NviePy
ReadMe

Vincenzo Schiano Di Cola

July 30, 2013

Contents

1 What is this? **4**

2 Theoretical part **5**
2.1 What is a Volterra integral equation? 5
2.2 What is a Runge-Kutta method for a VIE of second kind? 6
2.3 Order conditions for a VRK formula 8

3 The NviePy software **14**
3.1 How to start 15

Bibliography **17**

1 What is this?

This is only a short Read Me for the code *NviePy*. There is no external documentation, but the code is well documented as in NodePy (see [10]).

This document wants to be only a brief and simple summary on what it has been done and how to start from zero and use the software.

The code comes from a master thesis written for Math degree ([12]). The thesis, written in italian, is about Volterra trees and order conditions for Volterra Runge-Kutta type methods (VRK), and presents an open source software written in Python to obtain Volterra trees and the corresponding order conditions, for either VRK, Pouzet (PVRK) type and Bel'tyukov (BVRK) type. This study was based primarily on articles and books of Hairer and Brunner, while the software was inspired by NodePy, developed by Ketchson.

The thesis has been written since there are not much studies describing these numerical methods for Volterra integral equations of the second kind and their order conditions. Among this, as far as it is known, no software generating these order conditions has been published; and, in the open source world, Ketchson's software is the only one which studies Runge-Kutta methods for ordinary differential equations (ODEs). The elaboration of the thesis and the release of the software can help for further studies in these area of recherches.

This ReadMe consists of two section, apart from this.

The first section is theoretical, and explains what is done in a simple way, for further references you can look at the cited books.

The second section is basic step-by-step guide on how to start using the software.

If anyone finds some errors or wants to give me some advice, I will be extremely grateful. E-mail me at: vin (dot) schianodicola (at) gmail (dot) com

2 Theoretical part

NviePy is useful for studying order conditions for Runge-Kutta method for Volterra Integral Equations of second kind.

To understand this part it is required ad least a basic calculus course.

2.1 What is a Volterra integral equation?

A first example of integral equation is

$$y(x) = f(x) + \int_{\alpha(x)}^{\beta(x)} k(x, s, y(s))\,\mathrm{d}s,$$

where k is the kernel and $\alpha(x)$ and $\beta(x)$ are the limits of integration. The unknown function $y(x)$ appears under the integral sign and out of it. Notice that the functions: f, k, $\alpha(x)$ e $\beta(x)$ are known functions.

We define *Volterra integral equation* (or *VIE*) to be the following equation

$$\theta(x)y(x) = f(x) + \int_a^x k(x, s, y(s))\,\mathrm{d}s \tag{1}$$

where $f, \theta\colon I \to \mathbb{R}$ and $k\colon S \times \mathbb{R} \to \mathbb{R}$ are known functions; and $I := [a, b]$, with $a < b$, and

$$S := \{(x, s) : a \leq s \leq x \leq b\}.$$

The unknown function is $y(x)\colon I \to \mathbb{R}$.

Depending on the value of $\theta(x)$ we have three different kind of integral equations:

1. If $\theta(x) = 0$, $\forall x \in I$, equation (1) becomes

   $$f(x) = -\int_a^x k(x, s, y(s))\,\mathrm{d}s$$

   and is called *VIE of the first kind.*

2. When $\theta(x) = 1$, $\forall x \in I$, equation (1) becomes

   $$y(x) = f(x) + \int_a^x k(x, s, y(s))\,\mathrm{d}s \tag{2}$$

   and is called *VIE of the second kind.*

3. When $\theta(x)$ is a continuous function possessing a finite number of zeros in the interval I, then equation (1) is called *VIE of the third kind.*

We study, in particular, VIEs of the second kind.

For this fist part let us only assess the global existence and uniqueness theorem for a VIE of second kind see [2, Theorem 1.3.8].

Theorem 1 (Global existence and uniqueness). *Let $k(x,s,y)$ be continuous $\forall (x,s) \in S$ and $\forall y \in \mathbb{R}$, suppose that k satisfies the uniform Lipschitz condition with respect to y, i.e.*

$$|k(x,s,u_1) - k(x,s,u_2)| \leq L\,|u_1 - u_2|$$

where $L > 0$, and for all $(x,s) \in S$ e $u_1, u_2 \in \mathbb{R}$. Then for each $f \in C(I)$ the non linear equation (2) possesses a unique solution $y(x) \in C(I)$.

More details concerning this part can be found in [3] and [2, Chapter 1].

2.2 What is a Runge-Kutta method for a VIE of second kind?

In trying to solve an integral equation of the second kind

$$y(x) = f(x) + \int_a^x k(x,s,y(s))\,\mathrm{d}s \qquad x \in I := [a,b]. \tag{3}$$

one could find it really difficult to solve it, so we use a numerical technique. Among the various technique one can use a Runge-Kutta approach.

