Asynchronous finite-difference schemes for partial differential equations

Original paper by Diego A. Donzis & Kunduri Aditya
Asynchronous finite-difference schemes for partial differential equations
Journal of Computational Physics, accepted June 2014[1]
CES Seminar paper by Thomas Camminady*

January 12, 2015

Abstract

The numerical solution of partial differential equations in large-scale problems might require the usage of parallel computing. In this context, the computational domain is often split into smaller subdomains that are handled by seperate processors. Due to the non-local nature of the underlying problem, communication as well as synchronization between these processing elements (PEs) is required. Since the discretization by finite-difference schemes results in mostly simple arithmetic operations, the overhead due to communication and synchronization might become a bottleneck. In the original paper, Donzis and Aditya present a way to avoid the overhead in parallel computing that is required for the synchronization of finite-difference schemes. This is done by introducing asynchronous finite-difference schemes which avoid the synchroniation at each time layer, thus allowing the PEs to run almost independently. It is proven, that the convergence only depends on the original scheme, however consistency drops to first order. Numerical results are presented.

1 Introduction

Finite difference schemes are one way to solve partial differential equations. Consider for example the one dimensional heat equation in its simplest case

$$\partial_t u(x,t) = \partial_{xx} u(x,t) \tag{1}$$

Discretize the spatial domain by nodes $x_i := i \cdot \Delta x$ and the time domain by nodes $t^n := n \cdot \Delta t$. Then $u_i^n = u(i \cdot dx, n \cdot dt)$ can be used to approximate $u(x,t)$.

*This work is a summary of the original work by Donzis & Kunduri. The presented idea, as well as the theoretical analysis is only summarized. The numerical results are reproduced with the same test cases but a self written implementation. This report was produced for the CES Seminar at RWTH Aachen University.

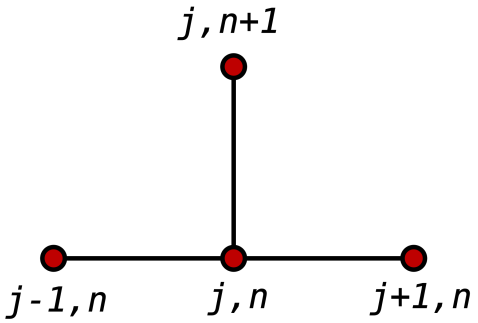


Figure 1: Standard explicit stencil for the heat equation. Image taken from[4]

Using the stencil in Figure 1, i.e. forward finite differences in time and central differences in space, we obtain

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + O(\Delta x^2, \Delta t) \tag{2}$$

Since the error is $O(\Delta x^2, \Delta t)$ we could increase the number of spatial and temporal nodes to increase the accuracy of the solution. The corresponding system can be written in matrix form as $u^{n+1} = Mu^n$ with corresponding matrix $M \in \mathbb{R}^{N_x \times N_x}$. Extending the problem into 3 dimensions the problem size grows since the linear system is of size $N_x^3 \times N_x^3$. Therefore, it might me helpful, to solve these systems on multiple processors. One common approach to split the workload onto multiple processors, is to decompose the computational domain accordingly to the physical domain which can be seen in Figure 2.

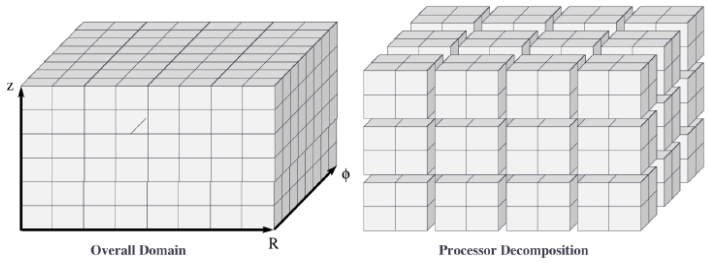


Figure 2: Domain decomposition for the three dimensional case. Split every part of the domain onto different PEs. Image taken from[2]

The problem, that arises in the context of domain decomposition is the need for information exchange on the boundary elements of each PE. Let us consider this problem in greater detail in the simplest case.

In Figure 3 we can see grid points of a one dimensional domain decomposition. To compute a quantity at the next time layer for node i with the stencil used for the heat equation, we need the neighbouring points $i-1$ and $i+1$. In part a), a serial implementation is considered, where all gridpoints are available at the PE. In b) however, we consider the case where we split the domain onto two PEs. PE0 has the information for all nodes up

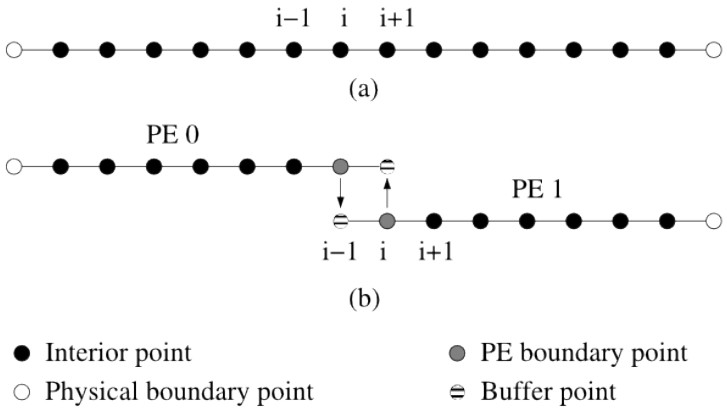


Figure 3: Domain decomposition and the problem with boundary points. Image taken from[1]

to node $i-1$ and in the memory of PE1, we have the nodes starting at i available. To make computation easier, Buffer points might be added at each boundary. If PE0 wants to compute u_{i-1}^{n+1}, the stencil needs the quantities u_{i-2}^{n}, u_{i-1}^{n} and u_{i}^{n}. Since u_{i}^{n} is not available on PE0, it has to communicate with PE1 to exchange the required quantitie. The same holds true for the computation of u_{i}^{n+1} on PE1. This simple example illustrates, that the different PEs can not perform their computation independently from each other, since communication and data exchange between neighbouring PEs is required at each time layer.

