Journal of the Nigerian Association of Mathematical Physics
Volume32,(November, 2015), **pp 175 – 188**

Forward-Forward Algorithm with Runge-Kutta Methodfor Control Problems with Initial State and Control Values

Oruh B.I. and Agwu E.U.

Department of Mathematics, Michael Okpara University of Agriculture, Umudike, Abia State, Nigeria.

Abstract

A Forward-Forward algorithm with Runge-Kutta method for solving optimal control problems governed by differential equations is proposed. The control and state variables together with the adjoint variable are considered as initial valued problems. Using the initial values, the optimal control problem reduces to initial valued problem which can be solved by Forward-Forward algorithm with Runge-Kutta method. Computational results, for some standard optimal control problems, show that this new algorithm for optimal control problems substantially outperforms some well-known algorithm for obtaining the numerical approximations for optimal control problems governed by differential equations.

1.0 Introduction

The human quest for highest profit making at lowest cost can only be solved by seeking for the best way to control the performance index. The task to determine the best, perfect and desirable way out of the possible alternatives or variables gives us the optimal control. The theory is formulated as an extension of calculus of variation for the purpose of preventing a system from lapsing into undesirable states. The historical development of optimal control theory cannot be discussed without mentioning the names of great Mathematicians like William Hamilton, Isaac Newton, Johann Bernoulli, Leonhard Euler and Ludovic Lagrange. The concept of optimal control theory improved in 1950s when Lev Pontryagin and his co-worker present the maximum principle, Pontryagin introduced the Hamiltonian function and adjoint function to attach to the differential equation to the objective functional. These functions serve a similar purpose as the Lagrangian function and Lagrangian multipliers respectively in differential calculus. Base on strong background laid by different authors, optimal control theory has developed into a well-established research area and finds its applications in many scientific fields such as biomedical, engineering, management sciences and geometry.

The optimal control problems we intend to solve in this paper is of the form:

Maximizes/minimize $$P = \int_0^T f\left(x(t), u(t), t\right)dt \tag{1.1}$$

subject to

$$\dot{x}(t) = g\left(x(t), u(t), t\right)$$

$$x(0) = x_0, u(0) = u_0, 0 \le t \le T$$

For the solution of optimal control problems, the principal method for the analytic solutions is given by Pontryagin which resolves a set of necessary conditions that an optimal control and the consistent state equation must satisfy. However, the analytic method fails as computational scheme when the problem is non-linear and is not restricted to being quadratic in x and u. As a result, it is necessary to employ numerical methods to solve optimal control problems. Recently, Several Researchers [1, 2, 3] have contributed to the theory of optimal control. In[1], the modified gradient method has been used while the Forward Backward Sweep, the Shooter Method, and an Optimization Method using the MATLAB Optimization Tool Box were compared in [2]. Also, Euler, Trapezoidal and Runge-Kutta using Forward Backward Sweep method (FBSM) were compared in [3]. Looking at the work done in [2] and [3], we find out that both of them fail to compare their numerical approximations

Corresponding author: Agwu E.U, E-mail: agwuemekauchendu@gmail.com, Tel.: +2348064486965

with analytic solutions and were focused only on problems with final value of the adjoint variable. Here, the control and state variables together with the adjoint variable are considered as initial valued problems. Using the initial values, the optimal control problem reduces to initial valued problem which can be solved by Forward-Forward Algorithm with Runge-Kutta method(FFARM) to generate the numerical approximations of both the control, state and adjoint variables instead of using Forward- Backward Sweep method with Runge-Kutta method (FBSM) as done in [2] and [3].

2.0 Methodology

2.1 Derivation of Hamiltonian Equations

Consider the basic optimal control problem of the form

$$J(u) = \int_{t_0}^{T} f(x(t), u(t), t)dt \tag{2.1}$$

subject to

$$\dot{x}_i(t) = g_i(x(t), u(t), t), \; i = 1, 2, ..., n \tag{2.2}$$

where we wish to find the optimal control vector u that minimizes or maximizes equation (2.1).

In (2.1), there are three variables: time t, the state variable x, and the control variable u. We now introduce a new variable, known as adjoint variable and denoted by λ (t). Like the Lagrange multiplier, the adjoint variable is the shadow price of the state variable. The adjoint variable is introduced into the optimal control problem by a Hamiltonian function, $H(t, x, u, \lambda) \equiv f(t, x, u) + \lambda(t)g(t, x, u)$, where H denotes the Hamiltonian and is a function of four variables: t, y, u, and λ.

