Asian Journal of Scientific Research

ISSN 1992-1454

OPEN ACCESS Asian Journal of Scientific Research

ISSN 1992-1454
DOI: 10.3923/ajsr.2018.553.559

Research Article
Softcode of Multi-Processing Milne's Device for Estimating First-Order Ordinary Differential Equations

[1]Jimevwo Godwin Oghonyon, [2]Olaide Adetola Adesanya, [1]Hudson Akewe and [1]Hilary Izuchukwu Okagbue

[1]Department of Mathematics, Covenant University, Ota, Ogun State, Nigeria
[2]Department of Mathematics, Modibbo Adama University of Technology, Yola, Nigeria

Abstract
Background and Objectives: Softcodes is a form of Mathematica language invented for the successful implementation of MPMD. Technical computing is an aspect of computing for the sole purpose of computation leading to better accuracy. This paper considers softcode of multi-processing Milne's device for estimating first-order Ordinary Differential Equations (ODEs). **Materials and Methods:** Multi-Processing Milne's Device (MPMD) is source from Adams collection of predicting-correcting scheme implemented via interpolation and collocation adopting multinomial finite sequence near resolution. This combination is mathematically assembled in MPMD pattern and analyzed to produce the order of the MPMD thereby setting up the chief local truncation errors. **Results:** The computational results generated were aided with Softcodes in Mathematica data format and setting the bounds of convergency. **Conclusion:** The calculated results are compared with subsisting methods to enhance the viability and effectiveness of the MPMD over others.

Key words: Softcode, MPMD, bounds of convergency, multi-processing, chief local truncation errors

Received: April 10, 2018 Accepted: July 26, 2018 Published: September 15, 2018

Citation: Jimevwo Godwin Oghonyon, Olaide Adetola Adesanya, Hudson Akewe and Hilary Izuchukwu Okagbue, 2018. Softcode of multi-processing Milne's device for estimating first-order ordinary differential equations. Asian J. Sci. Res., 11: 553-559.

Corresponding Author: Jimevwo Godwin, Oghonyon, Department of Mathematics, College of Science and Technology, Covenant University, P.M.B. 1023, Ota, Ogun State, Nigeria Tel: +234-8139724200

Competing Interest: The authors have declared that no competing interest exists.

Data Availability: All relevant data are within the paper and its supporting information files.

INTRODUCTION

Softcode for providing approximate results to Ordinary Differential Equations (ODEs) are very essential in technical computing, since it is greatly utilized to prototype real life applications[1-4]. Multi-processing Milne's device for estimating first-order differential equation is of the form Abell and Braselton[1], Ken *et al.*[3], Bakoji *et al.*[5] and Adejumo *et al.*[6]:

$$v' = g(t, z),\ z(u_0) = \alpha \tag{1}$$

Arising from Eq. 1, there is a need to look for numerical solution enclosed on $u\in[c, d]$ such that c and d are bounded with the assumption that z meets the considerations as seen in Akinfenwa *et al.*[7], Anake *et al.*[8], Anake and Adoghe[9], Jain *et al.*[10], Lambert[11,12], Sunday *et al.*[13] and Xie and Tian[14]. Thus, ensures that Eq. 1 possess a specific differential coefficient at every point.

The universal multi-processing Milne's device is instituted as:

$$\sum_{i=0}^{j}\alpha_i^{'} g_{m+i}^{'} = h\sum_{i=0}^{j}\beta_i^{'} z_{m+i}^{'} \tag{2}$$

where, $\alpha_i^{'}$ and $\beta_i^{'}$ are invariables implying that $\alpha_i^{'} \neq 0$, $\alpha_i^{'} + \beta_i^{'} \neq 0$ Adesanya *et al.*[15].

