Comparison of semi-analytical methods with solving surge tank problem

Kürşat Canevi[1], Sema Servi[2] and Galip Oturanç[3]

[1]Faculty of Science, Department of Mathematics, Karamanoğlu University, Karaman/Turkey
[2]Faculty of Technology, Department of Computer Engineering, Selcuk University, Konya/Turkey
[3]Faculty of Science, Department of Mathematics, Karamanoğlu University, Karaman/Turkey
E-mail: kursatcanevi@gmail.com[1], semaservi@selcuk.edu.tr[2], goturanc@kmu.edu.tr [3]

Abstract

Hydroelectric power plants are substantial alternative energy source among renewable energy sources. While these power plants are working, sudden pressure changes occur in the water transfer system that reveal a concept called water hammer. In these power plants, a structure called surge tank is usually built to prevent the water hammer from damaging any element in the water transfer system. The surge tank needs to be constructed according to the physical characteristics of the dam. The engineering problem revealed in the construction of the surge tank is expressed by a nonlinear differential equation system. This equation system does not have analytical solution. In this paper, four different numerical methods were used to find the solution of the surge tank problem, which was expressed as an initial value problem. The methods those are used and to be compared are differential transform method, Adomian decomposition method, variational iteration method and Runge-Kutta-Fehlberg method. Calculation algorithms were executed with Maple software. It has been observed that the obtained solutions with all four methods are consistent.

2010 Mathematics Subject Classification. **76-11**. 65L05, 65L06.
Keywords. differential transform method, Adomian decomposition method, variational iteration method, Runge-Kutta-Fehlberg method, surge tank.

1 Introduction

Many problems in mathematics, physics and engineering are expressed with differential equations. Although there are methods that give the analytical solution of some of these equations, it is often encountered with such differential equations that it is not possible to obtain the analytical solution. Numerical methods gain importance due to the fact that the differential equations expressing the problem are not linear or the existing analytical methods are not suitable for reaching solutions with the help of computers.

Since the 1980s, many studies have been carried out to reach numerical solutions of differential equations that cannot be solved analytically. Some of these methods are the Adomian decomposition method introduced by Adomian [2], the differential transformation method introduced by Zhou [6], and the variational iteration method introduced by He [3].

Advanced Studies: Euro-Tbilisi Mathematical Journal 16, supplement issue 4 (2023), pp. 115–131.
DOI: 10.32513/asetmj/1932200823410
Tbilisi Centre for Mathematical Sciences.
Received by the editors: 15 January 2023.
Accepted for publication: 05 June 2023.

1.1 The purpose

Hydroelectric power plants have an important place among alternative energy sources. In the water transfer pipes of the dams of these power plants, the opening and closing maneuvers of the turbine valves cause sudden changes in the flow rate of the water. Pressure fluctuations occur due to this sudden speed change. This phenomenon, called water hammer, can damage the pipe through which the water moves or other elements in the same line as the pipe. In the pipe system that carries water from the water reservoir to the turbines, water hammer can be prevented by a structure called a surge tank, which is placed before the turbine valve.

In this paper, the surge tank problem will be introduced and the solutions of the nonlinear differential equation system expressing this problem will be investigated by semi-analytical methods. Solutions will be obtained by applying the algorithms of the methods used with the codes written on the Maple 2021 package program. The methods to be used in solving the problem are differential transformation method, Adomian decomposition method, variational iteration method and Runge-Kutta-Fehlberg methods [7].

2 The surge tank problem

In this section, the surge tank problem, which will be solved in line with the purpose of the study, is introduced [1].