The idea is to consider the interval $[a,b]$ and suppose it is discretized by a uniform mesh on N points, defining $x_0 := a$ and

$$x_{n+1} := x_n + h, \qquad n = 0, 1, \ldots, N-1$$

where $h := (b-a)/N$. The given integral equation (3) can then be rewritten by relating it to the mesh, i.e.

$$y(x) = F_n(x) + h\Phi_n(x), \qquad x \in [x_n, b] \tag{4}$$

where F_n, called *lag term*), is given by

$$F_n(x) := f(x) + \int_{x_0}^{x_n} k(x,s,y(s))\,\mathrm{d}s, \qquad n = 0, \ldots, N-1; \tag{5}$$

while $\Phi_n(x)$ is the *increment function* with respect to the subinterval $[x_n, x_{n+1}]$ defined as

$$h\Phi_n(x) := \int_{x_n}^{x} k(x,s,y(s))\,\mathrm{d}s, \qquad x \in [x_n, b] \quad n = 0, \ldots, N-1. \tag{6}$$

A Runge-Kutta method for (4) is based on two approximations process which, in general, are independent of each other:

(i) An approximation scheme for the increment function $\Phi_n(x)$, i.e. a (local) approximation scheme on the subinterval $[x_n, x_{n+1}]$. The resulting discrete increment function, denoted by $\widetilde{\Phi}_n(x)$ will be called a *Volterra-Runge-Kutta formula* (*VRK formula*).

(ii) An approximation scheme for the lag term $F_n(x)$, i.e. an approximation scheme on the interval $[x_0, x_n]$. The resulting discrete lag term will be denoted by $\widetilde{F}_n(x)$ and will be referred as *lag term formula*.

Hence, the approximation y_{n+1} of the exact solution $y(x_{n+1})$ will be calculated at each grid point $x = x_{n+1}$, $n = 0, 1, \ldots, N-1$ using

$$y(x) = \widetilde{F}_n(x) + h\widetilde{\Phi}_n(x), \qquad n = 0, \ldots, N-1. \tag{7}$$

We shall call (7) a *VRK method* when both the VRK formula and lag term are specified.

Notice that the two terms on the right-hand side of (7) are referred as, respectively, the *tail approximation* and the *VRK part* of the VRK method.

The definition a a VRK formula is:

Definition 1. An *s-stage VRK formula* for (3) has the form

$$\widetilde{\Phi}_n(x) := \sum_{i=1}^{s} b_i k\left(x + (e_i - 1)h, x_n + c_i h, Y_i^{(n)}\right)$$

with $Y_i^{(n)}$ given by

$$Y_i^{(n)} := \widetilde{F}_n(x_n + \theta_i h) + h\sum_{j=1}^{s} a_{ij} k\left(x_n + d_{ij}h, x_n + c_j h, Y_j^{(n)}\right) \qquad i = 1, \ldots, s,$$

where the vectors $\boldsymbol{\theta} := (\theta_i)$, $\mathbf{c} := (c_i)$, $\mathbf{e} := (e_i)$, $\mathbf{b} := (b_i)$ and the square matrix $A := (a_{ij})$ e $D := (d_{ij})$ are given parameters.

Notice that we have $2(s^2+2s)$ parameters to assign. How do we assign them? Are there special choices? Yes, Pouzet and Bel'tyukov type. Are there any constrains that this parameters need to satisfy? Yes. The order conditions. In the next subsection we will see the order condition (among with the definition of order).

Definition 2. A VRK formula is of *Pouzet type* (or *PVRK formula*) if

$$d_{ij} = c_i, \quad e_i = 1, \quad \theta_i = c_i \qquad (i, j = 1, \ldots, s)$$

i.e. if its VRK part is completely characterized by the symbolic diagram

$$\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array},$$

and written explicitly, an s-stage PVRK formula is given by

$$\begin{cases} Y_i^{(n)} = \widetilde{F}_n(x_n + c_i h) + h\displaystyle\sum_{j=1}^{s} a_{ij} k\left(x_n + c_i h, x_n + c_j h, Y_j^{(n)}\right) & (i = 1, \ldots, s) \\ \\ y_{n+1} = \widetilde{F}_n(x_n + h) + h\displaystyle\sum_{i=1}^{s} b_i k\left(x_n + h, x_n + c_i h, Y_i^{(n)}\right) & (n = 0, \ldots, N-1). \end{cases} \tag{8}$$

Definition 3. A VRK formula if of *Bel'tyukov type* (or *BVRK formula*) if

$$d_{ij} = d_j, \quad e_i = d_i, \quad \theta_i = c_i \qquad (i, j = 1, \ldots, m).$$

Hence, the VRK part of a Bel'tyukov method is characterized by the digram

$$\begin{array}{c|c|c} \mathbf{d} & \mathbf{c} & A \\ \hline & & \mathbf{b}^T \end{array}.$$

An s-stage BVRK formula has thus the form

$$\begin{cases} Y_i^{(n)} = \widetilde{F}_n(x_n + c_i h) + h \displaystyle\sum_{j=1}^{s} a_{ij} k\left(x_n + d_j h, x_n + c_j h, Y_j^{(n)}\right) \quad (i = 1, \ldots, s) \\ y_{n+1} = \widetilde{F}_n(x_n + h) + h \displaystyle\sum_{i=1}^{s} b_i k\left(x_n + d_i h, x_n + c_i h, Y_i^{(n)}\right). \end{cases} \tag{9}$$

2.3 Order conditions for a VRK formula

Let $y(x)$ and $F_n(x)$ denote, respectively, the exact solution and the *exact lag term* of the equation (3), and suppose that the approximation $\bar{y}_n$ is defined by

$$\bar{y}_n := F_{n-1}(x_{n-1} + h) + h\overline{\Phi}_n(x_n + h), \tag{10}$$

where $\overline{\Phi}_n(x)$ is the *local increment function*

$$\overline{\Phi}_n(x) := \sum_{i=1}^{m} b_i k\left(x + (e_i - 1)h, x_n + c_i h, \overline{Y}_i^{(n)}\right), \tag{11}$$

with

$$\overline{Y}_i^{(n)} := F_n(x_n + \theta_i h) \sum_{j=1}^{m} a_{ij} k\left(x_n + d_{ij} h, x_n + c_j h, \overline{Y}_j^{(n)}\right), \quad (i = 1, \ldots, m). \tag{12}$$