According to Lambert[11], Dormand[16] and Faires and Burden[17], multi-processing Milne's device is seen as an alternative to multi-processing predicting-correcting scheme on the account of the numerical vantages it features over others. Generators such as Akinfenwa *et al.*[2], Bakoji *et al.*[5], Anake *et al.*[8], Anake and Adoghe[9], Adesanya *et al.*[15], Majid and Suleiman[18] and Oghonyon *et al.*[19-21] suggested multi-processing predictor-corrector scheme implemented on first-order ODEs. Multi-processing predicting-correcting scheme derives shortcomings during computation/execution and as such, unable to find a suitable length, resolve bounds of convergency and lack of error maximization.

The motivation of this research study is founded on the concept of generating certain qualities of the multi-processing Milne's device which are comparable to BDF for implementing stiff ODEs and vibration problem as discussed in Anake *et al.*[8], Anake and Adoghe[9], Jain *et al.*[10], Sunday *et al.*[13], Faires and Burden[17], Oghonyon *et al.*[19-21], Ascher and Petzold [22], Ngwane and Jator[23-25] and Ibrahim *et al.*[26]. Again, softcodes of Mathematica codes is projected for implementation[1,3].

The main aim of this research work is to develop softcodes of multi-processing Milne's device for computing first-order ODEs. Furthermore, this originality has been established on various body of literatures as cited Lambert[11,12], Dormand[16], Faires and Burden[17], Oghonyon *et al.*[19-21] and Ascher and Petzold[22] for more particulars. This includes some elements like; Adams type, multi-processing predicting-correcting scheme of the like range and chief local truncation errors as remarked above.

MATERIALS AND METHODS

Softcode for multi-processing Milne's device is a collection of multi-processing predicting-correcting scheme of Adams type. This requires Adams-Bashforth-Adams-Moulton (multi-processing predicting-correcting of ilk range) scheme. This involves u-length multi-processing predicting scheme and u-1-length multi-processing correcting scheme of ilk range. This compendium is established as:

$$g(t) = \sum_{i=0}^{j}\alpha_i g_{m-i} + h_1\sum_{i=0}^{j}\beta_i z_{m-i} \tag{3}$$

$$g(t) = \sum_{i=0}^{j}\alpha_i g_{m-i} + h_1\sum_{i=0}^{j}\beta_i z_{m+i} \tag{4}$$

Equation 3 and 4 determines the multi-processing predicting-correcting scheme of multi-processing Milne's device. Remarking $g(t_{m+i})\approx g_{m+i}$, $g(x_{m+i}, g_{m+i})\approx z_{m+i}$ having j = 0, 1, 2. To attain Eq. 3 and 4, the approximative function is penned below to evaluate the analytical resolution g(t) on clear-cut time intervals of $[t_n, t_{n-j}]$ by way of interpolation of the form:

$$g(t) = \sum_{i=0}^{j} x_i\left(\frac{t-t_n}{h_1}\right)^i \tag{5}$$

Revising Eq. 5 in softcode format produces the softcode approximate function as:

$$g[t\_] = x[0] + x[1]\frac{(t-t[n])}{h_1} + x[2] + x[3]\frac{(t-t[n])^3}{h_1^3} \tag{6}$$

where, x_0, x_1, x_2 and x_3 are parameters required to be settle in a special manner. Presuming that Eq. 6 corresponds with the precise result at approximately selected definite length of time interval t_n, t_{n-j} to yield approximation as:

$$g(t_n)\approx g_n,\ g(t_{n-j})\approx g_{n-j} \tag{7}$$

Taking that the approximating function (Eq. 6) gratifies (Eq. 1) at more or less chosen points t_{n+j}, j = 0, 1, 2 to obtain the following approximates as:

$$g'(t_{n+j}) \approx z_{n+j}, \quad j = 0, 1, 2 \tag{8}$$

Merging Eq. 7 and 8 will generate quadruplet formations which produces At = b:

$$\begin{aligned} &\text{matrixa} = \left\{ \begin{matrix} \{1,-1,1,-1\}, \\ \{0,1,0,0\}, \\ \{0,1,-2,3\}, \\ \{0,1,-4,12\}, \end{matrix} \right\}; \\ &b = \{g[n],\ z[n-1], z[n-2], z[n-3]\}; \\ &\{j,k,l,q\} = \text{Inverse}[\text{matrixa}].b \end{aligned} \tag{9}$$

