A non-smooth event-driven, accurate, adaptive time stepper for simulating switching electronic circuits.

Claude Lacoursière and Tomas Sjöström
HPC2N/UMIT, Umeå University
SE-901 87, Umeå, Sweden
claude@hpc2n.umu.se, tomas_sjostrom@hotmail.com

January 26, 2015

Abstract

We present an event driven adaptive time stepper for the simulation of the dynamics of switching electronic circuits with an application to power inverters. We show that in this context, variable time-stepping is both more efficient and more accurate than commonly used fixed-step integration methods using interpolation and extrapolation for event location and handling. We also show that impacts are needed to resolve certain types of transitions accurately. The stepping scheme is based on a nonsmooth variational method and can also handle complementarity conditions for nonsmooth elements such as diodes and transistors for instance. The combination of variable step size and impacts yields a method of second order according to numerical tests.

We also introduce a parallel, strongly coupled co-simulation method which allows to simulate many inverters with nearly linear complexity on multicore CPUs. The parallelization avoids the linear increase of switching events with the increase in number of phases or inverters, which leads to quadratic computational complexity. To do this, we used the effective admittances of subcircuits to enforce the voltages at common nodes. Our tests indicate that this is nearly as accurate as a fully coupled simulation. We have found that the combination of variable time-stepping and our parallelization technique allow for the simulation of several inverters in real-time.

We wrote prototype software which indicates that single inverters can be simulated faster than real-time, and slightly less than real-time for several ones, despite the additional numerical linear algebra necessary.

Contents

List of Figures

1 Introduction

Numerical simulation is essential for the development of switching circuits because they combine electronics and control systems. Interaction between the two can lead to systemic errors which are hard to find. This is even more important in the context of micro-grids because of strong interaction between inverters. Small grids are also sensitive to faults and so these have to be predicted.

The dynamics of switched circuits like inverters non-smooth, meaning that standard integration methods are not necessarily applicable since derivatives are not continuous at each switching event in the general case. Each transition must be treated with care in order to maintain good accuracy.

For the case of inverters, state transitions are governed by controllers whose function is to produce smooth AC signals from DC input, and they are tuned to respond quickly to small variations in frequency and to attenuate undesirable harmonics. The consequence is that simulations must run at very small time steps which make them slow. But as we show, the most commonly used methods are unnaturally slow by almost one order of magnitude. But there is a demand for simulations which can run in real time for hardware-in-the-loop for instance, and this motivates the present study for finding new, faster and more accurate methods.

The computational complexity of simulating inverters grows quadraticly with the number of phases which means that simulating all the inverters in a small wind farm, say, is prohibitive.

We introduce a high performance variable time step, event driven time integration method with impacts to improve both speed and accuracy. This contrasts with the state of the art methods which use interpolation and extrapolation to maintain a fixed step, and to locate events. There is currently no method which handles impacts either. As shown in our results section 6, variable time stepping is both faster and more accurate, reaching the real-time domain for two simple circuits, nearly one order of magnitude faster than the state of art commercial software packages.

We also developed a parallel algorithm allowing for the simulation of several inverters. Some accuracy is lost when doing this, but the parallel execution scales well enough that we can simulate very many coupled inverters with nearly linear scalability.

The report is organized as follows. We first discuss previous work in Sec. 2. We then continue with theory and numerical methods in Sec. 3, numerical methods in Sec. 4, and present our results and discussion in Sec. 6. The software is described in Sec. 5.

2 Previous work

The mathematical theory of differential equations with discontinuous right hand sides [8] is well established and for the case at hand, there is no difficulty from the perspective of existence and uniqueness of solutions for most cases. The problems we consider fit in the simpler category, and numerical handling of these is well-established [12, see p 196–200]. This is so because our in our case, events and switches are due to causes external to the dynamics of the system, i.e., they are *exogenous* events, i.e., they do

not depend directly on the internal states of the system. In other words, an event at a given time t cannot *cause* another one instantaneously, though several can happen simultaneously. Exogenous events are common in all numerical integration packages but since they work as black boxes, these are ill-suited for the case at hand where performance is crucial.

There is theory for cases of switching circuits with *endogenous* switches, i.e., state changes which arise due to the dynamics of the system [2, 26, 30] where the main problem is that solutions can be "trapped", meaning that it is impossible to cross from one state of a condition, where, say, a switch is in an ON state, to the other side where it is in the OFF state. This then requires sliding modes. This would be relevant for more general circuits but is outside of our scope. We have also worked on fixed step numerical methods which can handle such problems [29] but abandoned this avenue for the present study when tests showed that fixed step size integration is ill-suited here.

One could consider the complete circuits we study as a closed system, including the dynamics switches and filters, explained below in Sec. 3.1, but filters are weakly coupled to the rest of the dynamics, and are in fact delayed even in their physical realization.

When considering the simpler case only, a few questions remain namely, whether or not to use a variable step-size, and whether or not impacts must be resolved at transition time, i.e., where there is an instantaneous change of state when a event is triggered.

It appears that variable time step methods have not been used in the context we consider, though one can clearly use the standard methods offered in Matlab, but that is prohibitive in terms of performance, something we address here. One could presumably use one of the various implementations of [24] but we found very few literature about this.

The state of the art and previous work for the simplified event driven dynamics of electronic circuit then reduces to a handful of methods [see 21], all of which use fixed time step integration, but differ in how they locate transitions and how to reconcile fixed time step with the variable duration and uneven distribution in time of the events.

The industry standard software is currently PSCAD/EMCDT [6, 10]. This was originally designed for smooth circuits dynamics not subject to switching events according to the documentation. Fixed step is good here since the time integrator used is the implicit midpoint method [11] which is very stable even at fixed step. When fixed step is used, the linear algebra involved in computing voltages and currents is simplified since matrix factorizations can be reused. Adapting this to the case we study here poses difficulties, but revising the entire software package is presumably not advisable.

The five most common methods are illustrated in Fig. 2. We consider a circuit connected to a voltage source which can be switched ON or OFF. We assume that there is an event function $g(x)$ such that $g(x) > 0$ when the switch is ON, and $g(x) < 0$ when it is OFF, and x state of the system.

The fixed time grid used by all five methods is denoted $t_0, t_1, \ldots, t_5$ and an event is triggered at time between t_0 and t_1 and triggered again just before t_4, and has length less than the step size. This pulse is shown with a dotted black line, hidden by other

lines in some cases. In all cases, the orange rectangles are the excess energy put into the system because of the fixed step size, and blue is the energy unaccounted for. The reference is the middle panel on the left hand side.

For the simplest fixed step method in the top panel on the left hand side, time is advanced to t_1. A switch is then detection at an indeterminate time between t_0 and t_1. This event is then considered to have happened at t_1, and the switch is kept ON up to t_4. At that point, the switch is detected to be OFF, and the stepping continues with an OFF state.

Clearly, this makes sense only if the time step is very small. In the case of PWM circuits described below in Sec. 3.1, and in view of results in Sec. 6.3, high frequencies will not be resolved properly.

The post correction method on the top panel of the right hand side locates the ON event between t_0 and t_1, and then increases the voltage of the source to account for the missing energy. It does the opposite when the OFF switch is detected between t_4 and t_3. This clearly introduces anomalies.

FiXme Note: references?

For the Time Averaging Method (TAM) [21], an estimate of the energy in the duty cycle is computed, and the solution is then interpolated by varying the input voltage. This corresponds to recomputing the step from t_0 to t_1 in principle, but, without changing the state at t_0, only turning the switch ON at t_1. The same operation is repeated for the OFF event between t_3 and t_4.

Then comes the Interpolation and Extrapolation Method (IEM) which is the bottom panel on the right hand side column. This does locate the ON event before t_1, and interpolates to that point. It then turns the switch ON and takes a new step. This is not aligned with the grid however as it overshoots t_1. Instead of backtracking however, and new step is computed to reach a point before t_2. Since that undershoots t_2, a new step is taken, and the result is interpolated to t_2. Data for t_1 is then missing, or left at the wrong value.

Finally the Double Interpolation Method (DIM) used in PSCAD[9, 15, 17] is illustrated at the bottom panel of the left hand side. There the event is located by interpolation between t_0 and t_1 and interpolation is used to construct the state of the system there. A step is taken with the switch ON. Because of the fixed step, this overshoots and therefore, another interpolation is made to t_1 where a full step is now taken.

No matter how one looks at these methods, they can only be accurate for small step-size, even if the pulses are large on average. When the time step is large, there is severe attenuation of the high frequencies in the circuits as shown in Sec. 6.3.

But this is one aspect only. Common to all these methods is a simple restart of the integration when an event is located. However, as shown in Sec. 3.2, this is simply wrong and noticeably so for circuits which do not have inductance in them, such as non-ideal voltage sources.

PSCAD is the best representative of fixed-step methods we could find so it is the method we use for reference.

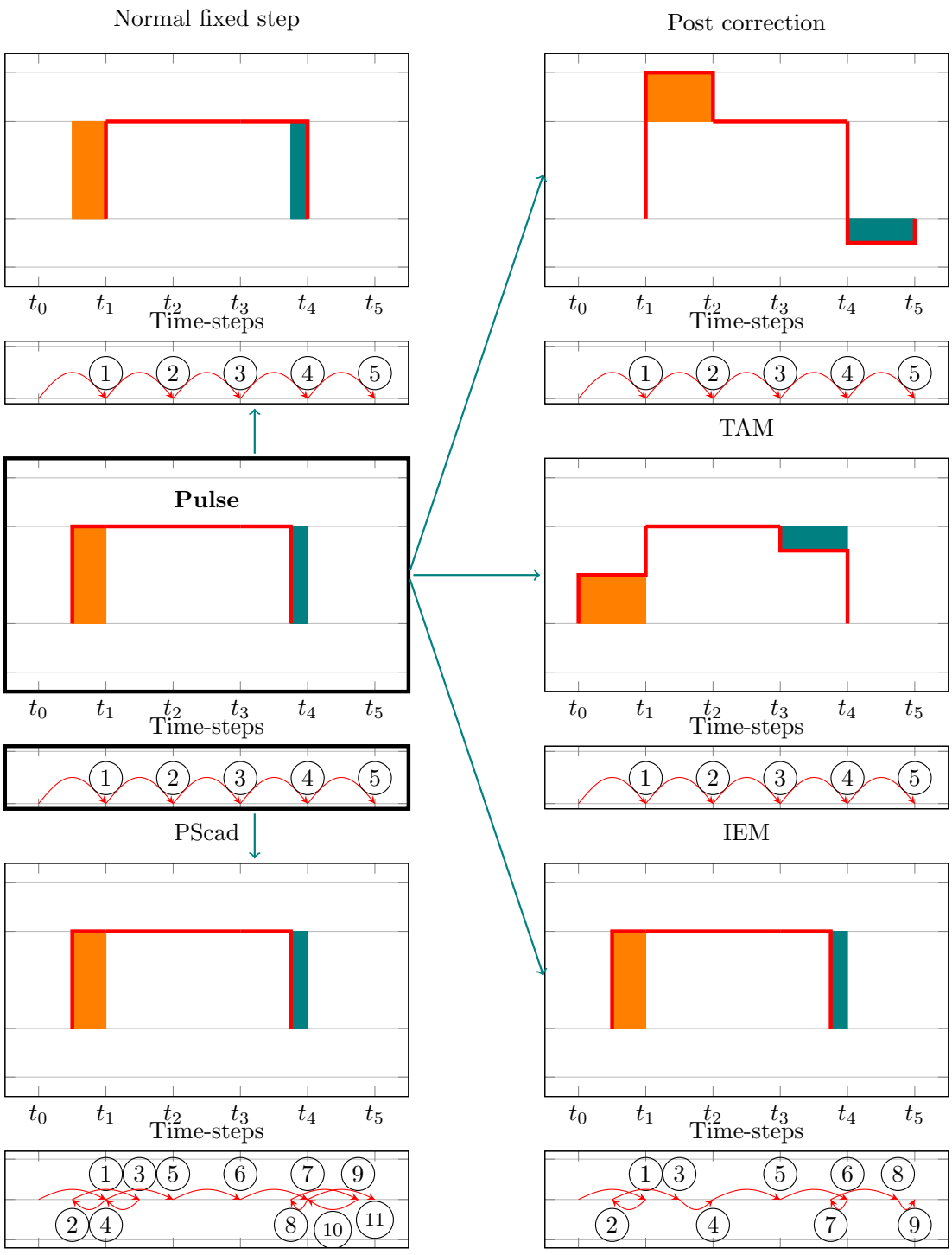

Figure 1: Different event location for fixed step methods

3 Theory

We present now the elements of theory needed to support our simulation techniques. We start with a general discussion of inverters, continue to the dynamics of electronic circuits. We then move on to discontinuities, and numerical time integration.

3.1 Inverters

Inverters are switching transformers which convert an DC signal to an AC one. These are used in frequency converters, also known as AC-AC transformers. Converters first transform an AC signal to a DC one using a rectifier. A DC transformer then converts this to the correct voltage, and an inverter is then used to produce AC at a given frequency using an array of switches which are turned ON and OFF in rapid sequences to produce a smooth signal of a given frequency. Frequency converters are essential in wind power generation for obvious reasons.

Going from AC to DC requires a filter, such as an LCL circuit. But modulating the input voltage, any wave form be generated. To produce a smooth sine wave however, the switches have to be controlled minutely. A fundamental problem with this technique is that any sudden change in the voltage causes undesirable high frequencies because of the $1/f$ nature of the spectrum of a pulse, as shown in Fig. 2. The LCL filter acts as a low pass filter though and greatly attenuates the higher frequencies but, being a linear filter, this attenuation is slow. The modulation of the natural frequency of a single phase filter is illustrated in Fig. 4.

Once a sine wave of the right frequency and amplitude is generated, there remains the problem of coupling this to a load such as the grid. For the latter case, an operator makes demands as to how much active and reactive power should be delivered. This requires the use of controllers which monitor the signal and modulate the pulse sequence.

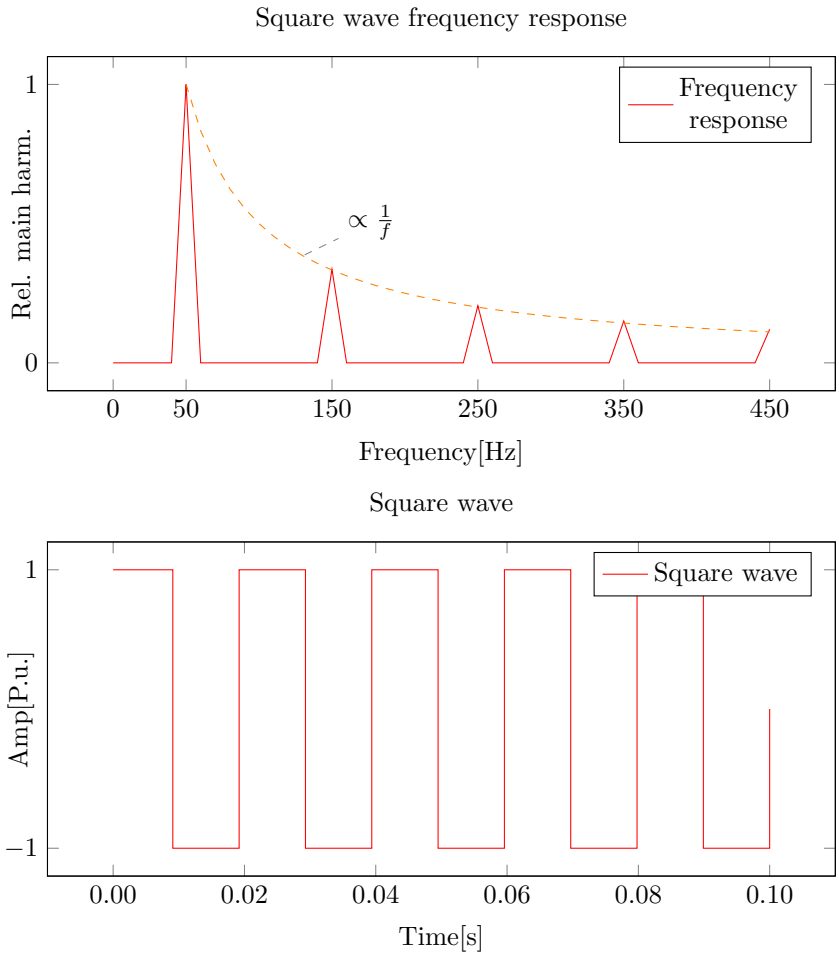

Figure 2: Frequency response of a square wave. The decay of the harmonics is very slow.

We consider two types of controllers, namely, those based on Pulse Width Modulation (PWM), and those based on Space Vector Modulation (SVM). We study both.

The working principle of PWM circuits is a comparator between a *carrier signal* and the output signal from a controller which monitors the signal. The carrier is a triangular wave at fixed frequency, much higher than that of the output wave. A typical value is 6KHz to control the 50Hz signal used in the grid. When the output signal crosses the carrier, an event is triggered. In this case, it would be to toggle a switch and change the voltage on one of the phases. To use PWM controllers in our simulation, we must detect the "collisions", i.e., the times at which the values we compute cross the carrier.

This is shown in Fig. 3 where the background grey triangular wave is the carrier signal, and the noisy sine waves are control signals for each of the phases. Comparator circuits triggers transitions whenever a control signal crosses the carrier, generating pulses of variable width. These pulses in turn activate switches on the voltage sources.

In our simulations, the time steps correspond to a fraction or the totality of a duty cycle which means that for PWM circuits, we must predict when a given control signal will cross the carrier. Given the noise in the signals, this poses a challenge we address in Sec. 4.7.

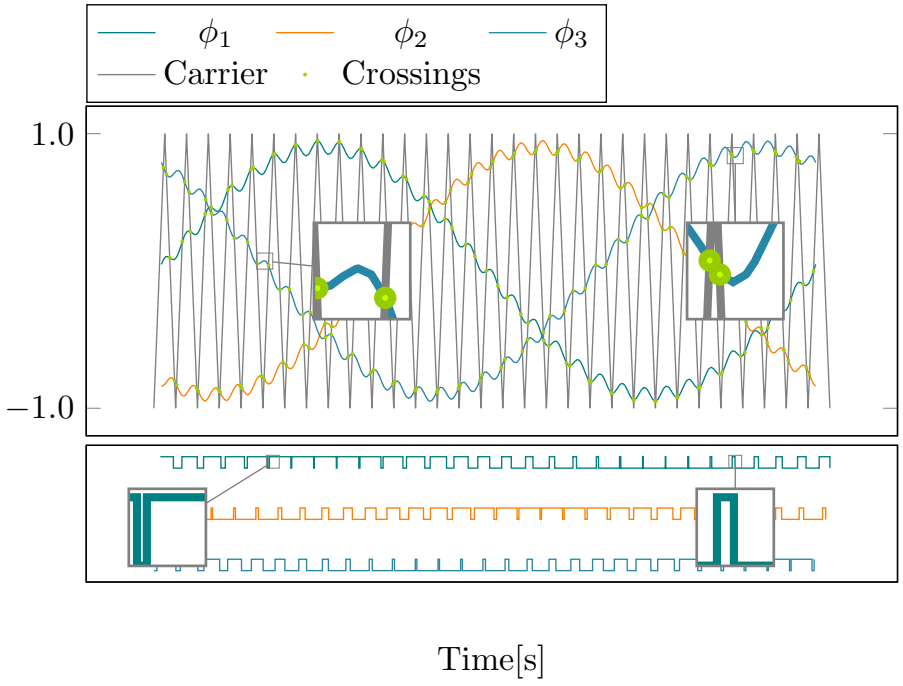

Figure 3: Details of a PWM controller

The SVM circuits on the other hand are based on a Clark transform which compute the phases and amplitude of the output signal from the controllers [1]. From this, a sequence of duty cycles is computed which should result in the desired signal. The SVM controller is essentially predictive, and we use this in our simulation to compute time steps.

A single phase inverter is described in Fig. 5. Here the filter is a simple LC circuit, and we have a load attached at the end of the inverter. The dynamics of the filter is shown in Fig. 4. The bottom curve is the natural oscillation of the LC filter, but the other two curves contain the current and voltage, respectively. Note that the current has discontinuous derivatives at the pulses.