In other words, we use the VRK formula

$$\begin{cases} Y_i^{(n)} = \widetilde{F}_n(x_n + \theta_i h) + h \displaystyle\sum_{j=1}^{m} a_{ij} k\left(x_n + d_{ij} h, x_n + c_j h, Y_j^{(n)}\right) \quad (i = 1, \ldots, m) \\ y_{n+1} = \widetilde{F}_n(x_n + h) + h \displaystyle\sum_{i=1}^{m} b_i k\left(x_n + e_i h, x_n + c_i h, Y_i^{(n)}\right) \end{cases} \tag{13}$$

with exact lag term , to solve (3) *locally* in $[x_n, x_{n+1}]$.

Definition 4. The VRK formula (13) is said to be *consistent of (local) order* p if, for all VIEs (3) with sufficiently differentiable f and $k$and for all sufficiently small $h > 0$, p is the largest integer for which

$$|y(x_n) - \bar{y}_n| \leq C h^{p+1} \qquad (x_n = x_0 + nh = x \text{ fixed, as } h \to 0)$$

holds, where C is a constant not depending on n and h.

From (10) and the expression of $\bar{y}_{n+1}$ in (7) we deduce that $\overline{\Phi}_n(x)$ is represents an approximation of order p of th exact increment function, provided that the local approximation on $[x_n, x_{n+1}]$ employs the exact lag term. More precisely the local increment function satisfies

$$h(\Phi_n(x_n + h) - \overline{\Phi}_n(x_n + h) = y(x_{n+1}) - \bar{y}_{n+1} = O(h^{p+1}).$$

The local order of consistency of a VRK formula ca be completely characterized in terms of the parameters of the formula; the resulting equations will be called the *order conditions for the VRK formula* (or *VRKOC*).

The theory for generating the VRKOC uses the V-trees, introduced by Brunner, Hairer and Nørsett in 1982 (cfr. [1]). This theory uses a more complex and general graph theory, the P-trees and P-series introduced by Hairer [6, 7].

We will only see a simple example of order conditions for $p \leq 3$. As in [5], we need to identify the coefficients of h^i in the Taylor expression of $y(x_{n+1})$ and $\bar{y}_{n+1}$, up to order 3.

Let us start with the series expansions of $y(x_{n+1})$ at x_n,

$$\begin{aligned} y(x_{n+1}) =& F_n(x_{n+1}) + hk + \frac{h^2}{2}\left[2k_x + k_s + k_y F_n' + k_y k\right] \\ &+ \frac{h^3}{6}\left[k_y F_n'' + 2k_x k_y + k_s k_y + k_y^2 F_n' + k_y^2 k + 3k_{xx} + k_{ss} + 3k_{xs} + k_{yy} F_n''\right. \\ &\left. + k_{yy}k^2 + 2k_{yy}F_n' k + 2k_{sy}F_n' + 2k_{sy}k + 3k_{xy}F_n' + 3k_{xy}k\right] + O(h^4). \end{aligned}$$

Notice that we omit the dependency of k evaluated in $(x_n, x_n, y(x_n))$.

Now, before expanding $\bar{y}_{n+1}$, let us write as a series $k(x_n + e_i h, x_n + c_i h, \overline{Y}_i^{(n)})$ at $(x_n, x_n, y(x_n))$

$$\begin{aligned} k(x_n + e_i h, x_n + c_i h, \overline{Y}_i^{(n)}) =& k + k_x \frac{e_i h}{1!} + k_s \frac{c_i h}{1!} + k_y \frac{\overline{Y}_i^{(n)} - y(x_n)}{1!} \\ &+ k_{xx}\frac{(e_i h)^2}{2!} + k_{ss}\frac{(c_i h)^2}{2!} + k_{yy}\frac{\left(\overline{Y}_i^{(n)} - y(x_n)\right)^2}{2!} \\ &+ 2k_{xs}\frac{(e_i h)(c_i h)}{2!} + 2k_{xy}\frac{(e_i h)\left(\overline{Y}_i^{(n)} - y(x_n)\right)}{2!} \\ &+ 2k_{sy}\frac{(c_i h)\left(\overline{Y}_i^{(n)} - y(x_n)\right)}{2!} + O(h^3); \end{aligned}$$

from which we obtain $\bar{y}_{n+1}$

$$\begin{aligned}\bar{y}_{n+1} =& F_n(x_{n+1}) + h\sum_{i=1}^{s} b_i k + h^2 \sum_{i=1}^{s} b_i \left[e_i k_x + c_i k_s + \frac{\left(\overline{Y}_i^{(n)} - y(x_n)\right) k_y}{h} \right] \\ &+ \frac{h^3}{2} \sum_{i=1}^{s} b_i \left[e_i^2 k_{xx} + c_i^2 k_{ss} + \frac{\left(\overline{Y}_i^{(n)} - y(x_n)\right)^2 k_{yy}}{h} \right. \\ &\left. + 2e_i c_i k_{xs} + 2\frac{\left(\overline{Y}_i^{(n)} - y(x_n)\right)(e_i k_{xy} + c_i k_{sy})}{h} \right] + O(h^4).\end{aligned}$$

