JOURNAL OF NUMERICAL ANALYSIS AND APPROXIMATION THEORY
J. Numer. Anal. Approx. Theory, vol. 45 (2016) no. 1, pp. 87–96
ictp.acad.ro/jnaat

ON THE CONVERGENCE OF THE VARIATIONAL ITERATION METHOD

ERNEST SCHEIBER*

Abstract. Convergence results are stated for the variational iteration method applied to solve an initial value problem for a system of ordinary differential equations.

MSC 2010. 65L20, 65L05, 97N80.

Keywords. variational iteration method, ordinary differential equations, initial value problem.

1. INTRODUCTION

The Ji-Huan He's Variational Iteration Method (VIM) was applied to a large range of problems for both ordinary and partial differential equations. The main ingredient of the VIM is the Lagrange multiplier used to improve an approximation of the solution of the differential problem [2].

The purpose of this paper is to prove a convergence theorem for VIM applied to solve an initial value problem for a system of ordinary differential equations.

The convergence of the VIM for the initial value problem of an ordinary differential equation may be found in D.K. Salkuyeh, A. Tavakoli [6]. For a system of linear differential equations a convergence result is given by D.K. Salkuyeh [5].

A particularity of the VIM is that it may be implemented both in symbolic (Computer Algebra System) and numerical programming environments. In the last section there are presented some results of our computational experiences. To make the results reproducible we provide some code. In [1] there is a pertinent presentation of the issues concerning the publishing of scientific computations.

2. THE CONVERGENCE OF VIM FOR A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS

Consider the following system of ordinary differential equations

*e-mail: scheiber@unitbv.ro.

$$(1)\qquad \begin{cases} x_1'(t) = f_1(t, x_1(t), \ldots, x_m(t)) & x_1(t_0) = x_1^0 \\ \vdots & \\ x_m'(t) = f_m(t, x_1(t), \ldots, x_m(t)) & x_m(t_0) = x_m^0 \end{cases}$$

where $t \in [t_0, t_f]$ with $t_0 < t_f < \infty$.

We shall use the notations

$$\mathbf{x} = (x_1, \ldots, x_m), \quad \|\mathbf{x}\|_1 = \sum_{j=1}^{m} |x_j|$$
$$\mathbf{x} = \mathbf{x}(t), \quad \|\mathbf{x}\|_\infty = \max_t \|\mathbf{x}(t)\|_1.$$

Thus, any equation of (1) may be rewritten as

$$x_i'(t) = f_i(t, \mathbf{x}(t)), \quad i \in \{1, \ldots, m\}.$$

The following hypothesis are introduced:

- The functions $f_1, \ldots, f_m$ are continuous and have first and second order partial derivatives in $x_1, \ldots, x_m$.
- There exists $L > 0$ such that for any $i \in \{1, \ldots, m\}$

$$|f_i(t, \mathbf{x}) - f_i(t, \mathbf{y})| \le L \sum_{j=1}^{m} |x_j - y_j| = L\|\mathbf{x} - \mathbf{y}\|_1, \quad \forall\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^m.$$

As a consequence

$$\Big|\frac{\partial f_i(t, \mathbf{x})}{\partial x_j}\Big| = |f_{i_{x_j}}(t, \mathbf{x})| \le L, \quad \forall (t, \mathbf{x}) \in [t_0, t_f] \times \mathbb{R}^m,\ \forall i, j \in \{1, \ldots, m\}.$$

According to the VIM the sequences of approximations are

$$(2)\qquad u_{n+1,i}(t) = u_{n,i}(t) + \int_{t_0}^{t} \lambda_i(s)(u_{n,i}'(s) - f_i(s, \mathbf{u}_n(s)))\mathrm{d}s, \quad n \in \mathbb{N},$$

$i \in \{1, \ldots, m\}$ and where $\mathbf{u}_n = (u_{n,1}, \ldots, u_{n,m})$.

It is supposed that $u_{n,i}(t_0) = x_i^0$ and that $u_{n,i}$ is a continuous differentiable function for any $i \in \{1, \ldots, m\}$.

In this case the VIM is a little trickier: the Lagrange multiplier attached to the i-th equation will act only on x_i [5].