A complete three-phase inverter is depicted in Fig. 6 which illustrates how the grid

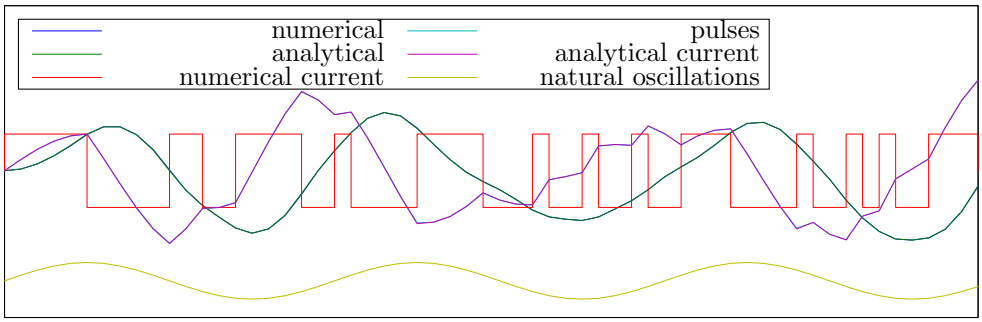

Figure 4: The dynamics of a nonsmooth harmonic oscillator

controller signal comes in to the inverter, filtered via numerous filters, from there, generating switches on the input voltage sources.

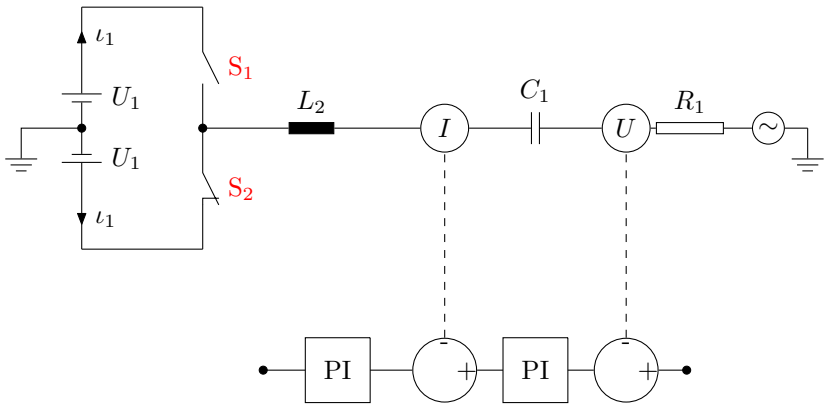

Figure 5: Single phase DCAC PWM driven inverter

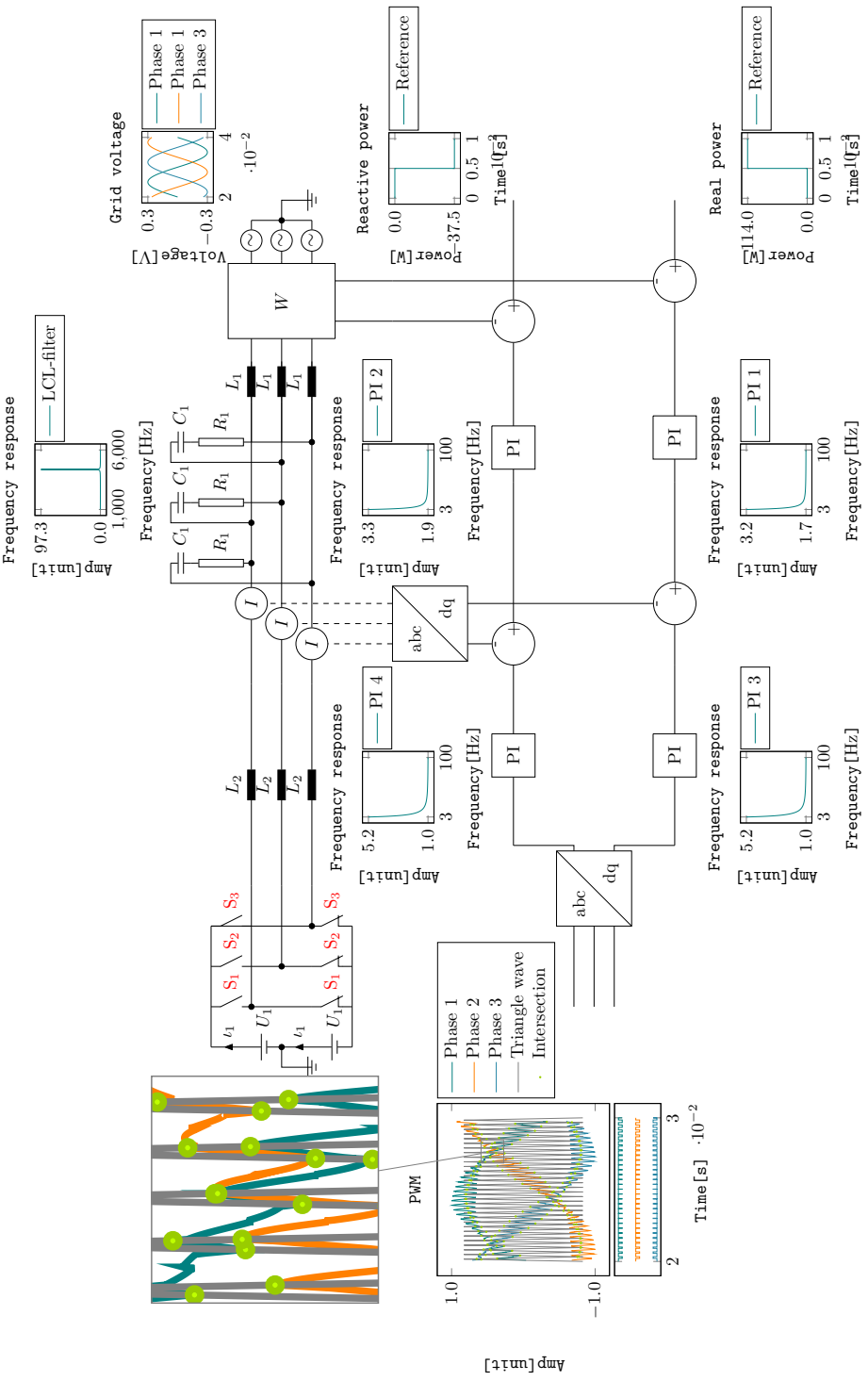

Figure 6: A complete PWM controlled three phase inverter circuit.

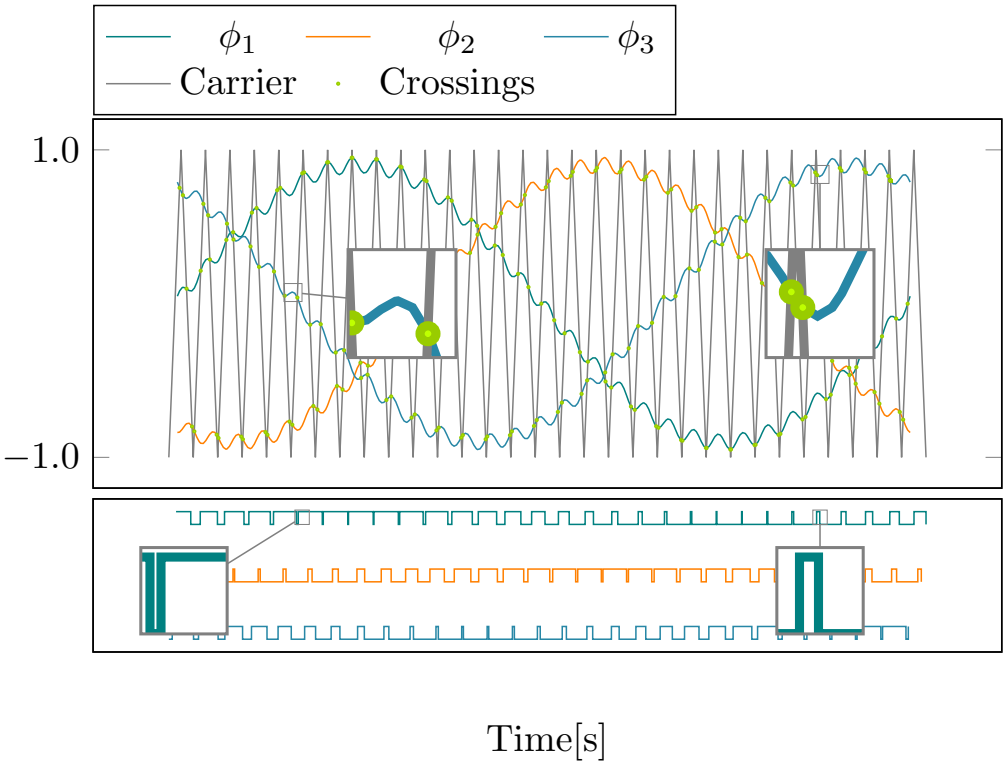

Figure 7: PWM control using a comparator between a triangle wave and a control signal.

3.2 Discontinuities

Switching circuits are discontinuous by definition and cause nearly instantaneous changes in voltages or currents. This can cause the currents or their derivatives to jump suddenly, though charges are continuous in time. We use simple serial LRC and RC circuits to illustrate this, and show that special care is needed for RC circuits or subcircuits.

A simple forced harmonic oscillator with damping with discontinuity at $t = 0$ reads

$$\begin{aligned} M\ddot{x}_- + B\dot{x}_- + K &= f_- \\ M\ddot{x}_+ + B\dot{x}_+ + K &= f_+, \end{aligned} \tag{1}$$

where $t_- < 0 < t_+$, and both are arbitrarily close to 0. Since a mass cannot change position instantaneously, $x_+ = x_- = x$. By integration we have

$$\int_{t_-}^{t_+} \mathrm{d}s\dot{x}(s) = x(t_+) - x(t_-) = 0. \tag{2}$$

This leads to

$$\ddot{x}_+ = \ddot{x}_- + (f_+ - f_-)/M. \tag{3}$$

This corresponds to a discontinuous derivative of the velocity. Since our time steppers only consider velocities and make no assumption on acceleration, see Sec. 4.1, this poses no problem.

When $M = 0$ however, Eqn. (1) implies

$$\dot{x}_+ = \dot{x}_- + (f_+ - f_-)/B. \tag{4}$$

This implies that there is an impact which needs to be resolved. Note however that the impact is due to a discontinuity on the forcing term which is entirely different from a an impact involving a boundary condition such as $x \geq 0$ for instance, which produces an instantaneous change in velocity. The latter needs an impact law for energy conservation but this is not the case here. Given the continuity assumptions, all that is needed is to compute a consistent acceleration after the impact, and this is unique since we already know all the other variables, so that know f_+ is enough to compute $\ddot{x}_+$ or $\dot{x}_+$ from Eqn. (1) or Eqn. (3) respectively.

Note also that for the case where $M = 0$, we have *two* values for $\dot{x}$ at time t, so that restarting the integrator must occur at $\dot{x}_+$, and that as far as the numerical solution is concerned, there must be data points that are either very close in time, or at the same time. The numerical ramifications are discussed in Sec. 4.2.

In the context of electronics, using $\dot{x} = i$ for current, $B = R$, the resistance, $M = L$ for inductance, $K = 1/C$ for capacity, and $f = V$, the voltage, this yields either

$$\frac{\mathrm{d}i_+}{\mathrm{d}t} = \frac{\mathrm{d}i_-}{\mathrm{d}t} + (V_+ - V_-)/L \tag{5}$$

or

$$i_+ = i_- + (V_+ - V_-)/R. \tag{6}$$

Once more, the continuity assumption on charge means that $q_+ = q_-$.

For switched circuits which do not have inductance on certain branches, these impacts must be resolved. The procedure for this is described in Sec. 4.1. The case of the LC circuit is illustrated in Fig. 4 where the discontinuities in the derivative of the current are apparent.

FiXme Note: write the fucking section

The RC circuit we chose was a battery that is switched ON and OFF, connected into a serial RC circuit. The current through the capacitor is shown in Fig. 8. The blue line is computed without using impacts and the black line is computed using impacts. PSCAD's solution is added for illustration. The impact solution is close to the analytic computation. But the solution without impacts overshoots significantly, and smooths out the dynamics. One should note that only q_- and $\dot{q}_-$ are reported by PSCAD which leads to lines with modest slopes instead of nearly vertical ones.

FiXme Note: for that picture, mention that the sine wave is different freq from the natural freq

The voltage on the battery and charge on the capacitor are shown in Fig. 9 and Fig. 10, respectively. It is clear that the naïve approach with the implicit midpoint method [13] gives catastrophic results, as seen from the blue line at the bottom. PSCAD's solution is better, but clearly, resolving impacts is necessary.

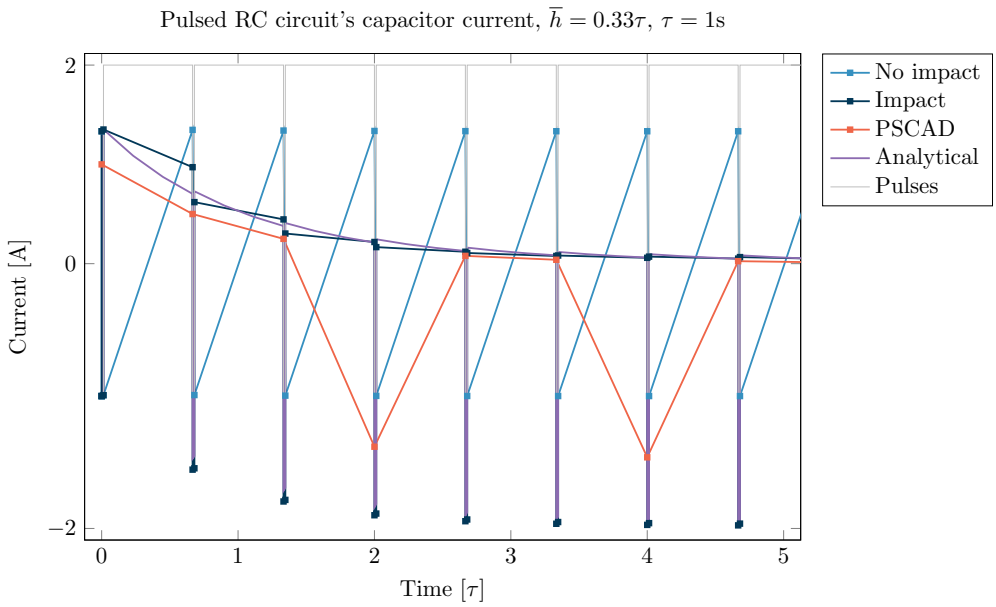

Figure 8: Impact dynamics for a pulsed, charging capacitor in an RC circuit: capacitor's current.

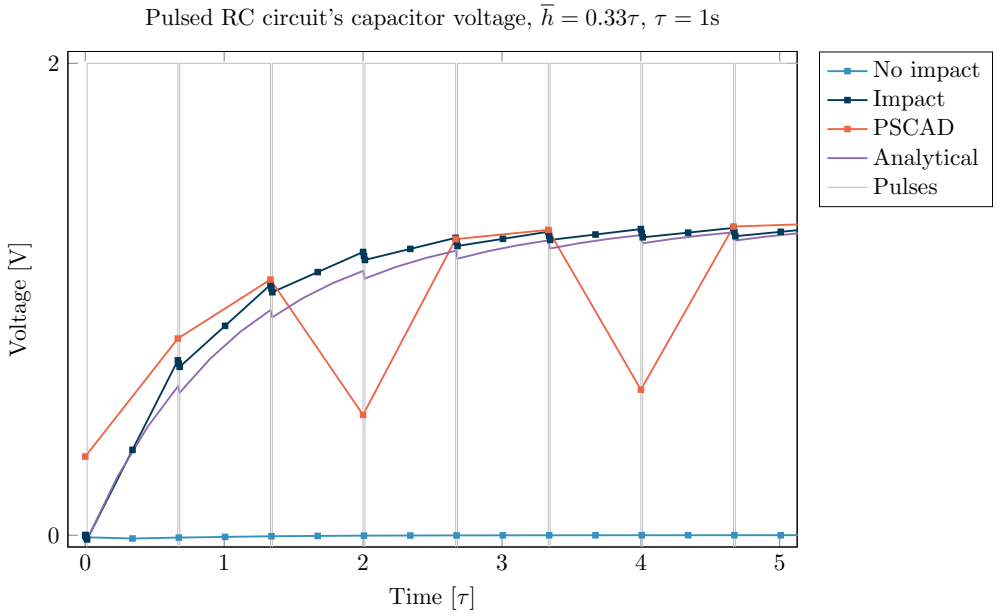

Figure 9: Impact dynamics for a pulsed, charging capacitor in an RC circuit: capacitor's voltage.

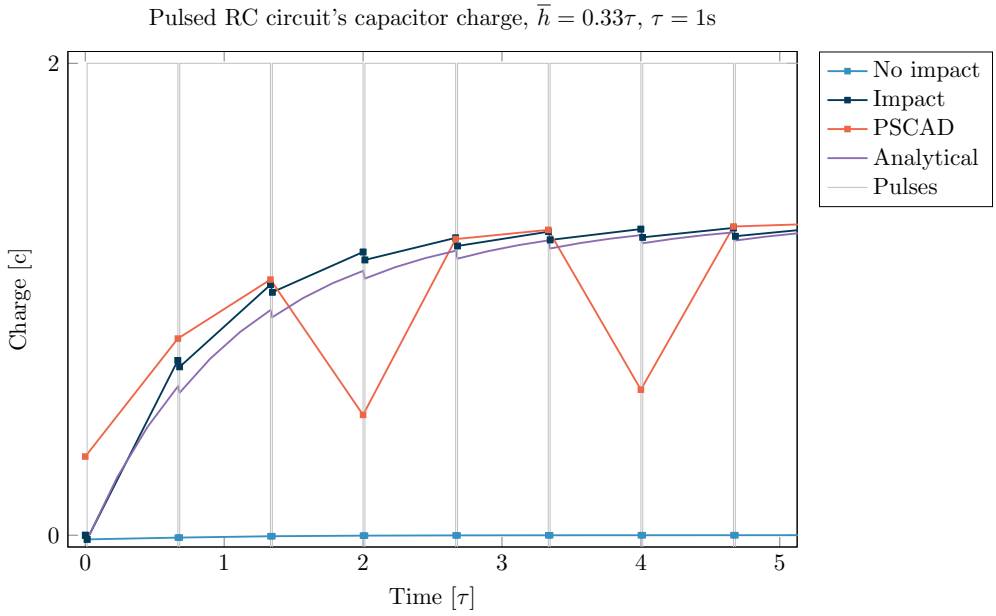

Figure 10: Impact dynamics for a pulsed, charging capacitor in an RC circuit: capacitor's charge.

3.3 The equations of motion

We briefly review the equations of motion of electrical circuits without discontinuities for the moment.

Consider a set of linear, discrete components such as capacitors C, resistors R and inductors L, each of which has a charge q and a current $\dot{q} = i$ running through it. These components are assembled in diagonal matrices with the same letters, namely, L, R, and C. These have different dimensions of course. Conservation of charge, or implies that

$$Ai = 0, \tag{7}$$

where A is the nodal matrix, with one row per node, and entries ± 1 corresponding to the components connected to the node. This is Kirchhoff's current law since for each row j of the matrix, we have

$$\sum_l a_{jl}\sigma_l i_l = 0, \text{ and } \sigma_l = \pm 1. \tag{8}$$

Restricting ourselves to linear components, the system's motion satisfies the following

Differential Algebraic Equations (DAEs)

$$\begin{bmatrix} L & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}i_L}{\mathrm{d}t} \\ \frac{\mathrm{d}i_R}{\mathrm{d}t} \\ \frac{\mathrm{d}i_C}{\mathrm{d}t} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_L \\ i_R \\ i_C \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{C} \end{bmatrix} \begin{bmatrix} q_L \\ q_R \\ q_C \end{bmatrix} = \begin{bmatrix} V_L \\ V_R \\ V_C \end{bmatrix}. \tag{9}$$
$$Ai = i_s,$$

where i_s are external current sources. The last line here is the algebraic condition, and one can show that in fact, the voltages can be separated as

$$\begin{bmatrix} V_L \\ V_R \\ V_C \end{bmatrix} = A^T V + V_e, \tag{10}$$

where V_l is the voltage at node l, and V_e are external sources.

The DAEs of motion in Eqn. (9) are degenerate which makes them difficult to solve numerically in this form. The most common method is called the Modified Nodal Analysis (MNA) [27] which reduces the system to a set of coupled differential equations which can be integrated numerically with standard methods. We choose a different method however.

4 Method

We now describe all the components of our algorithms.

