A new transient integration method for free-running oscillators

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computing Science

RANA 03-07
April 2003

A new transient integration method for free-running oscillators

by

S.H.M.J. Houben, E.J.W. ter Maten, M.F. Sevat

Reports on Applied and Numerical Analysis
Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven, The Netherlands
ISSN: 0926-4507

A new transient integration method for free-running oscillators

Stephan Houben; Magma Design Automation; Eindhoven; The Netherlands
Jan ter Maten; Philips Research Laboratories & Eindhoven University of Technology; The Netherlands
Marcel Sevat; Philips Research Laboratories; Eindhoven; The Netherlands

Abstract

This paper presents a new Runge-Kutta type integration method that is well-suited for time-domain simulation of oscillators. A unique property of the new method is that its damping characteristics can be controlled by a continuous parameter.

1 Introduction

In this paper, we focus on the behavior and the numerical simulation of oscillators. First we discuss the equations governing circuit behaviour. Second, we investigate time integration methods that are suitable for oscillatory circuits. We will show that BDF methods, which are widely used in circuit simulations, are less suitable for oscillators. Diagonal Runge-Kutta (DRK) methods will then be presented as alternatives. Implementation of these DRK methods in a circuit simulator is discussed as well, showing that BDF program code may be reused, leading to minimal implementation costs. A particular instance of a DRK method will be derived, and tested on two oscillator problems.

2 Circuit equations

The circuit equation, as formulated in the Modified Nodal Analysis (MNA) approach used in numerical circuit simulation (see e.g. [4]), takes the following form

$$\frac{d\boldsymbol{q}(t,\boldsymbol{x})}{dt} + \boldsymbol{j}(t,\boldsymbol{x}) = \boldsymbol{0} \tag{1}$$

in which $\boldsymbol{q}$ is a vector of charges and fluxes, $\boldsymbol{x}$ is a vector of nodal voltages and selected branch currents, $\boldsymbol{j}$ is a vector that contains branch currents and selected branch voltages along with current and voltage sources, and t denotes the time. The vector $\boldsymbol{x}(t)$ is to be solved from this equation.

Usually, this equation will be a Differential Algebraic Equation (DAE), because the Jacobian of $\boldsymbol{q}$ might be singular. Under normal circumstances, this DAE will be of index ≤ 1, implying that no time derivatives of the circuit sources are needed for a unique solution. More information about the properties of the DAE can be found in [4] as well.

In the case of a free-running oscillator circuit, the circuit equation (1) would carry no explicit time dependence. However, there are also driven circuits that contain an oscillator (e.g. a mixer with a VCO). To be able to cover these circuits as well, the explicit time dependence will be maintained in the analysis.

3 Methods for time integration

In this section, we investigate numerical methods to solve the circuit equations describing an electrical oscillator. In order to be useful for the simulation of oscillators, a time integration method should satisfy two conditions:

1. It should not damp out the oscillation. This can be relaxed somewhat by requiring that the method should not damp out the oscillation by more than a certain factor. If we are given a global relative simulation tolerance `reltol`, we should require a damping of less than α`*reltol` per waveform, with $0 < \alpha < 1$ a "security factor".

2. When starting in an inconsistent state, i.e. a computed circuit state that does not (yet) satisfy the circuit equation (1), the method should converge to a consistent state. This should happen sufficiently fast, i.e., the deviation from the consistent state should be damped in a characteristic time of $\tau > 0$. Typically, we would like to take τ a certain fraction of the oscillation period T, i.e., $\tau = \gamma T$, $0 < \gamma < 1$.

3.1 BDF methods

A popular family of time integration methods in circuit simulation are the Backward Difference Formulae (BDF) time integration methods. The reason for their popularity is that for an index-1 DAE, they produce a consistent state after each step. The characteristic time τ is thus in fact 0. However, BDF methods tend to damp out oscillations rather strongly. This makes them unsuitable for the simulation of oscillators, unless very small step-sizes are used. A lesser disadvantage of BDF methods is that they are multistep methods. A k-step method needs other methods to generate the first $k-1$ steps. Typically, start-up methods are of lower order. This is mainly a problem when the time integrator is called in an iterative way. Many restarts are then needed, which will make the effects of reduced efficiency and order more severe.