Denoting $\mathbf{x}(t) = (x_1(t), \ldots, x_m(t))$ the solution of the initial value problem (1), if $u_{n,i}(t) = x_i(t) + \delta u_{n,i}(t)$ and $u_{n+1,i}(t) = x_i(t) + \delta u_{n+1,i}(t)$ but $u_{n,j}(t) =$

$x_j(t)$, for $j \neq i$, then (2) implies

$$\delta u_{n+1,i}(t) =$$

$$= \delta u_{n,i}(t) + \int_{t_0}^{t} \lambda_i(s) \Big(x_i'(s) + \delta u_{n,i}'(s) -$$

$$- f_i(s, x_1(s), \ldots, x_{i-1}(s), x_i(s) + \delta u_{n,i}(s), x_{i+1}(s), \ldots, x_m(s))) \,\mathrm{d}s =$$

$$= \delta u_{n,i}(t) + \int_{t_0}^{t} \lambda_i(s) \Big(x_i'(s) + \delta u_{n,i}'(s) - f_i(s, \mathbf{x}(s)) - f_{i x_i}(s, \mathbf{x}(s)) \delta u_{n,i}(s) \Big) \,\mathrm{d}s +$$

$$+ \mathcal{O}((\delta u_{n,i})^2) =$$

$$= \delta u_{n,i}(t) + \int_{t_0}^{t} \lambda_i(s) \Big(\delta u_{n,i}'(s) - f_{i x_i}(s, \mathbf{x}(s)) \delta u_{n,i}(s) \Big) \,\mathrm{d}s + \mathcal{O}((\delta u_{n,i})^2).$$

After the integration by parts the above equality becomes

$$\delta u_{n+1,i}(t) =$$

$$(1 + \lambda_i(t)) \delta u_{n,i}(t) - \int_{t_0}^{t} \Big(\lambda_i'(s) + f_{i x_i}(s, \mathbf{x}(s)) \lambda_i(s) \Big) \delta u_{n,i}(s) \mathrm{d}s + \mathcal{O}((\delta u_n)^2).$$

In order that $u_{n+1,i}$ be a better approximation than $u_{n,i}$, it is required that λ_i is the solution of the following initial value problem

$$\lambda'(s) = -f_{i x_i}(s, \mathbf{x}(s)) \lambda(s), \quad s \in [t_0, t], \tag{3}$$

$$\lambda(t) = -1. \tag{4}$$

Because $\mathbf{x}(s)$ is an unknown function, the following problem is considered instead of (3)–(4)

$$\lambda'(s) = -f_x(s, \mathbf{u}_n(s)) \lambda(s), \quad s \in [t_0, t], \tag{5}$$

$$\lambda(t) = -1 \tag{6}$$

with the solution denoted $\lambda_{n,i}(s,t)$. The solution is

$$\lambda_{n,i}(s,t) = -e^{\int_s^t f_{i x_i}(\tau, \mathbf{u}_n(\tau)) \mathrm{d}\tau}$$

and

$$|\lambda_{n,i}(s,t)| \leq e^{L(t-s)} \leq e^{LT}, \qquad \forall\ t_0 \leq s \leq t \leq t_f \text{ and } T = t_f - t_0.$$

The recurrence formula (2) becomes

$$u_{n+1,i}(t) = u_{n,i}(t) + \int_{t_0}^{t} \lambda_{n,i}(s,t) (u_{n,i}'(s) - f_i(s, \mathbf{u}_n(s))) \mathrm{d}s, \quad n \in \mathbb{N}, \tag{7}$$

for any $i \in \{1, \ldots, m\}$.

The convergence result is:

Theorem 1. *If the hypotheses stated above are valid, then the sequence* $(\mathbf{u}_n)_{n \in \mathbb{N}}$ *defined by* (7) *converges uniformly to* $\mathbf{x}(t)$, *the solution of the initial value problem* (1).