We realize that we need the Taylor expansion of $(\overline{Y}_i^{(n)} - y(x_n))/h$. So consider the series of $\overline{Y}_i^{(n)}$ at x_n

$$\begin{aligned}\overline{Y}_i^{(n)} =& F(x_n) + F_n'(x_n)\theta_i h + F_n''(x_n)\frac{(\theta_i h)^2}{2!} + O(h^3) \\ &+ h\sum_{j=1}^{s} a_{ij} \left[k + k_x d_{ij} h + k_s c_j h + k_y(\overline{Y}_i^{(n)} - y(x_n)) + O(h^2) \right],\end{aligned}$$

and since $F(x_n) = y(x_n)$ and by omitting the dependency of F by x_n, we have

$$\begin{aligned}\frac{\overline{Y}_i^{(n)} - y(x_n)}{h} =& \theta_i F_n' + \sum_{j=1}^{s} a_{ij} k + h \left[\frac{1}{2}\theta_i^2 F_n'' + \sum_{j=1}^{s} a_{ij} d_{ij} k_x \right. \\ &\left. + \sum_{j=1}^{s} a_{ij} c_j k_s + \sum_{j=1}^{s} a_{ij} \theta_j k_y F_n' + \sum_{j,l=1}^{s} a_{ij} a_{jl} k_y k \right] + O(h^2),\end{aligned}$$

finally substitute it in $\bar{y}_{n+1}$,

$$\begin{aligned}\bar{y}_{n+1} =& F_n(x_{n+1}) + h\sum_{i=1}^{s} b_i k + h^2 \sum_{i=1}^{s} b_i \left[e_i k_x + c_i k_s + \theta_i F_n' k_y + \sum_{j=1}^{s} a_{ij} k k_y \right] \\ &+ \frac{h^3}{2} \sum_{i=1}^{s} b_i \left[\theta_i^2 F_n'' k_y + 2\sum_{j=1}^{s} a_{ij} d_{ij} k_x k_y + 2\sum_{j=1}^{s} a_{ij} c_j k_s k_y + 2\sum_{j=1}^{s} a_{ij} \theta_j k_y^2 F_n' \right. \\ &+ 2\sum_{j,l=1}^{s} a_{ij} a_{jl} k_y^2 k + e_i^2 k_{xx} + c_i^2 k_{ss} + \left(\theta_i F_n' + \sum_{j=1}^{s} a_{ij} k \right)^2 k_{yy} + 2e_i c_i k_{xs} \\ &\left. + 2e_i \theta_i k_{xy} F_n' + 2e_i \sum_{j=1}^{s} a_{ij} k_{xy} k + 2c_i \theta_i k_{sy} F_n' + 2c_i \sum_{j=1}^{s} a_{ij} k_{sy} k \right] + O(h^4).\end{aligned}$$

The order condition for a VRK formula are obtained by identifying the coefficients of h^i in the expansion of $\bar{y}_{n+1}$ and $y(x_{n+1})$. In Table 2 these conditions are listed in a *matrix formulation* (as in [13, 14]). Where we use two different type

Table 1: General VRK order conditions.

Order	VRK			
$\mathbf{p=1}$	$\mathbf{b}^T\mathbf{u}=1$			
$\mathbf{p=2}$	$\mathbf{b}^TE\mathbf{u}=1$			
	$\mathbf{b}^TC\mathbf{u}=\frac{1}{2}$	$\mathbf{b}^T\Theta\mathbf{u}=\frac{1}{2}$	$\mathbf{b}^TA\mathbf{u}=\frac{1}{2}$	
$\mathbf{p=3}$	$\mathbf{b}^TE^2\mathbf{u}=1$			
	$\mathbf{b}^TA\circ D\mathbf{u}=\frac{1}{3}$			
	$\mathbf{b}^TAC\mathbf{u}=\frac{1}{6}$	$\mathbf{b}^TA\Theta\mathbf{u}=\frac{1}{6}$	$\mathbf{b}^TA\circ(A\mathbf{u})=\frac{1}{6}$	
	$\mathbf{b}^TC^2\mathbf{u}=\frac{1}{3}$	$\mathbf{b}^T\Theta^2\mathbf{u}=\frac{1}{3}$	$\mathbf{b}^TC\Theta\mathbf{u}=\frac{1}{3}$	$\mathbf{b}^TA\circ A\mathbf{u}=\frac{1}{3}$
		$\mathbf{b}^TCA\mathbf{u}=\frac{1}{3}$	$\mathbf{b}^T\Theta A\mathbf{u}=\frac{1}{3}$	
	$\mathbf{b}^TEC\mathbf{u}=\frac{1}{2}$	$\mathbf{b}^TE\Theta\mathbf{u}=\frac{1}{2}$	$\mathbf{b}^TEA\mathbf{u}=\frac{1}{2}$	

of matrix multiplication, the usual *row-column matrix product* and the *Hadamard product* (or *point-wise product*, or *Schur product*) where the product of two matrices $A=(a_{ij})$ and $D=(d_{ij})$ of the same dimensions, is a matrix $A\circ D$ of the same dimensions, which has elements

$$(A\circ D)_{ij} := a_{ij}d_{ij}.$$

We also define

$$\mathbf{u} := [1,\dots,1]^T,$$

and C, E and Θ the $s\times s$ diagonal matrix such that $\mathbf{c}=C\mathbf{u}$, $\mathbf{e}=E\mathbf{u}$ and $\boldsymbol{\theta}=\Theta\mathbf{u}$.