For the ith constraint equation in (2.2) we form an augmented functional J^* as

$$J^* = \int_0^T \left[f + \sum_{i=1}^{n} \lambda_i (g_i - \dot{x}_i) \right] dt \tag{2.3}$$

Let the integrand be denoted by F

$$F = f + \sum_{i=1}^{n} \lambda_i (g_i - \dot{x}_i) \tag{2.4}$$

The Hamiltonian functional, H is defined as

$$H = f + \sum_{i=1}^{n} \lambda_i g_i \tag{2.5}$$

Hence,

$$J^* = \int_0^T \left[H - \sum_{i=1}^{n} \lambda_i \dot{x}_i \right] dt \tag{2.6}$$

Now the new integrand $F = F(x, u, t)$ becomes

$$F = H - \sum_{i=1}^{n} \lambda_i \dot{x}_i \tag{2.7}$$

We recall Euler-Lagrange equations

$$\frac{\partial F}{\partial x_i} - \frac{d}{dt}\left(\frac{\partial F}{\partial \dot{x}_i} \right) = 0, i = 1, 2, \ldots, n \tag{2.8}$$

$$\frac{\partial F}{\partial u_j} - \frac{d}{dt}\left(\frac{\partial F}{\partial \dot{u}_j} \right) = 0, \; j = 1, 2, \ldots, m \tag{2.9}$$

If we relate equations $(2.4), (2.8), (2.9)$, we have

$$\frac{\partial f}{\partial x_i}+\sum_{i=1}^{n}\lambda_i\frac{\partial g_i}{\partial x_i}+\dot{\lambda}_i=0 \qquad (2.10)$$

$$\frac{\partial f}{\partial u_i}+\sum_{i=1}^{n}\lambda_i\frac{\partial g_i}{\partial u_i}=0 \qquad (2.11)$$

If we relate equations (2.5), (2.10) ,(2.11),

we have

$$-\frac{\partial H}{\partial x_i}=\dot{\lambda}_i, i=1,2,...,n \qquad (2.12)$$

$$\frac{\partial H}{\partial u_i}=0,\ j=1,2,...,m \qquad (2.13)$$

where equation (2.12) is known as adjoint equation [4].

The optimum solutions for x,u,λ can be obtain by equation $(2.2),(2.12),(2.13)$.

We can now state the various components of the maximum principle for problem (2.1) as follows:

$H(t,x,u^*,\lambda)\geq H(t,x,u,\lambda)$ for all t∈ [0, T]

$\frac{\partial H}{\partial u}=0$ (Optimality condition)

$\dot{x}=\frac{\partial H}{\partial \lambda}$ (State equation)

$\dot{\lambda}=-\frac{\partial H}{\partial x}$ (Adjoint equation) (2.14)

$\lambda(T)=0$ (Transversality condition) [5].

Condition one and two in (2.14) state that at every time t the value of u(t), the optimal control, must be chosen so as to maximize the value of the Hamiltonian over all admissible values of u(t) .

Condition three and four of the maximum principle, $\dot{x}=\frac{\partial H}{\partial \lambda} and\ \dot{\lambda}=-\frac{\partial H}{\partial y}$, give us two equations of motion, referred to as the Hamiltonian systems for the given problem.

Condition five, $\lambda(T)=0,$ is the transversality condition appropriate for the free terminal state problem only. Therefore,if u*(t), x*(t) are optimal, then the above conditions hold.

2.2 Procedures to Analytical Solution

i. Form the Hamiltonian for the problem.

ii. Write the adjoint differential equation, transversality boundary condition, and the optimality condition in terms of three unknowns, u^*, x^*, and λ.

iii. Use the optimality equation $\frac{\partial H}{\partial u}=0$ to solve for u^* in terms of x^* and λ.

iv. Solve the two differential equations for x^* and λ with two initial conditions.

v. After finding the optimal state and adjoint, solve for the optimal control using the formula derived by third procedure.

2.3 Forward-Forward Algorithm with Runge-Kutta Method (Ffarm)

The FFARM is one of the iterative methods developed for solving optimal control problems with initial conditions. It begins by using the Maximum Principle to characterize the method as applied to the analytic process. This iterative method is named based on how the algorithm solves the problem of state and adjoint ODEs. Given an initial value of the state and control variables, the initial value of the adjoint variable can also be found using optimality equation. FFARM solves both the state and adjoint forward in time (from 0 to T) simultaneously. Once it has found the state and adjoint functions, the control is updated from the optimality equation. The algorithm starts the process over again using the updated control until it gets to

finial time T and then terminates the process, with the final approximations for the state, adjoint, and control functions considered as the solution to the optimal control problem. The Runge Kutta method of order 4, will be used as a solver to the two differential equations which arise from state and adjoint equations. Though there are many different adaptations of Runge Kutta method, only the method in its classical, fourth order will be used. The fourth order Runge- Kutta method approximates the solution to the problem of these forms

$$\dot{x} = f(t, x, u), \dot{\lambda} = f(t, \lambda, u, x)$$

This method is developed for solving ODE numerically and to avoid computation of derivatives [6]. Since optimal control problems are described by a set of ODE, we shall use this technique to obtain the numerical approximations to optimal control problems.