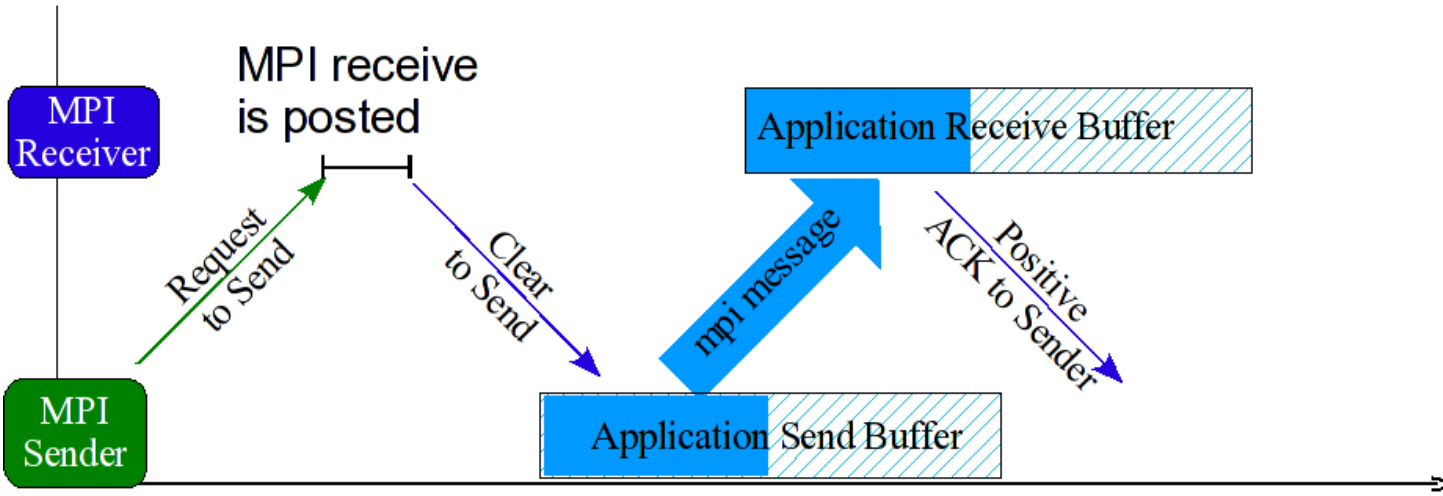


Figure 4: Timeline for MPI Send and MPI Receive. Image taken from[3]

One possible way to implement the data exchange between two PEs would be by using MPI Send and MPI Receive. Figure 4 illustrates the procedure of the two sided communication between the MPI Reciver and the MPI Sender to exchange the MPI message. Since the

process is two sided, a lot of overhead arises due to the request to send and the clearance to send in addition to the actual data transfer.

Furthermore, we do not only have to consider the costs for the data transfer itself, but for the synchronization between all different PEs after each time layer. By the definition of our stencil, all PEs have to compute quantities for the time layer, because we would have asynchronicity between the data that has to be exchanged. Therefore, we have to make sure that we synchronize all PEs after the computation of one time layer, possibly by the use of `MPI_Barrier`. The implementation of `MPI_Barrier` inserts a barrier in the algorithm. A PE can not continue its computation, until all PEs have reached the barrier. Since not all PEs finish the computation of a time layer in the exact same time, this results in possible large idle times for multiple PEs.

By using asynchronous finite-difference schemes, Donzis & Aditya propose a way to overcome this synchronization overhead and reduce the communication. Asynchronous schemes eliminate the synchronization after each time layer by allowing the differen PEs to run asynchronous in time. The question arises, under which conditions these methods are still convergent and whether the accuracy remains the same.

This paper is organised in the following way: In Section 2, we formally define asynchronous finite difference schemes for the heat equation. Section 3 presents the theoretical analysis of the schemes. Convergence criteria are presented and the order of consistency is derived. In Section 5 the implementation of the asynchronous scheme is described and numerical examples are presented, which are in accordance with the theoretical predictions, that where made in Section 3.

2 Asynchronous finite-difference schemes

Equation (2) was the discretized formulation of the heat equation. By omitting the error terms we are left with

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{3}$$

$$\Leftrightarrow u_i^{n+1} = u_i^n + \frac{\Delta t}{\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) \tag{4}$$

$$\Leftrightarrow u_i^{n+1} = \sigma u_{i+1}^n + (1 - 2\sigma)u_i^n + \sigma u_{i-1}^n \tag{5}$$

where $\sigma = \Delta t/\Delta x^2$ is the CFL number. As presented in Figure 3, we run into problems when node i is the leftmost (or rightmost) boundary node on a PE. In this case, node $i-1$ (or $i+1$) is no longer on the PE that computes u_i^{n+1}. Therefore, we will introduce the following notation for the leftmost node on a PE:

$$u_i^{n+1} = \sigma u_{i+1}^n + (1 - 2\sigma)u_i^n + \sigma u_{i-1}^{\tilde{n}} \tag{6}$$

and respectively

$$u_i^{n+1} = \sigma u_{i+1}^{\tilde{n}} + (1 - 2\sigma)u_i^n + \sigma u_{i-1}^n \tag{7}$$

for the rightmost node. Hereby, $\tilde{n}$ is the introduced asynchronicity. In the synchronous case, we have $\tilde{n} = n$, but now we allow $\tilde{n} = n, n-1, n-2, \ldots, n-(L-1)$ to be any of the prior

$L-1$ time layers. Here the concrete value of $\tilde{n}$ depends on the hardware characteristics, the network, workload balancing and additional possibly unpredictable factors. The bound on the maximal delay is needed for the theoretical analysis and the existence of such a bound can be considered a reasonable assumption. It also makes sense, to model $\tilde{n}$ as a discrete random variable that takes values between n and $n-(L-1)$. Let us rewrite $\tilde{n} = n - \tilde{k}$, then the random variable becomes $\tilde{k}$ with probabilities $p(\tilde{k})$ such that $\sum_{\tilde{k}=0}^{L-1} p(\tilde{k}) = 1$.

From the implementation side, we would now remove the `MPI_Barrier` and could possibly replace the two sided `MPI_Send` and `MPI_Recv` by the one sided `MPI_Put` which lets a PE write into the buffer point of a neighbour without synchronization. For the computation of boundary nodes, we would then take whatever value is present in the buffer point.

This algorithm removes the overhead due to synchronization after each time layer. Additionally, the idle time is smaller when using `MPI_Put` than for `MPI_Send` and `MPI_Recv`. However, it is not clear what happens to the convergence and the consistency of the method. This will be discussed in the following section.