We can notice that we have some redundant conditions, in fact wee need only the the ones presented in the first column of Table 2, if the suppose the *row sum condition* (or *RSC*)

$$\theta_i = c_i = \sum_{j=1}^{m} a_{ij} \qquad i=1,\dots,m,$$

i.e.

$$\boldsymbol{\theta}=\mathbf{c}=A\mathbf{u}.$$

In fact, the parameters θ_i have to satisfy the same conditions as the parameters c_i. Therefore, the conditions $\boldsymbol{\theta}=\mathbf{c}$.

Moreover, on substitution of $\Theta=C$, we see that the sum on the i-th row of A is coincides with c_i, i.e. $c_i=\sum_{j=1}^{m}a_{ij}$ or $C\mathbf{u}=A\mathbf{u}$.

So the RSC does not represent a loss of generality.

In Table 2 we list the order conditions for the VRK formulas (13) and PVRK (8) and BVRK (9) formulas up to order 4, while in Table 3 we list the order $p=5$.

Note that we use a matrix notation with a little modification. We suppose that the the Hadamard product $\circ$ is distributive respect the row-column product; moreover we suppose that in using the power of a matrix are repeating the Hadamard

Table 2: VRK, PVRK and BVRK order conditions u to order $p = 4$. This table can be generated with NviePy.

Order **p**	VRK	PVRK ($e_i = 1, d_{ij} = c_i$)	BVRK ($e_i = d_i, d_{ij} = d_j$)
1	$\mathbf{b}^T\mathbf{u} = 1$	1. $\mathbf{b}^T\mathbf{u} = 1$	1. $\mathbf{b}^T\mathbf{u} = 1$
2	$\mathbf{b}^T E\mathbf{u} = 1$		2. $\mathbf{b}^T D\mathbf{u} = 1$
	$\mathbf{b}^T C\mathbf{u} = \frac{1}{2}$	2. $\mathbf{b}^T C\mathbf{u} = \frac{1}{2}$	3. $\mathbf{b}^T C\mathbf{u} = \frac{1}{2}$
3	$\mathbf{b}^T E^2\mathbf{u} = 1$		4. $\mathbf{b}^T D^2\mathbf{u} = 1$
	$\mathbf{b}^T A\circ D\mathbf{u} = \frac{1}{3}$		5. $\mathbf{b}^T AD\mathbf{u} = \frac{1}{3}$
	$\mathbf{b}^T AC\mathbf{u} = \frac{1}{6}$	3. $\mathbf{b}^T AC\mathbf{u} = \frac{1}{6}$	6. $\mathbf{b}^T AC\mathbf{u} = \frac{1}{6}$
	$\mathbf{b}^T C^2\mathbf{u} = \frac{1}{3}$	4. $\mathbf{b}^T C^2\mathbf{u} = \frac{1}{3}$	7. $\mathbf{b}^T C^2\mathbf{u} = \frac{1}{3}$
	$\mathbf{b}^T CE\mathbf{u} = \frac{1}{2}$		8. $\mathbf{b}^T CD\mathbf{u} = \frac{1}{2}$
4	$\mathbf{b}^T E^3\mathbf{u} = 1$		9. $\mathbf{b}^T D^3\mathbf{u} = 1$
	$\mathbf{b}^T A\circ D^2\mathbf{u} = \frac{1}{4}$		10. $\mathbf{b}^T AD^2\mathbf{u} = \frac{1}{4}$
	$\mathbf{b}^T AA\circ D\mathbf{u} = \frac{1}{12}$		11. $\mathbf{b}^T AAD\mathbf{u} = \frac{1}{12}$
	$\mathbf{b}^T AAC\mathbf{u} = \frac{1}{24}$	5. $\mathbf{b}^T AAC\mathbf{u} = \frac{1}{24}$	12. $\mathbf{b}^T AAC\mathbf{u} = \frac{1}{24}$
	$\mathbf{b}^T AC^2\mathbf{u} = \frac{1}{12}$	6. $\mathbf{b}^T AC^2\mathbf{u} = \frac{1}{12}$	13. $\mathbf{b}^T AC^2\mathbf{u} = \frac{1}{12}$
	$\mathbf{b}^T A\circ DC\mathbf{u} = \frac{1}{8}$		14. $\mathbf{b}^T ACD\mathbf{u} = \frac{1}{8}$
	$\mathbf{b}^T CA\circ D\mathbf{u} = \frac{1}{4}$		15. $\mathbf{b}^T CAD\mathbf{u} = \frac{1}{4}$
	$\mathbf{b}^T CAC\mathbf{u} = \frac{1}{8}$	7. $\mathbf{b}^T CAC\mathbf{u} = \frac{1}{8}$	16. $\mathbf{b}^T CAC\mathbf{u} = \frac{1}{8}$
	$\mathbf{b}^T C^3\mathbf{u} = \frac{1}{4}$	8. $\mathbf{b}^T C^3\mathbf{u} = \frac{1}{4}$	17. $\mathbf{b}^T C^3\mathbf{u} = \frac{1}{4}$
	$\mathbf{b}^T EA\circ D\mathbf{u} = \frac{1}{3}$		18. $\mathbf{b}^T DAD\mathbf{u} = \frac{1}{3}$
	$\mathbf{b}^T EAC\mathbf{u} = \frac{1}{6}$		19. $\mathbf{b}^T DAC\mathbf{u} = \frac{1}{6}$
	$\mathbf{b}^T C^2E\mathbf{u} = \frac{1}{3}$		20. $\mathbf{b}^T C^2D\mathbf{u} = \frac{1}{3}$
	$\mathbf{b}^T CE^2\mathbf{u} = \frac{1}{2}$		21. $\mathbf{b}^T CD^2\mathbf{u} = \frac{1}{2}$

Table 3: VRK, PVRK e BVRK order condition of order $p = 5$. This Table has been generated with NviePy.