The algorithm for $\dot{x} = f(t, x, u)$ is given by

$$\begin{aligned} k_1 &= f(t_i, x_i, u_i) \\ k_2 &= f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2}k_1, u_i + \frac{h}{2}\right) \\ k_3 &= f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2}k_2, u_i + \frac{h}{2}\right) \\ k_4 &= f\left(t_i + h, x_i + hk_3, u_i + h\right) \\ x_{i+1} &= x_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), i = 0,1,2,3...N \end{aligned} \tag{2.15}$$

The algorithm for $\dot{\lambda} = f(t, \lambda, x, u)$ is given by

$$\begin{aligned} k_1 &= f(t_i, \lambda_{i,} x_i, u_i) \\ k_2 &= f\left(t_i + \frac{h}{2}, \lambda_i + \frac{h}{2}k_1, u_i + \frac{h}{2}, x_i + \frac{h}{2}\right) \\ k_3 &= f\left(t_i + \frac{h}{2}, \lambda_i + \frac{h}{2}k_2, u_i + \frac{h}{2}, x_i + \frac{h}{2}\right) \\ k_4 &= f\left(t_i + h, \lambda_i + hk_3, u_i + h, x_i + h\right) \\ \lambda_{i+1} &= \lambda_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), i = 0,1,2,3...N \end{aligned} \tag{2.16}$$

$x_{i+1} = x_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), i = 0,1,2,3...N$ is the iterative method for generating the next value for x; it is calculated using the current value of x_i plus the weighted average of four values of $K_j, j = 1,2...4$. Where $K_j, j = 1,2...4$ are functional relations. Note that h is the step size.

Table 1: The difference between the propose Forward-Forward Algorithm with Runge-Kutta Method (FFARM) and Forward-Backward Sweep Method with RK4 (FBSM)

The proposed FFARM for $\dot{x} = f(t, x, u)$ is given by $k_1 = f(t_i, x_i, u_i)$ $k_2 = f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2}k_1, u_i + \frac{h}{2}\right)$ $k_3 = f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2}k_2, u_i + \frac{h}{2}\right)$ $k_4 = f\left(t_i + h, x_i + hk_3, u_i + h\right)$ $x_{i+1} = x_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$ $i = 0,1,2,3...N$	The FBSM algorithm with RK4 for $\dot{x} = f(t, x, u)$ as done in [2] is given by $k_1 = f(t_i, x_i, u_i)$ $k_2 = f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2}k_1, \frac{1}{2}\left(u_i + u_{i+1}\right)\right)$ $k_3 = f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2}k_2, \frac{1}{2}\left(u_i + u_{i+1}\right)\right)$ $k_4 = f\left(t_i + h, x_i + hk_3, u_{i+1}\right)$ $x_{i+1} = x_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$ $i = 0,1,2,3...N$
The proposed FFARM for $\dot{\lambda} = f(t, \lambda, x, u)$ is given by $k_1 = f(t_i, \lambda_{i,} x_i, u_i)$ $k_2 = f\left(t_i + \frac{h}{2}, \lambda_i + \frac{h}{2}k_1, u_i + \frac{h}{2}, x_i + \frac{h}{2}\right)$ $k_3 = f\left(t_i + \frac{h}{2}, \lambda_i + \frac{h}{2}k_2, u_i + \frac{h}{2}, x_i + \frac{h}{2}\right)$ $k_4 = f\left(t_i + h, \lambda_i + hk_3, u_i + h, x_i + h\right)$ $\lambda_{i+1} = \lambda_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$ $i = 0,1,2,3...N$	The FBSM algorithm with RK4 for $\dot{\lambda} = f(t, \lambda, x, u)$ as done in [2] is given by $k_1 = f(t_j, \lambda_{j,} x_j, u_j)$ $k_2 = f\left(t_j - \frac{h}{2}, \lambda_j - \frac{h}{2}k_1, \frac{1}{2}\left(u_j + u_{j-1}\right), \frac{1}{2}\left(x_j + x_{j-1}\right)\right)$ $k_3 = f\left(t_j - \frac{h}{2}, \lambda_j - \frac{h}{2}k_2, \frac{1}{2}\left(u_j + u_{j-1}\right), \frac{1}{2}\left(x_j + x_{j-1}\right)\right)$ $k_4 = f\left(t_j - h, \lambda_j - hk_3, u_{j-1}, x_{j-1}\right)$ $\lambda_{j-1} = \lambda_j - \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$ $j = N + 2 - i$

3.0 Results

Here we will first take linear optimal control problem as test problem, which we will solve by analytical approach then obtain the numerical approximation by FFARM using Runge-Kutta schemes and executed by R - software.