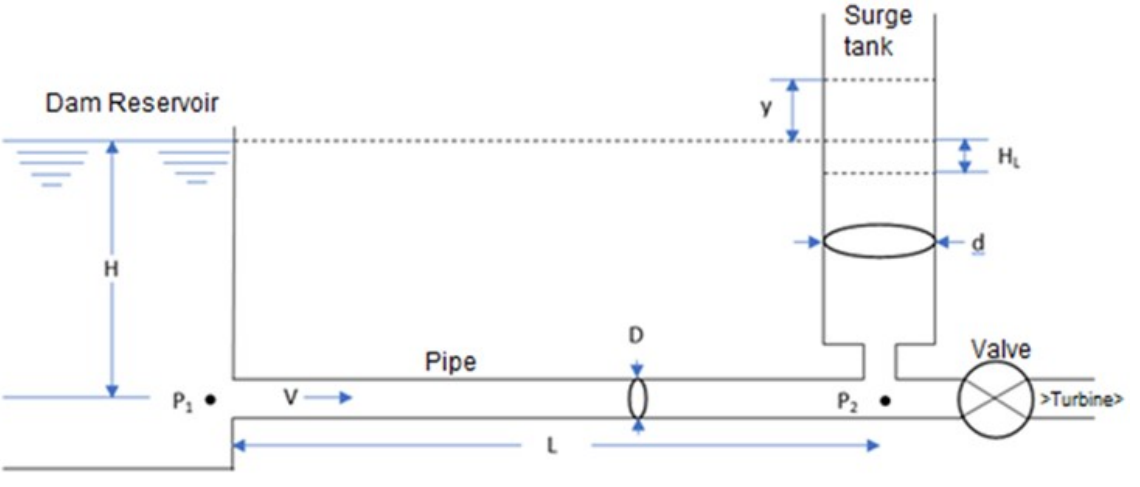


FIGURE 2.1. Powerplant water transfer system

A pipeline that brings water to a hydraulic turbine is shown in Figure 2.1. When the valve open at steady state, the velocity in the pipe is V_0. Due to the fluid friction in the long pipeline, the pressure p_2 is less than the pressure p_1, and the water level in the surge tank does not rise exactly

to the height H. If f is the friction factor and g is the gravitational acceleration, the head loss is expressed with the equation below [14, 15].

$$H_L = f\frac{L}{D}\frac{V_o^2}{2g} \tag{2.1}$$

Here, the propagation problem caused by the sudden closing of the valve at $t = 0$ will be examined.

The water level in the surge tank will move up and down until all excess kinetic energy is lost by fluid friction. The value of y reached in the first wave is of practical importance. The laws of physics on which the up and down movement depends are the continuity of the current and Newton's second law of motion. If the instantaneous level y and velocity dy/dt then for the continuity the equation 2.2 should be satisfied.

$$\frac{\pi d^2}{4}\frac{dy}{dt} = \frac{\pi D^2}{4}V \tag{2.2}$$

V is the instantaneous velocity in the pipe. The slowdown of the fluid in the pipe is due to the $p_2 - p_1$ pressure difference and friction. If ρ is the mass density of the liquid then

$$-\rho L\frac{\pi D^2}{4}\frac{dV}{dt} = (p_2 - p_1)\frac{\pi D^2}{4} + \tau\pi DL \tag{2.3}$$

can be written according to Newton's law. Here τ is the shear stress around the pipe. Because of the slowness of the water hammer, $p_2 - p_1$ can be approximated by the static pressure difference ρgy due to the y difference in hydraulic loads. The shear stress in the pipe wall is $\tau = f\rho V^2/8$. If these expressions are substituted in equation 2.4 we obtain

$$L\frac{dV}{dt} = -gy - f\frac{L}{D}\frac{V^2}{2} \tag{2.4}$$

Equations 2.2 and 2.4 are simultaneous ordinary differential equations for the variables y and V. The propagation problem consists of finding solutions that also satisfy the initial conditions (2.5) at $t = 0$.

$$\begin{aligned} V &= V_0 \\ g &= -f\frac{L}{D}\frac{V_0^2}{2g} \end{aligned} \tag{2.5}$$

A dimensionless formulation can be obtained using the following dimensionless variables:

$$x_1 = \frac{y}{f(L/D)(V_0^2/2g)} \qquad x_2 = \frac{V}{V_0} \qquad t' = \frac{t}{2D/fV_0} \tag{2.6}$$

Thus, the initial conditions 2.5 and the principal equations 2.2, 2.4 become as 2.7.

$$\begin{aligned} t = 0 &\begin{cases} x_1 &= -1 \\ x_2 &= 1 \end{cases} \\ t > 0 &\begin{cases} \frac{dx_1}{dt} &= \theta x_2 \\ \frac{dx_2}{dt} &= -x_1 - x_2^2 \end{cases} \end{aligned} \tag{2.7}$$

Here, the dimensionless parameter is specified as

$$\theta = \frac{g}{L}\left(\frac{2D^2}{fdV_0}\right)^2 \tag{2.8}$$

3 Semi-analytical methods

3.1 One-dimensional differential transform method (DTM)

When this method was introduced for the first time linear and non-linear initial value problems encountered in electrical circuit analysis were investigated [6]. Then, this method was developed by obtaining closed series solution forms for linear and non-linear initial value problems [11]. In the Taylor method, where the symbolic calculations of the derivatives of the given functions are required, the operations for large orders are quite laborious and require a long time. The differential transformation method, on the other hand, reduces the amount of the part to be calculated and can be easily applied to many problems. This solution method involves calculating the coefficients of the series solution using the differential equation and initial value information. The definition of transform and inverse transform functions of this method is given below.