VRK	PVRK	BVRK
$\mathbf{b}^T E^4 \mathbf{u} = 1$		22. $\mathbf{b}^T D^4 \mathbf{u} = 1$
$\mathbf{b}^T A \circ D^3 \mathbf{u} = \frac{1}{5}$		23. $\mathbf{b}^T A D^3 \mathbf{u} = \frac{1}{5}$
$\mathbf{b}^T AA \circ D^2 \mathbf{u} = \frac{1}{20}$		24. $\mathbf{b}^T AAD^2 \mathbf{u} = \frac{1}{20}$
$\mathbf{b}^T AAA \circ D\mathbf{u} = \frac{1}{60}$		25. $\mathbf{b}^T AAAD\mathbf{u} = \frac{1}{60}$
$\mathbf{b}^T AAAC\mathbf{u} = \frac{1}{120}$	9. $\mathbf{b}^T AAAC\mathbf{u} = \frac{1}{120}$	26. $\mathbf{b}^T AAAC\mathbf{u} = \frac{1}{120}$
$\mathbf{b}^T AAC^2 \mathbf{u} = \frac{1}{60}$	10. $\mathbf{b}^T AAC^2 \mathbf{u} = \frac{1}{60}$	27. $\mathbf{b}^T AAC^2 \mathbf{u} = \frac{1}{60}$
$\mathbf{b}^T AA \circ DC\mathbf{u} = \frac{1}{40}$		28. $\mathbf{b}^T AACD\mathbf{u} = \frac{1}{40}$
$\mathbf{b}^T ACA \circ D\mathbf{u} = \frac{1}{20}$		29. $\mathbf{b}^T ACAD\mathbf{u} = \frac{1}{20}$
$\mathbf{b}^T ACAC\mathbf{u} = \frac{1}{40}$	11. $\mathbf{b}^T ACAC\mathbf{u} = \frac{1}{40}$	30. $\mathbf{b}^T ACAC\mathbf{u} = \frac{1}{40}$
$\mathbf{b}^T AC^3 \mathbf{u} = \frac{1}{20}$	12. $\mathbf{b}^T AC^3 \mathbf{u} = \frac{1}{20}$	31. $\mathbf{b}^T AC^3 \mathbf{u} = \frac{1}{20}$
$\mathbf{b}^T A \circ DA \circ D\mathbf{u} = \frac{1}{15}$		32. $\mathbf{b}^T ADAD\mathbf{u} = \frac{1}{15}$
$\mathbf{b}^T A \circ DAC\mathbf{u} = \frac{1}{30}$		33. $\mathbf{b}^T ADAC\mathbf{u} = \frac{1}{30}$
$\mathbf{b}^T A \circ DC^2 \mathbf{u} = \frac{1}{15}$		34. $\mathbf{b}^T AC^2 D\mathbf{u} = \frac{1}{15}$
$\mathbf{b}^T A \circ D^2 C\mathbf{u} = \frac{1}{10}$		35. $\mathbf{b}^T ACD^2 \mathbf{u} = \frac{1}{10}$
$\mathbf{b}^T CA \circ D^2 \mathbf{u} = \frac{1}{5}$		36. $\mathbf{b}^T CAD^2 \mathbf{u} = \frac{1}{5}$
$\mathbf{b}^T CAA \circ D\mathbf{u} = \frac{1}{15}$		37. $\mathbf{b}^T CAAD\mathbf{u} = \frac{1}{15}$
$\mathbf{b}^T CAAC\mathbf{u} = \frac{1}{30}$	13. $\mathbf{b}^T CAAC\mathbf{u} = \frac{1}{30}$	38. $\mathbf{b}^T CAAC\mathbf{u} = \frac{1}{30}$
$\mathbf{b}^T CAC^2 \mathbf{u} = \frac{1}{15}$	14. $\mathbf{b}^T CAC^2 \mathbf{u} = \frac{1}{15}$	39. $\mathbf{b}^T CAC^2 \mathbf{u} = \frac{1}{15}$
$\mathbf{b}^T CA \circ DC\mathbf{u} = \frac{1}{10}$		40. $\mathbf{b}^T CACD\mathbf{u} = \frac{1}{10}$
$\mathbf{b}^T (A \circ D\mathbf{u}) \circ (A \circ D\mathbf{u}) = \frac{1}{5}$		41. $\mathbf{b}^T (AD\mathbf{u}) \circ (AD\mathbf{u}) = \frac{1}{5}$
$\mathbf{b}^T (A \circ D\mathbf{u}) \circ (AC\mathbf{u}) = \frac{1}{10}$		42. $\mathbf{b}^T (AD\mathbf{u}) \circ (AC\mathbf{u}) = \frac{1}{10}$
$\mathbf{b}^T (AC\mathbf{u}) \circ (AC\mathbf{u}) = \frac{1}{20}$	15. $\mathbf{b}^T (AC\mathbf{u}) \circ (AC\mathbf{u}) = \frac{1}{20}$	43. $\mathbf{b}^T (AC\mathbf{u}) \circ (AC\mathbf{u}) = \frac{1}{20}$
$\mathbf{b}^T C^2 A \circ D\mathbf{u} = \frac{1}{5}$		44. $\mathbf{b}^T C^2 AD\mathbf{u} = \frac{1}{5}$
$\mathbf{b}^T C^2 AC\mathbf{u} = \frac{1}{10}$	16. $\mathbf{b}^T C^2 AC\mathbf{u} = \frac{1}{10}$	45. $\mathbf{b}^T C^2 AC\mathbf{u} = \frac{1}{10}$
$\mathbf{b}^T C^4 \mathbf{u} = \frac{1}{5}$	17. $\mathbf{b}^T C^4 \mathbf{u} = \frac{1}{5}$	46. $\mathbf{b}^T C^4 \mathbf{u} = \frac{1}{5}$
$\mathbf{b}^T EA \circ D^2 \mathbf{u} = \frac{1}{4}$		47. $\mathbf{b}^T DAD^2 \mathbf{u} = \frac{1}{4}$
$\mathbf{b}^T EAA \circ D\mathbf{u} = \frac{1}{12}$		48. $\mathbf{b}^T DAAD\mathbf{u} = \frac{1}{12}$
$\mathbf{b}^T EAAC\mathbf{u} = \frac{1}{24}$		49. $\mathbf{b}^T DAAC\mathbf{u} = \frac{1}{24}$
$\mathbf{b}^T EAC^2 \mathbf{u} = \frac{1}{12}$		50. $\mathbf{b}^T DAC^2 \mathbf{u} = \frac{1}{12}$
$\mathbf{b}^T EA \circ DC\mathbf{u} = \frac{1}{8}$		51. $\mathbf{b}^T DACD\mathbf{u} = \frac{1}{8}$
$\mathbf{b}^T CEA \circ D\mathbf{u} = \frac{1}{4}$		52. $\mathbf{b}^T CDAD\mathbf{u} = \frac{1}{4}$
$\mathbf{b}^T CEAC\mathbf{u} = \frac{1}{8}$		53. $\mathbf{b}^T CDAC\mathbf{u} = \frac{1}{8}$
$\mathbf{b}^T C^3 E\mathbf{u} = \frac{1}{4}$		54. $\mathbf{b}^T C^3 D\mathbf{u} = \frac{1}{4}$
$\mathbf{b}^T E^2 A \circ D\mathbf{u} = \frac{1}{3}$		55. $\mathbf{b}^T D^2 AD\mathbf{u} = \frac{1}{3}$
$\mathbf{b}^T E^2 AC\mathbf{u} = \frac{1}{6}$		56. $\mathbf{b}^T D^2 AC\mathbf{u} = \frac{1}{6}$
$\mathbf{b}^T C^2 E^2 \mathbf{u} = \frac{1}{3}$		57. $\mathbf{b}^T C^2 D^2 \mathbf{u} = \frac{1}{3}$
$\mathbf{b}^T CE^3 \mathbf{u} = \frac{1}{2}$		58. $\mathbf{b}^T CD^3 \mathbf{u} = \frac{1}{2}$