Proof. Subtracting the equality

$$x_i(t) = x_i(t) + \int_0^t \lambda_{n,i}(s,t)\left(x_i'(s) - f_i(s,\mathbf{x}(s))\right)\mathrm{d}s$$

from (7) leads to

$$e_{n+1,i}(t) = e_{n,i}(t) + \int_{t_0}^t \lambda_{n,i}(s,t)\Big(e_{n,i}'(s) - (f_i(s,\mathbf{u}_n(s)) - f_i(s,\mathbf{x}(s)))\Big)\mathrm{d}s,$$

where $e_{n,i}(t) = u_{n,i}(t) - x_i(t),\ i \in \{1,\ldots,m\}$ and $\mathbf{e}_n(t) = (e_{n,1}(t),\ldots,e_{n,m}(t)) = \mathbf{u}_n(t) - \mathbf{x}(t),\ n \in \mathbb{N}$.

Again, an integration by parts gives

$$e_{n+1,i}(t) = \int_{t_0}^t \lambda_{n,i}(s,t)\Big(f_{i_{x_i}}(s,\mathbf{u}_n(s))e_{n,i}(s) - (f_i(s,\mathbf{u}_n(s)) - f_i(s,\mathbf{x}(s)))\Big)\mathrm{d}s.$$

The hypothesis on f_i implies the inequality

$$\Big|f_{i_{x_i}}(s,\mathbf{u}_n(s))e_{n,i}(s) - (f_i(s,\mathbf{u}_n(s)) - f_i(s,\mathbf{x}(s))\Big| \le L|e_{n,i}(s)| + L\|\mathbf{e}_n(s)\|_1$$

and consequently

$$|e_{n+1,i}(t)| \le Le^{LT}\int_{t_0}^t (|e_{n,i}(s)| + \|\mathbf{e}_n(s)\|_1)\mathrm{d}s.$$

Summing these inequalities, for $i = 1 : m$, we find

$$\|\mathbf{e}_{n+1}(t)\|_1 \le (m+1)Le^{LT}\int_{t_0}^t \|\mathbf{e}_n(s)\|_1\mathrm{d}s. \tag{8}$$

Let $M = (m+1)Le^{LT}$. From (8) we obtain successively:
For $n = 0$

$$\|\mathbf{e}_1(t)\|_1 \le M\int_{t_0}^t \|\mathbf{e}_0(s)\|_1\mathrm{d}s \le M(t-t_0)\|\mathbf{e}_0\|_\infty \ \Rightarrow\ \|\mathbf{e}_1\|_\infty \le MT\|\mathbf{e}_0\|_\infty.$$

For $n = 1$

$$\|\mathbf{e}_2(t)\|_1 \le M\int_{t_0}^t \|\mathbf{e}_1(s)\|_1\mathrm{d}s \le \tfrac{M^2(t-t_0)^2}{2}\|\mathbf{e}_0\|_\infty \ \Rightarrow\ \|\mathbf{e}_2\|_\infty \le \tfrac{M^2T^2}{2}\|\mathbf{e}_0\|_\infty.$$

Inductively, it results that

$$\|\mathbf{e}_n(t)\|_1 \le M\int_{t_0}^t \|\mathbf{e}_{n-1}(s)\|_1\mathrm{d}s \le \tfrac{M^n(t-t_0)^n}{n!}\|\mathbf{e}_0\|_\infty \ \Rightarrow\ \|\mathbf{e}_n\|_\infty \le \tfrac{M^nT^n}{n!}\|\mathbf{e}_0\|_\infty.$$

and hence $\lim_{n\to\infty}\|\mathbf{e}_n\|_\infty = 0$. □

A numerical implementation requires the usage of a quadrature method to compute the integral in (7) and the Lagrange multipliers.

3. COMPUTATIONAL RESULTS

The target of the given examples is twofold: to exemplify the convergence of the VIM and to obtain some clues about the usage of numerical vs. symbolical computations of VIM.

Example 1.

$$\begin{aligned} &x'(t) = 2x(t) + t \\ &x(0) = 0 \end{aligned}$$

The solution $x(t) = \frac{1}{4}(e^{2t} - 2t - 1)$ is obtained in an iteration with the *Mathematica* code provided in Appendix A. $\square$

Example 2.

$$\begin{aligned} &x'(t) = 1 - x^2(t) \\ &x(0) = 0 \end{aligned}$$

The initial value problem has the solution $x(t) = \frac{e^{2t}-1}{e^{2t}+1}$. The code used previously does not give an acceptable result in a reasonable time.