3.2 Runge-Kutta methods

To avoid the above mentioned disadvantages of BDF methods, we will look at one-step methods, in particular Runge-Kutta methods. We start with a brief review of Runge-Kutta methods; see [2] for details. Any Runge-Kutta method can be described by a matrix (a_{ij}), for $1 \leq i, j \leq s \in \mathbb{N}$, and a vector (b_i), for $1 \leq i \leq s$. We consider the case where a Runge-Kutta method is applied to a general DAE of the form (see [6])

$$\boldsymbol{g}(t, \dot{\boldsymbol{x}}, \boldsymbol{x}) = \boldsymbol{0}. \tag{2}$$

Given a step-size h and an initial value $\boldsymbol{x}_0$, the Runge-Kutta method computes a sequence $\{\boldsymbol{x}_n\}$, where $\boldsymbol{x}_n$ is an approximation to the solution at $t = nh$. Given (a_{ij}) and (b_i), we first define (c_i) as

$$c_i := \sum_{j=1}^{s} a_{ij}, \quad \text{for} \quad i = 1, \ldots, s. \tag{3}$$

The following formula computes $\boldsymbol{x}_{n+1}$, given $\boldsymbol{x}_n$: (see [1])

$$\boldsymbol{g}\left(t_n + hc_i, \boldsymbol{X}_n^{(i)}, \boldsymbol{x}_n + h\sum_{j=1}^{s} a_{ij}\boldsymbol{X}_n^{(j)}\right) = \boldsymbol{0}, \tag{4a}$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\sum_{i=1}^{s} b_i \boldsymbol{X}_n^{(i)}. \tag{4b}$$

The quantities $\boldsymbol{X}_n^{(i)}$ are called the stages of the Runge-Kutta method, and s is called the stage count. Not any arbitrary set of values (a_{ij}) and (b_j) will lead to a useful method; so-called order conditions have to be satisfied (see [2]). If (2) is an explicit ODE, i.e., $\boldsymbol{g}(t, \dot{\boldsymbol{x}}, \boldsymbol{x}) := \dot{\boldsymbol{x}} - \boldsymbol{f}(t, \boldsymbol{x})$, then (4) reduces to

$$\boldsymbol{X}_n^{(i)} = \boldsymbol{f}(t_n + hc_i, \boldsymbol{x}_n + h\sum_{j=1}^{s} a_{ij}\boldsymbol{X}_n^{(j)}), \tag{5a}$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\sum_{i=1}^{s} b_i \boldsymbol{X}_n^{(i)}. \tag{5b}$$

The order of method (4) applied to some DAE may be less than the order of (5) applied to an ODE. This phenomenon, called order reduction, is discussed in [1].

3.3 Diagonal Runge-Kutta Methods

Computing (4) means solving a large system, i.e., it requires many evaluations of $\boldsymbol{g}(t, \dot{\boldsymbol{x}}, \boldsymbol{x})$ and its derivatives. We will therefore look at a restricted class of Runge-Kutta methods, for which the computation is much cheaper. This class consists of the methods that have non-zero entries only on the diagonal of the matrix (a_{ij}), i.e. $a_{ij} = 0$ if $i \neq j$. Such methods will be called Diagonal Runge-Kutta methods (DRK). For DRK methods, (4) reduces to

$$\boldsymbol{g}\left(t_n + ha_{ii}, \boldsymbol{X}_n^{(i)}, \boldsymbol{x}_n + ha_{ii}\boldsymbol{X}_n^{(i)}\right) = \boldsymbol{0}, \tag{6a}$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\sum_{i=1}^{s} b_i \boldsymbol{X}_n^{(i)}. \tag{6b}$$

Note that (6a) constitutes an Implicit Euler step with step-size ha_{ii}. So in essence, a DRK method is a linear combination of Implicit Euler steps.

4 DRK methods

In this section we will investigate the order and stability properties of the DRK method and derive an instance of the DRK method dedicated to oscillatory problems.

4.1 Order of the DRK method

To investigate the order of general DRK methods, we will apply them to the test equation

$$\dot{x} = f(x), \quad f \in C^\infty(\mathbb{R}, \mathbb{R}). \tag{7}$$

which after reformulation as

$$g(t, \dot{x}, x) = \dot{x} - f(x) = 0 \tag{8}$$

leads to the following DRK procedure

$$X_n^{(i)} = f(x_n + ha_{ii}X_n^{(i)}), \quad \text{for} \quad i = 1 \ldots s, \tag{9a}$$

$$x_{n+1} = x_n + h\sum_{i=1}^{s} b_i X_n^{(i)}. \tag{9b}$$

The order conditions up to order k are now found by equating, for arbitrary f, the following terms at the point $h = 0$, see also [2]:

$$\frac{d^j x_{n+1}}{dh^j} = \frac{d^j x(t_n + h)}{dh^j}, \quad \text{for} \quad j = 0 \ldots k. \tag{10}$$

For $j = 0 \ldots 3$, the right-hand side of (10) is given by

$$\frac{d^0 x(t_n + h)}{dh^0} = x_n, \tag{11a}$$

$$\frac{d^1 x(t_n + h)}{dh^1} = f(x_n), \tag{11b}$$

$$\frac{d^2 x(t_n + h)}{dh^2} = f'(x_n)f(x_n), \tag{11c}$$

$$\frac{d^3 x(t_n + h)}{dh^3} = f''(x_n)f^2(x_n) + (f'(x_n))^2 f(x_n). \tag{11d}$$