Problem 1: Consider the problem

Minimizes $\int_0^1 \left(x^2 + u^2\right)dt$ (3.1)

Subject to

$$\dot{x}(t) = -x + u$$

$$x(0) = 1, u(0) = 0.5, 0 \le t \le 1$$

Taking the Hamiltonian equation, we have

$$H = f + \lambda g = u^2 + x^2 - \lambda(x - u) \tag{3.2}$$

Taking the necessary condition for optimality, we have

$$\lambda - 2x = \dot{\lambda} \tag{3.3}$$

$$2u + \lambda = 0 \tag{3.4}$$

If we differentiate equation (3.4), we have

$$-2\dot{u} = \dot{\lambda}$$

$$\Rightarrow -2\dot{u} = \lambda - 2x$$

$$\Rightarrow -2\dot{u} = -2u - 2x \tag{3.5}$$

But, $\dot{x} = -x + u \Rightarrow \ddot{x} = -\dot{x} + \dot{u}$, equation (3.5) becomes

$$\ddot{x} - 2x = 0$$

Therefore the 2nd order ODE with initial conditions gives

$$x(t) = 0.3232e^{\sqrt{2}t} + 0.6768e^{-\sqrt{2}t}$$

$$u(t) = 0.7803e^{\sqrt{2}t} - 0.2803e^{-\sqrt{2}t}$$

From (3.1), we have

$$u_i = -\frac{\lambda_i}{2}$$

$$\dot{x} = -x + u$$

$$\dot{\lambda} = \lambda - 2x, x_0 = 1, u_0 = 0.5 \Rightarrow \lambda_0 = -1$$

Now applying the FFARM, we have the iterative formula for x to be

$$k_1 = f(t_i, x_i, u_i) = -x_i + u_i$$

$$k_2 = f\left(t_i + \frac{h}{2}, u_i + \frac{h}{2}, x_i + \frac{h}{2}k_1\right) = \left(u_i + \frac{h}{2} - x_i + \frac{hx_i}{2} - \frac{hu_i}{2}\right)$$

$$k_3 = f\left(t_i + \frac{h}{2}, u_i + \frac{h}{2}, x_i + \frac{h}{2}k_2\right) = \left(u_i + \frac{h}{2} - x_i - \frac{hu_i}{2} - \frac{h^2}{4} + \frac{hx_i}{2} - \frac{h^2x_i}{4} + \frac{h^2u_i}{4}\right)$$

$$k_4 = f\left(t_i + h, u_i + h, x_i + hk_3\right) = \left(u_i + h - x_i - hu_i - \frac{h^2}{2} + hx_i + \frac{h^2u_i}{2} + \frac{h^3}{4} - \frac{h^2x_i}{2} + \frac{h^3x_i}{4} - \frac{h^3u_i}{4}\right)$$

$$x_{i+1} = x_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) = x_i + \frac{h}{6}\left(-6x_i + 6u_i + 3hx_i - 3hu_i - h^2x_i + h^2u_i + \frac{h^3x_i}{4} - \frac{h^3u_i}{4} + 3h - h^2 + \frac{h^3}{4}\right)$$

$$i = 0, 1, 2, 3 \ldots N$$

The iterative formula for λ is

$$k_1 = f(t_i, \lambda_i, x_i, u_i) = -2x_i + \lambda_i$$

$$k_2 = f\left(t_i + \frac{h}{2}, \lambda_i + \frac{h}{2}k_1, u_i + \frac{h}{2}, x_i + \frac{h}{2}\right) = \left(-2x_i - h + \lambda_i - hx_i + \frac{h\lambda_i}{2}\right)$$

$$k_3 = f\left(t_i + \frac{h}{2}, \lambda_i + \frac{h}{2}k_2, u_i + \frac{h}{2}, x_i + \frac{h}{2}\right) = \left(-2x_i - h + \lambda_i - hx_i - \frac{h^2}{2} + \frac{h\lambda_i}{2} - \frac{h^2x_i}{2} + \frac{h^2\lambda_i}{4}\right)$$

$$k_4 = f\left(t_i + h, \lambda_i + hk_3, u_i + h, x_i + h\right) = \left(-2x_i - 2h + \lambda_i - 2hx_i - h^2 + h\lambda_i - h^2x_i - \frac{h^3}{2} + \frac{h^2\lambda_i}{2} - \frac{h^3x_i}{2} + \frac{h^3\lambda_i}{4}\right)$$

$$\lambda_{i+1} = \lambda_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) = \lambda_i + \frac{h}{6}\left(-12x_i + 6\lambda_i - 6hx_i + 3h\lambda_i - 2h^2x_i + h^2\lambda_i - \frac{h^3x_i}{2} + \frac{h^3\lambda_i}{4} - 6h - 2h^2 - \frac{h^3}{2}\right)$$