4.1 Variational integration

Using a Lagrangian formulation and discrete-time variational mechanics[22], we arrive at a discretization which reads as follows

$$\begin{bmatrix} Z(h) & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} i_{k+1} \\ V \end{bmatrix} = \begin{bmatrix} a_k \\ i_e \end{bmatrix} \tag{11}$$

where $k$is the discrete time, i_e contains external current sources, and

$$Z(h) = \begin{bmatrix} \frac{1}{h}L & 0 & 0 \\ 0 & \frac{h}{4C} & 0 \\ 0 & 0 & R \end{bmatrix}, \quad i_k = \begin{bmatrix} i_k^{(L)} \\ i_k^{(C)} \\ i_k^{(R)} \end{bmatrix}, \quad a_k = \begin{bmatrix} \frac{1}{h}Li_k + V_k^{(L)} \\ -\frac{h}{4C}i_k - \frac{1}{C}q_k + V_k^{(C)} \\ V_k^{(R)} \end{bmatrix}. \tag{12}$$

Here, $V_k^{(j)}, j = L, C, R$ are external voltage source relative to ground. We update the capacity's charges according to

$$q_{k+1}^{(C)} = q_k^{(C)} + \frac{h}{2}(i_{k+1}^{(C)} + i_k^{(C)}), \tag{13}$$

which is the trapezoidal rule. We will write the system matrix as follows

$$H = \begin{bmatrix} Z & -A^T \\ A & 0 \end{bmatrix}, \quad \text{and } \bar{H} = \begin{bmatrix} Z & A^T \\ A & 0 \end{bmatrix}. \tag{14}$$

Either H or $\bar{H}$ can be used to solve the stepping scheme in Eqn. (11), as H can be used to solve for i_{k+1}, V, and $\bar{H}$ can solve for $i_{k+1}, -V$. Not that H is non-symmetric so it is generally more efficient to use $\bar{H}$ which is semi-definite.

A similar method is present in the literature [25], but this is seldom used on the grounds that the matrix H or $\bar{H}$ in Eqn. (14) large and would be inefficient in a numerical context in comparison to the MNA formulation which uses smaller, positive definite matrices, but which are more dense. But matrix $\bar{H}$ is very sparse. Much like the midpoint method, the time stepping scheme above is very stable [18] at fixed step. This formulation also allows for inequalities which then includes diodes and transistors among other non-smooth components. This can also be done at fixed step without any difficulty [18, 20]. This differs from standard stiff nonlinear functions for diodes, and is much more efficient [29].

This stepping scheme then is the basis of our numerical method. The matrix H in Eqn. (14) depends on the time step so factorization is needed whenever this changes. But provided this matrix has the same structure at each factorization, the cost can be made small given suitable sparse linear algebra routines. This is discussed further below in Sec. 4.3.

4.2 Impacts for complete circuits

We think now of crossing singularities for an entire system, i.e., stepping with a vanishing step when there is a sudden, large change in input voltage. As said previously, suffice to compute a new current according to an equivalent of Eqn. (1) or Eqn. (3). Though it suffices to restart the integrator after the discontinuity, taking a standard step would blur the sudden change which is undesirable as discussed in Sec. 3.2. The change should be instantaneous or at least we should have values for the currents, charges and voltages at the instant infinitesimally close to the discontinuity. This does not matter for circuits with inductances since the changes in the current, charges and voltages are not abrupt.

But looking at the numerical impedance matrix in Eqn. (12), it is clear that we cannot let the time step be too small because of the factor $1/h$ in the inductances, yet we want as sharp corners as possible.

To study this, we are thinking of applying a voltage $V_s = V_+ - V_-$ on the components with a zero time step. We then compute the current i_+ after the impact using the currents i_- before. To unify notation, we denote $i_- = i_k$ and $i_{k+1} = i_+$.

Starting with the inductors, we take the Schur complement of $\frac{1}{h}L$ to get

$$\begin{bmatrix} R & 0 & A_R^T \\ 0 & \frac{h}{4C} & A_C^T \\ A_R & A_C & hA_L L^{-1} A_L^T \end{bmatrix} \begin{bmatrix} i_{k+1}^{(R)} + V_s^{(R)} \\ -\frac{h}{4C} i_{k+1}^{(C)} - q_{k+1}^{(C)} + V_s^{(C)} \\ -A_L(L^{-1} i_k^{(L)} + V_s^{(L)}) \end{bmatrix}. \tag{15}$$

It is clear now that the matrix on the RHS in Eqn. (15) has a good limit as $h \to 0$, and that the inductors now act essentially as current sources. The remaining system

matrix has some issues to. As $h \to 0$ we have

$$\tilde{H} = \begin{bmatrix} R & 0 & -A_R^T \\ 0 & 0 & -A_C^T \\ A_R & A_C & 0 \end{bmatrix} \tag{16}$$

This matrix will not necessarily have full row rank, which corresponds to a circuit made of disconnected subcircuits. However we know the voltage drops across capacitors which means that we can remove them. We therefore remove A_C and reconstruct A_R so that components that were coupled on either side of a capacitor are now connected together. The capacitors become voltage sources if they are connected to resistors on both sides so that for each one of them, we put $V^{(j)}$ on one of the components and 0 on the other, where

$$V^{(j)} = -\frac{q_k^j}{C^j} - V_C^{(j)}. \tag{17}$$

Assuming that a capacitor j is connected to nodes m, n, the A matrix is then modified by summing these two, and then one of two nodes, m, say. The voltage in Eqn. (17) is then added to all components connected previously connected to m.

All the nodal voltages are known at this point but not the capacitor or resistor currents. For inductors we have $i_{k+1}^{(L)} = i_k^{(L)}$ since the voltages are finite by construction. For the capacitors the difference

$$A_{m_j,\bullet} i_{k+1} - A_{n_j,\bullet} i_{k+1} \tag{18}$$

using the original rows of matrix A. Note that these impact matrices can be computed and factored once and then stored for reuse. The complete procedure is listed in Alg. 4.1.

Algorithm 4.1 Computing impacts for a full circuit

Assemble current sources from the inductors: $-A_L(L^{-1} i_k^{(L)} + V_s^{(L)})$
for all capacitors $j = 1, 2, \ldots, n_c$ **do**
  Sum rows $A_{m_j,\bullet} \leftarrow A_{m_j,\bullet} + A_{n_j,\bullet}$
  Delete row n_j from A
  Delete column c_j from A
  Add voltage $V^{(j)}$ to components in row m_j
end for
Add current to the RHS of each new submatrices
Solve new systems for $i_{k+1}^{(R)}$
Compute the capacitor currents according to Eqn. (18)

4.3 Numerical linera algebra

Matrix H in Eqn. (14) is non-symmetric but positive definite and this structure is necessary when dealing with non-smooth system for solvability of the Linear Complementarity Problems (LCP)s[23] which arise when diodes and transistors are present[29].

We use $\bar{H}$ in the computations because it is symmetric which leads to computational savings. However, since $\bar{H}$ is indefinite, some dynamic pivoting is generally needed [7] we instead relied on an a-priori safe pivoting technique [19].

Factorizing a sparse matrix requires permutations to avoid fill-ins [5] which is a costly process. But $\bar{H}$ has fixed structure for any given circuit. Though widely available packages such as UMFPACK[4] and CHOLMOD [3] mention that the permutations can be reused, they do not reuse all information and spend considerable amount of time repeating work at each factorization. The most problematic being that for a factorization

$$\bar{H} = LDL^T, \tag{19}$$

where L is unit lower triangular, and D is diagonal, the solution of a linear system $\bar{H}x = b$ is computed as follows

Solve $Lz = b$ for z
Solve $Dy = z$ for y
Solve $L^T x = y$ for x

each step being relatively easy to perform [5]. The issue here is that transposition is a costly operation. Our solution here is to construct L^T so that it holds pointers to the elements in L. This involves memory indirection but the rationale here is that sorting, permuting and copying is slower than a single memory lookup, especially if the factorization is used only once, which is the case for variable time-stepping.

4.4 Splitting

The complexity of simulating N coupled inverters grows at least as fast N^2. If we consider a single inverter with M components and p phases, simulating two separate inverters takes twice as much work. If they are coupled however, this might be even be larger because of the linear algebra which can have complexity larger than linear, even when using good software for sparse matrices. However, the number of events doubles. Unless one takes many steps between events, this leads to twice as many steps. Let h be the average time step, $n = T/h$ the number of steps per period T, and n_c the number of crossing. The number of steps taken is simply $n_s = n + n_c$ in general, or just n_c for a variable step size assuming the filter frequencies are low enough. The number of crossing is then twice the number of phases so $n_c = 2Np$. The total amount of work w_t per period is then $w_t = w(N)n_s$, where $w(N)$ the work to step N inverters. With the optimistic case of linear scaling to perform one step in terms of the number of inverters, we have $w(N) = \alpha N$. Asymptotically, $n_s \to n_c = 2n_p = 2pN$. Putting all this together, we have

$$w_t(N) = O(w(N)n_s) = O(\alpha N \cdot n_c) = O(\alpha \cdot N \cdot 2pN) = O(N^2). \tag{20}$$

Clearly, this is an undesirable situation. The only opportunity for parallelism in this case is in the numerical linear algebra and this is rarely advantageous except for very large systems.

One can decouple the inverters in principle by simulating them separately, each with its own grid model, and then exchange currents periodically so they share a common

voltage at the grid coupling. The problem then is that each inverter must have an estimate of the nodal voltage, as well as the impedance of the others. This type of weak coupling is notoriously unstable and this only gets worse as the number if subsystems increases.

We present two strategies to split a system and estimate the current through and voltage at a common node.

4.5 Weak coupling

We consider two systems each with impedance $Z^{(i)}$, coupled at a common node. With the discretization mentioned in This can be written as

$$\begin{bmatrix} Z_{11} & 0 & A_1^T \\ 0 & Z_{22} & A_2^T \\ A_1 & A_2 & 0 \end{bmatrix} \begin{bmatrix} i_k^{(1)} \\ i_k^{(2)} \\ V \end{bmatrix} = \begin{bmatrix} a_k^{(1)} \\ a_k^{(2)} \\ 0 \end{bmatrix}, \tag{21}$$

where $i^{(j)}$ are currents Z_{jj} are impedances, and $a^{(j)}$ are vectors which depend on discretization. What the last line in the matrix says is

$$A_1 i_k^{(1)} + A_2 i_k^{(2)} = 0, \tag{22}$$

which is Kirchhoff's law for the common node where only few currents are involved. In this case, A_1, A_2 are simply projection matrices, i.e., this equation simply says that a few currents of the first subsystem should be identical to a few currents in the second. In other words, some components are duplicated, and the constraint ensures that their currents match. The Lagrange multiplier λ is then simply the voltage at the common node, which is an input to each of the subsystems. With simple manipulations, it is easy to show that

$$\left[A_1 Z_{11}^{-1} A_1^T + A_2 Z_{22}^{-1} A_2^T\right] \lambda = W\lambda = -A_1 Z_{11}^{-1} a_k^{(1)} - A_2 Z_{22}^{-1} a_k^{(2)}. \tag{23}$$

There are methods to simulate weakly coupled systems, a popular one being the Transmission Line Model[16, 28] but in all our tests, this failed to converge properly for reasonable steps.

4.6 Strong coupling

We choose a "strong coupling" strategy in which an estimate of the admittance of the coupled inverters is performed numerically. To do this, we perform the time integration of each subcircuits – inverter in this case – $p+1$ times where p is the number of phase. In each of these time integration, we vary the input slightly on one of the phases, away from a starting estimate used for all subcircuits and all phases. One integration serves as a reference. Using finite differences, we can then compute the admittance of a circuit, i.e., a matrix which tells us the current-voltage relationship. Once this information is collected for each subcircuit we can compute the approximate input current for each so they agree on the voltage at the common node at the end of a

given time interval. At that point, each subcircuit is integrated one more time. Note that this is multirate as well since the effective admittance can be computed for a time interval that is commensurate with the oscillation period of the grid.

In between the synchronization points, each inverter can also use the admittance matrix to compute the impedance of a local grid model.

This technique then requires $p+2$ integrations per inverter per step. However, the admittance computations which require $p+1$ time integrations over a given interval are embarrassingly parallel. The overall cost is then only twice as large assuming enough computing resources. In principle, this can scale linearly. In terms of raw computational cost, this method is advantageous from six inverters onward, but taking parallelism into account and assuming sufficient resources, this is faster in terms of wall clock time from three inverters onward.

We consider two inverters with p phases and a grid serving as an energy sink with resistance R_g and voltage

$$V_g(t) \approx \sin(\omega t + \phi). \tag{24}$$

For a short interval of time, the voltage at the node connecting the inverters and the grid can be approximated with

$$V \sin(\omega_g t + \phi_p) \tag{25}$$

for each phase p. The grid itself, approximated by an impedance R_g is simulated with a fixed integration step of h.

For each inverter there is an operator which maps initial condition $z_0^{(j)}$, which includes currents and changes, and the grid parameter V so that

$$z_h^{(j)} \leftarrow \Phi_h^{(j)}(z_0^{(j)}, V). \tag{26}$$

Note here that several internal steps are needed to go from $t=0$ to $t=h$. For two phases now we need to solve the nonlinear system of equations

$$\begin{aligned} z_h^{(1)} &= \Phi_h^{(1)}(z_0^{(1)}, V) \\ z_h^{(2)} &= \Phi_h^{(2)}(z_0^{(2)}, V) \\ A^{(1)} z_h^{(1)} + A^{(2)} z_h^{(2)} &= 0. \end{aligned} \tag{27}$$

The last line can be rewritten thus

$$A^{(1)} \Phi_h^{(1)}(z_0^{(1)}, V) + A^{(2)} \Phi_h^{(2)}(z_0^{(2)}, V) = 0. \tag{28}$$

We need to know V such that this is satisfied and this we do by first using an estimate $\bar{V}$ so that $V = \bar{V} + \delta V$. We first write

$$J^{(j)} = \frac{\partial \Phi_h^{(j)}(z_0^{(j)}, V)}{\partial V} \tag{29}$$

So that we can write

$$A^{(j)} \Phi_h^{(j)}(z_0^{(j)}, \bar{V}) = \bar{i}^{(j)} \tag{30}$$

and

$$A^{(j)}\Phi_h^{(j)}(z_0^{(j)}, V) \approx \bar{i}^{(j)} + J^{(j)}\delta V, \tag{31}$$

so that Eqn. (28) is approximated by

$$A^{(1)}\bar{i}^{(1)} + A^{(2)}\bar{i}^{(2)} + [J^{(1)} + J^{(2)}]\delta V \approx 0. \tag{32}$$

This is then a simple linear system of equations for δV. Remains to compute the matrices $J^{(j)}$ which must be done procedurally since there are multiple switches between 0 and h. To do this, we first simulate the inverter with $\bar{V}$ and obtain $\bar{z}_h^{(j)}$. Then for each phase we introduce a perturbation $\delta V^{(p)}$ and compute the resulting states $\bar{z}_h^{(j,p)}$. We then have an approximation of the column p of $J^{(j)}$ as

$$J_{\bullet,p}{}^{(j)} \approx \frac{\bar{z}_h^{(j,p)} - z_h^{(j)}}{\delta V^{(p)}}. \tag{33}$$

Only the current variables in $z^{(j)}$ are used. We then construct the matrix

$$K = J^{(1)} + J^{(2)}. \tag{34}$$

Note that if we did just one step of equal length for each inverter, we would have

$$J^{(j)} = [Z^{(j)}]^{-1} A^{(j)^T}, \tag{35}$$

and so K would be symmetric and positive definite. This is not the case here however and though there is no theory available at this time, this is suspected to be a source of instability so there are limits on the time step h.

The full listing is found in Alg. 4.2.

4.7 Collision detection

The speed of our method depends critically on the collision detection for PWM control circuits. Here, each phase of the control signal crosses the carrier twice per carrier period. There therefore six crossings for a three phase circuit with PWM control. The carrier for a three phase circuit with PWM control, as in Fig. 6. Each crossing causes changes in the switches, therefore changing the configuration of the circuit. In the best possible case assuming that the filter dynamics is slow enough, we could in principle take only six steps per carrier period assuming that collisions between the signals of the phases and that of the carrier can be detected accurately enough. There is no collision detection in the case of SVM, though the switches depend on the accuracy of the simulation, and this decreases when proceeding at fixed step. We concentrate on PWM circuits with triangular carriers in what follows.

Our examples had relatively slow dynamics, i.e., a filter at 400Hz and a carrier at 6KHz. This means that we can take large steps and in that context, linear interpolation for collision detection is sufficient. Quadratic interpolation would be used if the filter frequency was higher.

Algorithm 4.2 Strong coupling, multirate simulation

Given n inverters with p phases each, a global step h, and a grid voltage of V
while Not done **do**
  for all Inverters $j = 1, 2, \ldots, n$ **do** ▷ parallel
    Simulate for time h with node with grid approximation $\bar{V}$
    Store $bari^{(j)}$
    for all Phases $l = 1, 2, \ldots, p$ **do** ▷ Parallel
      Simulate for time h with node with grid approximation $\bar{V} + \delta V^{(l)}$
    end for
    Compute $J^{(j)}$
  end for
  Compute $K = \sum_j J^{()} j$
  Compute $\bar{i} = \sum_j A^{(j)} \bar{i} j$
  Solve $KV = \bar{i}$
  for all inverters $j = 1, 2, \ldots, n$ **do** ▷ Parallel
    Simulate for time h with node with grid approximation V
    Store $bari^{(j)}$
  end for
  Update grid voltages with $bari^{(j)}$
  $\bar{V} \leftarrow V + V_s$ ▷ V_s is an estimate of noise
end while

The stepping scheme here uses a nominal step h_*, then checks for collisions in the past. If there is a collision on one of the phases, the step is rejected and a new steps is taken to the estimated crossing time. Collision detection is performed again to see if other phases crossed the carrier, but the step is not rejected.

Algorithm 4.3 Collision detection

```
while not done do
    Take a step of length h_*
    Detect collisions on all phases
    if collision on phase p then
        Compute h < h_* for the earliest collision
        Take a step of length h
        Switch the state of phase p
        for all phases j ≠ p do
            Detect collision
            if collision then
                Switch the state of phase j
            end if
        end for
    end if
end while
```

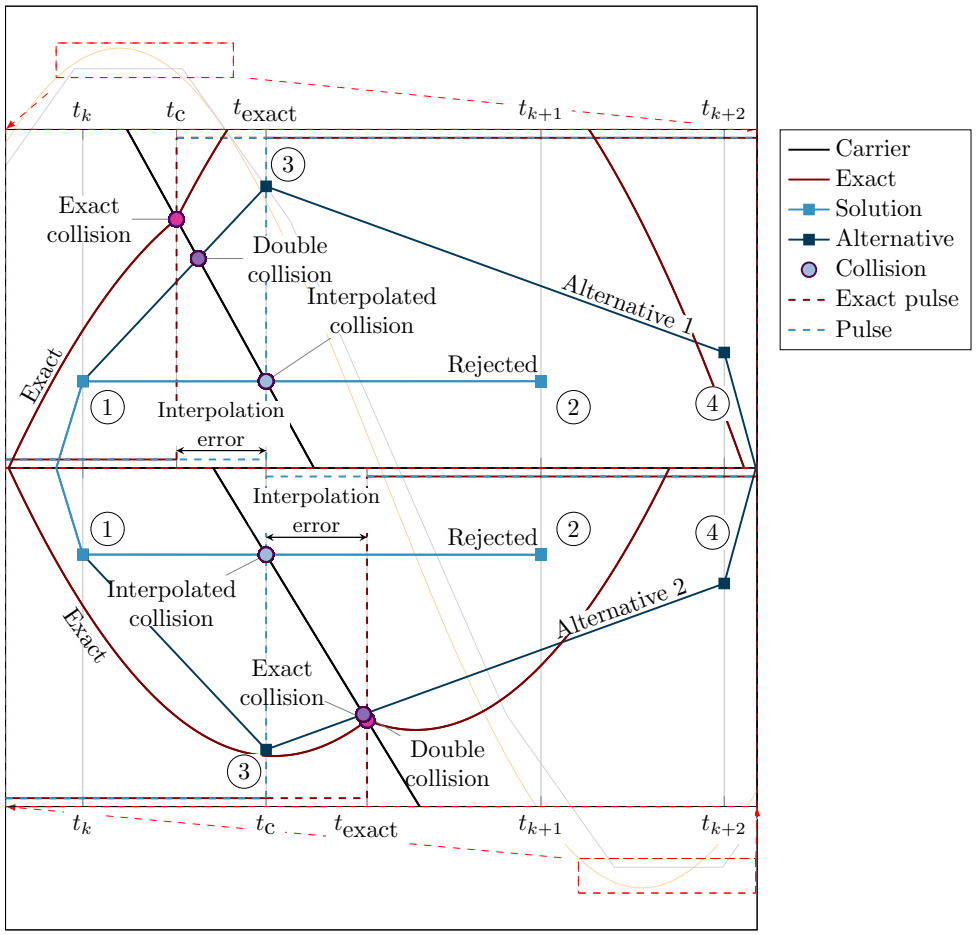

Figure 11: The dynamics of variable time stepping and collision detection. The difference between the analytic solution and the numerical one is grossly exaggerated for illustrative purposes only.