The left-hand side of (10) for those values of j is given by

$$\frac{d^0 x_{n+1}}{dh^0} = x_n, \tag{12a}$$

$$\frac{d^1 x_{n+1}}{dh^1} = \sum_{i=1}^{s} b_i f(x_n), \tag{12b}$$

$$\frac{d^2 x_{n+1}}{dh^2} = 2\sum_{i=1}^{s} b_i a_{ii} f'(x_n) f(x_n), \tag{12c}$$

$$\frac{d^3 x_{n+1}}{dh^3} = 3\sum_{i=1}^{s} b_i a_{ii}^2 (f''(x_n) f^2(x_n) + 2(f'(x_n))^2 f(x_n)). \tag{12d}$$

By comparing (11d) to (12d) it becomes immediately clear that it is impossible to find a_{ii} and b_i which satisfy (10) for $j = 3$. So the maximum order that can be reached by a DRK method is 2. For this maximum order to be reached, the following order conditions should be satisfied:

$$\sum_{i=1}^{s} b_i = 1, \tag{13a}$$

$$\sum_{i=1}^{s} b_i a_{ii} = \frac{1}{2}. \tag{13b}$$

The fact that DRK methods are limited to such low orders, appears to make them quite unappealing. In contrast to the common approach in Runge-Kutta theory, as presented in e.g. [2], we do not aim for maximising of the order of the method. Rather we balance the desire for a high order against the goal of obtaining a method that does not damp out oscillations.

4.2 Stability of the DRK method

To investigate the stability of the DRK method, the increment function ζ is introduced. The increment function $\zeta : \mathbb{C} \to \mathbb{C}$ of a linear one-step method M is defined as follows. Let M be applied to the Dahlquist test equation

$$\dot{x} = \lambda x, \quad \lambda \in \mathbb{C},$$

Let $x_n \in \mathbb{C}\backslash\{0\}$ and let the stepsize $h > 0$. Compute x_{n+1} using method M. Then the fraction $\frac{x_{n+1}}{x_n}$ will only depend on the choice of method M and on the product $h\lambda$ (see [2]). Therefore we can define

$$\zeta(h\lambda) := \frac{x_{n+1}}{x_n}. \tag{14}$$

When applying this definition to (9) with $f(x) = \lambda x$, we obtain:

$$\zeta(h\lambda) = 1 + \sum_{i=1}^{s} \frac{b_i h\lambda}{1 - a_{ii} h\lambda} \tag{15}$$

or, when defining $z = h\lambda$,

$$\zeta(z) = 1 + \sum_{i=1}^{s} \frac{b_i z}{1 - a_{ii} z}. \tag{16}$$

For a DRK method to be usable as an integration method for oscillator problems, $\zeta(z)$ should satisfy the following conditions:

$$|\zeta(j\omega)| \lesssim 1\ , \ \omega \in \mathbb{R}, \tag{17a}$$

$$|\zeta(z)| < 1 \ \text{ for } \ Re(z) < 0, \tag{17b}$$

$$\lim_{Re(z)\to-\infty} \zeta(z) = 0, \tag{17c}$$

Condition (17a) demands that for any oscillatory problem the oscillating solution is not appreciably damped by the method. Condition (17b) demands that the method be stable for any damped problem. Condition (17c) requires that for strongly damped problems (i.e. $Re(\lambda) \to -\infty$), the method also generates a strong damping solution.

In mathematical terms, condition (17a) requires the method to be I-stable. The condition (17b) demands that the method be A-stable, the additional condition (17c) makes the method also L-stable. It can be shown that, once condition (17a) is satisfied, and $\zeta(z)$ is analytic in the left half of the complex plane, then also condition (17b) is satisfied (see [3]).

Assuming for the moment that condition (17a) is satisfied, then to also satisfy condition (17b) the poles of $\zeta(z)$ need to be in the right half of the complex plane. This leads to the following restriction on the coefficients a_{ii}:

$$a_{ii} > 0 \quad \text{for } i = 1, \ldots, s \tag{18}$$

Note that even with this restriction satisfied, we still need to check on any proposed set of coefficients whether (17a) is satisfied.

Applying condition (17c) to (16) leads to:

$$\lim_{Re(z)\to-\infty} \zeta(z) = 1 - \sum_{i=1}^{s} \frac{b_i}{a_{ii}} = 0, \tag{19}$$

or, equivalently,

$$\sum_{i=1}^{s} \frac{b_i}{a_{ii}} = 1 \tag{20}$$

which embodies another restriction on the DRK coefficients.