$$i = 0, 1, 2, 3 \ldots N$$

Table 2: The numerical results to problem 1 at h = 0.05

S/N.	Time	u	x	λ	u .exact	x .exact
1	0	0.5	1	-1	0.49227	1
2	0.05	0.578178	0.976844	-1.15636	0.569106	0.977477
3	0.1	0.659177	0.95863	-1.31835	0.648788	0.959844
4	0.15	0.743394	0.945255	-1.48679	0.731716	0.947011
5	0.2	0.831244	0.93664	-1.66249	0.818303	0.938916
6	0.25	0.923157	0.932729	-1.84631	0.908984	0.935516
7	0.3	1.019581	0.933492	-2.03916	1.004212	0.936797
8	0.35	1.120988	0.93892	-2.24198	1.104462	0.942763
9	0.4	1.227873	0.949029	-2.45575	1.210238	0.953445
10	0.45	1.340757	0.963858	-2.68151	1.322066	0.968896
11	0.5	1.460188	0.983469	-2.92038	1.440508	0.989193
12	0.55	1.586748	1.007948	-3.1735	1.566155	1.014439
13	0.6	1.721052	1.037406	-3.4421	1.699636	1.044758
14	0.65	1.863752	1.071977	-3.7275	1.841618	1.080304
15	0.7	2.015541	1.111822	-4.03108	1.992813	1.121253
16	0.75	2.177155	1.157126	-4.35431	2.153975	1.167811
17	0.8	2.349379	1.208103	-4.69876	2.325911	1.22021
18	0.85	2.533046	1.264993	-5.06609	2.509482	1.278712
19	0.9	2.729047	1.328066	-5.45809	2.705605	1.343611
20	0.95	2.93833	1.397622	-5.87666	2.915261	1.41523
21	1	3.16191	1.473993	-6.32382	3.1395	1.493928

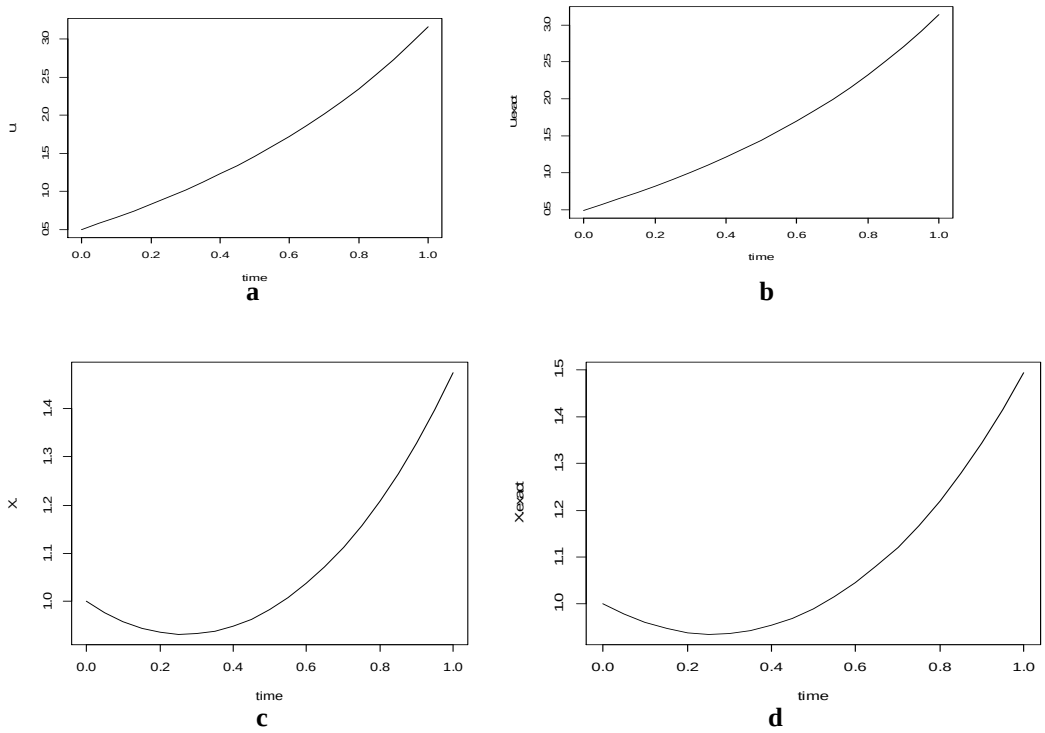


Fig. 1: Optimal state and control values for problem 1 at h = 0.05

Note. a representsthe graph of u_{FFARM} against time, b represents the graph of u_{exact} against time , c represents the graph of x_{FFARM} against time and d represents the graph of x_{exact} against time

CODE USED FOR PROBLEM 1

```
function(u.old,x.old,lambda.old,h,itra){
#h is d step size
#u.old is d initial guess for u
#x.old is d initial guess for x
#lambda.old is d initial guess for lambda
uby=c(u.old)
xby=c(x.old)
lambdaby=c(lambda.old)
time=seq(0,by=h,length=itra+1)
itra=itra+1
U.exact=0.7803*(exp(1.4142*time))-(0.28803*(exp(-1.4142*time)))
X.exact=0.3232*(exp(1.4142*time))+(0.6768*(exp(-1.4142*time)))
if (itra==1)return(c(u.old,x.old,lambda.old,U.exact,X.exact))
else
for ( i in 2:itra){
xby[i]=xby[i-1]  +  ((h/6)*((-6*xby[i-1])+(6*uby[i-1])+(3*h*xby[i-1])-(3*h*uby[i-1])-((h^2)*xby[i-1])+((h^2)*uby[i-1])+(((h^3)/4)*xby[i-1])-(((h^3)/4)*uby[i-1])+(3*h)-(h^2)+((h^3)/4)))
lambdaby[i]=lambdaby[i-1+((h/6)*((-12*xby[i-1])+(6*lambdaby[i-1])-(6*h*xby[i-1])+(3*h*lambdaby[i-1])-(2*(h^2)*xby[i-1])+((h^2)*lambdaby[i-1])-(((h^3)/2)*xby[i-1])+(((h^3)/4)*lambdaby[i-1])-(6*h)-(2*(h^2))-((h^3)/2)))
uby[i]=((-(lambdaby[i]))/2)  }
fgh<- data.frame(time,U.=uby,X.=xby,Lambda=lambdaby,U.exact,X.exact)
return(fgh)
}
```