3 Theoretical analysis

To prove convergence of the asynchronous scheme, we will start with the synchronous formulation. Let us start with equation (5). We can extend this formulation to compute the u_i^{n+1} at all nodes $i = 1, \ldots, I$ for the time layer $n+1$. For a periodic domain, we can explicitely write

$$\begin{pmatrix} u_1^{n+1} \\ \vdots \\ \vdots \\ u_I^{n+1} \end{pmatrix} = \begin{pmatrix} 1-2\sigma & \sigma & 0 & 0 & \ldots & 0 & 0 & \sigma \\ \sigma & 1-2\sigma & \sigma & 0 & \ldots & 0 & 0 & 0 \\ & & \ddots & & & & & \\ \sigma & 0 & 0 & \ldots & 0 & 0 & \sigma & 1-2\sigma \end{pmatrix} \begin{pmatrix} u_1^{n} \\ \vdots \\ \vdots \\ u_I^{n} \end{pmatrix} \tag{8}$$

or

$$u^{n+1} = Mu^n \tag{9}$$

with M being the matrix of equation (8). Since M is time-independet, we can recursively write

$$u^{n+1} = Mu^n = M^2u^{n-1} = \ldots = M^{n+1}u^0 \tag{10}$$

Thus stability results from $||M|| \leq 1$. Let $||\cdot|| = ||\cdot||_\infty$ by the infinity norm, defined as $||A||_\infty = \max_i \sum_j |A_i j|$. Given the definition of the matrix M, our method is stable, if

$$|\sigma| + |1-2\sigma| + |\sigma| \leq 1 \tag{11}$$

which is fulfilled for $0 \leq \sigma \leq 1/2$. This result is the well known CFL condition for the given problem and its discretization.

3.1 The simplified analysis

We will now consider a simplified case of the asynchronous method. Assume, that our scheme is either snychronized or has delay 1, i.e. $\tilde{n} = n$ or $n-1$. Furthermore we will assume, that we only need data exchange for the rightmost boundary nodes. Additionally we will assume, that the number of PEs is the number of spatial points, which is no realisitic assumption, but does not influence the validity of the proof. Under the given assumtions, we can write

$$\begin{pmatrix} u^{n+2} \\ u^{n+1} \end{pmatrix} = C \begin{pmatrix} u^{n+1} \\ u^{n} \end{pmatrix} \tag{12}$$

where $(u^{n+2}, u^{n+1})^T$ is of dimension $2I \times 1$, as well as $(u^{n+2}, u^n)^T$. The matrix C is of size $2I \times 2I$ and is defined as

$$C = \begin{pmatrix} A_0 & A_1 \\ I & 0 \end{pmatrix} \tag{13}$$

with the identity matrix I and

$$A_0 = \begin{pmatrix} (1-2\sigma) & (1-\tilde{k}_2)\sigma & 0 & \dots & \sigma \\ \sigma & (1-2\sigma) & (1-\tilde{k}_3)\sigma & \dots & 0 \\ \vdots & \ddots & & \vdots & \\ (1-\tilde{k}_1)\sigma & 0 & \dots & \sigma & (1-2\sigma) \end{pmatrix} \tag{14}$$

as well as

$$A_1 = \begin{pmatrix} 0 & \tilde{k}_2\sigma & 0 & \dots & 0 \\ 0 & 0 & \tilde{k}_3\sigma & \dots & 0 \\ \vdots & \ddots & & \vdots & \\ \tilde{k}_1\sigma & 0 & \dots & 0 & 0 \end{pmatrix} \tag{15}$$

And the $\tilde{k}_i \in \{0, 1\}$ which switch between $\tilde{n} = n$ and $\tilde{n} = n-1$.

Given this formulation and the time independency of the matrix C, we can write

$$\begin{pmatrix} u^{n+2} \\ u^{n+1} \end{pmatrix} = C \begin{pmatrix} u^{n+1} \\ u^{n} \end{pmatrix} = \dots = C^{n+1} \begin{pmatrix} u^{1} \\ u^{0} \end{pmatrix} \tag{16}$$

Let us define $W^n = (u^{n+1}, u^n)^T$. For stability, we want $||W^n|| \leq ||W^0||$ for a given norm. Thus, the norm of C has to be bounded by unity. Let us consider the infinity norm, that is $||M||_\infty = \max_i \sum_j |M_{ij}|$. Given the structure of the matrix C, we can deduce, that

$$||C||_\infty \leq 1 \Leftrightarrow |\sigma| + |1-2\sigma| + |(1-\tilde{k}_i)\sigma| + |\tilde{k}_i\sigma| \leq 1 \tag{17}$$

since $\tilde{k}_i$ is either 1 or 0, this simplifies to

$$||C||_\infty \leq 1 \Leftrightarrow |\sigma| + |1-2\sigma| + |\sigma| \leq 1 \tag{18}$$

which is exactly the stability criterion for the synchronous case in equation (11).

Therefore, in this simplified case, the asynchronous scheme is stable, if the initial scheme is stable.

3.2 The general case

First note, that due to the choice of the infinity norm, only those rows that include delay are relevant, because all other rows are bounded by unity due to the assumption that the synchronous scheme is stable.

Each finite difference stencil of size $2S + 1$, that includes only two time layers can be written as

$$u_i^{n+1} = \sum_{j=-S}^{S} c_j u_{i+j}^n \tag{19}$$

Therefore, the stability criteria when using the infinity norm turns out to be

$$\sum_{j=-S}^{S} |c_j| \leq 1 \tag{20}$$

which is a sufficient condition for stability.