4.3 Example of a DRK method

To have a DRK method suitable for use in oscillator simulation, the coefficients a_{ii} and b_i should satisfy order and stability conditions as derived in the preceding

sections. For convenience we repeat them here:

$$\sum_{i=1}^{s} b_i = 1, \tag{21a}$$

$$\sum_{i=1}^{s} b_i a_{ii} = \frac{1}{2}, \tag{21b}$$

$$\sum_{i=1}^{s} \frac{b_i}{a_{ii}} = 1, \tag{21c}$$

$$a_{ii} > 0 \quad \text{for} \quad i = 1, \ldots, s. \tag{21d}$$

It turns out that already for two stages (21) has solutions with one degree of freedom. For this particular case, the set of equations to be solved reduces to:

$$b_1 + b_2 = 1, \tag{22a}$$

$$b_1 a_{11} + b_2 a_{22} = \frac{1}{2}, \tag{22b}$$

$$\frac{b_1}{a_{11}} + \frac{b_2}{a_{22}} = 1, \tag{22c}$$

$$a_{11} > 0, a_{22} > 0. \tag{22d}$$

In the sequel, we denote $\gamma := a_{22}$ as the degree of freedom. We then find the following solution to (22):

$$b_1 = \frac{2\gamma^2 - 3\gamma + 1}{2\gamma^2 - 4\gamma + 1}, \tag{23a}$$

$$b_2 = \frac{-\gamma}{2\gamma^2 - 4\gamma + 1}, \tag{23b}$$

$$a_{11} = \frac{2\gamma - 1}{2\gamma - 2}, \tag{23c}$$

$$a_{22} = \gamma, \tag{23d}$$

$$\gamma \in (0, \frac{1}{2}) \cup (1, \infty), \tag{23e}$$

$$\gamma \neq \frac{1}{2 \pm \sqrt{2}}. \tag{23f}$$

It remains to be checked if this set of coefficients satisfies the condition (17a). To that end, we investigate the value of $|\zeta(j\omega)|$. We have

$$\zeta(j\omega) = \frac{b_1/a_{11}}{1 - j a_{11}\omega} + \frac{b_2/a_{22}}{1 - j a_{22}\omega}, \tag{24}$$

from which we find that

$$|\zeta(j\omega)|^2 = \frac{(b_1/a_{11})^2}{1 + a_{11}^2\omega^2} + \frac{(b_2/a_{22})^2}{1 + a_{22}^2\omega^2} + 2\frac{(b_1/a_{11})(b_2/a_{22})(1 + a_{11}a_{22}\omega^2)}{(1 + a_{11}^2\omega^2)(1 + a_{22}^2\omega^2)}. \tag{25}$$

Using (23) we find

$$|\zeta(j\omega)|^2 = 1 - \frac{\omega^4\gamma^2(1 - 2\gamma)^2}{(1 + \gamma^2\omega^2)[4(1 - \gamma)^2 + \omega^2(1 - 2\gamma)^2]}, \tag{26}$$

which shows that $|\zeta(j\omega)| \leq 1$, thereby satisfying condition (17a). Note that by making γ small enough, $|\zeta(j\omega)|$ can be brought as close to 1 as we want. The DRK method we have derived is an A-stable, L-stable one-step method of order 2, of which the damping behaviour can be completely controlled by the parameter γ. Note that if we take $\gamma \to 0$, we have $b_1 \to 1$, $b_2 \to 0$, $a_{11} \to \frac{1}{2}$ and $a_{22} \to 0$. We then find:

$$\boldsymbol{g}\left(t_n + \frac{1}{2}h, \boldsymbol{X}_n^{(1)}, \boldsymbol{x}_n + \frac{1}{2}h\boldsymbol{X}_n^{(1)}\right) = \boldsymbol{0}, \tag{27a}$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\boldsymbol{X}_n^{(1)}, \tag{27b}$$

which is the midpoint rule. So for $\gamma \to 0$ the method approaches the midpoint rule.

The stability diagrams for various values of γ are shown in Figure 1. This figure clearly illustrates the fact that γ can be used to control the amount of damping on the imaginary axis. Furthermore, it shows that by sufficiently decreasing the value of γ the amount of damping can be brought as close to zero as we require.

As experimental verification, we will make a numerical comparison of the derived method with the Implicit Euler method. Both methods are L-stable implicit Runge-Kutta methods. We apply these methods to the simple harmonic oscillator defined by $\dot{x} = -y,\ \dot{y} = x$ which has a constant-amplitude periodic solution with period $T = 2\pi$. This gives us a good insight in the damping behaviour of these methods. For the DRK method, we will use $\gamma = \frac{1}{5}$. We integrate from $t = 0$ to $t = 25T$. As a stepsize, we take $h = T/40$. The results for $x(t)$ and $y(t)$ are shown in Figure 2. This figure clearly illustrates that the Implicit Euler method effects an appreciable damping in the computed solution, whereas the new method does not.