The collision detection algorithm in Fig. 12. The upper half represents the case where the computed collision time is greater than the real impact, and the lower half covers the opposite case. The alternative curves are zoom-ins of the background sine wave.

In both cases, the dark red line is the exact solution. The blue line is the computed trajectory from the last step to point (1), when the first computation of the state (2) at t_{k+1} is computed. Since the trajectory crosses the carrier however, the approximate location of the collision t_c is computed by intersecting lines. Then the step is rejected.

Setting the time step to $h = t_c - t_k$, the new state (3) is computed and a flag is set to indicate that the collision is handled. Since the collision time t_c is approximated according to the rejected trajectory, the new state (3) could be above or below the carrier, i.e., ahead or after the actual collision, corresponding to the top and bottom half, respectively. In the first case, we detect that we passed the carrier. Since the flag is ON at this point, we assume that this is the collision we had to consider. We accept the step and unset the flag. We do not backtrack however, but fire the pulse and take a new step to state (4) at time t_{k+2}. We could do a refinement, but this first order method has proved sufficiently accurate.

The alternative is shown in the bottom have in which case the value of the signal at (3) to t_c is not yet passed the carrier. Here we assume that we are close to the carrier and therefore, keep the flag ON. Again, the pulse is fired and the step is accepted to reach state (4). The collision detected between t_c and t_{k+2} is then ignored due to the flag being set, meaning that it is assumed to be the same one which was detected previously. The flag is then unset, since we are guaranteed to have past the carrier.

The value of the pulse is determined by looking at state (1). If the control signal is below the carrier, the value pulse is set to the upper value and vice versa.

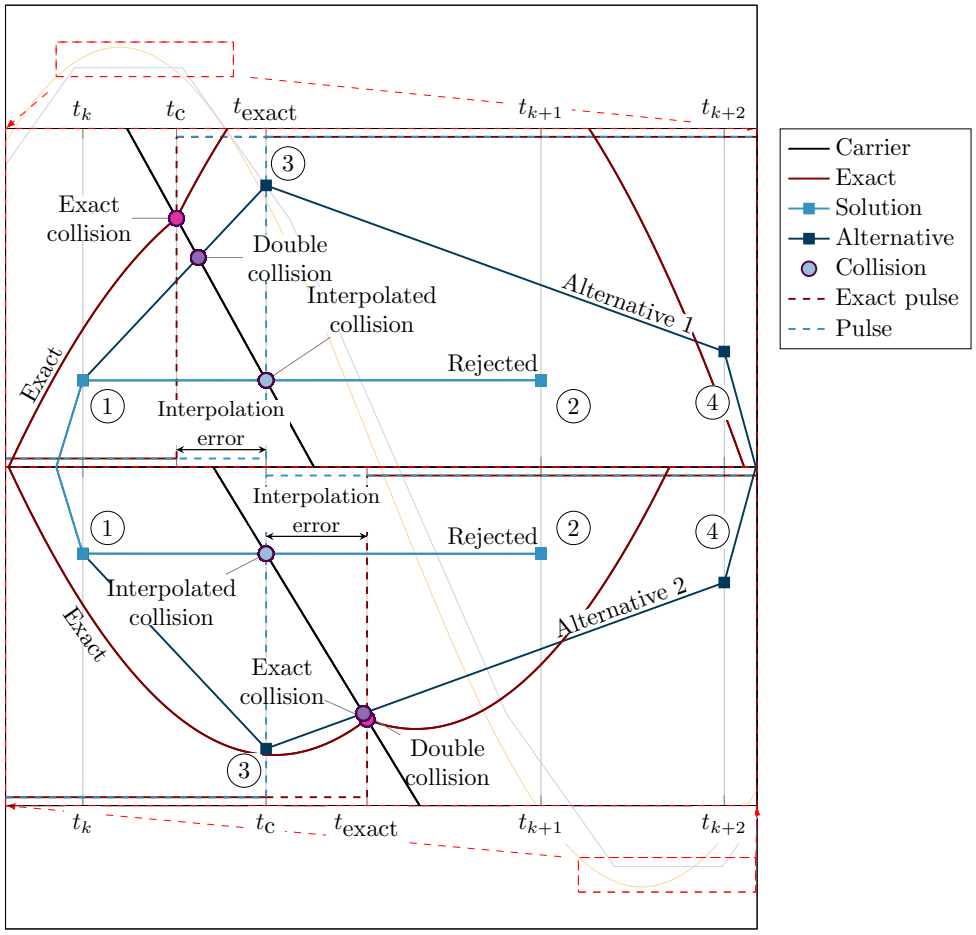

Figure 12: Detailed view of collision detection.

All possible types of steps are presented in Fig. 13, where we used straight lines only for clarity. We start from the top of the figure which describes situations involving a peak, and the bottom half is a mirror image corresponding to a dip. The dotted vertical lines at t_4 and t_{10} are walls which prevent double crossing, meaning that only collision is with the carrier is possible, as illustrated in traces +1 to +5. For trace +1 for instance, we want to step all the way to t_8 which would mean that two collisions are missed. However, the wall at t_4 means that we will search only between t_2 and t_4, during which interval we are guaranteed to have a maximum of one collision. The wall at t_7 is not even considered. Trace +6 is clear so step t_2 is taken. Traces +7 and +8 are also stopped at a wall, but trace +9 does not need this. Traces +10, +11 and +12 start at a wall and collide with the carrier, and stopped by a wall for +10 and

+11, but not +12. Traces from +13 to +18 we start from a safe point and continue forward to either a wall, a safe point, or a carrier position. The other cases from −1 to −18 are mirror images.

Essentially, we never integrate past a wall, i.e., if we do go past, we restart with a suitable time step. Since we keep the factorization of the stepping matrix for the nominal step, the carrier period in the optimal case which is exactly the time between two walls, this is more efficient overall, instead of adjusting the step to avoid passing them.

However there is a still a problem since this scheme has a minimum of 8 steps per carrier period. A better collision detection scheme could reduce the number of accepted steps per period to 6, and the number of rejected steps would also be 6. This requires more logic to handle double crossing which then allows to remove the walls. We did not implement this.

The case of several phases is exemplified in Fig. 14 with two phases. This is much like Fig. 12 except that the exact solution is not drawn. The main issue here is that after re-doing the step, another phase could have crossed the carrier. This is illustrated in "Alternative 1" phase 1 is detected to have crossed the carrier before phase 2 so we backtrack to t_k and step to t_c. However, during that step, phase 2 crosses the carrier between t_k and $t_{col.1}$. However, we never reject small steps in our current implementation so $t_{col.1}$ is accepted. Therefore, we trigger the switch on both phases 1 and 2 at $t_{col.1}$. A step is then taken to t_{k+2}, where the phases are at the very top and very bottom. For "Alternative 2", the step to t_c causes phase 1 to cross, but phase 2 does not. In this case, we trigger the switch on phase 1 only.

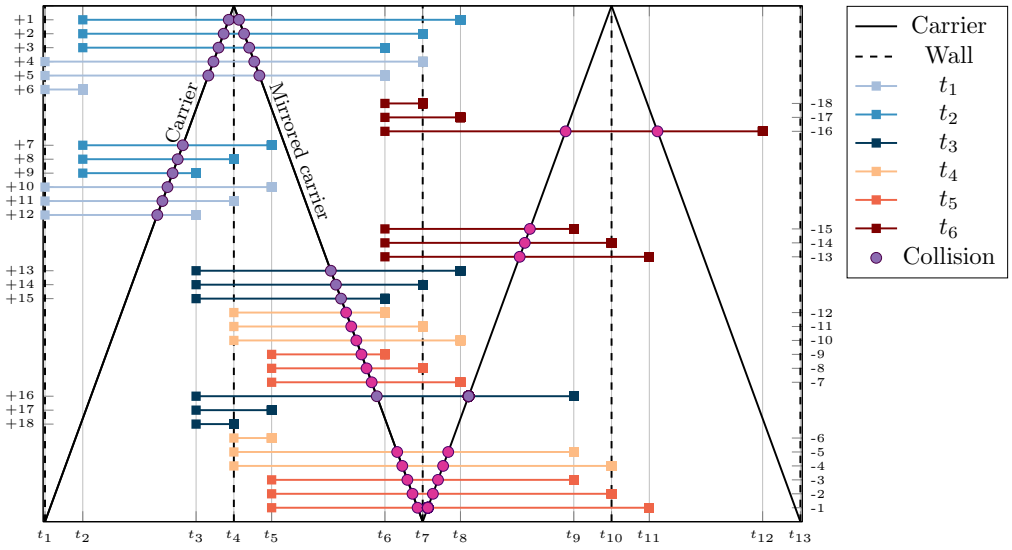

Figure 13: Description of all possible types of collisions between signal and carrier.

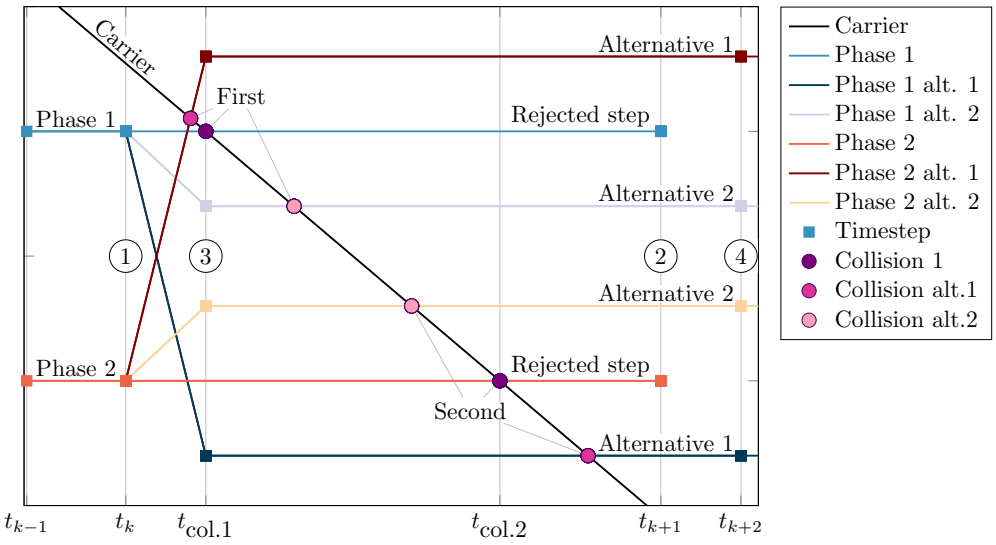

Figure 14: Collision detection for two phases.

Finally, the crossing times computed by our collision detection algorithm converges quadraticly to the exact collision times with the nominal time step h_*, from the maximum allowable step down to zero as shown in Fig. 15.

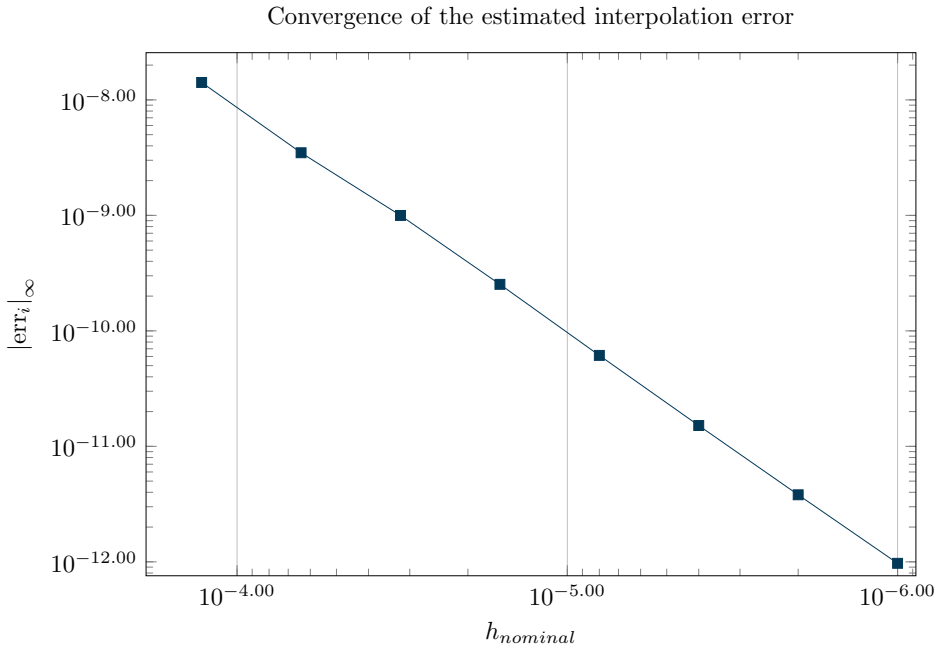

Figure 15: Convergence of event location

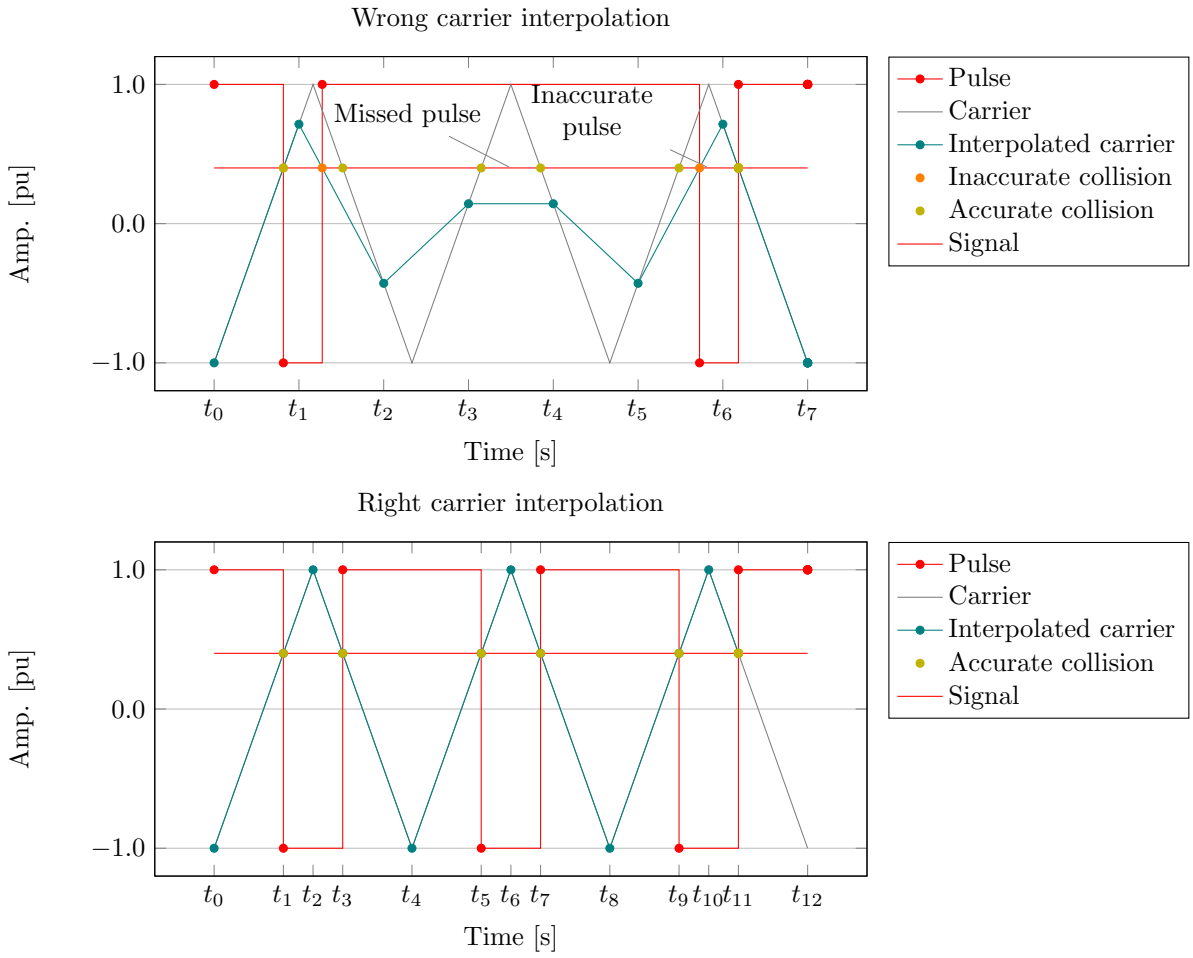

Figure 16: Shows the carrier interpolated with fixed time steps and variable time steps. The above pictures correspondence to how PSCAD interpolates the carrier and the bottom one how it is done in our method.

4.8 Time step selection

FiXme Note: describe graph better: average vs nominal step etc.

The maximum nominal time step is the minimum of half the carrier period and 1/20 of the filter period, because we must resolve all six collisions with the carrier signal, and should resolve the filter dynamics accurately enough. For the example circuit we used with 6Khz carrier frequency, and 400Hz filter frequency, the carrier period is the smallest so that our nominal step corresponds to approximately 40 steps per period of the filter. For the implicit midpoint integration method, which has second order accuracy, this is short enough to produce a high quality approximation of a sine wave.

The average step changes with the nominal step as illustrated in Fig. 17, where it is seen to increase up to critical points where increasing h_* has no influence. Here we consider two different filters of varying periods on the x-axis. We choose two representative filters with periods t_{d_1} and t_{d_2}, and a carrier with period t_{ca} such that $t_{d_1} < t_{ca} < t_{d_2}$ to illustrate the different regimes. We write t_{cr} for the optimal average step, i.e., the carrier frequency divided by the number of crossings, six in our example circuit.

From the origin to the red point Dyn.1, the average step is not dependent on the carrier frequency but we cannot go beyond t_{d_1} for stability and accuracy reasons. The for a lower frequency filter t_{d_2}, the average step increases linearly with the nominal step up to the critical point which is the limitation due to the carrier. The linear relation is an idealization, assuming high accuracy collision detection. If the noise level is very high however, that may not be the case.

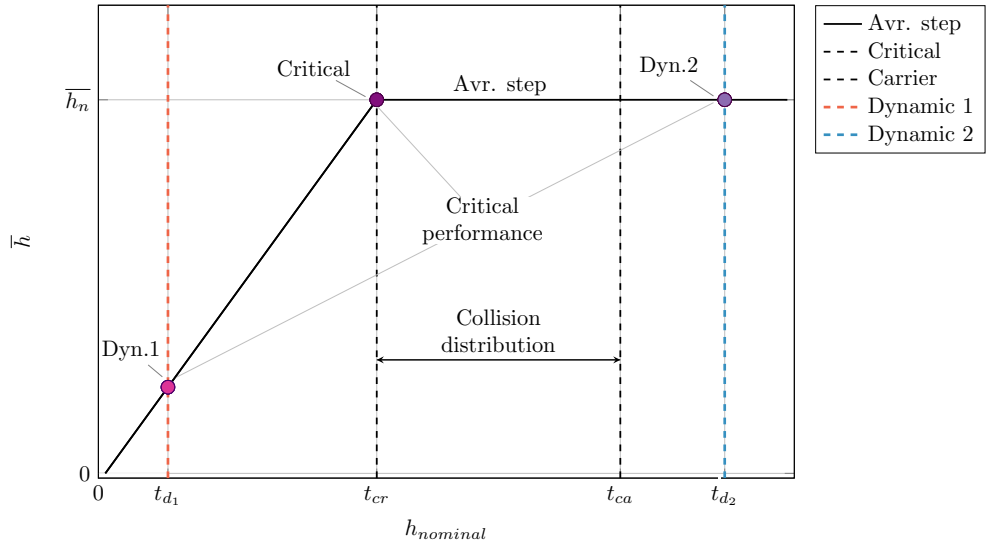

Figure 17: Critical performance of time step control.

5 Software implementation

The first guiding principle in designing the software is that a switched circuit is in fact a collection of similar circuits, each using the same components. The second principle is that in order to achieve performance, one must have good memory alignment as well as a minimum number of copy operations, if any.

At the basis of the software is the concept of "stacked objects" which contain multiple copies of the same variable. Depending on the state of a circuit, the suitable value is reported, used, written to.

What this means is that, say, a `stacked_double` variable is in fact a collection

of `double` variables. The object containing these acts and behaves like a `double` in all aspects, but it changes layer when needed. Layers are changed when there is a switching event.