One-dimensional differential transform of function $u(x)$ is defined by 3.1.

$$U(k) = \frac{1}{k!}\left[\frac{d^k}{dx^k}u(x)\right]_{x=0} \tag{3.1}$$

If $U(k)$ is the transform function, then the differential inverse transform function of $U(k)$ is defined by 3.2.

$$u(x) = \sum_{k=0}^{\infty} U(k)x^k \tag{3.2}$$

Combining 3.1 and 3.2, it can be obtained that

$$u(x) = \sum_{k=0}^{\infty} \frac{1}{k!}\left[\frac{d^k}{dx^k}u(x)\right]_{x=0} x^k \tag{3.3}$$

For fundamental operations of one-dimensional differential transform method see references [8, 9, 10, 11, 13]. Some of the operations those may be used with 2.7 are listed in Table 3.1.

TABLE 3.1. The operations for the one-dimensional differential transform method.

Original Function	**Transformation Function**
$w(x) = u(x) \pm v(x)$	$W(k) = U(k) \pm V(k)$
$w(x) = c \cdot u(x) \quad (c \in \mathbb{R})$	$W(k) = c \cdot U(k)$
$w(x) = \frac{d}{dx}u(x)$	$W(k) = (k+1)U(k+1)$
$w(x) = u(x) \cdot v(x)$	$W(k) = \sum_{r=0}^{k} U(r)V(k-r)$

3.2 Adomian decomposition method (ADM)

Equation 3.4 is considered to introduce the method structurally.

$$F[u(t)] = g(t) \tag{3.4}$$

In this equation, $u(t)$ is the unknown function, $g(t)$ is a continuous function, and F is a nonlinear differential operator with linear and non-linear terms. The linear term is decomposed as $L + R$. Equation 3.4 can be given as 3.5, where R is the remainder of the linear operator and L is a higher order and inversible differential operator.

$$Lu + Ru + Nu = g \tag{3.5}$$

Since N is a nonlinear differential operator and L is an inversible operator, the inverse operator L^{-1} can be applied to both sides of 3.5. We get

$$L^{-1}Lu = L^{-1}g - L^{-1}Ru - L^{-1}Nu \tag{3.6}$$

ADM calculates the solution of $u(t)$ as series form that shown as 3.7 and decomposes the nonlinear Nu terms like as 3.8

$$u(t) = \sum_{n=0}^{\infty} u_n \tag{3.7}$$

$$Nu = \sum_{n=0}^{\infty} A_n \tag{3.8}$$

Here, A_n denote the polynomials those called Adomian polynomials and depend on $u_0, u_1, \ldots\ldots, u_n$ terms. u and Nu terms can be obtained as

$$\begin{aligned} u &= \sum_{i=0}^{\infty} \lambda^i u_i \\ N(u) &= N\left(\sum_{i=0}^{\infty} \lambda^i u_i\right) = \sum_{i=0}^{\infty} \lambda^i A_i \end{aligned} \tag{3.9}$$

Here λ is a parameter taken for congruence. A_n polynomials can be found from 3.9 as

$$n! A_n = \frac{d^n}{d\lambda^n}\left[N\left(\sum_{n=0}^{\infty} \lambda^n u_n\right)\right]_{\lambda=0} \tag{3.10}$$

If we write 3.7 and 3.8 expressions at 3.6 we get

$$\sum_{n=0}^{\infty} u_n = \theta + L^{-1}g - L^{-1}R\left(\sum_{n=0}^{\infty} u_n\right) - L^{-1}\left(\sum_{n=0}^{\infty} A_n\right) \tag{3.11}$$

Note that θ=u(0). Terms of $\sum_{n=0}^{\infty} u_n$ series can be written with reduction formulas as

$$\begin{aligned} u_0 &= \theta + L^{-1}g \\ u_1 &= -L^{-1}Ru_0 - L^{-1}A_0 \\ &\vdots \\ u_{n+1} &= -L^{-1}Ru_n - L^{-1}A_n, \quad n \geq 0 \end{aligned} \tag{3.12}$$

Hence, the solution of 3.5 is found in series form. However, in practice it is difficult to calculate all terms of the series $\sum_{n=0}^{\infty} u_n$. So, the approximate solution can be found like

$$\Phi_n = \sum_{i=0}^{\infty} u_n \tag{3.13}$$

or

$$\begin{aligned} \Phi_1 &= u_0 \\ \Phi_2 &= u_0 + u_1 \\ \Phi_3 &= u_0 + u_1 + u_2 \\ &\vdots \\ \Phi_{n+1} &= u_0 + u_1 + u_2 + ... + u_n, \quad n \geq 0 \end{aligned} \tag{3.14}$$

3.3 Variational decomposition method (VIM)

The variational iteration method introduced by He [3] is a successful approach that gives solutions that converge rapidly to analytical solutions. The applications of the method have increased due to the low computer processing power required in the use of the method and the low amount of error in the numerical results obtained. This method was developed and also used to investigate the solutions of integro differential equations [4], Burger and coupled Burger, Schrödinger-KdV, generalized KdV and shallow water equations [12].