4.4 Alternative formulation

We will now derive an alternative formulation of the DRK method, starting from the standard formulation (6), through the following transformation:

$$\widetilde{\boldsymbol{X}}_n^{(i)} = \boldsymbol{x}_n + h a_{ii} \boldsymbol{X}_n^{(i)} \quad \text{for} \quad i = 1, \ldots, s. \tag{28}$$

When used in (6) this leads to:

$$\boldsymbol{g}\left(t_n + h a_{ii}, \frac{\widetilde{\boldsymbol{X}}_n^{(i)} - \boldsymbol{x}_n}{h a_{ii}}, \widetilde{\boldsymbol{X}}_n^{(i)}\right) = \boldsymbol{0}, \tag{29a}$$

$$\boldsymbol{x}_{n+1} = (1 - \sum_{i=1}^{s} \frac{b_i}{a_{ii}})\boldsymbol{x}_n + \sum_{i=1}^{s} \frac{b_i}{a_{ii}} \widetilde{\boldsymbol{X}}_n^{(i)}, \tag{29b}$$

which can be used as an alternative to (6).

If now condition (20) is enforced, this simplifies to:

$$\boldsymbol{g}\left(t_n + h a_{ii}, \frac{\widetilde{\boldsymbol{X}}_n^{(i)} - \boldsymbol{x}_n}{h a_{ii}}, \widetilde{\boldsymbol{X}}_n^{(i)}\right) = \boldsymbol{0}, \tag{30a}$$

$$\boldsymbol{x}_{n+1} = \sum_{i=1}^{s} \frac{b_i}{a_{ii}} \widetilde{\boldsymbol{X}}_n^{(i)}, \tag{30b}$$

For the two-stage case, considered in the previous section, the coefficients (23) satisfy condition (20). So (30) holds with the following expressions for its coefficients:

$$\frac{b_1}{a_{11}} = \frac{2(\gamma-1)^2}{2\gamma^2-4\gamma+1}, \tag{31a}$$

$$\frac{b_2}{a_{22}} = \frac{-1}{2\gamma^2-4\gamma+1}, \tag{31b}$$

$$a_{11} = \frac{2\gamma-1}{2\gamma-2}, \tag{31c}$$

$$a_{22} = \gamma, \tag{31d}$$

$$\gamma \in (0, \frac{1}{2}) \cup (1, \infty), \tag{31e}$$

$$\gamma \neq \frac{1}{2 \pm \sqrt{2}}. \tag{31f}$$

The use of the coefficients (31) in (30), therefore leads to the same order and stability properties as the coefficients (23) used in (6).

5 Implementation of the method

5.1 Standard implementation

We now consider applying method (6) to the circuit equations

$$\frac{d\boldsymbol{q}(t,\boldsymbol{x})}{dt} + \boldsymbol{j}(t,\boldsymbol{x}) = \boldsymbol{0}. \tag{32}$$

In doing this, we have to rewrite (32) into the standard form (6). There are several ways to do this; we will consider two possibilities.

$$\boldsymbol{g}(t,\dot{\boldsymbol{x}},\boldsymbol{x}) := \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} + \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{t}} + \boldsymbol{j}(t,\boldsymbol{x}), \tag{33a}$$

or

$$\boldsymbol{g}\left(t, \begin{bmatrix}\dot{\boldsymbol{x}}\\ \dot{\boldsymbol{y}}\end{bmatrix}, \begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{bmatrix}\right) := \begin{bmatrix}\dot{\boldsymbol{y}} + \boldsymbol{j}(t,\boldsymbol{x})\\ \boldsymbol{q}(t,\boldsymbol{x}) - \boldsymbol{y}\end{bmatrix}. \tag{33b}$$

In (33b), $\boldsymbol{y} := \boldsymbol{q}(t,\boldsymbol{x})$. It should be noted that these two different choices lead to different numerical schemes. In general, (33b) is preferable over (33a), for the following two reasons.

1. In (33b) the accuracy results of the method apply to the whole vector $[\boldsymbol{x}^T, \boldsymbol{y}^T]^T$. Therefore not only the error in $\boldsymbol{x}$ but also the error in $\boldsymbol{y} = \boldsymbol{q}(t,\boldsymbol{x})$ is under control, provided that we have accounted for order reduction due to the DAE character of (33b).

2. In general, numerical methods to compute (6) use a Newton scheme, and therefore require the Jacobian $\partial \boldsymbol{g}/\partial \boldsymbol{x}$ to be computed. For (33a), this is problematic, since this requires the second derivative $\partial^2 \boldsymbol{q}/\partial \boldsymbol{x}^2$. This quantity is typically not readily available in circuit simulators. It would also be quite expensive to compute, since it is a three-tensor and therefore has N^3 elements.