What this means in particular is that during circuit construction, a single instance of a capacitor would be created and then inserted into one or several layers. This means there is absolutely no chance to inadvertently insert two different capacitors in two different layers where they should be identical. Circuit creation has proven nearly error prone.

From that point on, there is no difference between the different configuration, only an active circuit.

Then we reuse the linear algebra as much as possible and even though our variable stepping method requires refactorization, the configuration of each state of the global circuit does not change. We therefore use one matrix per configuration, and store the fill-in reducing reordering as soon as it is computed. This is used once only. Also, once this is computed, the address in the matrix where an object will write is given to that object, meaning that no copy is ever made. On top of that, we use a pointer access to the triangular factors of the matrix so that the transpose operation is performed once only, avoiding all data copy.

We wrote our own LDLT factorization matrix for the saddle point matrices described in Sec. 4.1 but used a proprietary implementation of a preemptive pivoting method [19], which is why the software cannot be published at this time.

Also in line with direct pointer access, we designed a submatrix class which behaves like a matrix in every respect, but is only a view into the system matrix. It allows writing data into the system matrix without any knowledge of the permutation or even the size of the full matrix.

The rest of the design is a standard box and arrow architecture. At the basis are filters which have ports. The directionality of these is irrelevant to the linear algebra operations so there is no assumption of causality here. We also used direct pointer access here so that a filter owns it's output ports, but the input ports are in fact pointers to another blocks' output ports. The danger of this design is alleviated with the smart pointers in the `C++` language and standard library.

For data handling, we used the HDF5 library systematically along with various buffering mechanisms to alleviate the poor performance when writing to disk. Because of the class hierarchy in which everything ultimately resolves to a block, we get systematic access to all the data and can read and write anything from and to a file.

Finally we used Intel's thread building blocks `tbb`[14] for parallelism which proved to be surprisingly easy to use and very effective.

We did not implement a parser to process netlists and this is a drawback. But not in the context of the present study.

6 Results and discussion

We now present results obtained from the prototype code briefly described in the previous Sec. 5.

6.1 Energy conservation

We start with a one phase circuit to simplify the analysis and illustrate the worse case scenario. We consider a linear LC filter of frequency 150Hz and a carrier frequency of 10,000Hz without any feedback control and used instead a regular pulse train with two pulses per carrier period, i.e., regular pulses of 50μs. We take the period of the pulse train as the representative time scale of the system, namely, $\tau = 100\mu$s. This configuration allows for the evaluation of the event handling, time step adaptation or interpolation, since the period of oscillation of the filter itself is much slower than the carrier's period.

In Fig. 18, we present an exact solution without any resistance which gives a base line for the performance of the variable time-step integrator. This is a good reference for our technique since we use ideal bi-directional switches meaning there is no resistance at all. The switches in PSCAD use IGBT models which require small resistances of the order of mΩ, 10mΩ being the default value. We set this to $10^{-9}\Omega$ which is possible for this circuit.

We measure the rate of energy loss as a function of time step by simulating over 10,000 carrier period. For the interpolation method, we used time steps from $\frac{1}{4}\tau \cdot 10^{-3}$, four times less than the recommended value from the users's manual of $\tau/100$, and up to $\tau/24$. Our method used τ throughout as nominal step, but that could have been as high as 70τ. We then divided the simulated amplitude by the reference solution. The decay rate is then the logarithm of this ratio with the simulation time.

Our software implementation of the adaptive time stepping technique does not loose energy, despite the inexact event location. Of course, the result is independent of the nominal step as long as the dynamics of the filters is slow compared to the carrier. However, as the time step increases, the interpolation method quickly looses energy as shown in Fig. 18. The energy decays exponentially with a time constant essentially proportional to the time step. We believe this is not specific to PSCAD but representative of any interpolation method. Using the recommended value of $\tau/100$, there is still a significant loss of energy. Our method takes four steps per period which is a twenty five fold increase in performance. Only at a step of $\frac{1}{4} \cdot 10^{-3}$ is the energy loss commensurate with the small resistor in the switches. But at this point, the performance ratio is then a thousand folds in favor of variable stepping.

We also computed the exact solution for an LRC circuit subject to regular square pulses as seen in Fig. 19 to see what is the effect of the parasitic resistances. We experimented with very small values of 1nΩ to see if dissipation was due to this resistance or the time stepping scheme itself. Our results show that indeed, the parasitic resistance plays very little role in the decay of the signal, but that interpolation causes rapid energy losses even for moderate time steps.

The same experiment is presented in Fig. 19 in the time domain for selected values of time step. We plotted the rolling RMS of the capacitance voltage to remove the overall high amplitude sine wave. The zoom-in shows how the simulated dynamics between adaptive and fixed step methods converge. The difference with the analytic solution is due to sampling errors.

We also simulated an LRC circuit to see what happens when more practical values

of resistances are used for the non-ideal switches. We used PSCAD's default value with R = 10mΩ. For that case, the artificial dissipation due to the interpolations during event detection is not as severe, differing by only a few percent from the exact solution, unless unreasonably large steps are taken. This is shown in Fig. 18 where we used a rolling RMS value of the voltages to remove the overall sine wave in the signal.

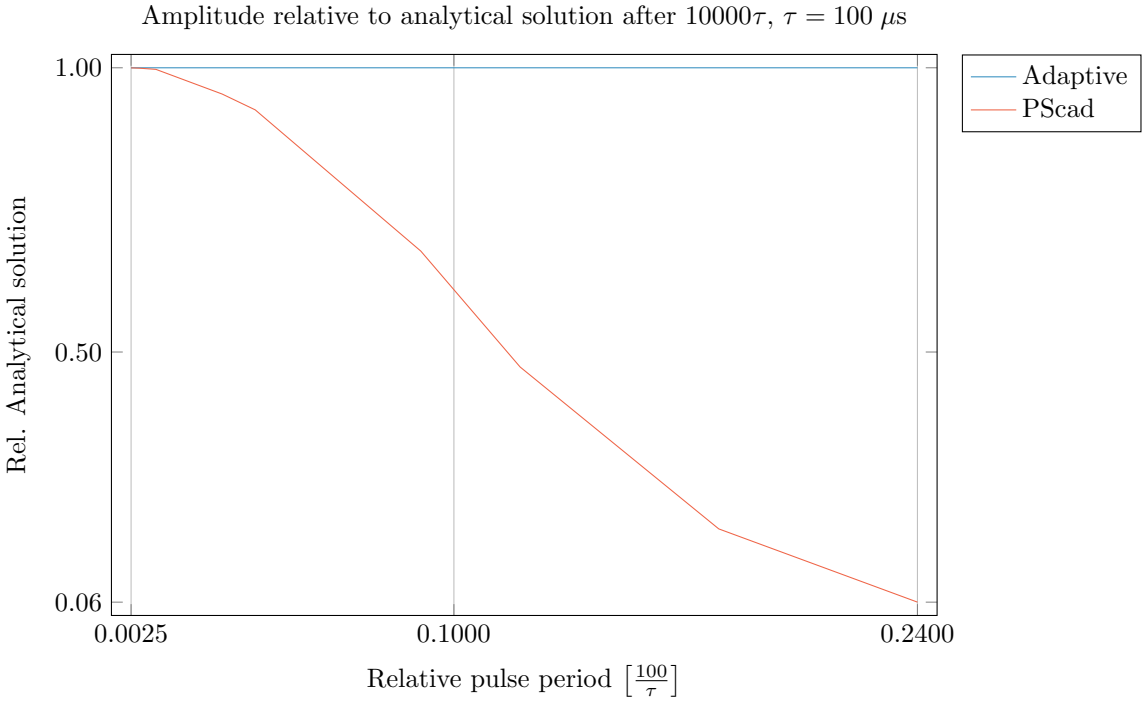

Figure 18: Energy conservation in a single phase circuit subjected to regular pulses.

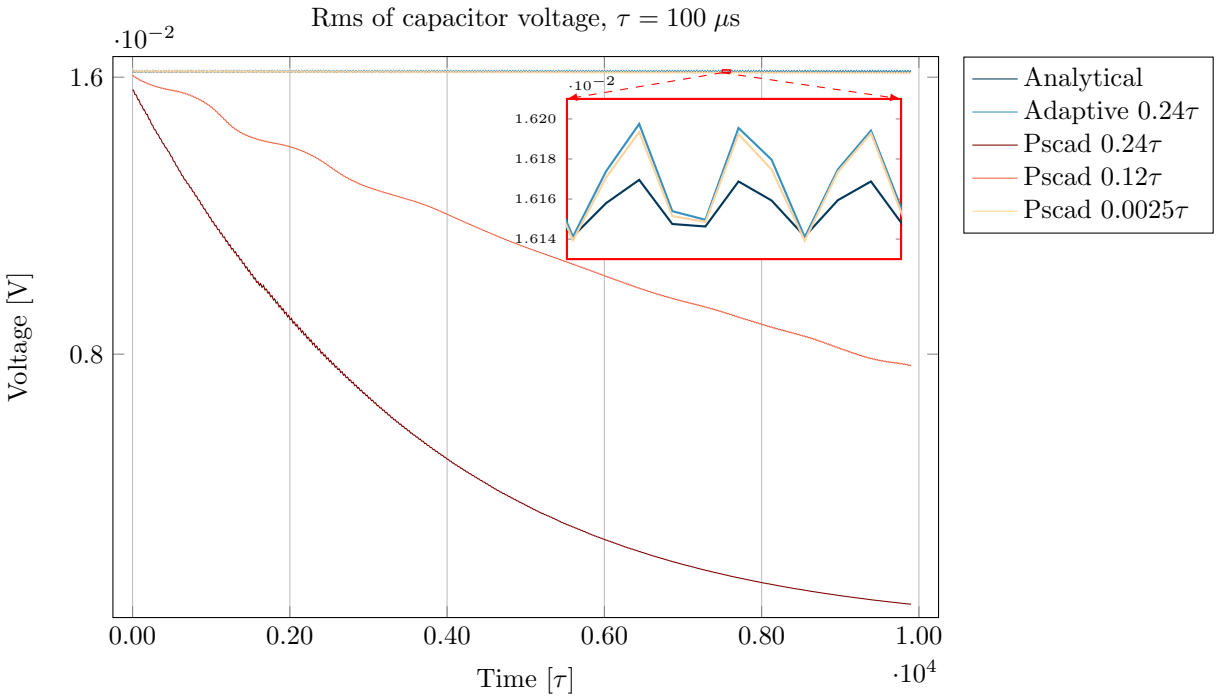

Figure 19: Energy conservation and dissipation for different time step. Rolling RMS value of the voltages are plotted to remove the overall large amplitude sine wave.

Our conclusion is that there is excessive energy dissipation non-ideal switches are used, and especially when interpolation is used unless unnecessarily small steps are used and this shows the usefulness of a variable time step method. In particular, as long as event location is performed correctly, the variable time step method can preserve energy with relatively large average step, the size of which depends only on the desired accuracy for the dynamics of the filters. The use of ideal switches helps also, even though real circuits would have resistance. We believe that a circuit should be simulated faithfully and that ideal switches must be available when desired. Energy preservation has repercussions on the spectrum as shown below.

6.2 Time domain analysis

We compared the output our method with that of PSCAD which is a good representative of fixed step methods. Since it uses interpolation, we used the effective average step by counting the interpolation steps as well as the full steps. This yields an average step by dividing the total integration time by the total number of steps taken. We did the same for our method. In Fig. 20 we present time domain data for the grid current.

We adjusted PSCADs step size as well as our nominal step so we could compare the accuracy for similar amount of numerical work. It is clear from the zoom-in windows is that we do not converge to exactly the same solution. This could be due to small difference in the circuit configurations and the non-ideal vs ideal switches. Nevertheless, the time-steps taken by PSCAD have to be much smaller before the results are comparable which was expected. The convergence rate also appears to be slower for PSCAD, especially in the transient regime. We believe this is due to missing pulses because of the method described in Sec. 4.7, even at relatively small steps. There is also a much slower convergence of the reference signal as seen in Fig. 22, for reasons unknown. On the figure for the reference signal in Fig. 22, the left most zoom-in contains the carrier signals in grey lines. We see here that the data from PSCAD is not at the right location in time because of the interpolations. This has consequences in the spectrum shown below in Sec. 6.3.

The real power delivered on the grid is in Fig. 21 and the convergence rate of the PSCAD solution is much worse here than for the current of the inverter or the control signal. Ours converge quickly and from this graph at least, the 21μs average step seems sufficient to produce a good signal, given the relative errors between the different simulations.

Because of the qualitative converge rate, we can only conclude that variable time stepping gives more accurate solutions, and can do so even at large step size.