We now introduce the asynchronicity by adding an additional sum over all possible delays from 0 to $L-1$. Since only one of these terms is non-zero, we furthermore use the Kronecker delta to accomplish this. Let the Kronecker delta be given by

$$\delta_{k_i,m} = \begin{cases} 1, & \text{iff } k_i = m \\ 0, & \text{else} \end{cases} \tag{21}$$

We can now write

$$u_i^{n+1} = \sum_{m=0}^{L-1} \sum_{j=-S}^{S} c_j \delta_{k_i,m} u_{i+j}^{n-m} \tag{22}$$

Generalizing the formulation of equation (12) to all possible time layers, we can deduce the stability criteria for the general asynchronous case. That is, our scheme is stable, if

$$\sum_{m=0}^{L-1} \sum_{j=-S}^{S} |c_j \delta_{k_i,m}| \leq 1 \tag{23}$$

Since the Kronecker dela is either 0 or 1, and adds up to unity when summed over m we get

$$\sum_{m=0}^{L-1} \sum_{j=-S}^{S} |c_j \delta_{k_i,m}| = \sum_{m=0}^{L-1} \sum_{j=-S}^{S} |c_j| \delta_{k_i,m} \tag{24}$$

$$= \sum_{j=-S}^{S} |c_j| \sum_{m=0}^{L-1} \delta_{k_i,m} \tag{25}$$

$$= \sum_{j=-S}^{S} |c_j| \tag{26}$$

which is exactly the condition for the synchronous scheme to be stable. We see, that the asynchronous scheme is stable, if the synchronous scheme is stable. □

3.3 Remarks

The theoretical analysis, showed that the question of convergence does not depend on the statistics of the delay $\tilde{k}_i$ nor on the number of PEs. The first fact is important, since the $\tilde{k}_i$ are in principle random variables and assumptions on their statistics can not be made in the general case. The latter ensures, that the method is stable independent of the domain decomposition, as well of the underlying architecture. Furthermore, stability is kept when the problem is scaled to a finer grid decomposition and a higher number of PEs.

A further remark should be made, regarding the choice of the stability criteria. In principle, our formulation implies, that the scheme is damped at each single step. However, in practice there exist convergent schemes that include amplification during time layers. The choice of the infinity norm applied to the iteration matrix might therefore be too restrictive.

3.4 Consistency

By performing Taylor expansion, one can easily show, that for the synchronous case and by using the above stencil, the error between the correct solution and the approximated solution can is given by

$$E_i^n = -\frac{u_{tt}}{2}\Delta t + \frac{\alpha u_{xxxx}}{12}\Delta x^2 + O(\Delta t^2, \Delta x^4) \tag{27}$$

Where we extend the original problem $u_t = u_{xx}$ to $u_t = \alpha u_{xx}$ by introducing the diffusion coefficient α. We see, that for $(\Delta t, \Delta x) \to 0$ the error vanishes, therefore, the numerical approximation is consistent with the analytical problem at a given point i with first order in time and second order in space.

Consider now the asynchronous case. For abitrary delay k at grid point $i+1$, we deduce with the help of Taylor expansion

$$\tilde{E}_i^n|_{\tilde{k}_{i+1}=k} = -\frac{u_{tt}}{2}\Delta t + \frac{\alpha u_{xxxx}}{12}\Delta x^2 - \alpha k u_t \frac{\Delta t}{\Delta x^2} + \alpha k u_{tx}\frac{\Delta t}{\Delta x} - \alpha k u_{txx}\frac{\Delta t}{2} \tag{28}$$

$$+ O(\Delta x^3, \Delta t^2, \Delta x^p \Delta t^q) \tag{29}$$

where the parameters p and q depend on the bound on the delay. By introducing the CFL number $\sigma_\alpha = \alpha \Delta t / \Delta x^2$ we can rewrite the equation and get

$$\tilde{E}_i^n|_{\tilde{k}_{i+1}=k} = -\frac{\sigma_\alpha u_{tt}}{2\alpha}\Delta x^2 + \frac{\alpha u_{xxxx}}{12}\Delta x^2 - \sigma_\alpha k u_t + \sigma_\alpha k u_{tx}\Delta x - \sigma_\alpha k u_{txx}\frac{\Delta x^2}{2} \tag{30}$$

$$+ O(\Delta x^3, \Delta t^2, \Delta x^p \Delta t^q) \tag{31}$$

We see, that this scheme is therefore not consistent with the initial scheme, when σ_α is kept constant. Since the error contains zeroth order contributions, the error does not vanish under grid refinement. However, this error analysis is only valid for nodes where two PEs work asynchronous. For the majority of points, this is not the case. It is therefore necessary to perform a statistical error analysis, that includes both, the statistics of the delays as well as the number of PEs.

In the following, let I_B denote the set of boundary nodes and I_I the set of interiror nodes. That is, $i \in I_B$ if we need communication between PEs for the computation of u_i^{n+1}

and $i \in I_I$ otherwise. Clearly, I_B and I_I are disjoint. We introduce the spatial averages $\langle f \rangle_S = 1/N_S \sum_{i \in S} f_i$ for a set S of size N_S. Furthermore, let $\bar{f}$ denote the ensemble average over the statistical properties. The combination of the spatial and ensemble average for a given time layer n is

$$\langle \bar{E} \rangle = \frac{1}{N} \sum_{i=1}^{N} \bar{E}_i^n \tag{32}$$

since we only need to take an ensemble average over nodes that belong to I_B this is equivalent to

$$\langle \bar{E} \rangle = \frac{1}{N} \left[\sum_{i \in I_I} E_i^n + \sum_{i \in I_B} \bar{E}_i^n \right] \tag{33}$$

From equation (27), we know that we have

$$\sum_{i \in I_I} E_i^n \approx \sum_{i \in I_I} \left(-\frac{u_{tt}}{2} \Delta t + \alpha \frac{u_{xxxx}}{12} \Delta x^2 \right) \tag{34}$$

$$= \sum_{i \in I_I} \left(-\frac{u_{tt}}{2\alpha} \sigma_\alpha + \alpha \frac{u_{xxxx}}{12} \right) \Delta x^2 \tag{35}$$

$$= \Delta x^2 \sum_{i \in I_I} K_S \tag{36}$$

$$= \Delta x^2 N_I \langle K_S \rangle_{I_I} \tag{37}$$

The number of interior grid points for P PEs and a stencil of size S under the assumption of onesided delay is given as $N_I = N - SP$

For the computation of the second term in equation (33), we have to take into account the statistical nature of the delays. This rather lengthy computation is performed in great detail in the original paper. The final result is then computed to be

$$\sum_{i \in I_B} \bar{E}_i^n \approx N_B \langle K_s \rangle_{I_B} \Delta x^2 + \left(-\sigma_\alpha N_B \langle u_t \rangle_{I_B} + \sigma_\alpha N_B \langle u_{xt} \rangle_{I_B} \Delta x - \frac{N_B}{2} \langle u_{txx} \rangle_{I_B} \right) \overline{\tilde{k}} \tag{38}$$