In the application of the variational iteration method, the solution sought is handled in the form of differential equation 3.15, where L is the linear operator, N is the non-linear operator, and $g(x)$ is the nonhomogeneous term.

$$Lu + Nu = g(x) \tag{3.15}$$

According to the variational iteration method, the variation function of the equation is formed as 3.16.

$$u_{n+1}(x) = u_n(x) + \int_0^x \lambda\{Lu_n(s) + N\tilde{u}_n(s) - g(s)\}ds \tag{3.16}$$

The λ value here is the Lagrange multiplier [16] and is calculated using some programs such as Mathematica and Maple, based on the variation theory. Expression $\tilde{u}_n$ is limited variation [5]. Also $\delta\tilde{u}_n = 0$. By rearranging the variation function according to the λ multiplier found, the recurrence relation for the function that will give the solution is obtained. By choosing the function

given as the initial condition as u_0, approximations for u_n terms for $n > 0$ are obtained. In the end, the solution function is obtained from 3.17.

$$u = \lim_{n\to\infty} u_n \tag{3.17}$$

The integration factor needed in the application of this method is given in Table 3.2 for some functions [8].

TABLE 3.2. Lagrange multipliers for some equations.

Equation	**Lagrange Multiplier**
$y' = f(t)$	$\lambda(s) = -1$
$y'' = f(t)$	$\lambda(s) = s - t$
$y' + y = f(t)$	$\lambda(s) = -e^{s-t}$
$y' - y = f(t)$	$\lambda(s) = -e^{-s+t}$

3.4 Runge-Kutta-Fehlberg method (RKF45)

In this section, a brief about RKF45 from literature is given [18].

Fehlberg's method is actually a variation of Runge-Kutta(RK) method which was designed to approximate the Taylor series method without requiring explicit evaluations of derivatives except the first one while preserving the accuracy of the Taylor method. Runge-Kutta method sets up a problem with undetermined parameters and uses evaluations of $f(x, y)$ within the interval (x_n, y_n) and (x_{n+1}, y_{n+1}). The general form of the RK is in the form as

$$y_{n+1} = y_n + \sum_{i=1}^{m} w_i \cdot k_i \tag{3.18}$$

where m is number of $f(x, y)$ substitutions, w_i are weighting coefficients and the k_i satisfy the sequence

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + c_2 h, y_n + a_{21} k_1) \\ k_3 &= hf(x_n + c_3 h, y_n + a_{31} k_1 + a_{32} k_2) \\ &\vdots \end{aligned} \tag{3.19}$$

The values of k can be thought of as estimates of the change in y when x changes a value of h. Thus, it now becomes one of determining the coefficients w_i,c_i and a_{ij}. Each set of parameters will specify the points (x, y) where $f(x, y)$ is to be evaluated. Therefore, this method calculates y_{n+1} using only a value for y_n and evaluations of $f(x, y)$ at points between x_n and x_{n+1}. To obtain specific values for the coefficients, a value for m is chosen and y_{n+1} is expanded in powers of h such that it agrees as well as possible with the solution of the ordinary differential equation found using the Taylor series method.

For a given value of y_n , Fehlberg's method computes two estimates of y_{n+1} using fourth and fifth order Runge-Kutta formulas. In order to obtain an estimate of local error, the two values of y_{n+1} are compared. Fehlberg uses

$$y_{n+1} = y_n + \sum_{i=1}^{6} c_i \cdot k_i \tag{3.20}$$

where the k_i satisfy

$$k_i = h_n f\left(x_n + a_i h_n, y_n + \sum_{j=1}^{i-1} \beta_{ij} k_i\right) \quad i = 1, \ldots, 6 \tag{3.21}$$

This method requires six function evaluations per step. The c_i are found by expanding y_{n+1} powers of h_n so that it agrees as well as possible with the Taylor series solution. Fehlberg found that the two expansions match until the h_n^6 term, thus, making the method fifth-order. This is a departure from the behavior of the nth-order Runge-Kutta methods where $n = 1, 2, 3, 4$ which produce $(n + 1)$st order error. This partially explains the popularity of the classical fourth-order method; it takes two more function evaluations to obtain one more order of accuracy. Fehlberg's method, however, exploits the sixth function evaluation by determining a second value y^*_{n+1} using

$$y^*_{n+1} = y_n + \sum_{i=1}^{6} c^*_i \cdot k_i \tag{3.22}$$

This value was found to be fourth-order using the method described above. The local error is then estimated by

$$D_n = \sum_{i=1}^{6} (c_i - c^*_i) k_i \tag{3.23}$$

which is used for stepwise control.