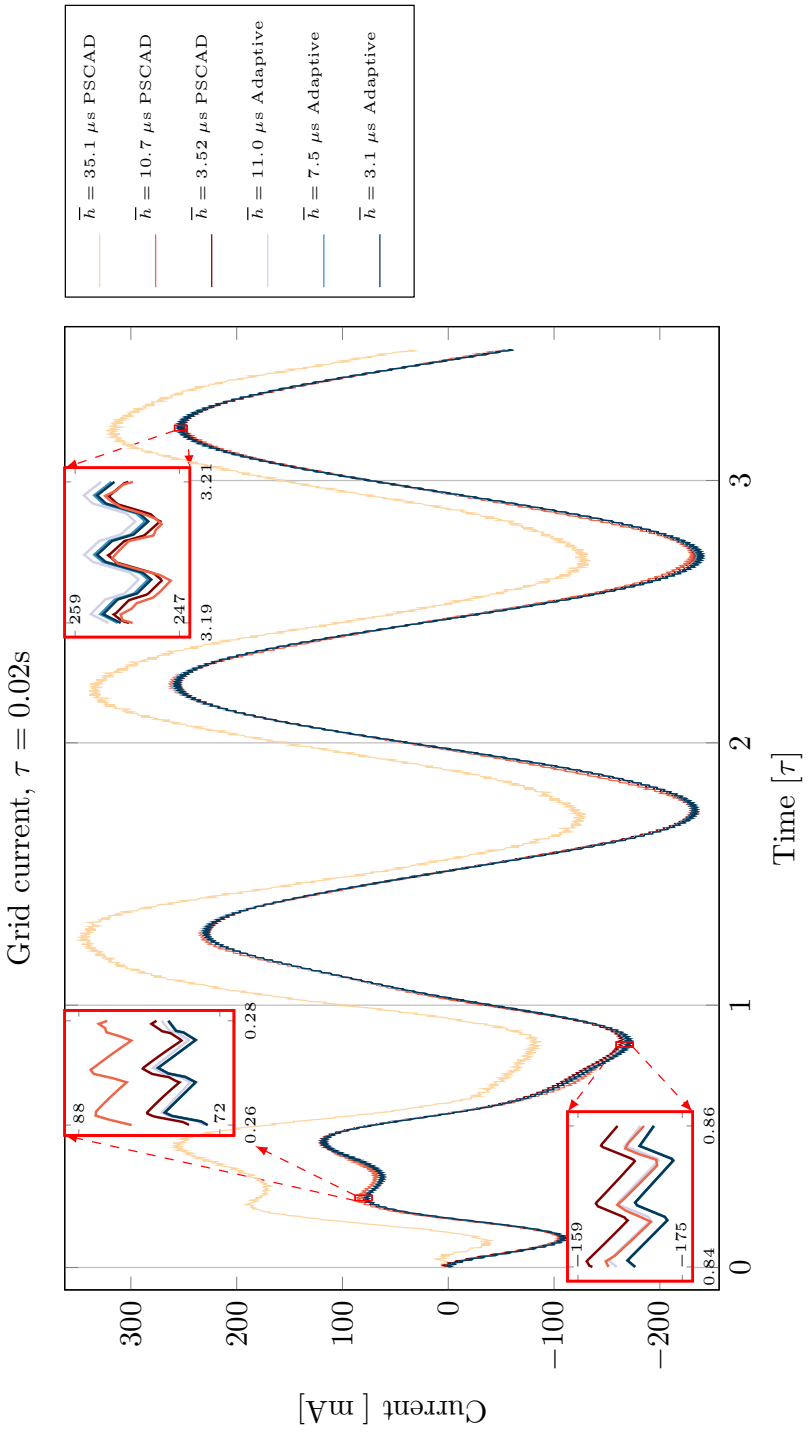

Figure 20: Grid current

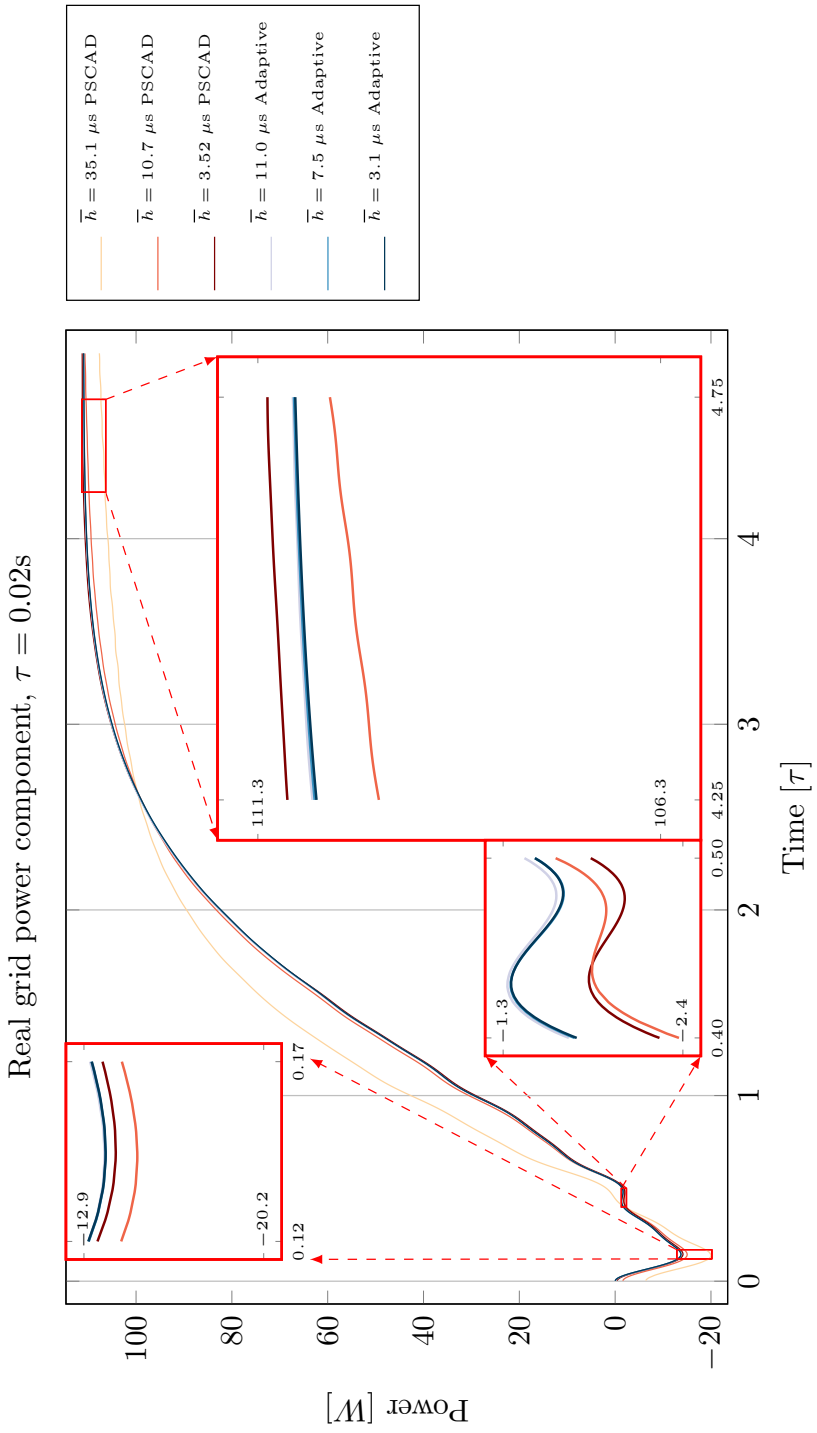

Figure 21: Power delivered to the grid

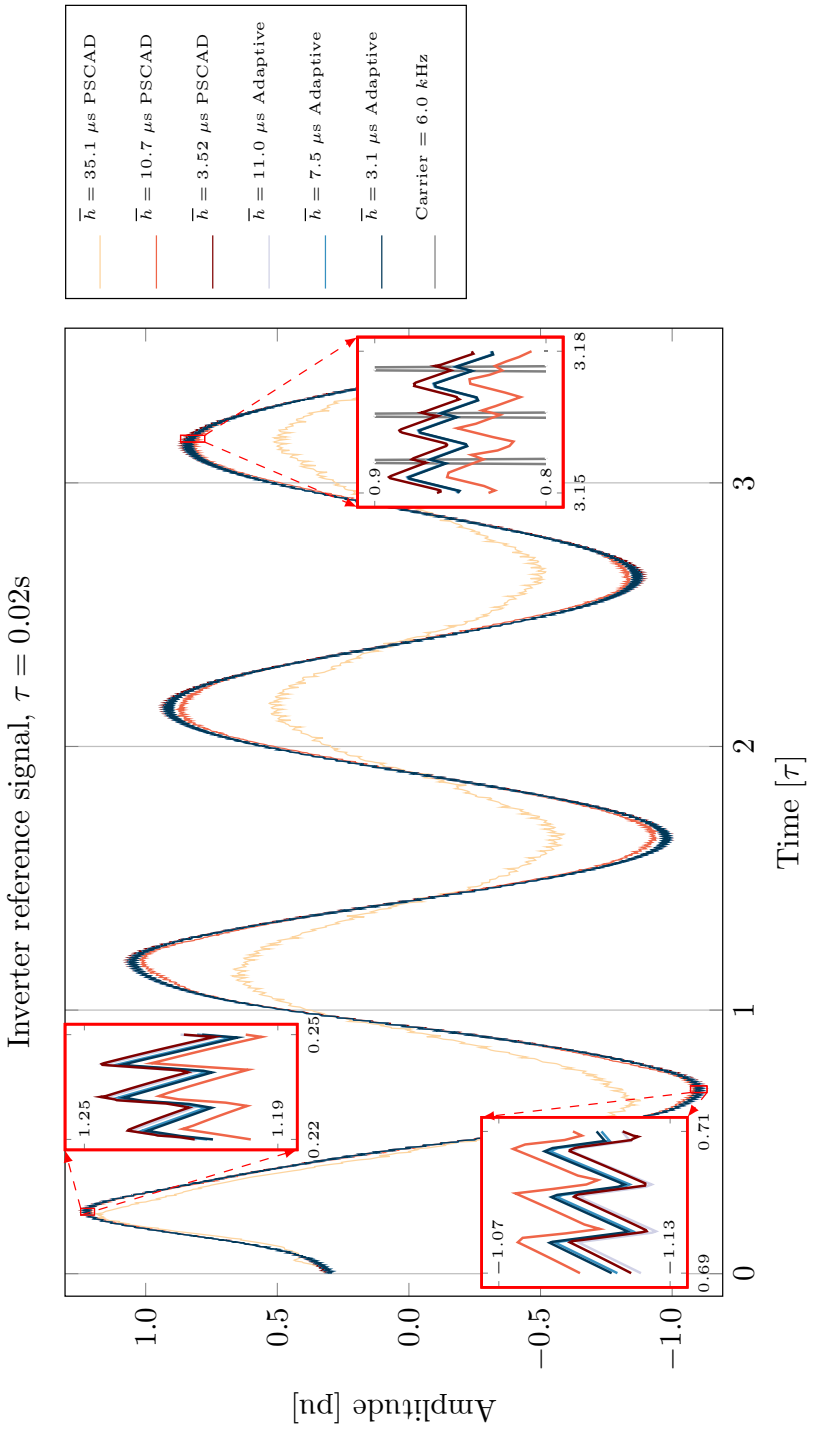

Figure 22: Transients of the reference signal

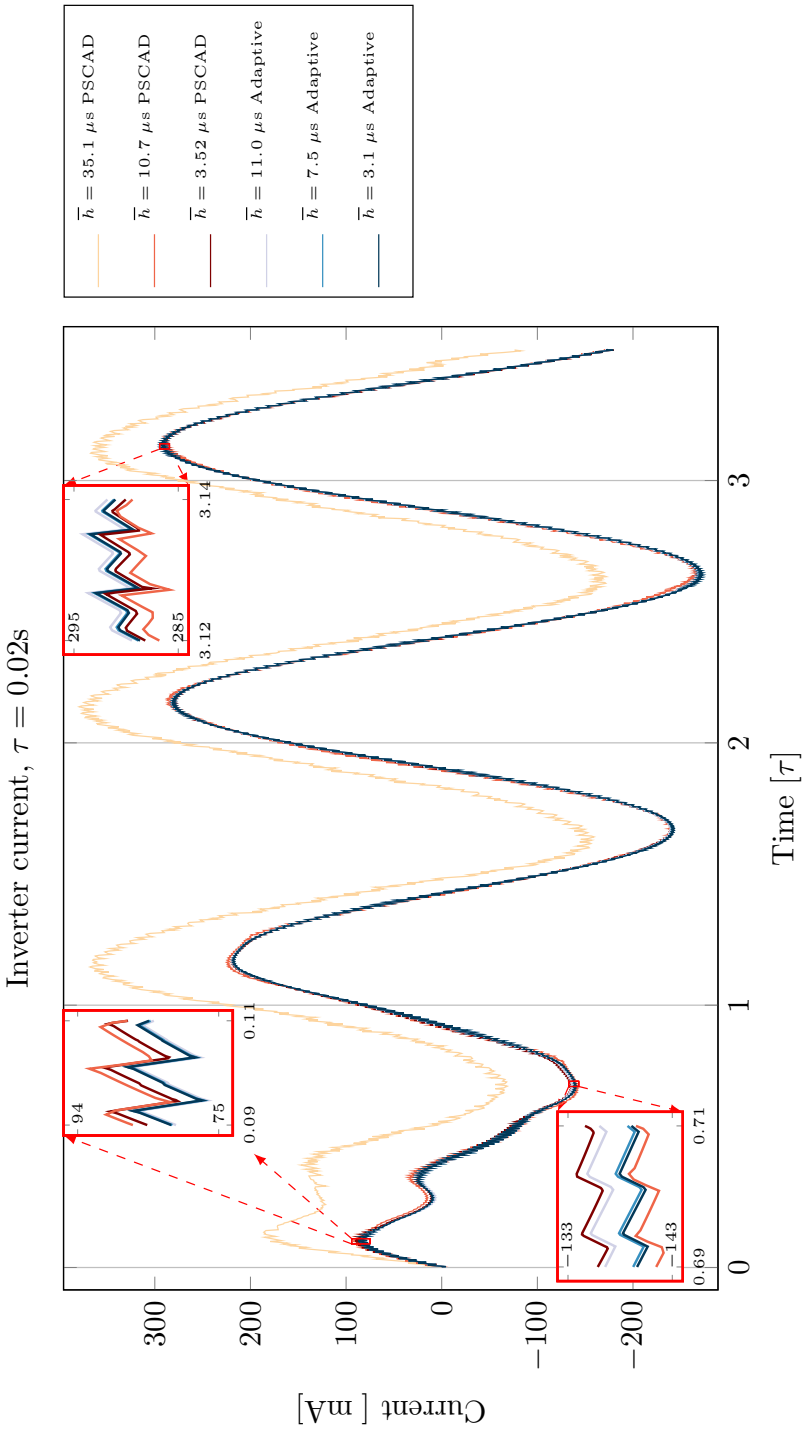

Figure 23: Transients of the reference signal

6.3 Spectral analysis

The properties of the spectrum of the output signal is the most important aspect of the simulation output. The noise and harmonic levels of the simulated data should match those of the real circuit being designed. The numerical method should not hide defects or add anomalies.

Fourier analysis of irregularly sampled signals is difficult. Non-uniform Fourier transforms using NUFFT produced too many artifacts and spurious results. We opted instead for the Fast Fourier Transform using a linear interpolation to project our data onto a regular grid, choosing an suitably small time interval.

We present the spectra of the grid current in Fig. 24, the inverter current in Fig. 25, and reference signal in Fig. 26.

For the grid current spectrum, the adaptive stepping method has very low amplitude in the low frequencies as it should be, without artificial filtering. Since we have less noise, we have up to 0.2dB amplitude at the fundamental and harmonics. For the PSCAD signal, the slow rate of convergence is even more striking than seen in the time domain. However, the peaks are not shifted.

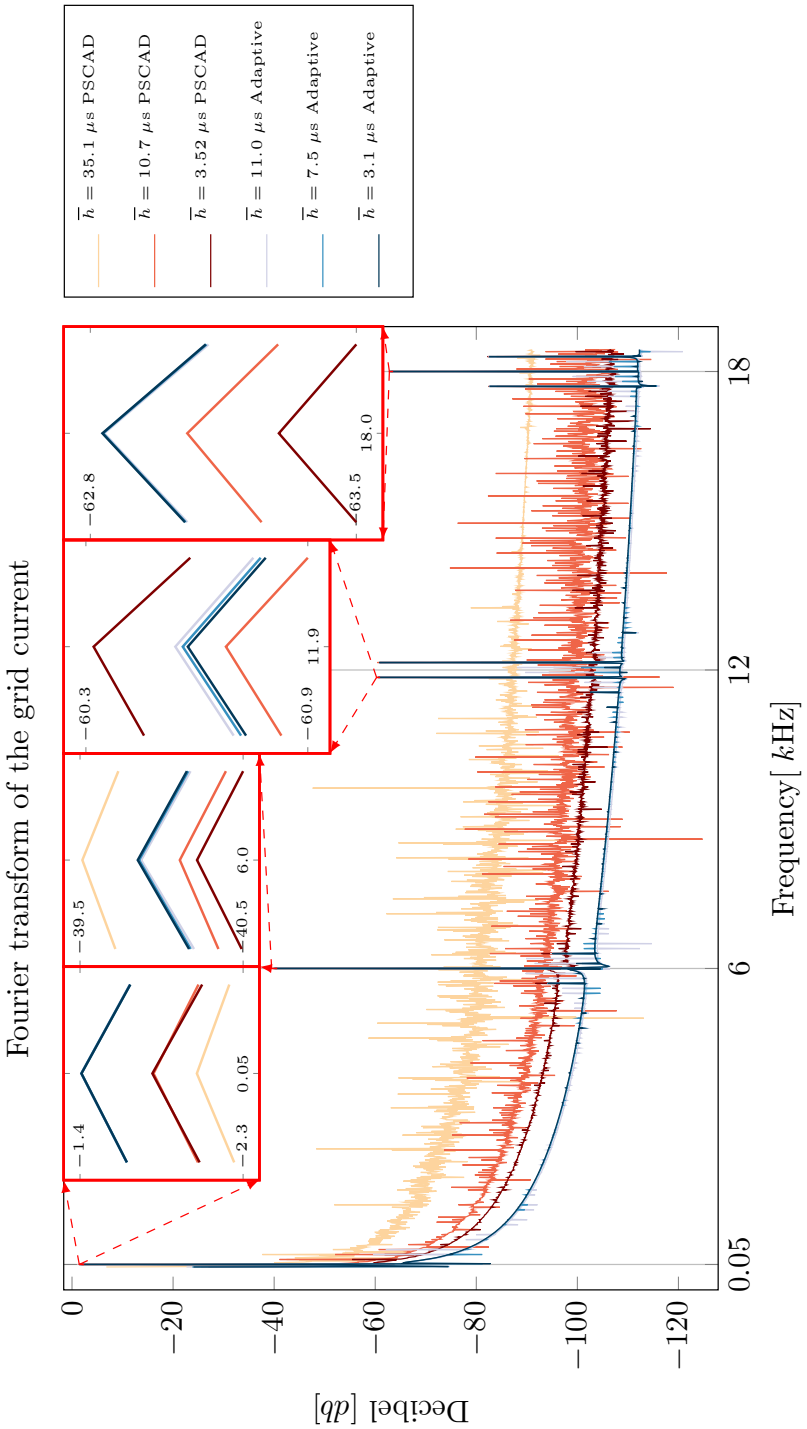

Figure 24: Spectrum of the grid current

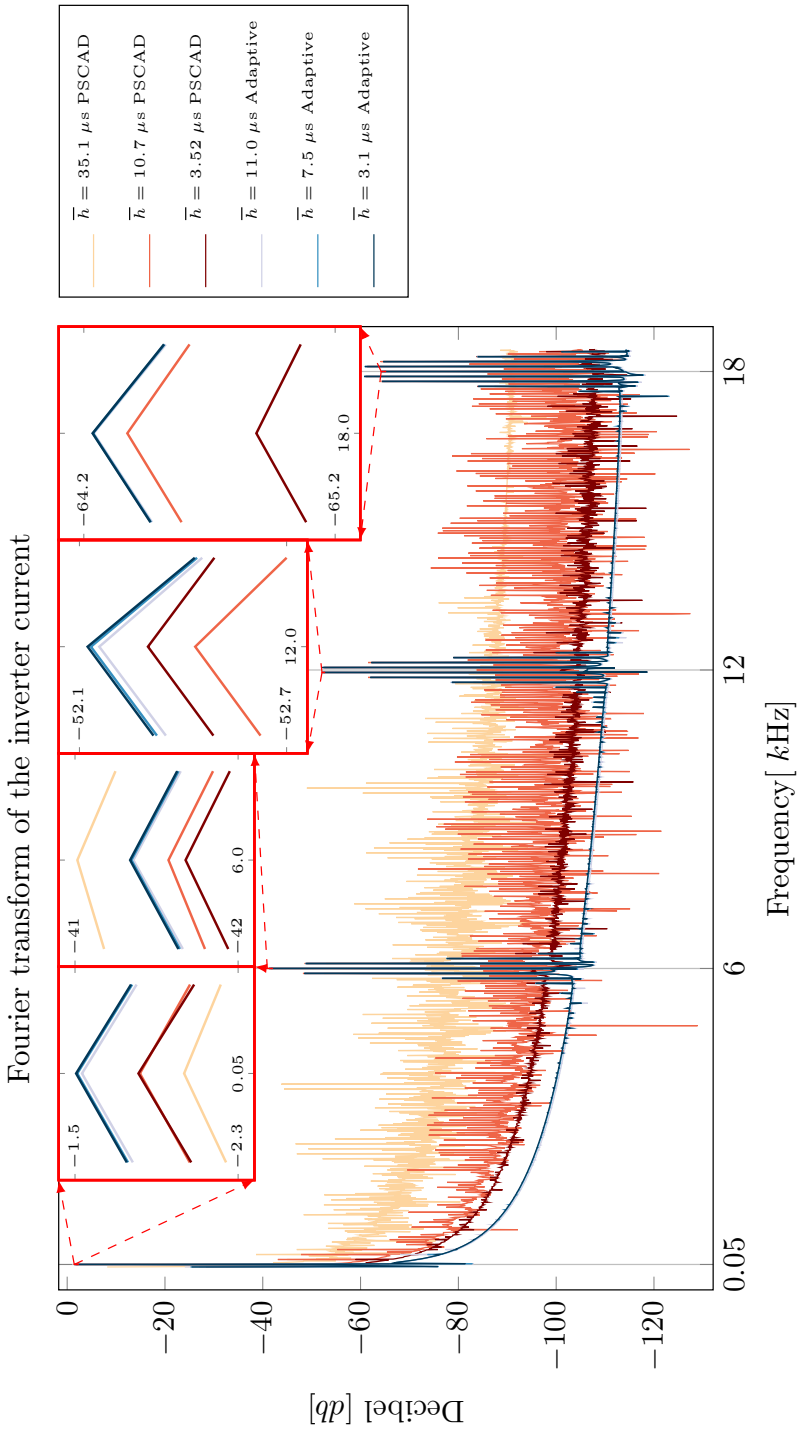

Figure 25: Spectrum of the interver current

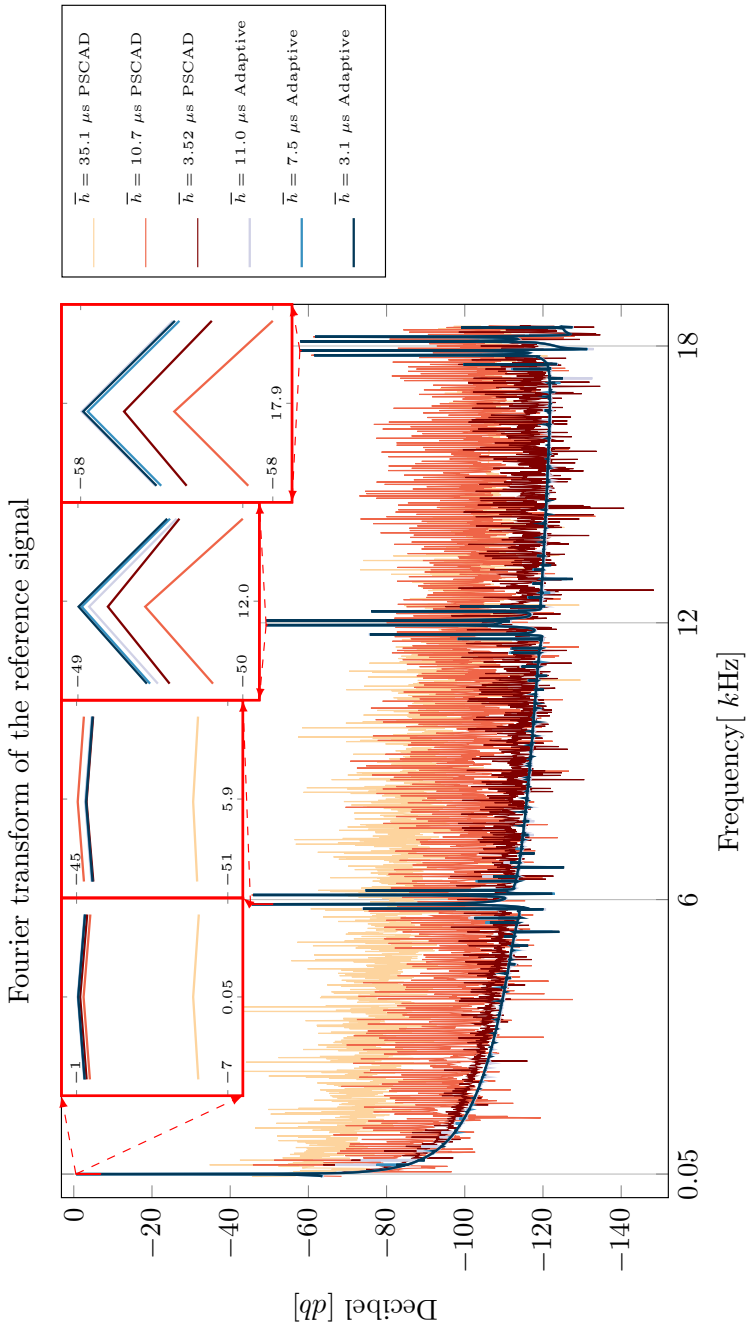

Figure 26: Spectrum of the reference signal

6.4 SVM circuits

SVM controllers do not require collision detection since the controller generates sequences of switching times. These are therefore easier to implement and simulate since no backtracking is necessary for the case of a variable stepping scheme, unlike for fixed step and interpolation methods. Our circuit uses a non-ideal power source with a finite capacity and resistance. In this case, switching produces impacts because RC sub-circuits become disconnected from the inductive part of the circuit, i.e., LCL filters. The reason for the impacts is explained in Sec. 3.2. We found that this introduced significant difference in the dynamics of the circuit, closely matching analytic solutions when these could be computed. We used a Simulink implementation as a reference which ran at fixed step. This has obvious limitations since the SVM filter can produce switching sequences arbitrarily close. We could not do this implementation in PSCAD because the free version we used was too limited and we could not get access to a full one in this project. We chose time steps of $1, 2$ and 10μs in the Simulink simulations to test convergence. Steps below 1μs are unreasonable because these simulations are very slow, taking more than five minutes to simulate just one second on a commodity desktop computer, compared to less than one second for our software.

The treatment of impacts requires the solution of an extra linear system of equations but in this case, the matrices are smaller, and their factorizations can be reused and so that resolving impacts comes at little cost.

The power spectrum of the grid signal appears in Fig. 27. Our method shows good convergence and a large average step. Impacts reduced this by half of the theoretical value, as would be the case if the voltage source was ideal, which does not require impacts. Moreover, the fixed step method does not appear to converge to our solution either in the asymptotic regime. Differences in the early parts of the transients are due to slightly different conditions which depend on the software implementation. What is obvious from Fig. 27 is that there is more noise in the SVM circuit, and that the peaks at the fundamental and harmonics are spread, unlike the case for the PWM circuit in Fig. 24. The fixed step implementation however converges much more slowly, even with steps twenty times smaller. In particular, there is a large amount of spurious noise near the fundamental, as well as between the peaks. Less energy is lost between the harmonics at smaller step, but the noise near the fundamental is large. This is explained by the fact that fixed steps act as filters, since they cannot resolve the high frequencies. But even at the fundamental, the amplitude is nearly 12dB lower for fixed step compared to variable stepping, i.e., an underestimate of the peak by a factor of more than 15. Similar difference appear all the harmonics.