Problem 2: **THE REAL LIFE PROBLEM (NON-LINER PROBLEM)**

A certain community in Ohafia L.G.A where $x(t)$ represents the number of people infected by malaria parasite (*Plasmodium*) surfers the problem of controlling the rate of reduction $u(t)$ of people infected over a period of one year. At the beginning, only two people were infected and the reduction rate was half of the number of people infected. At time goes on, the instantaneous rate of people infected is directly proportional to reduction rate and the total number of people infected in the community is $x^4 + u^4$. At what rate of reduction will minimize the total number of people infected in the community over the stated period.

Solution: from the problem stated above, the performance index is given by

Minimizes $\int_0^1 \left(x^4(t) + u^4(t) \right) dt$ (3.6)

Subject to

$$\dot{x}(t) \alpha u(t) \Rightarrow \dot{x}(t) = ku(t)$$

$$x(0) = 1, u(0) = 0.5, 0 \le t \le 1$$

Let the constant of proportionality be 1, then taking the Hamiltonian equation, we have

$$H = f + \lambda g = u^4 + x^4 + \lambda u \quad (3.7)$$

Taking the necessary conditions for optimality, we have

$$\dot{\lambda} = -4x^3 \quad (3.8)$$

$$4u^3 + \lambda = 0 \quad (3.9)$$

If we differentiate equation (3.9), we have

$$-12u^2 \dot{u} = \dot{\lambda}$$

$$\Rightarrow -12u^2 \dot{u} = -4x^3 \quad (3.10)$$

But, $\dot{x} = u \Rightarrow \ddot{x} = \dot{u}$, equation (3.10) becomes

$$3\dot{x}^2 \ddot{x} = x^3$$

Therefore, problem 2 has no analytic solution. We will now seek for numerical approximation to problem2

From (3.6), we have

$$u_i = \sqrt[3]{\frac{-\lambda_i}{4}}$$

$$\dot{x} = u$$

$$\dot{\lambda} = -4x^3, x_0 = 1, u_0 = 0.5 \Rightarrow \lambda_0 = -0.5$$

Now applying the FFARM, we have the iterative formula for x to be

$$k_1 = f(t_i, x_i, u_i) = u_i$$

$$k_2 = f\left(t_i + \frac{h}{2}, u_i + \frac{h}{2}, x_i + \frac{h}{2}k_1\right) = (u_i + \frac{h}{2})$$

$$k_3 = f\left(t_i + \frac{h}{2}, u_i + \frac{h}{2}, x_i + \frac{h}{2}k_2\right) = (u_i + \frac{h}{2})$$

$$k_4 = f\left(t_i + h, u_i + h, x_i + hk_3\right) = u_i + h$$

$$x_{i+1} = x_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) = x_i + \frac{h}{6}\left(6u_i + 3h\right)$$

$$i = 0,1,2,3...N$$

The iterative formula for λ is

$$k_1 = f(t_i, \lambda_{i,} x_i, u_i) = -4x_i^3$$

$$k_2 = f\left(t_i + \frac{h}{2}, \lambda_i + \frac{h}{2}k_1, u_i + \frac{h}{2}, x_i + \frac{h}{2}\right) = -4\left(x_i + \frac{h}{2}\right)^3$$

$$k_3 = f\left(t_i + \frac{h}{2}, \lambda_i + \frac{h}{2}k_2, u_i + \frac{h}{2}, x_i + \frac{h}{2}\right) = -4\left(x_i + \frac{h}{2}\right)^3$$

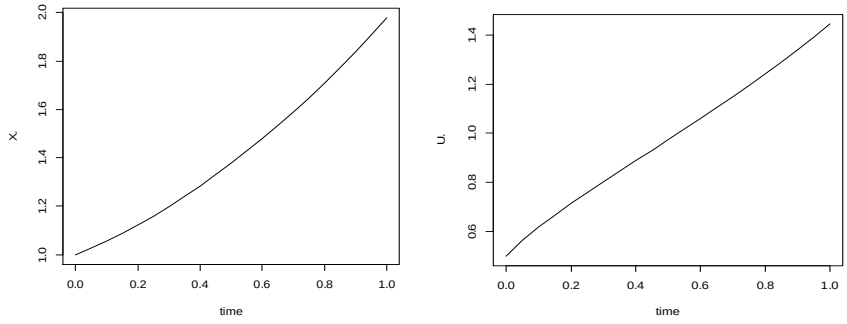
$$k_4 = f\left(t_i + h, \lambda_i + hk_3, u_i + h, x_i + h\right) = -4\left(x_i + h\right)^3$$