4 Applications and results

In the equation 2.7, If we choose θ=1 then the surge tank problem can be rewritten as equation 4.1.

$$\begin{aligned} t = 0 &\begin{cases} x_1 &= -1 \\ x_2 &= 1 \end{cases} \\ t > 0 &\begin{cases} \frac{dx_1}{dt} &= x_2 \\ \frac{dx_2}{dt} &= -x_1 - x_2^2 \end{cases} \end{aligned} \tag{4.1}$$

4.1 Application of DTM

To solve the system 4.1 with DTM, the Maple codes given with Table 4.1 can be used [17].

TABLE 4.1. Maple program codes to solve the surge tank problem with DTM

```
restart:
y1[0] := -1: y2[0] := 1:
m:=100: #number of steps
for k from 0 to m do y1[k + 1] := y2[k]/(k + 1): y2[k + 1] := (-y1[k]
- sum(y2[r]*y2[k - r], r = 0 .. k))/(k + 1): end do:
s1 := 0: s2 := 0:
for k from 0 to m do s1 := s1 + y1[k]*x^k: s2 := s2 + y2[k]*x^k: end do:
print(s1): print(s2): #Optional codes. If the number of step is high, output may flood the screen.
for q from 0 by 0.05 to 1 do evalf[15](subs(x = q,s1)) end do;
for q from 0 to 1 by .05 do evalf[15]( subs(x=q,s2)) end do;
plot([s1, s2], x = 0 .. 1, color=[red, blue], legend = ["x1 DTM solution", "x2 DTM solution"]);
#Warning: An execution error may occur. Re-type quotation marks( " ) in legend section
after copy-paste
```

The graph of the DTM solution is given with Figure 4.1. All solution values given together with Table 4.5 and Table 4.6 [17].

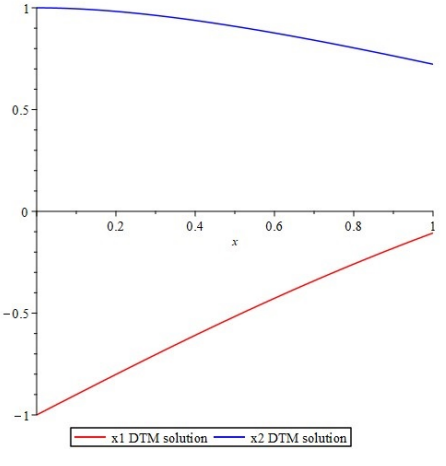


FIGURE 4.1. Solution graph with DTM

4.2 Application of ADM

To solve the system 4.1 with ADM, the Maple codes given with Table 4.2 can be used [17].

TABLE 4.2. Maple program codes to solve the surge tank problem with ADM.

```
restart:
NF := y(t)^2:
m := 6:
y[t] := sum(y[b]*t^b, b = 0 .. m):
NF[y] := subs(y(t) = y[t], NF):
s := expand(NF[y], t):
dt := unapply(s, t):
for i from 0 to m do a[i] := (D@@i)(dt)(0)/i!: print(A[i] = a[i]): end do:
x[0] := -1 + int(1, t = 0 .. t):
y[0] := 1 + int(0, t = 0 .. t):
for k from 0 to m do x[k + 1] := -1 + int(y[k], t = 0 .. t):
y[k + 1] := 1 + int(-y[k]^2 - x[k], t = 0 .. t): end do:
with(plots):
fig1 := plot(x[k], t = 0 .. 1, color = blue, style = line, legend = "x1 ADM solution"):
fig2 := plot(y[k], t = 0 .. 1, color = red, style = line, legend = "x2 ADM solution"):
for q from 0 by 0.05 to 1 do evalf[15](subs(t = q, x[k])): end do;
for q from 0 by 0.05 to 1 do evalf[15](subs(t = q, y[k])): end do;
display({fig1, fig2});
#Warning: An execution error may occur. Re-type quotation marks( " ) in legend section
after copy-paste
```

The graph of the ADM solution is given with Figure 4.2. All solution values given together with Table 4.5 and Table 4.6 [17].