Thus the overall statistical error can be given by

$$\langle \bar{E} \rangle = \langle K_S \rangle \Delta x^2 + \frac{N_B}{N} \overline{\tilde{k}} \left(-\sigma_\alpha N_B \langle u_t \rangle_{I_B} + \sigma_\alpha N_B \langle u_{xt} \rangle_{I_B} \Delta x - \frac{N_B}{2} \langle u_{txx} \rangle_{I_B} \right) \tag{39}$$

We see, that for the synchronous case, $\overline{\tilde{k}} = 0$ and we have second order consistency in space. However, as soon as we consider the asynchronous case, for $\Delta x \to 0$, we are left with

$$\langle \bar{E} \rangle \approx -\frac{N_B}{N} \overline{\tilde{k}} \sigma_\alpha \langle u_t \rangle_{I_B} = -S \frac{P}{N} \overline{\tilde{k}} \sigma_\alpha \langle u_t \rangle_{I_B} \tag{40}$$

And therefore, with all other parameters kept constant,

$$\langle \bar{E} \rangle \in O(P \overline{\tilde{k}} \Delta x) \tag{41}$$

and the original scheme drops to first order consistency in space, when the number of PEs is constant. However, for the case of weak scaling, where P/N is kept constant, the error is of zeroth order and the scheme becomes inconsistent.

It is important to remark, the the asynchronous scheme will always drop to first order consistency, regardless of the initial synchronous scheme.

4 Implementation and numerical results

To reproduce the numerical results a C-code was written to simulate the problem. Since the authors focussed on the theoretical analysis and did not consider a performance analysis, we decided to artifically simulate the delay and the communication between different PEs to gain full control over the statistial nature of the delays. The code uses MPI to communicate between the different PEs. However the communication is synchronous. To allow an analysis of the influence of the delay, we let each PE compute the required quantities on its own and only perform communication between the PEs after a given number of time layers haven been computed asynchronous. Let `lag` denote the variable for number of time layers between synchronization, then the time loop of the algorithm that is performed by each PE can be given in pseudo code as follows:

```
for (t =0;t<NT;t++){
   if (t%lag==0 ){
      MPI_Send(...);  // send boundary nodes
      MPI_Recv(...);  // receive boundary nodes
   }
   for(i=0;i<n;i++){
      u[i] = ...;  // perform the computation for all nodes on the PE
   }
}
```

The testproblem is the advection-diffusion equation

$$u_t + cu_x = \alpha u_{xx} \tag{42}$$

with $u = u(t, x)$. Further we assume a periodic domain of length 2π and the initial condition are superimposed sinusoidal waves, i.e.

$$u(x, 0) = \sum_{\kappa} A(\kappa) \sin(\kappa x \phi_\kappa) \tag{43}$$

which has the analytical solution

$$u(x, t) = \sum_{\kappa} e^{-\alpha\kappa^2 t} A(\kappa) \sin(\kappa x \phi_\kappa - ct) \tag{44}$$

which means, that the waves are shifted by $-ct$ and damped by $\exp(-\alpha\kappa^2 t)$. The advection-diffusion equation is discretized for all interior nodes by

$$\frac{1}{\Delta t}(u_i^{n+1} - u_i^n) + \frac{c}{\Delta x}(u_{i+1}^n - u_{i-1}^n) = \frac{\alpha}{\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) \tag{45}$$

and for left boundary nodes

$$\frac{1}{\Delta t}(u_i^{n+1} - u_i^n) + \frac{c}{\Delta x}(u_{i+1}^n - u_{i-1}^{\tilde{n}}) = \frac{\alpha}{\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^{\tilde{n}}) \tag{46}$$

and analogously for right boundary nodes.

During the numerical experiment, we vary the number of PEs, the statistics of the delay as well as the number of grid points. We keep σ_α constant and fix T_{end}.

The results can be seen in figures 5 to 8b. In figures 6a and 6b the number of PEs is fixed to 4 and 16, respectively. For each number of PEs we simulate the case where the delay is in $\{0, 2, 4, 6, 8\}$, meaning that we have a synchronous simulation if the delay is zero and are asynchronous in all other cases. In the log-log plot in figure 5, we can see, that we have second order convergence in space for the synchronous simulation and drop to first order convervence for the delay, independently of the number of PEs or the magnitude of the delay. However, higher delay results in a larger (but still first order) error. Figures 7a and 7b keep the number of grid points fixed at 512 and 128, respectively. For each case, we perform the simulation for a different number of PEs and vary the delay for each scenario. We observe the same phenomenological behavior, independent of the number of grid points. As expected, the error is the same for all number of PEs if the delay is zero. This is required, since we basically perform a synchronous simulation that performs communication between PEs after each step and the error is therefore independent of the number ofPEs. As soon, as we increase the delay, we can observe two things. First, the higher the delay, the larger the error, which is plausible since a higher delay results in increase number of steps where the computation takes place asynchronously. Second, we see that the error is larger for a greater number of PEs. This is in agreement with $\langle \bar{E} \rangle \in O(P\Delta x)$. In figures 8a and 8b the delay is fixed to 8 and 4 and the simulation are performed for different number of PEs with varying number of grid points. Here we again observe first order convergence for all number of PEs and an error increase when the number of PEs increases.

Summarizing the results, we can see that the numerical experiments are in agreement with the theoretical predictions from the prior sections. The second order method in space for the synchronous case drops to first order convergence as soon as delay is introduced. For all other parameters fixed, the error grows with an increase of the number of PEs (except for the synchronous case). Furthermore, with again all other parameters fixed, the error grows with an increase of the delay.

We would like to remark, that even though the theoretical analysis has been taken from the original paper by Donzis and Aditya, the numerical results are produced by our own simulation and agree with the numerical results that are presented by Donzis and Aditya.

5 Conclusion

In the context of parallel computing, the problem of communication is becoming a bottleneck in the efficiency of algorithms. Domain decomposition for finite difference schemes for partial differential equations require communication at the domain boundaries between processors. Since especially for simple computations, the costs of communication outweigh those for arithmetic operations, it is desirable to reduce communication as much as possible. One

way to accomplish this goal is by using asynchronous schemes. Instead of synchronization at each time layer, it is possible that processors compute independently from each other in an asynchronous fashion. The question remains, whether convergence and consistency behavior of the original method changes when it is no longer synchronous.