The time domain signals in Fig. 28 shows once more than our method converges quickly and does not require very small nominal time step, i.e., performs nicely at large average step. The fixed step version does not converge to the same value as our method. This is also visible on the time plot of the voltage source in Fig. 29. These anomalies are significant if one wants to rely on simulation to validate a design, especially if they are visible on very simple circuits.

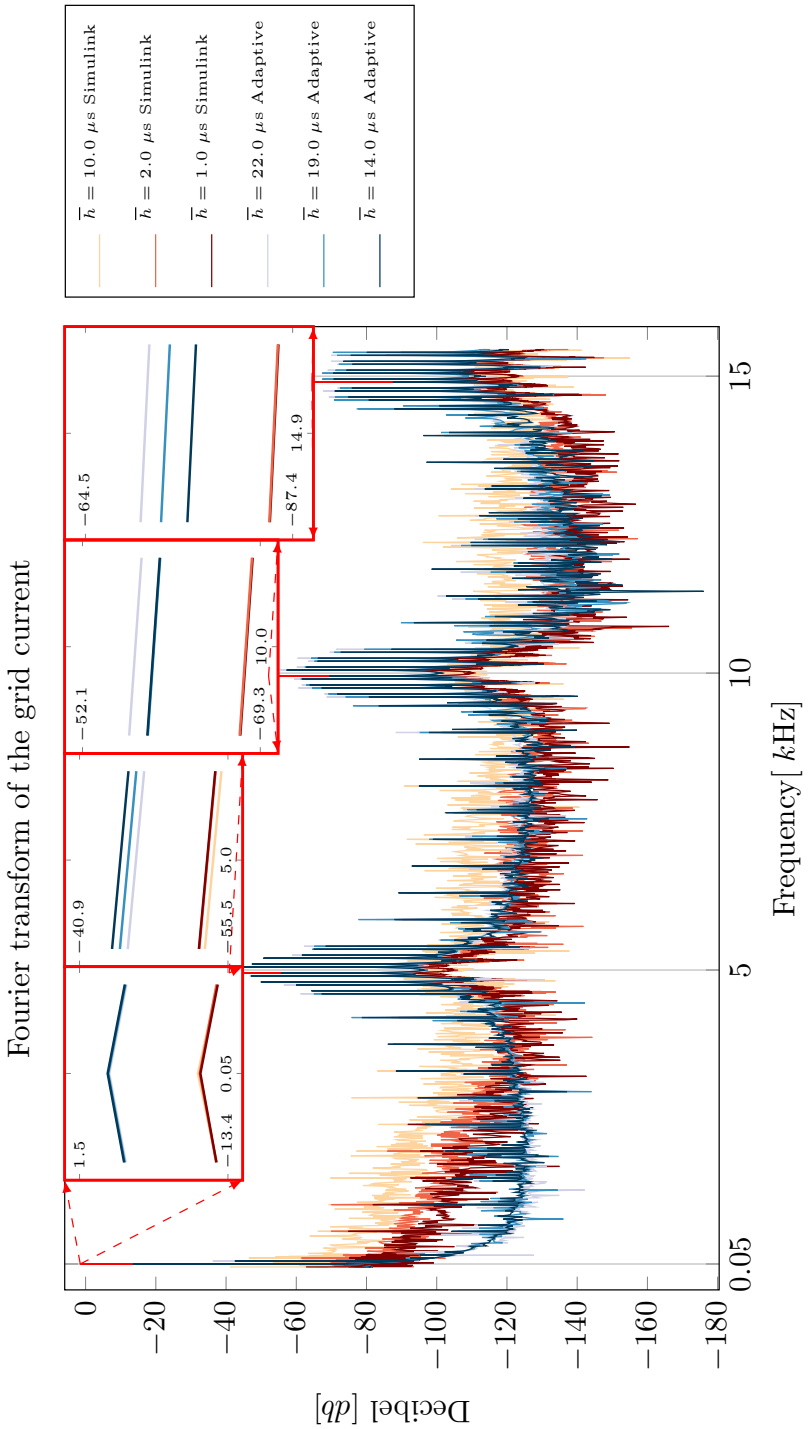

Figure 27: Frequency spectrum of the grid signal for an SVM inverter.

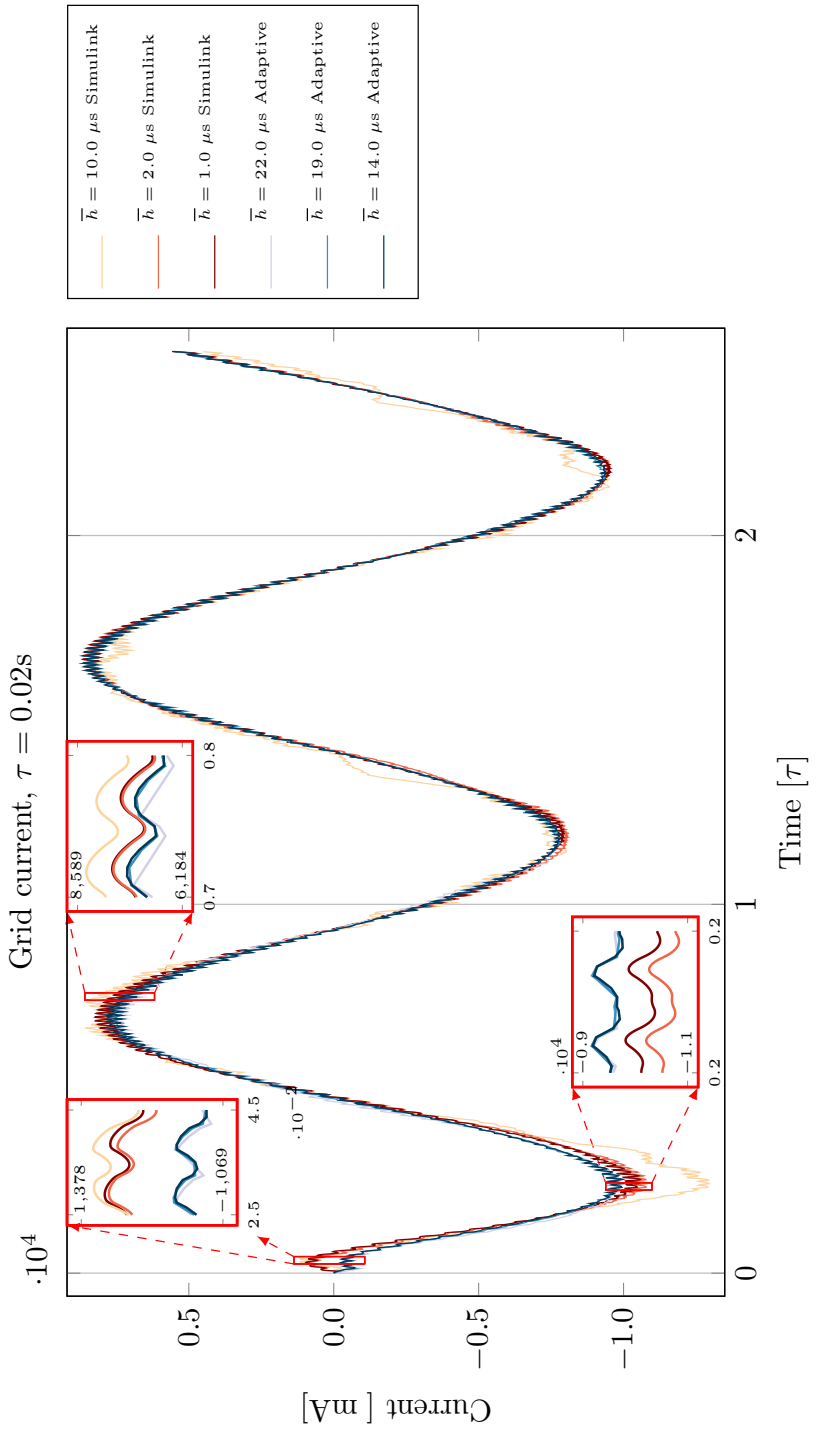

Figure 28: Time domain grid signal for an SVM inverter.

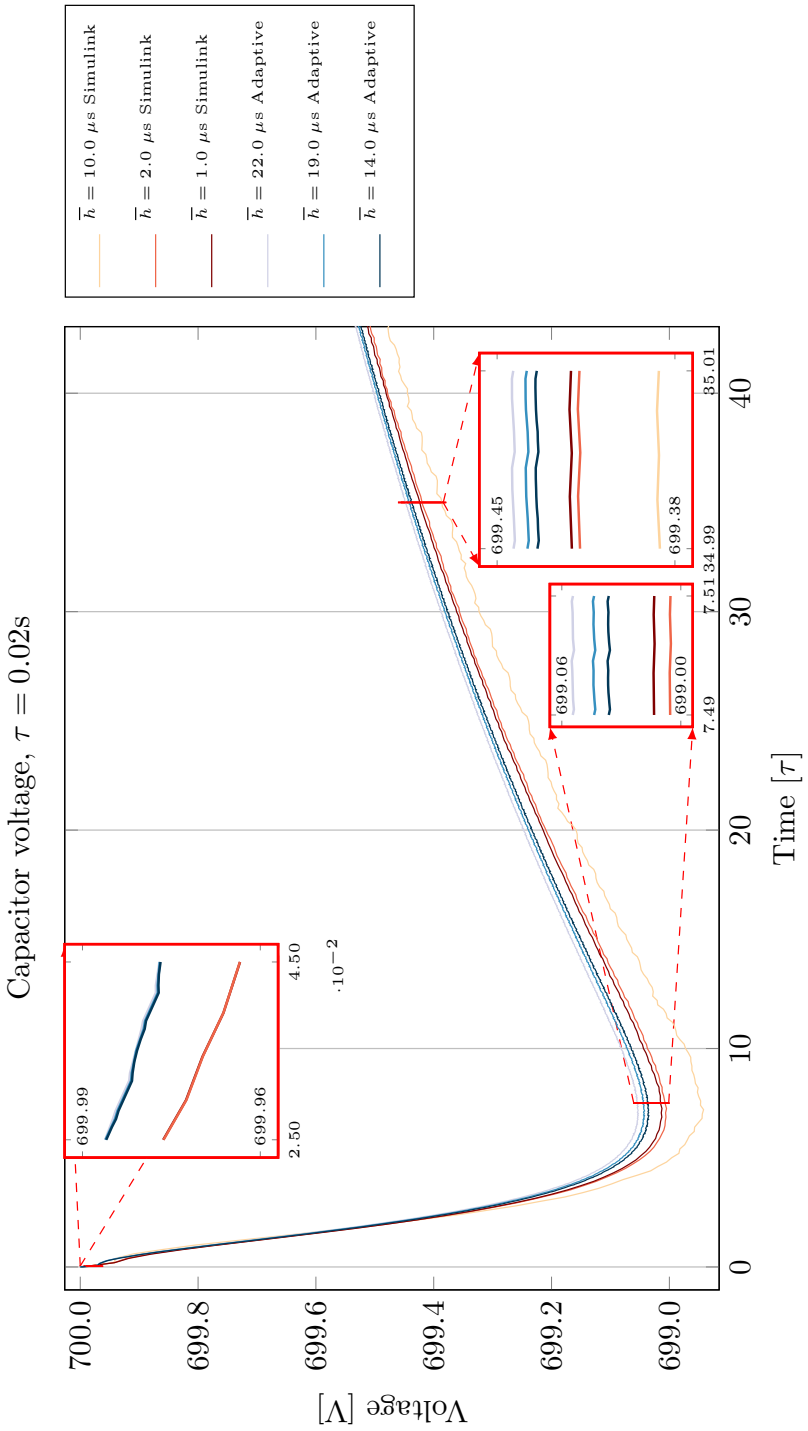

Figure 29: Time domain voltage of the DC voltage source for an SVM inverter.

6.5 Timing results

We simulated both PWM and SVM circuits, and used the same filters in either case with 350Hz natural frequency since this does not matter up to approximately 1000Hz as per Fig. 17.

Since we used ideal voltage sources for PWM, we did not perform impact computations. Our SVM circuit however used a non-ideal source which means that impacts were computed at every step.

We simulated 1s of dynamics and the performance of our method is summarized in Tab. 1. This shows that real-time performance is achievable for SVM circuits in particular. There are many opportunities for optimization which we did not pursue.

In table 1, the "Control freq." corresponds to either the carrier frequency for PWM, or operating frequency for SVM.

Inverter model	Control freq.	Filter freq.	$h_*(\mu s)$	time (s)
PWM variable	2 KHz	350 Hz	500	0.4
PWM variable	5 KHz	350 Hz	200	1.0
PWM variable	6 KHz	350 Hz	166	1.2
PWM PSCAD	6 KHz	350 Hz		
SVM variable	5 KHz	350 Hz		0.5

Table 1: Timing data for PWM and SVM interters

6.6 Parallel performance

We implemented the strong coupling algorithm Alg. 4.2 in Sec. 4.6 and performed tests on the Akka and Abisko supercomputers at HPC2N `www.hpc2n.umu.se` which are up-to-date machines, having reached high rank in the 100 list when they were inaugurated. Details of the configuration is on the website. An important feature is that Abisko which is more recent than Akka is lower, with clock frequencies of 2.6GHz and 2.5GHz respectively, but Abisko has more cores per node than Akka, 48 and 8 cores per node, respectively. This affects performance in different aspects depending on the amount of cache available, 6MB L2 and 6MB L3 for Abisko, compared to 12MB of L2 but no L3 for Akka. We could only use one node because we did not do an MPI implementation.

We only used an SVM inverter with the properties described in Sec. 6.5.

The run-time performance on an SVM circuit starts in Fig. 30. Given that each inverter needs 4 threads to compute the Jacobian according to Alg. 4.2, the number of threads needed for 1, 12 and 24 inverters are 4, 48 and 96 respectively. The reason for the slowdown in comparison to the results in Tab. 1 is likely due to memory traffic.

We then fixed the number of cores and tested the scalability of the algorithm itself and results are shown in Fig. 31. We have very nearly linear scalability when the number of threads is sufficient. This indicates that our memory utilization is good, and demonstrates that our stacked design is effective, despite complexity.

Speedup data is shown in Fig. 32 and shows that parallelization is effective at least up to 24 inverters. The fluctuations before 10 inverters or so are presumably due to memory traffic.

Finally, Fig. 33 is the relative speedup with respect to processing resources showing that we can accelerate our method up to 20x on current CPUs using threads for 24 inverters.

An MPI implementation would of course go beyond.

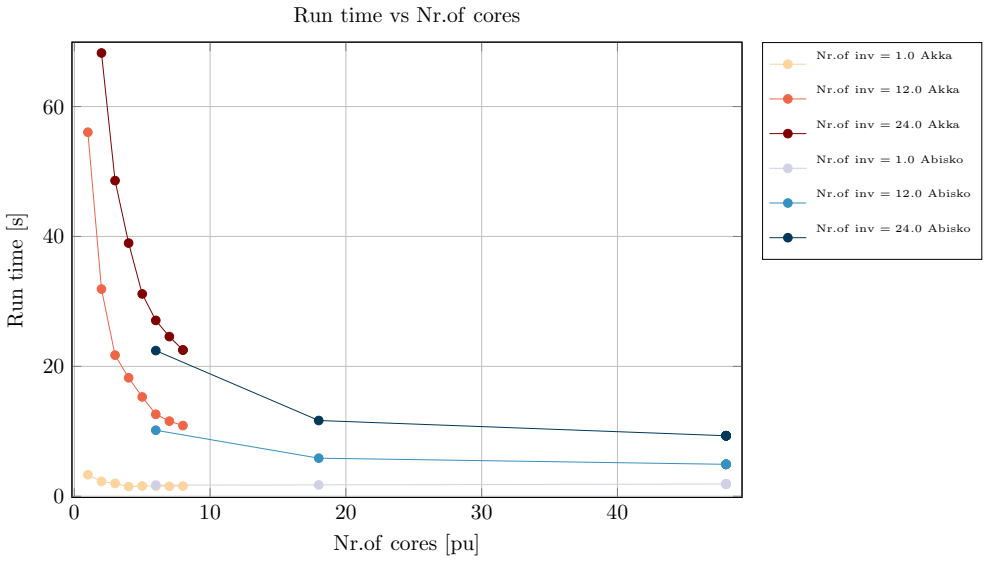

Figure 30: Parallel runtime performance

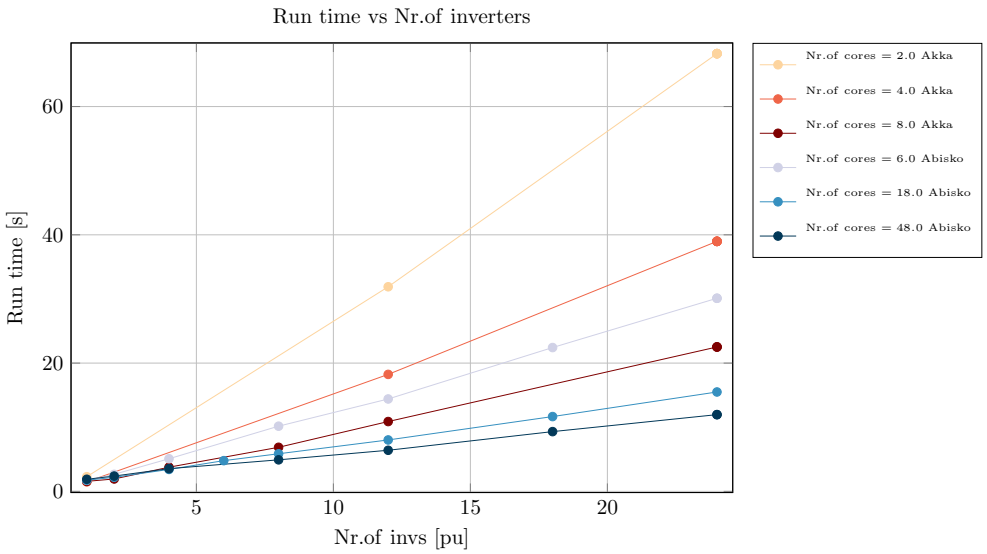

Figure 31: Timing for multiple inverters

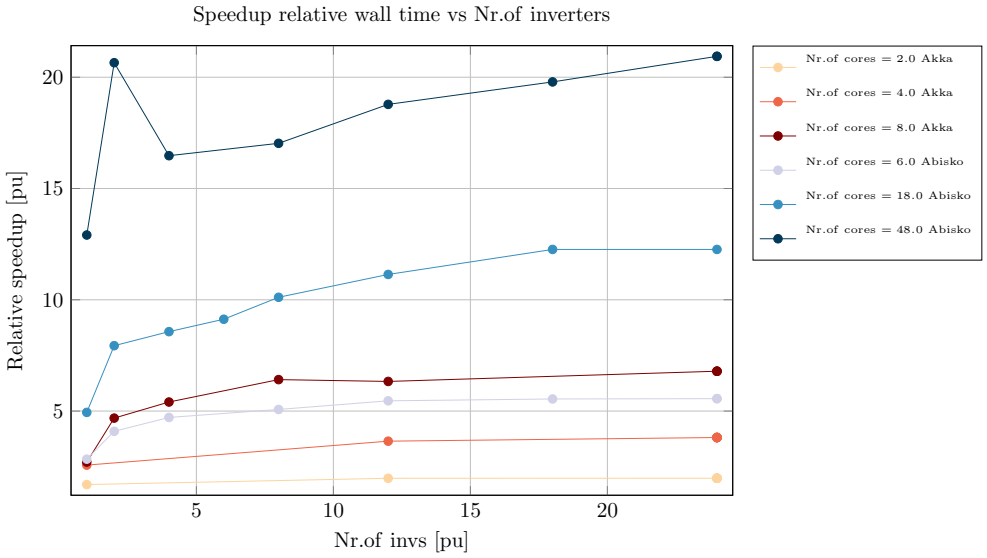

Figure 32: Speedup

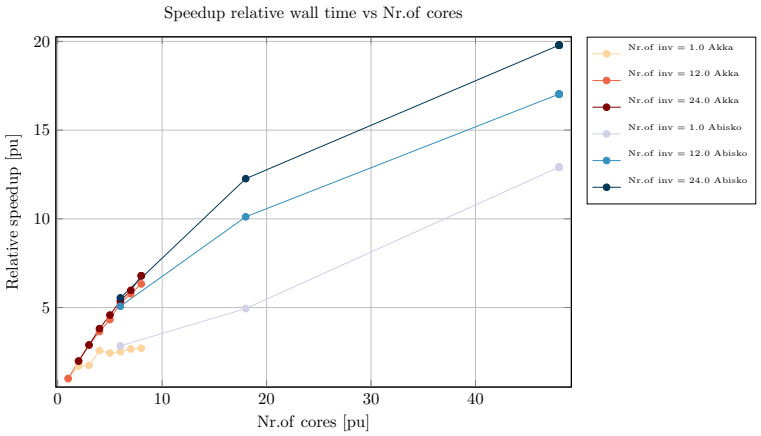

Figure 33: Speedup

6.7 Accuracy of the multirate method

We now describe the quality of the solution of the multirate method. The grid current is found in Fig. 34 and shows good agreement between a circuit with two inverters simulated with the standard method, we call this global henceforth, and the same circuit simulated with the multirate method. Note that the average step for the global simulation is one half that used in each inverter in the multirate since all collisions of all six phases are detected and processed. This means that it is nearly impossible to converge to the exact same solution, though this is to be expected. What can be seen in the zoom-in is that there is convergence however.