Table 4: The number of order conditions for PVRK and formulas for $p \leq 10$.

Order p		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
PVRK	n_p	1	1	2	4	9	20	48	115	286	719
	N_p	1	2	4	8	17	37	85	200	486	1205
BVRK	n_p	1	2	5	13	37	108	332	1042	3360	11019
	N_p	1	3	8	21	58	166	498	1540	4900	15919

product, this clarification is unnecessary for diagonal matrix, but is relevant for the A matrix.

We can notice how much the software is useful by observing that the number of order conditions increases rapidly. Table 4 gives a comparison of the number of order conditions for the two classes of VRK formulas: PVRK and BVRK; n_p denotes the number of order conditions necessary to increase the order from $p-1$ to p, while N_p is the total number of order conditions to attain order p.

3 The NviePy software

NviePy (Numerical VIEs in Python) is a Python package for obtaining order conditions for a Runge-Kutta method for VIE, and could be used to obtain new methods or check the order of a given one. It can be the base for a more general and complex software. NviePy is released under a modified BSD license, and is based on *NodePy* (crf. [10]).

The code can be found at *GitHub*: `http://github.com/MathBear/nviepy`.

I depends on: *Python, Numpy, Matplotlib and SymPy* [8, 11].

Using NviePy 0.1 you can generate Volterra tress and Butcher trees as strings or symbolically and do a plot of these trees. If you select a string output you ca choose between: Octave, Python or LaTeX ; whereas by choosing a symbolic output you can manipulate these order conditions using Sympy or more simplifying function available in NviePy.

The NviePy package consists of a folder `nviepy` containing four listings: two are useful to work with strings and generate the partition of an integer (`strmainip`, `utilities`); while for the remaining other two, one creates trees and order conditions `volterra_rooted_3s` and the other is a collection of functions to create and manipulate symbolically order conditions in order to obtain further methods `vrk_methods`.

There is the directory `examples`, containing some examples on how to generate order conditions; and a directory `notebook`, containing notebook files `.ipynb` that generate VRK formulas reported in [1, 2, 9].

The software is documented using the markup language *reStructuredText* (*reST*), and so an external documentation is available using *Sphinx*.

Since you can find more in the code, let us see only how to start using NviePy.