In this paper, the idea of asynchronous finite difference schemes is presented. The requirement for synchronization at boundary nodes is removed and the computation can continue asynchronous. In section (3) we used the matrix method together with the inifinity norm, to prove, that the asynchronous scheme is stable, if the original scheme is stable. This result was independent of the characteristics of the delay, the number of PEs and the way, the domain is decomposed. In section (3.4) we derived an averaged error description for the consistency of the asynchronous method. We were able to show, that any higher order method drops to first order consistency in the asynchronous case. The averaged error description is required due to the statistical nature of the delays, as well as the spatial position of the boundary nodes. We derived $\langle \bar{E} \rangle \in O(\overline{\tilde{k}} P \Delta x)$, which shows, that the error is of first order in space and proportional to the number of PEs and proportional to the average delay which is in principle a random variable. Numerical experiments where reproduced on the problem description of Donzis and Aditya and are in agreement with the theoretical description of section (3.4), as well as with the original results.

Even though, the analysis was restricted to the one-dimensional case, the analysis can be extended towards multi-dimensional problems.

The question remains, whether the drop to first order consistency can be compensated by the faster computation due to the loosened requirements on processor communication given a speficic implementation on an existing architecture layout.

References

[1] Diego A. Donzis and Konduri Aditya. Asynchronous finite-difference schemes for partial differential equations. *Journal of Computational Physics*, 274(0):370 – 392, 2014.

[2] Dan Reynolds. SMU HPC Winter Workshop. In *Introduction to Parallel Computing*, 2014.

[3] Michael E. Thomadakis. Hardware and software architecture operations and optimizations for parallel message-passing applications. 2011.

[4] Wikipedia. Finite difference method, 2014. [Online; accessed 21-Dec-2014].

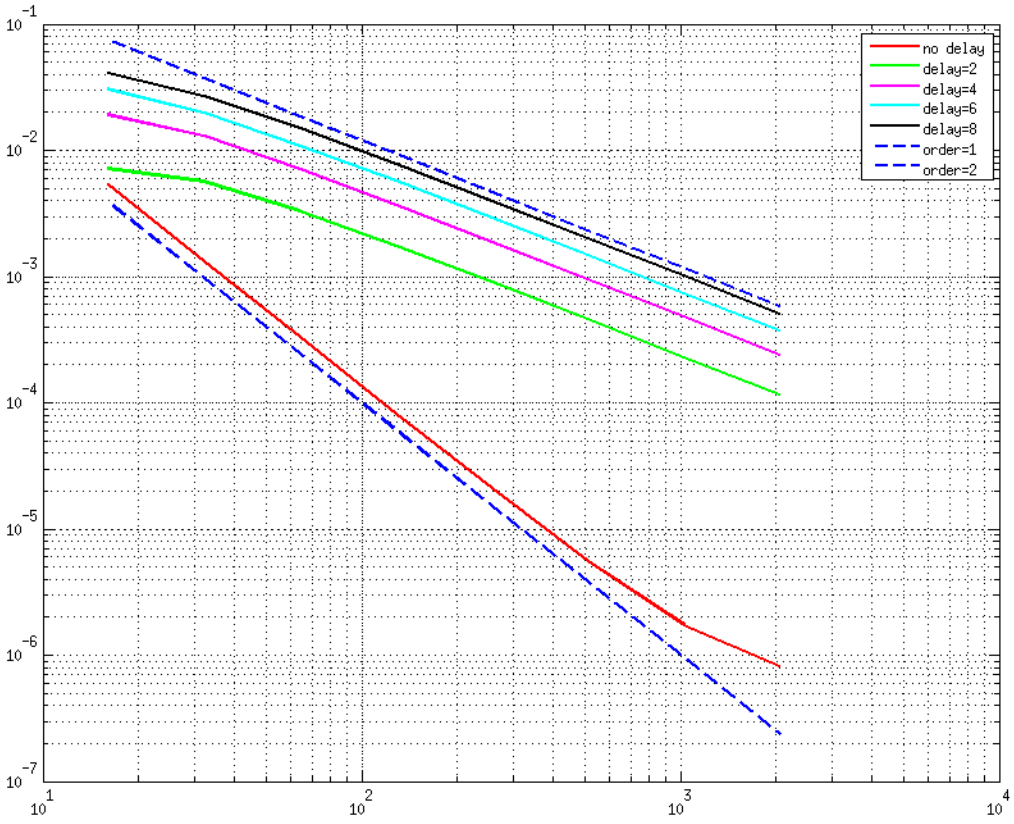
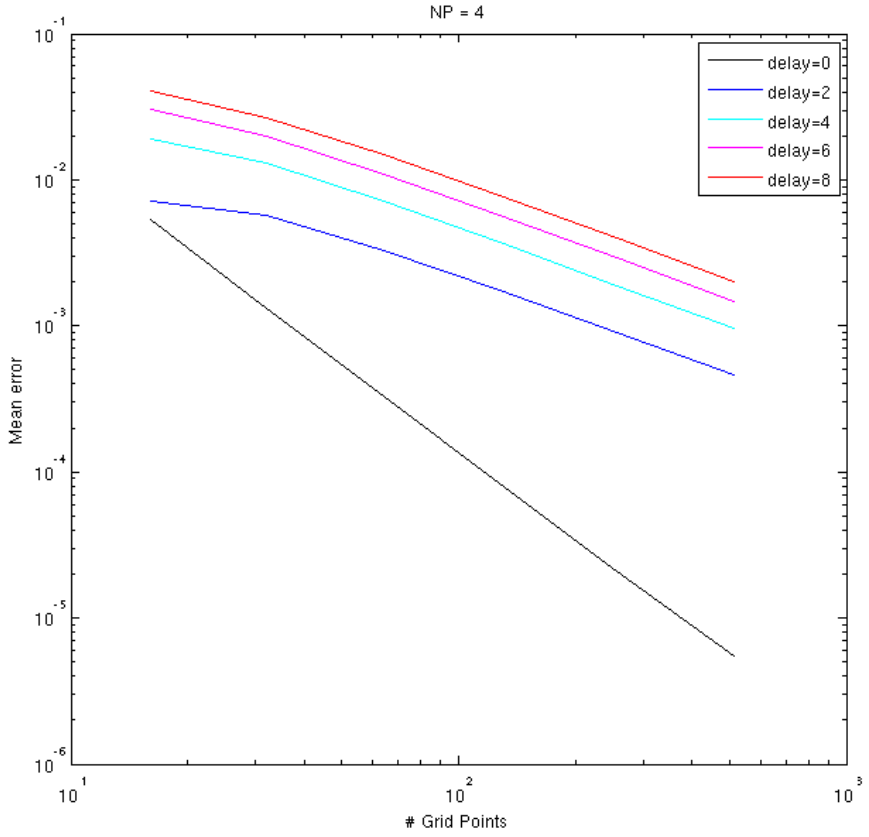
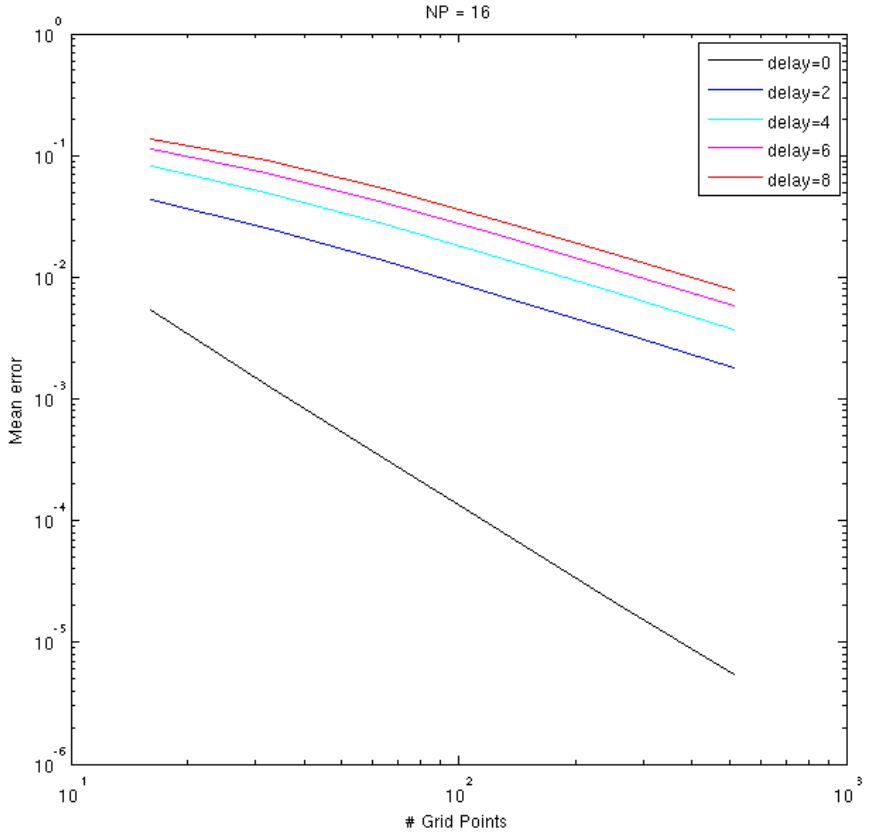