Moving on numerical computation we obtain practical results. The relation (7) is transformed into

$$u_{n+1,i}(t) = \int_{t_0}^{t} \Big(f_i(s, \mathbf{u}_n(s)) - f_{ix_i}(s, \mathbf{u}_n(s)) u_{n,i}(s) \Big) e^{\int_s^t f_{ix_i}(\tau, \mathbf{u}_n(\tau)) \mathrm{d}\tau} \mathrm{d}s + \\ + e^{\int_{t_0}^t f_{ix_i}(\tau, \mathbf{u}_n(\tau)) \mathrm{d}\tau} x^0. \tag{9}$$

Let $(t_i)_{0 \le i \le I}$ be an equidistant grid on $[t_0, t_f]$ and denote by $\mathbf{u}_i$ an approximation of $\mathbf{x}(t_i)$. Furthermore, the recurrence relation (9) is used only to compute $\mathbf{u}_{i+1}$ from $\mathbf{u}_i$, i.e. on a $[t_i, t_{i+1}]$ interval. The integrals in (9) will be computed with the trapezoidal rule using a local equidistant grid.

The iterations are done until the distance between two consecutive approximation of $\mathbf{u}_{i+1}$ is less then a given tolerance.

The final approximate solution is a first order spline function defined by the points $(t_i, \mathbf{u}_i)_{0 \le i \le I}$.

This procedure requires a single passage from t_0 to t_f.

All our numerical results were computed using the *Scilab* code presented in Appendix B [11].

For $t_f = 1,\ I = 100$ we obtained $\max_{0 \le i \le I} |u_i - x(t_i)| \approx 0.4 \times 10^{-6}$. $\square$

Example 3. [5]

$$\begin{aligned} &\dot{x}_1 = 4x_1 + 6x_2 + 6x_3 && x_1(0) = 7 \\ &\dot{x}_2 = x_1 + 3x_2 + 2x_3 && x_2(0) = 2 \\ &\dot{x}_3 = -x_1 - 5x_2 - 2x_3 && x_3(0) = -\tfrac{9}{2} \end{aligned}$$

The solution is $x_1(t) = 4e^t + 3(1+t)e^{2t},\ x_2(t) = e^t + (1+t)e^{2t},\ x_3(t) = -3e^t - (\frac{3}{2} + 2t)e^{2t}$.

For $t_f = 1,\ I = 100$ we found $\max_{0 \le i \le I} \|\mathbf{u}_i - \mathbf{x}(t_i)\| \approx 0.8363 \times 10^{-3}$. $\square$

Example 4.

$$\begin{array}{ll} x_1'(t) = x_2(t), & x_1(0) = \frac{1}{2} \\ x_2'(t) = \frac{x_1(t)x_2^2(t)}{x_1^2(t)-1}, & x_2(0) = \frac{\sqrt{3}}{2} \end{array} \quad \Leftrightarrow \quad \begin{array}{l} (x^2(t)-1)x''(t) = x(t)x'^2(t) \\ x(0) = \frac{1}{2} \\ x'(0) = \frac{\sqrt{3}}{2} \end{array}$$

The solution is $x_1(t) = \sin{(t + \frac{\pi}{6})},\ x_2(t) = \cos{(t + \frac{\pi}{6})}$.
For $t_f = \frac{\pi}{3},\ I = 100$ the result is $\max_{0 \le i \le I} \|\mathbf{u}_i - \mathbf{x}(t_i)\| \approx 0.62 \times 10^{-5}$. □

Example 5. Van der Pol equation

$$\begin{array}{ll} x_1'(t) = x_2(t), & x_1(0) = 0.5 \\ x_2'(t) = (1 - x_1^2(t))x_2(t) - x_1(t), & x_2(0) = 0 \end{array} \quad \Leftrightarrow$$

$$\Leftrightarrow \quad \begin{array}{l} x''(t) - (1 - x^2(t))x'(t) + x(t) = 0 \\ x(0) = 0.5 \\ x'(0) = 0 \end{array}$$

In this case we do not have a closed form of the solution. We compare the VIM approximation with the solution $\mathbf{v}$ obtained with `ode`, a *Scilab* numerical integration function.