Using (33b) in (6), we obtain

$$\boldsymbol{Y}_n^{(i)} = -\boldsymbol{j}\left(t_n + ha_{ii}, \boldsymbol{x}_n + ha_{ii}\boldsymbol{X}_n^{(i)}\right), \tag{34a}$$

$$\boldsymbol{q}\left(t_n + ha_{ii}, \boldsymbol{x}_n + ha_{ii}\boldsymbol{X}_n^{(i)}\right) = \boldsymbol{q}(t_n,\boldsymbol{x}_n) + ha_{ii}\boldsymbol{Y}_n^{(i)}, \tag{34b}$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\sum_{i=1}^{s} b_i \boldsymbol{X}_n^{(i)}. \tag{34c}$$

Equations (34a) and (34b) can be combined into

$$\frac{\boldsymbol{q}\left(t_n + ha_{ii}, \boldsymbol{x}_n + ha_{ii}\boldsymbol{X}_n^{(i)}\right) - \boldsymbol{q}(t_n,\boldsymbol{x}_n)}{ha_{ii}} = -\boldsymbol{j}\left(t_n + ha_{ii}, \boldsymbol{x}_n + ha_{ii}\boldsymbol{X}_n^{(i)}\right). \tag{35}$$

At first sight it might not appear straightforward to implement this DRK method in an existing circuit simulator. Since many existing simulators use BDF methods, internal data structures of the simulator are typically optimised for these methods (see e.g. [5]). As it turns out, DRK methods can easily be implemented by reusing the code for the BDF method. The key observation is that the main equation to be solved in the DRK method is (35). Note that (35) has a structure similar to the Euler Implicit method, which for (32) takes on the form

$$\frac{\boldsymbol{q}\left(t_n + h, \boldsymbol{x}_{n+1}\right) - \boldsymbol{q}(t_n,\boldsymbol{x}_n)}{h} = -\boldsymbol{j}\left(t_n + h, \boldsymbol{x}_{n+1}\right). \tag{36}$$

In particular, observe that we get (35) from (36) by substituting ha_{ii} for h and $\boldsymbol{x}_n + ha_{ii}\boldsymbol{X}_n^{(i)}$ for $\boldsymbol{x}_{n+1}$. Now suppose that a circuit simulator contains a routine `EulerImp`, which computes $\boldsymbol{x}_{n+1}$ from (36):

$$\boldsymbol{x}_{n+1} = \texttt{EulerImp}(t_n, \boldsymbol{x}_n, h).$$

Then, given a time t_n, a circuit state $\boldsymbol{x}_n$ and a step-size $h > 0$, the next step of a DRK method can be computed as follows:

Algorithm 1 (Standard).
$\boldsymbol{x}_{n+1} := \boldsymbol{x}_n$
for $i = 1, \ldots, s$ ***do***
  $\boldsymbol{X}_n^{(i)} := (\texttt{EulerImp}(t_n, \boldsymbol{x}_n, a_{ii}h) - \boldsymbol{x}_n)/ha_{ii}$
  $\boldsymbol{x}_{n+1} := \boldsymbol{x}_{n+1} + hb_i\boldsymbol{X}_n^{(i)}$

We conclude that when Euler Implicit is already implemented, a DRK method can be implemented as a simple loop around the Euler Implicit algorithm. It follows that if the Euler Implicit code is already written to handle special sparse structures of the circuit, e.g., a hierarchical structure (as e.g. described in [5]), then the DRK method can also handle such sparse structures with little additional effort.

5.2 Alternative implementation

Based on the alternative formulation of the DRK method in the previous section, also an alternative implementation of this method in a circuit simulator can be derived. From a numerical point of view, this latter implementation will be shown to be more stable than its counterpart in the standard approach. To get to this alternative implementation, we assume that condition (20) holds. Method (30) can then be applied to the circuit equations

$$\frac{d\boldsymbol{q}(t,\boldsymbol{x})}{dt} + \boldsymbol{j}(t,\boldsymbol{x}) = \boldsymbol{0}. \tag{37}$$

Here we make the same choice for $\boldsymbol{g}(t,\dot{\boldsymbol{x}},\boldsymbol{x})$ as in the previous section:

$$\boldsymbol{g}\left(t, \begin{bmatrix}\dot{\boldsymbol{x}}\\ \dot{\boldsymbol{y}}\end{bmatrix}, \begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{bmatrix}\right) := \begin{bmatrix}\boldsymbol{y} + \boldsymbol{j}(t,\boldsymbol{x})\\ \boldsymbol{q}(t,\boldsymbol{x}) - \boldsymbol{y}\end{bmatrix} \tag{38}$$

which, when used in (30), leads to:

$$\frac{\boldsymbol{Y}_n^{(i)} - \boldsymbol{y}_n}{ha_{ii}} = -\boldsymbol{j}\left(t_n + ha_{ii}, \boldsymbol{X}_n^{(i)}\right), \tag{39a}$$

$$\boldsymbol{q}\left(t_n + ha_{ii}, \boldsymbol{X}_n^{(i)}\right) = \boldsymbol{Y}_n^{(i)}, \tag{39b}$$

$$\boldsymbol{x}_{n+1} = \sum_{i=1}^{s} \frac{b_i}{a_{ii}} \boldsymbol{X}_n^{(i)}. \tag{39c}$$

Equations (39a) and (39b) can be combined into

$$\frac{\boldsymbol{q}\left(t_n + ha_{ii}, \boldsymbol{X}_n^{(i)}\right) - \boldsymbol{q}(t_n, \boldsymbol{x}_n)}{ha_{ii}} = -\boldsymbol{j}\left(t_n + ha_{ii}, \boldsymbol{X}_n^{(i)}\right). \tag{40}$$