3.1 How to start

I used Ubuntu 12.04 LTS, so the steps I will be showing are for this distro, but you can do similar steps for other Linux distro, Win or Mac.

First of all we need the packages to develop with Python, and the scientific ones

```
$ sudo apt-get install python python-dev python-numpy python-
    scipy python-matplotlib python-sympy
```

furthermore I advise to use a nice interactive interface as Mathematica

```
$ sudo pip install ipython ipython-notebook ipython-qtconsole
```

Notice that I have used the following versions of the software SymPy 0.7.2, IPython 0.13.2, Python 2.7.5, Matplotlib 1.2.1.

You might want to install more packages to develop with Python

```
$ sudo apt-get install python-pip python-qt4  python-
    distutils-extra python-setuptools build-essential pyflakes
     python-rope pylint pep8
```

And I also advise you to install

```
$ sudo apt-get install python-sphinx spyder git
```

the first one, Sphinx, is for generate external documentation; the second is an IDE, and stands for *Scientific PYthon Development EnviRonment* necessary if you want to do scientific programming with Python; and at least Git so you can contribute to NviePy.

Now we have to download NviePy 0.1. We can do this by going to the link of the project and downloading the .zip file containing the code. Or going directly to `http://github.com/MathBear/nviepy/zipball/master/`. Otherwise if you installed Git, simply execute

```
  $ git clone git://github.com/MathBear/nviepy.git
Cloning into 'nviepy'...
remote: Counting objects: 982, done.
remote: Compressing objects: 100% (410/410), done.
remote: Total 982 (delta 572), reused 955 (delta 557)
Receiving objects: 100% (982/982), 1.15 MiB | 408 KiB/s, done.
Resolving deltas: 100% (572/572), done.
```

Now we have a folder `nviepy`, located in a certain `path/to/package/` containing the program. Instead of moving this folder in one of the Python's default directory, we can set the `PYTHONPATH` of our OS. To modify the `PYTHONPATH`, in Ubuntu, just open the `.profile` that you have in your `home` and paste

```
export PYTHONPATH="path/to/package/"
```

Finally to apply the change just restart the session.

I we do not set the path each time we execute `python` we need to give the following commands

```
import sys
sys.path.append("//path/to/package/")
```

Notice that `sys.path` gives the list of all directories in which Python will search for `.py` programs or modules.

Finally we can use NviePy in the terminal, for example

```
  $ python
Python 2.7.3 (default, Sep 26 2012, 21:53:58)
[GCC 4.7.2] on linux2
Type "help", "copyright", "credits" or "license" for more
   information.
>>> from nviepy import *
>>> print vrt.list_trees(3)
['{x^2}', '{{x}}', '{{a}}', '{aa}', '{xa}']
>>>
```

Instead of the terminal you can have different type of interface experience, I suggest *Spyder* an IDE Matlab-like, or *IPython* or his *notebook* for a Mathematica experience. In order to use a notebook just execute from the terminal

```
  $ ipython notebook
```

a browser will open an you can execute `.ipynb` files.

For more on Python see `http://docs.python.org/2.7/`, for NumPy `http://www.scipy.org/NumPy_for_Matlab_Users`, and for SymPy `http://github.com/sympy/sympy/wiki/SymPy-vs.-Mathematica` .

References

[1] H. Brunner, E. Hairer, and S. P. Nørsett. Runge-Kutta Theory for Volterra Integral Equations of the Second Kind. *Mathematics of computation*, 39(159):147–163, 1982.

[2] H. Brunner and P. J. Houwen. *The numerical solution of Volterra equations*. CWI monograph. North-Holland, 1986.

[3] T. A. Burton. *Volterra Integral And Differential Equations*. Mathematics in Science and Engineering. Elsevier Science Limited, 2005.

[4] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2008.

[5] C. Chiapparelli. *Metodi V_0-stabili per la risoluzione di equazioni integrali di Volterra di seconda specie*. PhD thesis, Università degli Studi di Napoli Federico II, 2011.

[6] E. Hairer. Order Conditions for Numerical Methods for Partitioned Ordinary Differential Equations. *Numerische Mathematik*, 36(4):431–445, 1981.

[7] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, 2008.

[8] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[9] G. Izzo, E. Russo, and C. Chiapparelli. Highly stable Runge-Kutta methods for Volterra integral equations. *Appl. Numer. Math.*, 62(8):1002–1013, 2012.

[10] D. I. Ketcheson. NodePy software version 0.4. `http://numerics.kaust.edu.sa/nodepy/`. [Online; in data 8-giugno-2013].

[11] F. Pérez and B. E. Granger. IPython: a System for Interactive Scientific Computing. *Comput. Sci. Eng.*, 9(3):21–29, May 2007.

[12] V. Schiano Di Cola. Volterra Trees and Order Conditions for Volterra Runge-Kutta type methods. Master's thesis, Università degli Studi di Napoli Federico II, Naples, 2013.

[13] P. W. Sharp and J. H. Verner. Extended explicit Bel'tyukov pairs of orders 4 and 5 for Volterra integral equations of the second kind. *Appl. Numer. Math.*, 34(2-3):261–274, 2000.

[14] P. W. Sharp and J. H. Verner. Some Extended Explicit Bel'tyukov Pairs for Volterra Integral Equations of the Second Kind. *SIAM J. Numer. Anal.*, 38(2):347–359, 2000.

[15] M. Sofroniou. Symbolic Derivation of Runge-Kutta Methods. *Journal of Symbolic Computation*, 18(3):265 – 296, 1994.