The obtained results are given in Table 1. □

t_f	I	$\max_{0 \le i \le I} \|\mathbf{u}_i - \mathbf{v}_i\|$
10	100	0.0001988
20	100	0.0033857
30	100	0.0142215
40	100	0.0384137
50	100	0.0777549
100	100	0.6410885
100	1000	0.0069729

Table 1. Numerical results for the Van der Pol equation.

4. CONCLUSIONS

Despite the convergence properties of the method the amount of the computation is greater than of the usual methods (*e.g.* Runge-Kutta, Adams type methods). Even so the numerical solution can be taken into consideration. The numerical implementation can be improved by an adaptive approach and using some parallel techniques (*e.g.* OpenCL / CUDA) in an appropriate environment.

Although the VIM may be implemented for symbolic computation our experiments show disappointing results.

The VIM offers a way to obtain a symbolic approximation of the solution of the initial value problem. But such an approximation may also be obtained from a numerical solution with the *Eureqa* software [10], [7]. A better symbolic implementation would be useful.

Acknowledgements. The author want to thank the referee for his helpful suggestions for the improvement of this paper.

REFERENCES

[1] T. Daly, *Publishing Computational Mathematics*, Notices of the AMS, **59** (2012) no. 2, pp. 320–321.

[2] M. Inokuti, H. Sekine, T. Mura, *General use of the Lagrange multiplier in Nonlinear Mathematical Physics*, In Variational Methods in Mechanics and Solids, ed. Nemat-Nasser S., Pergamon Press, pp. 156–162, 1980.

[3] J.H. He, *Variational iteration method - Some recent results and new interpretations*, J. Comput. Appl. Math., **207** (2007), pp. 3–17.

[4] Z.M. Odibat, *A study on the convergence of variational iteration method*, Math. Computer Modelling, **51** (2010), pp. 1181–1192.

[5] D. K. Salkuyeh, *Convergence of the variational iteration method for solving linear systems of ODE with constant coefficients*, Comp. Math. Appl., **56** (2008), pp. 2027–2033.

[6] D. K. Salkuyeh, A. Tavakoli, *Interpolated variational iteration method for initial value problems*, arXiv:1507.01306v1, 2015.

[7] E. Scheiber, *From the numerical solution to the symbolic form,* Bull. Transilvania University of Braşov, Series III, Mathematics, Informatics, Physics, **8**(**57**) (2015) no. 1, pp. 129–137.

[8] M. Tatari, M. Dehghan, *On the convergence of He's variational iteration method.* J. Comput. Appl. Math., **207** (2007), pp. 121–128.

[9] M. Torvattanabun, S. Koonprasert, *Convergence of the variational iteration method for solving a first-order linear system of PDEs with constant coefficients*, Thai J. of Mathematics, Special Issue, pp. 1–13, 2009.

[10] * * *, `www.nutonian.com`

[11] * * *, `www.scilab.org`

Appendix A. *MATHEMATICA* CODE

The *Mathematica* procedure used to solve an initial value problem.

```
In[1]:=
 VIM[f_, U0_, m_] := Module[{V, U = U0, df, Lambda},
  df[t_, x_] := D[f[t, x], x];
  Lambda[U_] := -Exp[
      Integrate[df[w, x] /. {x -> U, t -> w}, {w, s, t}]];
  For[i = 0, i < m, i++,
   V = U +
      Integrate[Lambda[U] ((D[U, t] - f[t, U]) /. t -> s), {s, 0, t}];
   U = V; Clear[V]]; U]
```

For Example 1 the calling sequence is

```
In[2]:= f[t_, x_] := {2 x[[1]] + t}
In[3]:= U0 = {0}
In[4]:= VIM[f,U0,1]
Out[4]={(1/4)*(-1 + E^(2*t) - 2*t)}
```