As it turns out, also this formulation of the DRK method can easily be implemented by reusing the code for the BDF method. The key observation is that the main equation to be solved is (40). Note that (40) has a structure similar to the Euler Implicit method, which for (37) takes on the form

$$\frac{\boldsymbol{q}\left(t_n + h, \boldsymbol{x}_{n+1}\right) - \boldsymbol{q}(t_n, \boldsymbol{x}_n)}{h} = -\boldsymbol{j}\left(t_n + h, \boldsymbol{x}_{n+1}\right). \tag{41}$$

In particular, observe that we get (40) from (41) by substituting ha_{ii} for h and $\boldsymbol{X}_n^{(i)}$ for $\boldsymbol{x}_{n+1}$. Again suppose that a circuit simulator contains a routine `EulerImp`, which computes $\boldsymbol{x}_{n+1}$ from (41):

$$\boldsymbol{x}_{n+1} = \texttt{EulerImp}(t_n, \boldsymbol{x}_n, h).$$

Then, given a time t_n, a circuit state $\boldsymbol{x}_n$ and a step-size $h > 0$, the next step of a DRK method can be computed as follows:

Algorithm 2 (Alternative).
$\boldsymbol{x}_{n+1} := 0$
for $i = 1, \ldots, s$ ***do***
  $\boldsymbol{X}_n^{(i)} := \texttt{EulerImp}(t_n, \boldsymbol{x}_n, a_{ii}h)$
  $\boldsymbol{x}_{n+1} := \boldsymbol{x}_{n+1} + \frac{b_i}{a_{ii}} \boldsymbol{X}_n^{(i)}$

From a numerical point of view, this algorithm is to be preferred over the standard algorithm, as it avoids computing the difference of nearly equal quantities, and thereby the unnecessary loss of accuracy.

5.3 Oscillator test circuit

We will now make a numerical comparison of the DRK method with the Implicit Euler and a two-stage third-order SDIRK2 method (see [2]). For the DRK method we choose the alternative approach (30), i.e. Algorithm 2 with the coefficients (31). In this approach we create two instances, one with $\gamma = \frac{1}{5}$ and one with $\gamma = \frac{1}{100}$. All four methods are L-stable implicit Runge-Kutta methods. We apply them to Colpitt's oscillator, shown in Figure 3. We integrate from $t = 0$ to $t = 10^{-3}$. As a stepsize, we take $h = 10^{-6}$. The solutions produced with the various methods are shown in Figure 4. The strong damping effect of Euler Implicit and, to a lesser extent, SDIRK2 is clearly visible. The best results are obtained with the DRK methods.

6 Conclusions

We developed a Diagonal Runge-Kutta algorithm that is particularly suited for transient simulation of oscillatory circuits, in the sense that it does not damp out any oscillation present in the solution of the circuit equation. In fact it has been shown that its damping characteristics can be controlled by a dedicated parameter. The new algorithm allows designers to better simulate oscillators, or to detect unwanted oscillation earlier than would be the case with standard BDF methods.

References

[1] E. Griepentrog and R. März. *Differential-Algebraic Equations and Their Numerical Treatment*. Teubner, 1986.

[2] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I, Non-stiff problems*. Second edition, Springer, 1993.

[3] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II, Stiff and Differential Algebraic problems*. Second edition, Springer, 1996.

[4] Stephan H.M.J. Houben. *Circuits in Motion: the numerical simulation of electrical oscillators; PhD Thesis*. Eindhoven University of Technology, 2003.

[5] J.G. Fijnvandraat, S.H.M.J. Houben, E.J.W. ter Maten, J.M.F. Peters, *"Time domain analog circuit simulation", presented at the International Workshop on Computational codes: Technological Aspects of Mathematics*, Bari, December 2002.

[6] P.J. Rabier and W.C. Rheinboldt. Theoretical and Numerical Analysis of Differential Algebraic Equations. In P.G. Ciarlet and J.L. Lions, editors, *Handbook of Numerical Analysis, Volume VIII*. Elsevier, 2002.