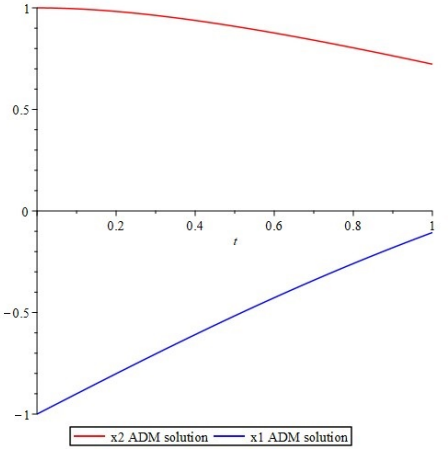


FIGURE 4.2. Solution graph with ADM

4.3 Application of VIM

To solve the system 4.1 with VIM, the Maple codes given with Table 4.3 can be used [17]. Notice that the Lagrange multipliers are chosen as $\lambda_1 = \lambda_2 = -1$.

TABLE 4.3. Maple program codes to solve the surge tank problem with VIM.

```
restart:
x[1][0] := -1: x[2][0] := 1:
m := 7:
for k from 0 to m do
x[1][k + 1] := x[1][k] + int((-1)*(diff(x[1][k], t) - x[2][k]), t = 0 .. t):
x[2][k + 1] := x[2][k] + int((-1)*(diff(x[2][k], t) + x[1][k] + x[2][k]^2), t = 0 .. t):
end do:
with(plots):
fig1 := plot(x[1][k], t = 0 .. 1, color = blue, legend = "x1 VIM sol."):
fig2 := plot(x[2][k], t = 0 .. 1, color = red, legend = "x2 VIM sol."):
display({fig1, fig2});
for q from 0 by 0.05 to 1 do evalf[15](subs(t = q, x[1][k])); end do;
for q from 0 by 0.05 to 1 do evalf[15](subs(t = q, x[2][k])); end do;
#Warning: An execution error may occur. Re-type quotation marks( " ) in legend section
after copy-paste
```

The graph of the VIM solution is given with Figure 4.3. All solution values given together with Table 4.5 and Table 4.6 [17].

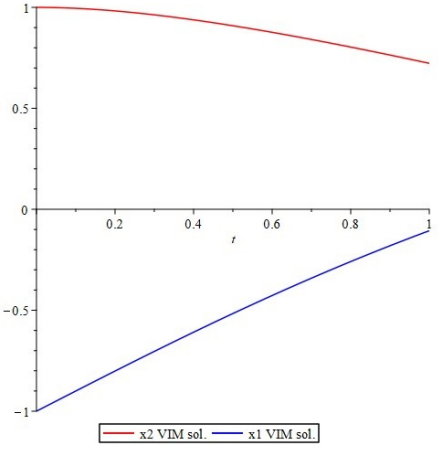


FIGURE 4.3. Solution graph with VIM

4.4 Application of RKF45

To solve the system 4.1 with RKF45, the Maple codes given by Table 4.4 can be used [17]. This is a built-in method in Maple program. An algorithm wasn't developed for calculations.

TABLE 4.4. Table 4.4: Maple program codes to solve the surge tank problem with RKF45.

```
restart:
dsys1 := {diff(x1(t), t) = x2(t), diff(x2(t), t) = -x1(t) - x2(t)^2, x1(0) = -1, x2(0) = 1}:
init1 := {x1(0) = -1, x2(0) = 1}:
dsol1 := dsolve(dsys1 union init1, numeric, method = rkf45, range = 0 .. 1):
with(plots):
for q from 0 by 0.05 to 1 do dsol1(q): end do;
odeplot(dsol1, [[t, x1(t), color = blue], [t, x2(t), color = red]]);
```

The graph of the RKF45 solution is given with Figure 4.4. All solution values given together with Table 4.5 and 4.6.

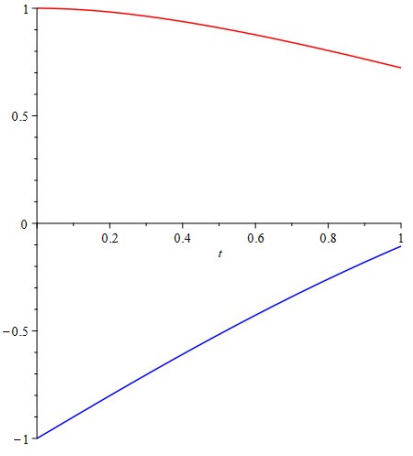


FIGURE 4.4. Solution graph with RKF45

4.5 Results

Approximate solutions of the system 4.1 for $t \in [0, 1]$ with .05 steps are shown in Table 4.5 and Table 4.6.