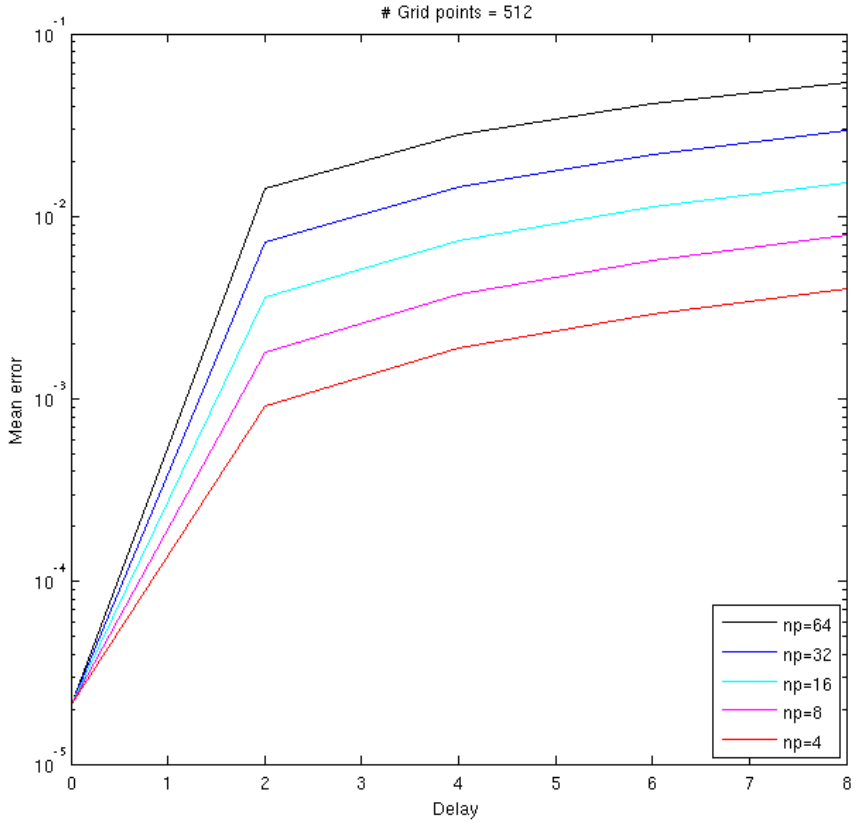
Figure 5: Synchronous vs. asynchronous simulation with order 1 and 2.