$$\lambda_{i+1} = \lambda_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) = \lambda_i + \frac{h}{6}\left(-4x_i^3 - 16\left(x_i + \frac{h}{2}\right)^3 - 4\left(x_i + h\right)^3\right)$$

$$i = 0,1,2,3...N$$

Table 3: The numerical results to problem 2 at h=0.05

S/N	time	u.	x.	λ.
1	0	0.5	1	-0.5
2	0.05	0.563445	1.02625	-0.71551
3	0.1	0.618448	1.055672	-0.94617
4	0.15	0.668444	1.087845	-1.19469
5	0.2	0.715279	1.122517	-1.46382
6	0.25	0.760077	1.159531	-1.75644
7	0.3	0.803588	1.198785	-2.07568
8	0.35	0.846342	1.240214	-2.42492
9	0.4	0.888738	1.283781	-2.8079
10	0.45	0.931084	1.329468	-3.22869
11	0.5	0.97363	1.377272	-3.69183
12	0.55	1.016582	1.427204	-4.2023
13	0.6	1.060119	1.479283	-4.76566
14	0.65	1.104394	1.533539	-5.38806
15	0.7	1.149546	1.590008	-6.07631
16	0.75	1.195701	1.648736	-6.83799
17	0.8	1.242975	1.709771	-7.68152
18	0.85	1.291475	1.77317	-8.61625
19	0.9	1.341306	1.838993	-9.65259
20	0.95	1.392567	1.907309	-10.8021
21	1	1.445352	1.978187	-12.0776



a b

Fig. 2: Optimal state and control values for problem 2 at h = 0.05

Note a represents the graph of x_{FFARM} against time and b represents the graph of u_{FFARM} against time

> CODE USED FOR PROBLEM 2

```
function(u.old,x.old,lambda.old,h,itra,k){
#h is d step size
#u.old is d initial guess for u
#x.old is d initial guess for x
#lambda.old is d initial guess for lambda
uby=c(u.old)
xby=c(x.old)
lambdaby=c(lambda.old)
time=seq(0,by=h,length=itra+1)
itra=itra+1
U.exact=lambda.old/sqrt(1-(lambda.old)^2)
X.exact=time + k
if (itra==1)return(c(u.old,x.old,lambda.old,U.exact,X.exact))
else
for ( i in 2:itra){
xby[i]=xby[i-1] + (h/6)*((6*uby[i-1]) + (3*h))
lambdaby[i]=lambdaby[i-1]+(h/6)*(-4*xby[i-1]-16*(xby[i-1]+(h/2))^(3)-4*(xby[i-1]+h)^3)
uby[i]=(-lambdaby[i]/4)^(1/3)
        }
fgh<- data.frame(time,U.=uby,X.=xby,Lambda=lambdaby,U.exact,X.exact)
return(fgh)
}
```

[7].

Note that the next problem has been done in [3] using FBSM, we wish to solve this problem by FFARM, so as to compare the two methods.

PROBLEM 3: maximizes $$\int_0^2 \left(x - \frac{1}{2}u^2 \right) dt + 2x(2) \tag{3.11}$$

subject to

$$\dot{x}(t) = -x + u$$

$$x(0) = 2, u(0) = 1.1342, 0 \le t \le 2$$

Taking the Hamiltonian equation, we have

$$H = f + \lambda g = x - \frac{1}{2}u^2 - \lambda(x - u) \tag{3.12}$$

Taking the necessary conditions for optimality, we have

$$\dot{\lambda} = -1 + \lambda \tag{3.13}$$

$$-u + \lambda = 0 \tag{3.14}$$

If we differentiate equation (3.14), we have

$$\dot{u} = \dot{\lambda}$$

$$\Rightarrow \dot{u} = \lambda - 1 = u - 1 \tag{3.15}$$

But, $\dot{x} = -x + u \Rightarrow \ddot{x} = -\dot{x} + \dot{u}$, equation (3.15) becomes

$$\ddot{x} - x = -1$$

Therefore the 2nd order ODE with initial conditions gives

$$x(t) = 0.0671e^{t} + 0.9329e^{-t} + 1$$

$$u(t) = 0.1342e^{t} + 1$$

From (3.11), we have

$$u_i = \lambda_i$$

$$\dot{x} = -x + u$$

$$\dot{\lambda} = -1 + \lambda, x_0 = 2, u_0 = 1.1342 \Rightarrow \lambda_0 = 1.1342$$

Now applying the FFARM, we have the iterative formula for λ to be

$$k_1 = f(t_i, \lambda_i, x_i, u_i) = -1 + \lambda_i$$

$$k_2 = f\left(t_i + \frac{h}{2}, \lambda_i + \frac{h}{2}k_1, u_i + \frac{h}{2}, x_i + \frac{h}{2}\right) = (\lambda_i - 1)(1 + \frac{h}{2})$$

$$k_3 = f\left(t_i + \frac{h}{2}, \lambda_i + \frac{h}{2}k_2, u_i + \frac{h}{2}, x_i + \frac{h}{2}\right) = -1 + \lambda_i + \frac{h}{2}(\lambda_i - 1)(1 + \frac{h}{2})$$