TABLE 4.5. Approximate solution of $x_1(t)$ with four different methods.

t	$\sim x_1(t)$ **DTM**	$\sim x_1(t)$ **ADM**	$\sim x_1(t)$ **VIM**	$\sim x_1(t)$ **RKF45**
0	-1	-1	-1	-1
.05	-0.950020320348868	-0.950020320348868	-0.950020320348868	-0.950020320488403
.10	-0.900158585241291	-0.900158585241272	-0.900158585241271	-0.900158651137040
.15	-0.850522228223087	-0.850522228222394	-0.850522228222400	-0.850522288138037
.20	-0.801208073887312	-0.801208073878586	-0.801208073878586	-0.801208101084884
.25	-0.752303259403008	-0.752303259341823	-0.752303259341813	-0.752303340819832
.30	-0.703886117410602	-0.703886117116899	-0.703886117116894	-0.703886181579279
.35	-0.656027009420721	-0.656027008339784	-0.656027008339786	-0.656027051147845
.40	-0.608789103276202	-0.608789100013168	-0.608789100013173	-0.608789191450155
.45	-0.562229091595942	-0.562229083162373	-0.562229083162357	-0.562229161038078
.50	-0.516397850598531	-0.516397831396051	-0.516397831396061	-0.516397899747668
.55	-0.471341040474416	-0.471341001207698	-0.471341001207690	-0.471341127533067
.60	-0.427099649686862	-0.427099576652400	-0.427099576652411	-0.427099722630276
.65	-0.383710486362194	-0.383710361896721	-0.383710361896700	-0.383710538687662
.70	-0.341206620383321	-0.341206425649566	-0.341206425649572	-0.341206702983327
.75	-0.299617780013107	-0.299617501717176	-0.299617501717167	-0.299617854959021
.80	-0.258970706912843	-0.258970349928809	-0.258970349928821	-0.258970760418445
.85	-0.219289473338398	-0.219289081504976	-0.219289081504980	-0.219289549757939
.90	-0.180595765133170	-0.180595452609756	-0.180595452609743	-0.180595839518595
.95	-0.142909133922470	-0.142909129383027	-0.142909129383047	-0.142909188026089
1	-0.106247221671221	-0.106247927206731	-0.106247927206749	-0.106247292337712

TABLE 4.6. Approximate solution of $x_2(t)$ with four different methods.

t	$\sim x_2(t)$ **DTM**	$\sim x_2(t)$ **ADM**	$\sim x_2(t)$ **VIM**	$\sim x_2(t)$ **RKF45**
0	1	1	1	1
.05	0.998790881188479	0.998790881188480	0.998790881188470	0.998790881299406
.10	0.995320726949663	0.995320726949650	0.995320726949640	0.995320802179370
.15	0.989811113087334	0.989811113086570	0.989811113086570	0.989811188210914
.20	0.982464897919403	0.982464897909980	0.982464897909990	0.982464917193271
.25	0.973466850749209	0.973466850685320	0.973466850685320	0.973466963037912
.30	0.962984549871985	0.962984549577860	0.962984549577870	0.962984623490116
.35	0.951169449751527	0.951169448723980	0.951169448723980	0.951169466749572
.40	0.938158040180724	0.938158037274130	0.938158037274130	0.938158155044822
.45	0.924073039641673	0.924073032722260	0.924073032722260	0.924073103177451
.50	0.909024580900489	0.909024566739180	0.909024566739150	0.909024583806395
.55	0.893111359476081	0.893111334411080	0.893111334411090	0.893111450830977
.60	0.876421725452623	0.876421687665090	0.876421687665080	0.876421774339811
.65	0.859034706613830	0.859034661164970	0.859034661164970	0.859034692213251
.70	0.841020956490296	0.841020924490630	0.841020924490640	0.841021014907688
.75	0.822443625006561	0.822443658334960	0.822443658334960	0.822443658047400
.80	0.803359152319770	0.803359355093610	0.803359355093610	0.803359122853020
.85	0.783817988427547	0.783818545865580	0.783818545865590	0.783818013002297
.90	0.763865242411313	0.763866456775180	0.763866456775180	0.763865257141036
.95	0.743541265949987	0.743543597881250	0.743543597881250	0.743541226269680
1	0.722882176127569	0.722886287944110	0.722886287944130	0.722882173782271

5 Conclusion

In this study, effectiveness of semi-analytical methods was investigated by solving an engineering problem. DTM, ADM and VIM methods put forward polynomials with more than 100 terms. The number of terms and degrees of the polynomials are shown at Table 5.1.

