007.

[3] Björn Engquist, Henrik Holst, and Olof Runborg. Analysis of HMM for one dimensional wave propagation problems over long time, 2011.

[4] Björn Engquist, Henrik Holst, and Olof Runborg. Multiscale methods for one dimensional wave propagation with high frequency initial data. Technical report, School of Computer Science and Communication, KTH, 2011. TRITA-NA 2011:7.

[5] Björn Engquist, Henrik Holst, and Olof Runborg. Multiscale methods for the wave equation. *Comm. Math. Sci.*, 9(1):33–56, Mar 2011.

[6] Björn Engquist, Henrik Holst, and Olof Runborg. Multiscale methods for wave propagation in heterogeneous media over long time. In Björn Engquist, Olof Runborg, and Richard Tsai, editors, *Numerical Analysis and Multiscale Computations*, volume 82 of *Lect. Notes Comput. Sci. Eng.*, pages 167–186. Springer Verlag, 2011.

[7] Agnes Lamacz. Dispersive effective models for waves in heterogeneous media. *Math. Models Methods Appl. Sci.*, 21:1871–1899, 2011.

[8] R.J. LeVeque. *Numerical Methods for Conservation Laws.* Lecture Notes in Mathematics. Birkhäuser Verlag AG, 1994. ISBN 9783764327231.

[9] Maplesoft. *Maple 12 User Manual*, 2008.

[10] MATLAB. *7.12.0.635 (R2011a).* The MathWorks Inc., Natick, Massachusetts, 2011.

[11] Fadil Santosa and William W. Symes. A dispersive effective medium for wave propagation in periodic composites. *SIAM J. Appl. Math.*, 51(4): 984–1005, 1991. ISSN 0036-1399.

[12] W. A. Stein *et al. Sage Mathematics Software (Version 4.2.1).* The Sage Development Team, 2009. `http://www.sagemath.org`.

[13] John C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations (2nd ed.).* SIAM, 2004. ISBN 0898715679.