The differences are more noticeable on the voltage in Fig. 35 where much of the noise is removed because of the simplistic approximation of the grid signal in each inverter between synchronization point. We believe this can be fixed and will present solutions in the future.

A problem appears however as shown in Fig. 36, which is due both to the connection of several inverters at a common node and to the multirate method where the issue is aggravated.

Assuming an inverter with output impedance of 1Ω. Coupling n such inverters to a common node will decrease the effective impedance to $1/n\Omega$ for each, meaning that they influence each other strongly, because they can easily accept currents from each other. This happens quickly. This is bad for the multirate case because of the extrapolation used between the large steps. In practical terms, since changing the output impedance of the inverters, on must adjust the grid impedance to a sufficiently small value given a number of inverters. At zero grid impedance though, the inverters do not affect each other making the simulation pointless.

What is seen in Fig. 36 is that the global simulation can fail, and the multirate one fails similarly for small global step. But at big global step, the multirate method becomes unstable, not surprisingly.

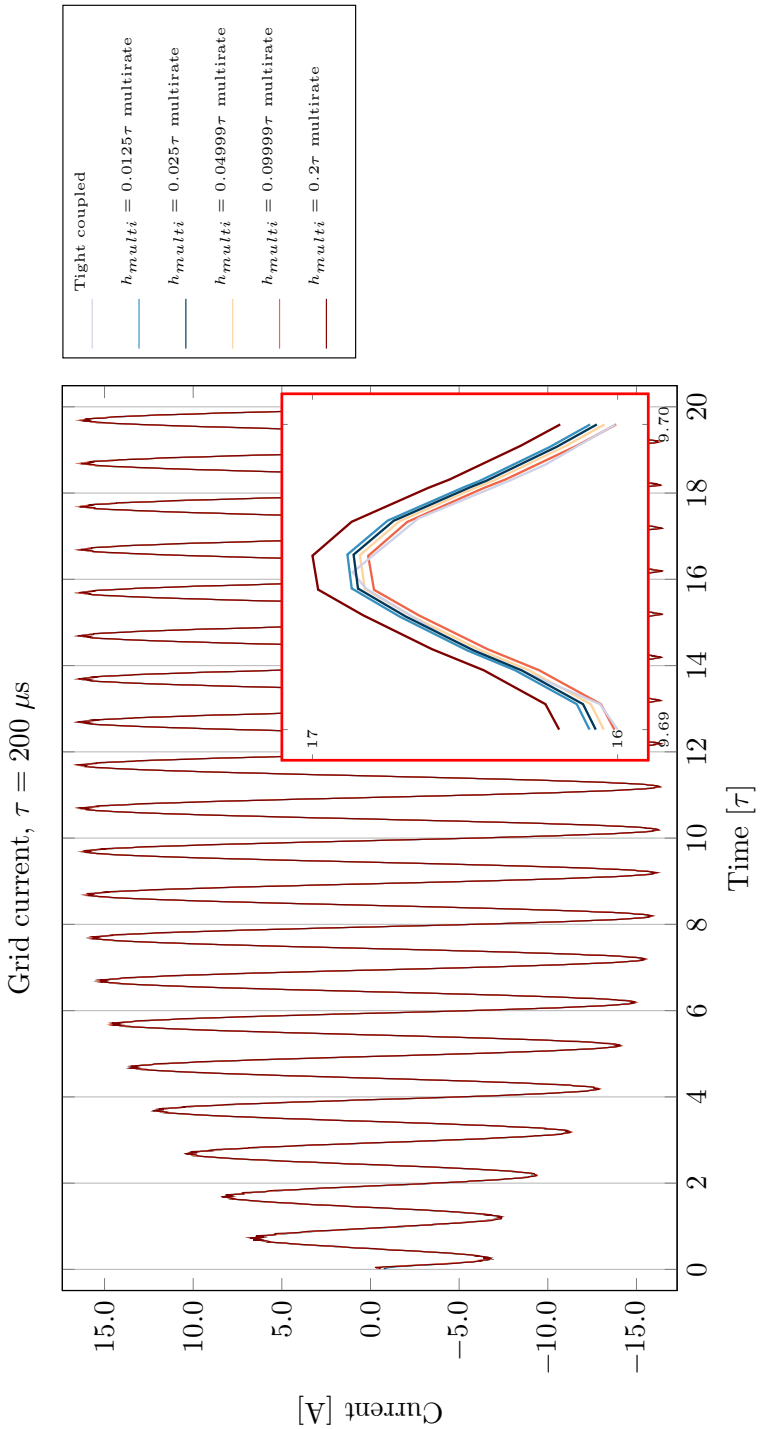

Figure 34: Grid current of the multirate method

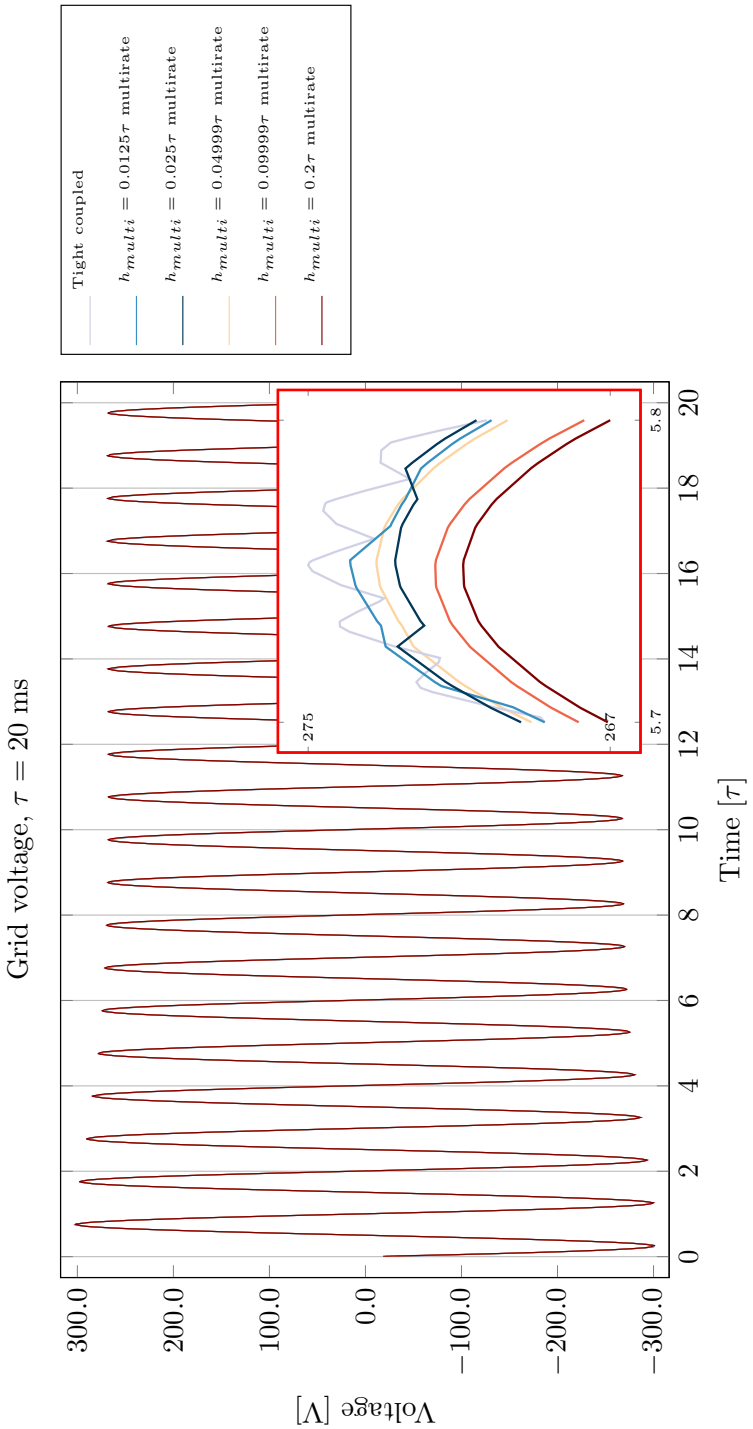

Figure 35: Time domain of multirate

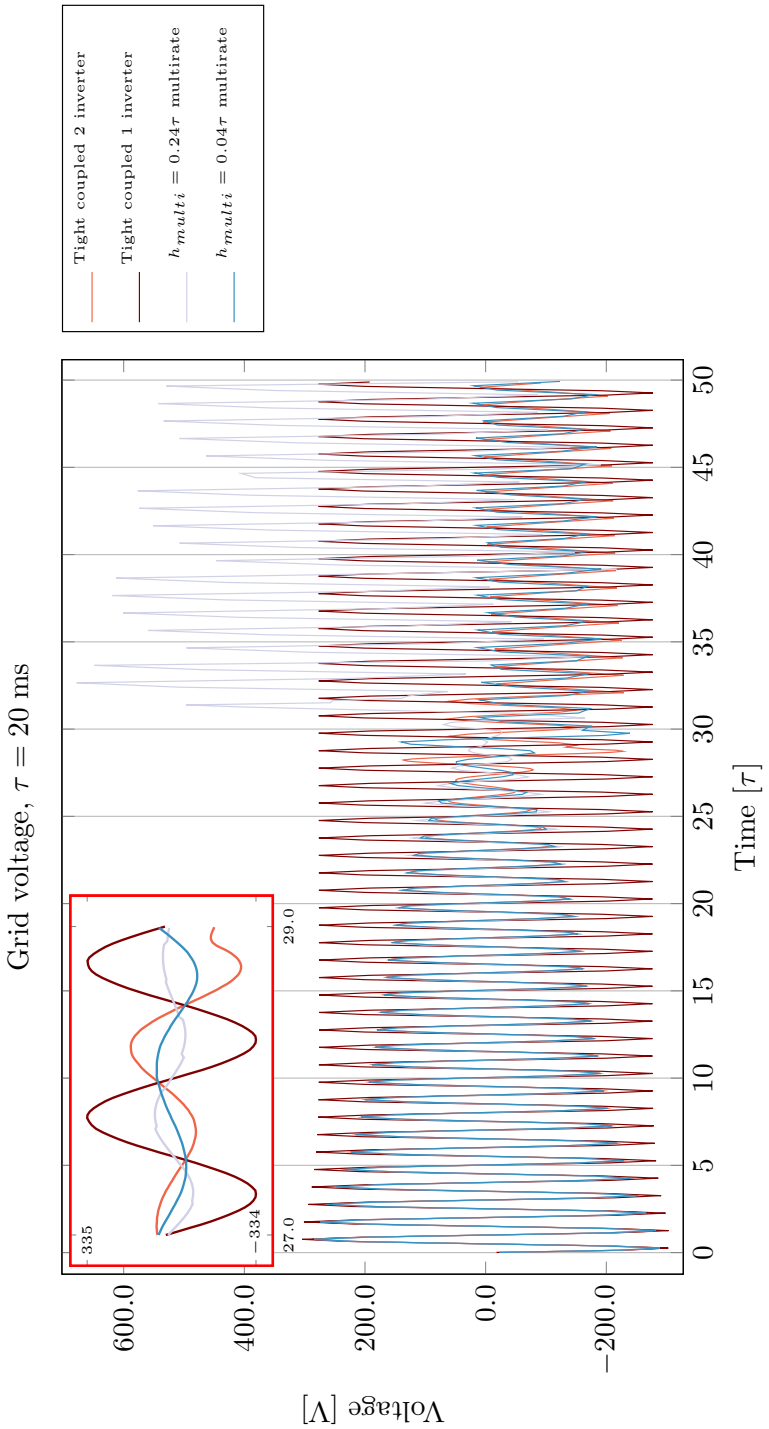

Figure 36: Failure case of the multirate method

6.8 Limitations

Collision detection first order only: can't resolve only one collision per carrier crossing. A dramatic change in slope could in principle collide with a carrier twice by shooting upwards after the first collision. Fixing this is a technical problem we did not consider yet.

Another limitation here is that we only considered exogenous events, and have not harmonized this with other circuits with other types of discontinuities i.e., endogenous events.

7 Conclusion

Our investigations demonstrate clearly that variable time stepping methods are superior to fixed step ones in the context of externally switched electronic circuits, both in speed in accuracy. We have also shown that resolving impacts is essential to get numerical accuracy whenever the currents are discontinuous.

We also introduced an effective and accurate multirate method which, though expensive numerically, is well suited for parallel computing.

Our software implementation, not freely available now for technical reasons, also demonstrates that real-time performance is achievable for simple inverters.

Possible improvements include better collision detection, a more refined model of the grid for the multi-rate method, e.g., considering the global admittance computed in Eqn. (34) for the multi-rate algorithm, as well as the inclusion of diodes, transistors, and other non-smooth components for the same.

We should also extend this work to see how such externally switched circuits can be simulated simultaneously with endogenously switched ones, i.e., circuits with diodes, op-amps, internally triggered switches, and transistors to name a few.

The code could be improved with further optimizations as well as adding MPI capabilities for large scale parallelism,

Acknowledgments

This research was conducted using the resources of High Performance Computing Center North (HPC2N). This project was partially funded by Botnia Atlantica with the NOSEG grant, ProcessIT, and the Vinnova funded project Simovate.

References

[1] Ahmed Abdalrahman, Abdalhalim Zekry, K. L. Ahmed Alshazlyian, and P.W. Lehn. Simulation and Implementation of Grid-connected Inverters. *International Journal of Computer Applications*, 60(4):42–43, 2012. URL `www.researchgate.net/publication/234101300_Simulation_and_`

Implementation_of_grid-connected_inverters/file/d912f50f16da90c316.pdf.

[2] Vincent Acary, Olivier Bonnefon, and Bernard Brogliato. *Nonsmooth Modeling and Simulation for Switched Circuits*, volume 69 of *Lecture Notes in Electrical Engineering*. Springer-Verlag, 2011. ISBN 978-90-481-9680-7.

[3] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35(3):1–14, 2008. ISSN 0098-3500. doi: http://doi.acm.org/10.1145/1391989.1391995. cholmod.

[4] Timothy A. Davis. Algorithm 832: UMFPACK — an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2):196–199, June 2004.

[5] Timothy A. Davis. *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2)*. SIAM, Philadelphia, PA, USA, 2006. ISBN 0898716136.

[6] Herman W. Domer. *Electro-magnetic transients program EMTP*. Bonneville Power Administration, 1984 wp5.1 edition, 1981. URL www.dee.ufrj.br/ipst/EMTPTB.PDF.

[7] Iain S. Duff and Stephane Pralet. Towards stable mixed pivoting strategies for the sequential and parallel solution of sparse symmetric indefinite systems. *SIAM J. on Mat. Anal.and Appl.*, 29(3):1007–1024, 2007. doi: 10.1137/050629598.

[8] A. F. Filippov. *Differential Equations with Discontinuous Righthand Sides*. Kluwer Academic Publishers, 1988.

[9] A.M. Gole, I.T. Fernando, G.D. Irwin, and O.B. Nayak. Modeling of power electronic apparatus: Additional interpolation issues. In *Proc. of the Intl. Conference on Power System Transients, Seatle 2001*, pages 23–28. IPST, June 2001.

[10] Dr. Ani Gole, Garth Irwin, Dr. Om Nayak, and Dennis Woodford. *USER'S GUIDE A Comprehensive Resource for EMTDC*. Manitoba HVDC Research Centre, Winnipeg, 5 edition, 2010. URL https://hvdc.ca/uploads/ck/files/reference_material/EMTDC_User_Guide_v4_3_1.pdf.

[11] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential Algebraic Problems*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second revised edition edition, 1996. ISBN 3-540-60452-9.

[12] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second revised edition, 1991. ISBN 3-540-56670-8.

[13] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2001.

[14] *Intel Thread Building Blocks 4.3*. Intel, 2014. URL `http://tinyurl.com/inteltbb`.

[15] *Precision Simulation of PWM Controllers*, June 2001. IPST.

[16] Peter B. Johns and Mark O'Brien. Use of the transmission-line modelling (t.l.m.) method to solve non-linear lumped networks. *Radio and Electronic Engineer*, 50 (1.2):59–70, January 1980. ISSN 0033-7722. doi: 10.1049/ree.1980.0006.

[17] P. Kuffel, K. Kent, and G.D. Irwin. The implementation and effectiveness of linear interpolation within digital simulation. In *Proc. of the Intl. Conference on Power System Transients, Lisbon, 1995*, pages 499 – 504. IPST, Sept 1995.

[18] Claude Lacoursière. *Ghosts and Machines: Regularized Variational Methods for Interactive Simulations of Multibodies with Dry Frictional Contacts*. PhD thesis, Dept. of Computing Science, Umeå University, June 2007.

[19] Claude Lacoursière, Mattias Linde, and Olof Sabelström. Direct sparse factorization of blocked saddle point matrices. In *PARA 2010, Part II*, volume 7134 of *LNCS*, pages 324–335. Springer, 2012.

[20] R. I. Leine, U. Aeberhard, and C. Glocker. Hamilton's principle as variational inequality for mechanical systems with impact. *Journal of Nonlinear Science*, 19 (6):633–664, Dec 2009. ISSN 0938-8974. doi: {10.1007/s00332-009-9048-z}.

[21] K. L. Lian and P.W. Lehn. Real Time Simulation of Voltage Source Converters based on Time Average Method. *Power Systems, IEEE Transactions on*, 20(1): 110–118, 2005. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=\&arnumber=1388500`.

[22] J. E. Marsden and M. West. Discrete mechanics and variational integrators. *Acta Numer.*, 10:357–514, 2001. ISSN 0962-4929.

[23] Katta G. Murty and Feng-Tien Yu. *Linear Complementarity, Linear and Nonlinear Programming*. Self-published: internet edition, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, http://www-personal.engin.umich.edu/~murty/book/LCPbook/index.html, 1997.

[24] Laurence W. Nagel. *SPICE2: A Computer Program to Simulate Semiconductor Circuits*. PhD thesis, EECS Department, University of California, Berkeley, 1975.

[25] Sina Ober-Blöbaum, Molei Tao, Mulin Cheng, Houman Owhadib, and Jerrold E. Marsdenb. Variational integrators for electric circuits. *Arxiv.org*, (1130.1859v1), 2011.

[26] Petri T. Piiroinen and Yuri A. Kuznetsov. An event-driven method to simulate filippov systems with accurate computing of sliding motions. *ACM Trans. Math. Softw.*, 34(3):13:1–13:24, May 2008. ISSN 0098-3500. doi: 10.1145/1356052.1356054. URL `http://doi.acm.org/10.1145/1356052.1356054`.

[27] G Rizzoni. *Principles and Applications of Electrical Engineering*. McGraw Hill, 2003.

[28] Martin Sjölund, Robert Braun, Peter Fritzson, and Peter Krus. Towards efficient distributed simulation in modelica using transmission line modeling. In Peter Fritzson, Edward Lee, François Cellier, and David Broman, editors, *Proceedings of the 3rd International Workshop on Equation-Based Object-Oriented Languages and tools*, pages 71–80, Oct 2010. URL `http://www.ep.liu.se/ecp/047/ecp10047.pdf`.

[29] Tomas Sjöström. Discrete time variational mechanics of multidomain systems: Applications to coupled electronic, hydraulic, and multibody systems. Master's thesis, Dept. of Physics, Umeå university, 2012.

[30] David Stewart. A high accuracy method for solving ODEs with discontinuous right-hand side. *Numerische Mathematik*, 58:299–328, 1990.