TABLE 5.1. Number of terms and degrees of the solutions with DTM, ADM and VIM.

Methods	$x_1(t)$		$x_2(t)$	
	#of terms	Degree	#of terms	Degree
DTM	100	100	100	100
ADM	95	96	190	191
VIM	95	96	190	191

Table 4.4 and Table 4.5 shows that solutions were very close to each other. Increasing the number of steps in the trials for the differential transformation method, the Adomian decomposition method, and the variational iteration method also increased the precision in the solutions. However, increasing the number of steps for the Adomian decomposition method and the variational iteration method resulted in a very rapid increase in the number of terms. This caused Maple responding very late and failing graph sketching. For this reason, order preference has been made so that the Adomian decomposition method and the variational iteration method will give the solution with the same number of terms and the program will be able to draw graphs.

All methods offered consistent solutions with each other. In addition, advantages or disadvantages of the methods compared to each other have been observed. For example, the differential transformation method had to be applied in many more steps compared to ADM or VIM solutions in order to give results closer to other methods. However, this method needed less computing performance but more step. Therefore, the use of computers eliminates this negative situation. It has been observed that the application of the variational iteration method on other problems where the Lagrange multiplier is difficult to determine makes it difficult to apply this method and a consistent solution cannot be obtained without the correct Lagrange multiplier. However, it was concluded that this method is easy to understand and program if the Lagrange multiplier can be determined. It was seen that the solutions obtained by increasing the number of steps of the other methods approached the RKF45 solution and it was concluded that the Runge-Kutta-Fehlberg method was the most effective solution.

References

[1] S. Crandall, *Engineering Analysis: A Survey of Numerical Procedures*, New York: McGraw-Hill (1956)

[2] G. Adomian, *A new approach to nonlinear partial differential equations*, Journal of Mathematical Analysis and Applications (1984) 420–434.

[3] J. He, *A new approach to nonlinear partial differential equations*, Communications in Nonlinear Science and Numerical Simulation (1997) 203–205.

[4] S. Wang and J. He, *Variational iteration method for solving integro-differential equations*, Physics Letters A (2007) 188–191.

[5] J. He, *Variational iteration method – a kind of non-linear analytical technique: some examples*, International Journal of Non-Linear Mechanics (1999) 699–708.

[6] J. K. Zhou, *Differential transformation and its applications for electrical circuits*, Wuhan: Huazong University Press (1986).

[7] E. Fehlberg, *Klassische Runge-Kutta-Formeln vierter und niedrigerer Ordnung mit Schrittweiten-Kontrolle und ihre Anwendung auf Wärmeleitungsprobleme*, Computing 6 (1970) 61–71.

[8] Y. Keskin and G. Oturanç, *Diferansiyel Dönüşüm Yöntemiyle Diferansiyel Denklemlerin Çözülmesi*, Konya: Selçuk Üniversitesi Fen Fakültesi Matematik Bölümü (2010).

[9] A. Kurnaz and G. Oturanç, *The differential transform approximation for the system of ordinary differential equations*, International Journal of Computer Mathematics (2005) 709–719.

[10] C. Chen and S. Ho, *Application of Differential transformation to eigenvalue problems*, Applied Mathematics and Computation (1996) 173–188.

[11] C. Chen and S. Ho, *Solving partial differential equations by two-dimensional differential transform method*, Applied Mathematics and Computation (1999) 171–179.

[12] M. Abdou and A. Soliman, *New applications of variational iteration method*, Physica D: Nonlinear Phenomena (2005) 1–8.

[13] A. Arıkoğlu and İ. Özkol, *Solution of boundary value problems for integro-differential equations by using differential transform method*, Applied Mathematics and Computation (2005) 1145–1158.

[14] J. Hunsaker and B. Rightmire, *Engineering Applications of Fluid Mechanics*, New York: McGraw-Hill Book Company Inc. (1947).

[15] G. Rich, *Hydraulic Transients*, New York: McGraw-Hill Book Company Inc. (1951).

[16] M. Inokuti, *General Use of the Lagrange Multiplier in Nonlinear Mathematical Physics.*, N. Nasser, S., Ed., Variational Method in the Mechanics of Solid, Oxford: Pergamon Press (1978) 156–162.

[17] K. Canevi, *Yarı Analitik Yöntemlerin Hidroelektrik Santrali Denge Bacası Probleminin Çözümü ile Kıyaslanması*, Karamanoğlu Mehmetbey University, Turkey (2022).

[18] Colin F. Mayo, *Implementation of the Runge-Kutta-Fehlberg method for solution of ordinary differential equations on a parallel processor*, Monterey, California. Naval Postgraduate School (1987).