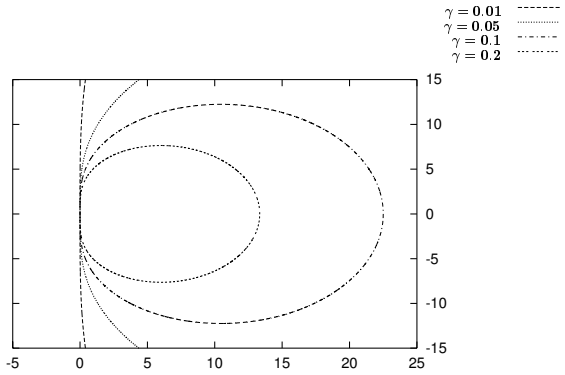
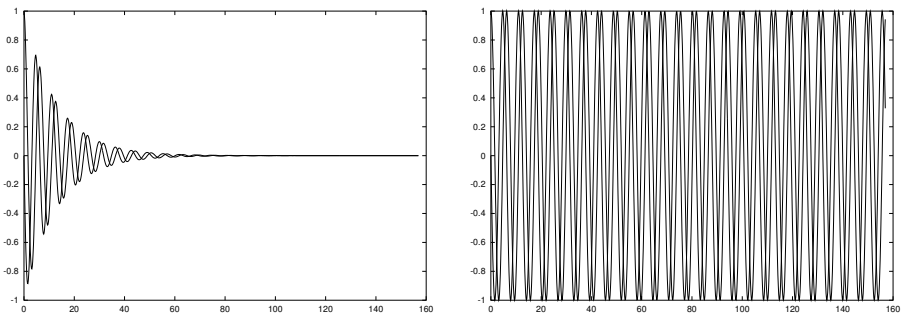


Figure 1: Stability diagram of the DRK method in the complex $h\lambda$-plane.



(a) Euler Implicit

(b) DRK($\gamma = \frac{1}{5}$)

Figure 2: Simulation of the harmonic oscillator $\dot{x} = -y, \dot{y} = x$ with 2 different methods.

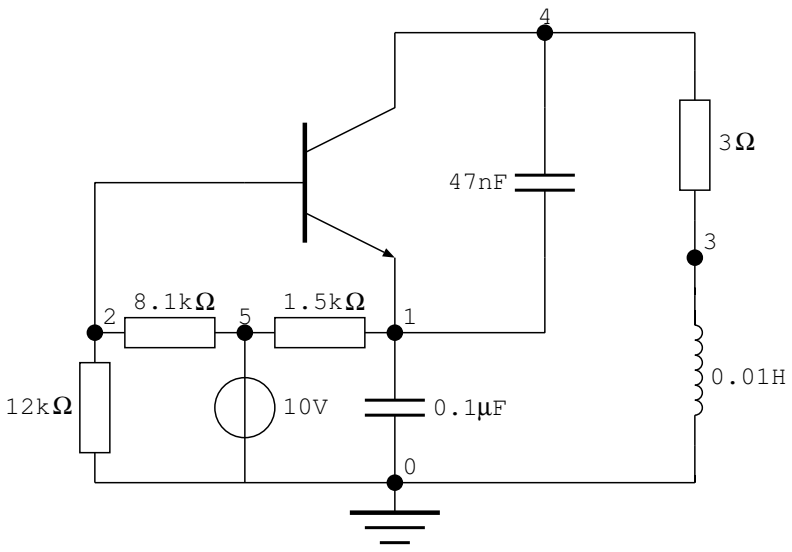
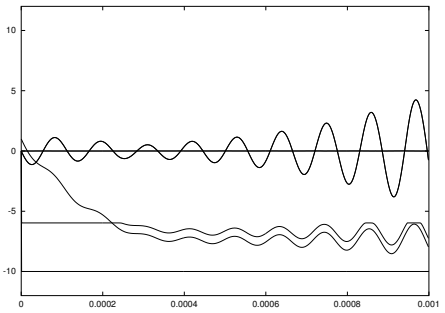
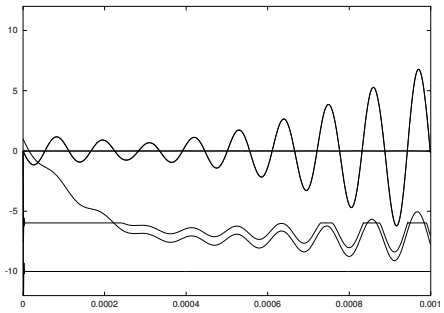


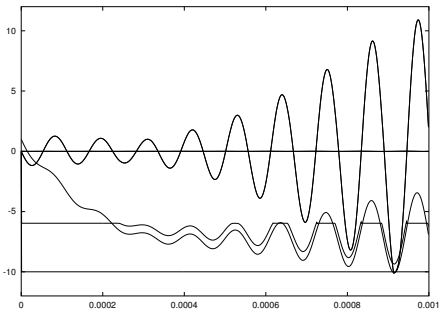
Figure 3: Colpitt's oscillator.



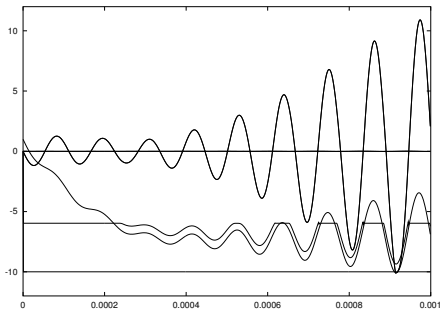
(a) Euler Implicit



(b) SDIRK2



(c) DRK($\gamma = \frac{1}{5}$)



(d) DRK($\gamma = \frac{1}{100}$)

Figure 4: Simulation of Colpitt's oscillator with 4 different methods.