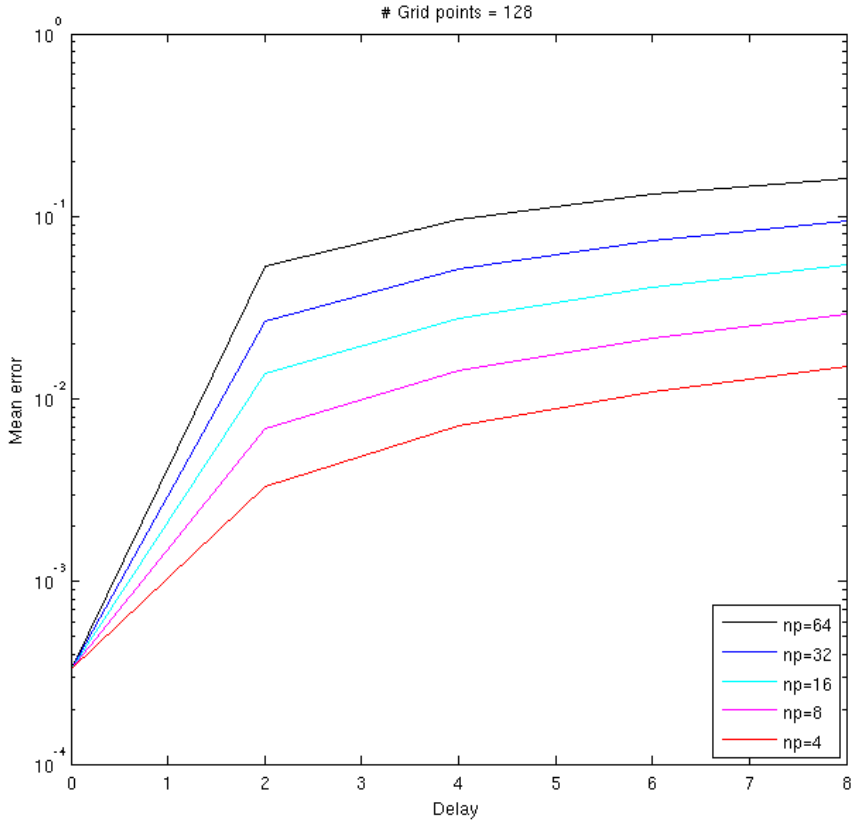
(a) Synchronous vs. asynchronous simulation. 4 PEs.



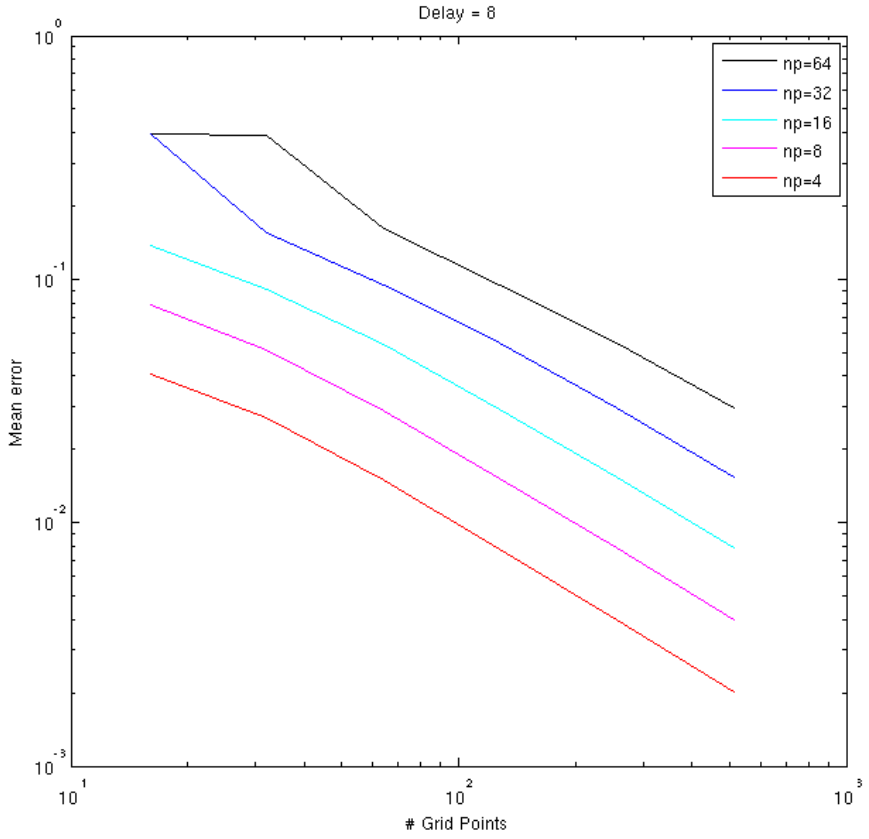
(b) Synchronous vs. asynchronous simulation. 16 PEs.



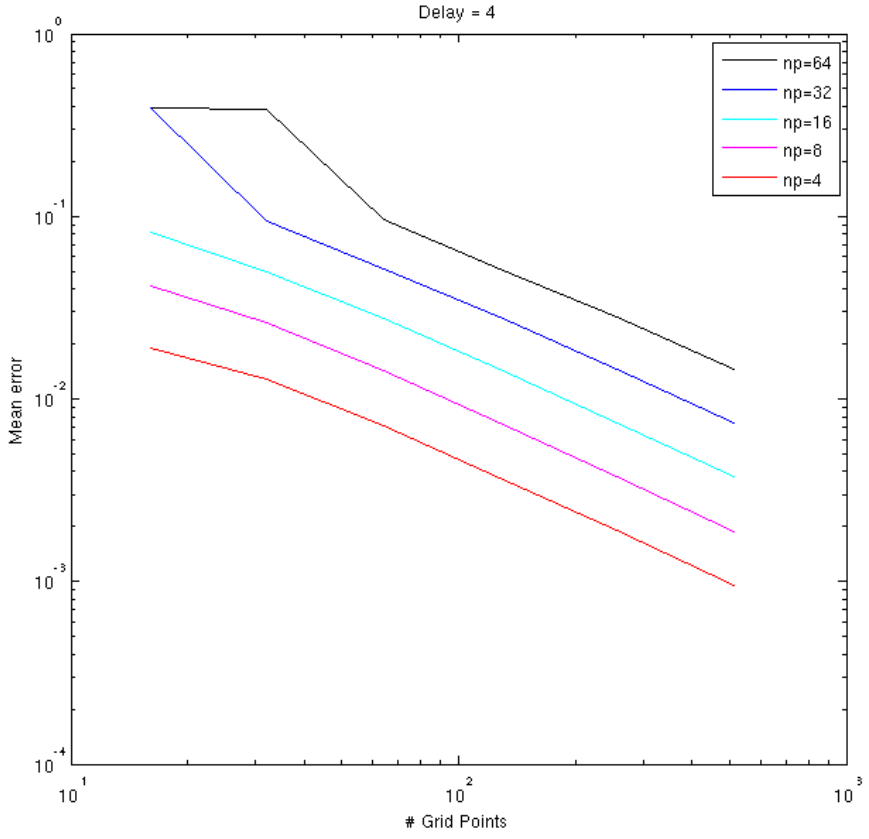
(a) Influence of delay for different number of PEs. 512 Grid points



(b) Influence of delay for different number of PEs. 128 Grid points



(a) Influence of number of grid points for different number of PEs. Delay 8



(b) Influence of number of grid points for different number of PEs. Delay 4