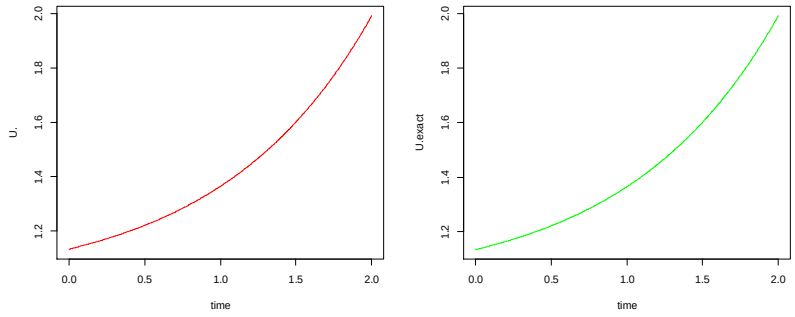
$$k_4 = f\left(t_i + h, \lambda_i + hk_3, u_i + h, x_i + h\right) = -1 + \lambda + h\left(-1 + \lambda_i + \frac{h}{2}(\lambda_i - 1)(1 + \frac{h}{2})\right)$$

$$\lambda_{i+1} = \lambda_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) = \lambda_i + \frac{h}{6}\left(-6.006004002 + 6.006004002\lambda_i\right)$$

$$i = 0, 1, 2, 3 \ldots N$$

Table 4: The numerical results to problem 3 at h=0.002

S/N	Selected valve of Time	u . FFA With RK M	u . FBSM With RK in [3]	u .exact
1	0.0000	1.1324	1.1324	1.1324
2	0.008	1.1353	1.1353	1.1353
3	0.0380	1.1394	1.1395	1.1394
4	0.0980	1.1480	1.1481	1.1480
5	0.1260	1.1522	1.1524	1.1522
6	0.3080	1.1826	1.1830	1.1826
7	0.8160	1.3035	1.3048	1.3035
8	1.0200	1.3722	1.3740	1.3722
9	1.2040	1.4473	1.4497	1.4473
10	1.8220	1.8299	1.8351	1.8299
11	1.9500	1.9432	1.9493	1.9432
12	2.0000	1.9916	1.9880	1.9916



a b

Fig. 3: Optimal control values for problem 3 at h = 0.002

Note. a represents the graph of u_{FFARM} against time and b represents the graph of u_{exact} against time

4.0 Discussion

In this paper, FFARM as a powerful tool for approximating the solutions to optimal control problems is presented. Some numerical examples have solved to show the efficiency of the proposed algorithm.

The first test problem considered has its exact optimal trajectory and control functions as

$$x(t) = 0.3232e^{\sqrt{2}t} + 0.6768e^{-\sqrt{2}t}$$

$$u(t) = 0.7803e^{\sqrt{2}t} - 0.2803e^{-\sqrt{2}t}$$

respectively. The computed results of applying the proposed algorithm have been shown in Table 2. Also the obtained approximate optimal control and state trajectory which has been compared to the exact ones can be seen in Fig. 1.

For the test problem 2, the analytic solution is not obvious. Numerical approximation to the problem is shown in Table 3.

The numerical approximation to the solution to test problem 3 is indicated in Table 4, while Fig. 3 shows the graphical representation. The exact optimal trajectory and control functions are

$$x(t) = 0.0671e^{t} + 0.9329e^{-t} + 1$$

$$u(t) = 0.1342e^{t} + 1$$

In all the examples, we see that the proposed numerical scheme is simple compared with the FBSM and the obtained results show that the approximate solutions are near to exact solutions. Therefore, we conclude that FFARM is simple and gives error that is negligible.

5.0 References

[1] Ejieji, C.N, *A modified conjugate gradient method for solving discrete optimal control problems*: *PhD thesis*, University Ilorin, Ilorin, Nigeria, 2005.

[2] Garret, R. R, *Numerical methods for solving optimal control problems***:** *M.sc thesis*, The University of Tennessee, Knoxville, 2015.

[3] Saleem, R, Habib, M. and Manaf, A, *"Review of forward backward Sweep method for bounded and unbounded control problem with payoff term"*, Science Int. Lahore0, 27(1). 69-72. 2014

[4] Singiresu, S. R, *Engineering optimization: theory and practices*, New age international limited, West Lafayette, Indiana, 2008.

[5] Eugene, S. and Wing, S, *The structure of Economics*, Irwin McGraw–Hill, New York, 2000.

[6] Francis, S., *Numerical Analysis*, Irwin McGraw–Hill, New York, 1968

[7] R Core Team. R: *A language and environment for statistical computing. R Foundation for Statistical Computing,*Vienna, Austria, 2015. URL https://www.R-project.org/.