Appendix B. *SCILAB* code

The code used to obtain numerical results

```
function [t,u,info]=vim(f,df,x0,t0,tf,n,nmi,tol)
// Computes the solution of an initial value problem
// of a system of ordinary differential equations
// using the Variational Iteration Method -
// Dispatcher function.
//
// Calling Sequence
//     [t,u,info]=vim(f,df,x0,t0,tf,n,nmi,tol)
//
// Output arguments
//     t    : a real vector, the times at which the solution is computed.
//     u    : a real vector or matrix, the numerical solution.
//     info: an integer, error code (0 - OK).
// Input arguments
//     f    : a Scilab function, the right hand size of the
//            differential system.
//     df   : a Scilab function, df=(df_1/dx_1,df_2/dx_2,...).
//     x0   : a real vector, the initial conditions.
//     t0   : a real scalar, the initial time.
//     tf   : a real scalar, the final time.
//     n    : a positive integer, the global discretization parameter.
//     nmi  : maximum allowed number of local iterations iteration.
//     tol  : a positive real number, a tolerance.
//
    t=linspace(t0,tf,n)
    d=length(x0)
    u=zeros(d,n)
    u(:,1)=x0'
    m=10   // the discretization parameter on [t_i,t_{i+1}]
    u0=x0
    errorMarker=%t
    for i=1:n-1 do  // the passage from t_0 to t_f
        // Computes u0:=u_{i+1} from u_i
        [u0,eM,iter]=vimstep(f,df,u0,t(i),t(i+1),m,nmi,tol)
        u(:,i+1)=u0'
        errorMarker=errorMarker & eM
    end
    if errorMarker then
        info=0
    else
        info=1
    end
endfunction
```

with

```
function [u,eM,iter]=vimstep(f,df,x0,t0,tf,n,nmi,tol)
// Variational Iteration Method.
//
// Output arguments
//     u     : a real vector or matrix, the numerical solution.
//     eM    : a boolean value, error marker, (%t - OK).
//     iter  : an integer, the number of local performed iterations.
// Input arguments
//     f     : a Scilab function, the right hand size of the
               differential system.
//     df    : a Scilab function, df=(df_1/dx_1,df_2/dx_2,...).
//     x0    : a real vector, the initial conditions.
//     t0    : a real scalar, the initial time.
//     tf    : a real scalar, the final time.
//     n     : a positive integer, the local discretization parameter.
//     nmi   : maximum allowed number of local iterations iteration.
//     tol   : a positive real number, a tolerance.
//
    t=linspace(t0,tf,n)
    h=(tf-t0)/(n-1)
    d=length(x0)
    u_old=zeros(d,n)
    u_old(:,1)=x0'
    sw=%t
    iter=0
    while sw do
        iter=iter+1
        u_new=zeros(d,n)
        u_new(:,1)=x0'
        f0=zeros(d,n)
        df0=zeros(d,n)
        for j=1:n do
            p=u_old(:,j)
            q=f(t(j),p)
            dq=df(t(j),p)
            f0(:,j)=q
            df0(:,j)=dq
        end
        for i=2:n do
            z=zeros(d,n)
            for j=i-1:-1:1 do
                z(:,j)=0.5*h*(df0(:,j)+df0(:,j+1))+z(:,j+1)
            end
            w=(f0-df0.*u_old).*exp(z)
            s=zeros(d,1)
            s=w(:,1)+w(:,i)
            if i>2 then
                for j=2:i-1 do
                    s=s+2*w(:,j)
                end
            end
            u_new(:,i)=0.5*h*s+exp(z(:,1)).*x0'
        end
        nrm=max(abs(u_new-u_old))
        u_old=u_new
```

```
          if nrm<tol | iter>=nmi then
              sw=%f
          end
      end
      u=u_old(:,n)'
      if nrm<tol then
          eM=%t
      else
          eM=%f
      end
endfunction
```

The calling sequence for Example 4 is

```
deff('q=f(t,p)',
     ['x1=p(1)','x2=p(2)','q(1)=x2','q(2)=x1.*x2.^2.0./(x1.^2-1)'])
deff('q=df(t,p)',
     ['x1=p(1)','x2=p(2)','q(1)=0','q(2)=2*x1.*x2./(x1.^2-1)'])
x0=[0.5,0.5*sqrt(3)]
t0=0
tf=%pi/3
n=100
nmi=50
tol=1e-5
[t,u,info]=vim(f,df,x0,t0,tf,n,nmi,tol)
```

Received by the editors: October 5, 2015.