$$\begin{aligned} &\text{matrixa} = \left\{ \begin{matrix} \{1,-1,1,-1\}, \\ \{0,1,2,3\}, \\ \{0,1,4,12\}, \\ \{0,1,6,27\} \end{matrix} \right\}; \\ &b = \{g[n-1],\ z[n+1], z[n+2], z[n+3]\}; \\ &\{j,k,l,q\} = \text{Inverse}[\text{matrixa}].b \end{aligned} \tag{10}$$

Figuring out the systems of equation applying Mathematica 9 kernel, softcodes gives x_j, j = 0, 1, 2, 3 and putting back values of x_j's into Eq. 6 will generates the uninterrupted multi-processing prediction scheme and multi-processing correcting scheme of Milne's device as:

$$\begin{aligned} g[t\_] = (1)g[n-1] + &\left(\frac{5}{12} + \frac{(t-t[n])^1}{h} + \frac{3(t-t[n])^2}{4h^2} + \frac{(t-t[n])^3}{6h^3}\right) \\ f[n]h + &\left(\frac{2}{3} - \frac{(t-t[n])^2}{h^2} - \frac{(t-t[n])^3}{3h^3}\right) f[n-1]h + \\ &\left(\frac{-1}{12} + \frac{3(t-t[n])^2}{4h^2} + \frac{(t-t[n])^3}{6h^3}\right) f[n\text{-}2]h \end{aligned} \tag{11}$$

$$\begin{aligned} g[t\_] = (1)g[n-1] + &\left(\frac{53}{12} + \frac{3(t-t[n])^1}{h} - \frac{5(t-t[n])^2}{4h^2} + \frac{(t-t[n])^3}{6h^3}\right) \\ f[n+1]h + &\left(-\frac{16}{3} - \frac{3(t-t[n])^1}{h} + \frac{2(t-t[n])^2}{h^2} - \frac{(t-t[n])^3}{3h^3}\right) f[n+2] + \\ &\left(\frac{23}{12} + \frac{(t-t[n])^1}{h} - \frac{3(t-t[n])^2}{4h^2} + \frac{(t-t[n])^3}{6h^3}\right) f[n+3] \end{aligned} \tag{12}$$

Assessing the uninterrupted multi-processing prediction scheme and multi-processing correcting scheme of Milne's device at some favourable grids, t_{n+j}, j = 1, 2, 3 will originate the multi-processing prediction Milne's device and multi-processing correcting Milne's device as:

$$\begin{aligned} g(t) &= g_{n-1} + h_1(\mu_1 z_i + \mu_2 z_{i-1} + \mu_3 z_{i-2}), \\ g(t) &= g_{n-1} + h_1(\beta_1 z_{i+1} + \beta_2 z_{i+2} + \beta_3 z_{i+3}) \end{aligned} \tag{13}$$

where, β_1, β_2, β_3, μ_1, μ_2 and μ_3 are parametric quantity[1,4,11,12,16,17,23-25,27] for more details.

Devising bounds of convergence for multi-processing Milne's device: To set in motion numeric operation of multi-processing Milne's device, the r-length multi-processing predicting scheme and r-1-length multi-processing correcting scheme are put to use as multi-processing predicting-correcting scheme owns alike range Locate[11,12,16,17,19-22] for more. Uniting Lambert[11,12], Dormand[16], Faires and Burden[17], Oghonyon *et al.*[19-21] and Ascher and Petzold[22], it is workable to find approximative chief local truncation error of multi-processing predicting-correcting scheme in absentia of higher order differential coefficients, g(t). What is more, $p_1 = c_1$ where, p_1 and c_1 represents range of multi-processing predicting and correcting schemes. Straightaway, scheme of range p_1, taking apart multi-processing predicting r-length gives rise to the chief principal local truncation errors:

$$\begin{aligned} P^{[1]}_{p_1+4} h_1^{p_1+4} g^{(p_1+4)}(t_n) &= g(t_{n+1}) - g^{[q_1]}_{n+1} + O\left(h_1^{p_1+5}\right), \\ P^{[1]}_{p_2+4} h_2^{p_2+4} g^{(p_2+4)}(t_n) &= g(t_{n+2}) - g^{[q_2]}_{n+2} + O\left(h_2^{p_2+5}\right) \\ P^{[3]}_{p_3+4} h_3^{p_3+4} g^{(p_3+4)}(t_n) &= g(t_{n+3}) - g^{[q_3]}_{n+3} + O\left(h_3^{p_3+5}\right) \end{aligned} \tag{14}$$

Likewise, looking into multi-processing correcting scheme r-1-step brings forth chief local truncation errors as:

$$\begin{aligned} C^{[1]}_{c_1+4} h_1^{c_1+4} g^{(c_1+4)}(t_n) &= g(t_{n+1}) - g^{[s_1]}_{n+1} + O\left(h_1^{c_1+5}\right), \\ C^{[2]}_{c_2+4} h_2^{c_2+4} g^{(c_2+4)}(t_n) &= g(t_{n+2}) - g^{[s_2]}_{n+2} + O\left(h_2^{c_2+5}\right), \\ C^{[3]}_{c_3+4} h_3^{c_3+4} g^{(c_3+4)}(t_n) &= g(t_{n+3}) - g^{[s_3]}_{n+3} + O\left(h_3^{c_3+5}\right), \end{aligned} \tag{15}$$

where, $P^{[1]}_{p_1+4}$, $P^{[2]}_{p_2+4}$, $P^{[3]}_{p_3+4} C^{[1]}_{c_1+4}$, $C^{[2]}_{c_2+4}$ and $C^{[3]}_{c_3+4}$ continues as classified quantity of length h_1 and g(t) behave as analytic resolution to higher derived function conforming to the initial stipulation $g(t_n) \approx g_n$. Look into Lambert[11,12], Dormand[16], Faires and Burden[17], Oghonyon *et al.*[19-21] and Ascher and Petzold[22] more items.

Further advancement for less precondition measures of length h_1 is reached $g^{(4)}(t_n) \approx g^{(4)}(t_n)$ and the potency of multi-processing Milne's device trusts instantly on this presumption stated over.

Reducing in advance the chief the principal local truncation errors of Eq. 14 and 15 over besides dismissing considerations of range $O(h^{p_1+5})$. Thus, introduces no concern achieving the numerical formulation of chief local truncation errors of the multi-processing Milne's device:

$$\begin{aligned} C^{[1]}_{p_1+4} h^{p_1+4} g^{(p_1+4)}(t_n) &\approx \frac{C^{[1]}_{p_1+4}}{P^{[1]}_{p_1+4} - C^{[1]}_{p_1+4}} \left[g^{[q_1]}_{n+1} - g^{[s_1]}_{n+1} \right] < \tau_1, \\ C^{[2]}_{p_2+4} h^{p_2+4} g^{(p_2+4)}(t_n) &\approx \frac{C^{[2]}_{p_2+4}}{P^{[2]}_{p_2+4} - C^{[2]}_{p_2+4}} \left[g^{[q_2]}_{n+2} - g^{[s_2]}_{n+2} \right] < \tau_2, \\ C^{[3]}_{p_3+4} h^{p_3+4} g^{(p_3+4)}(t_n) &\approx \frac{C^{[3]}_{p_3+4}}{P^{[3]}_{p_3+4} - C^{[3]}_{p_3+4}} \left[g^{[q_3]}_{n+3} - g^{[s_3]}_{n+3} \right] < \tau_3 \end{aligned} \tag{16}$$

Referring the avouchment that $g^{[q_1]}_{n+1} \neq g^{[s_1]}_{n+1}$, $g^{[q_2]}_{n+2} \neq g^{[s_2]}_{n+2}$ and $g^{[q_3]}_{n+j} \neq g^{[s_3]}_{n+j}$ are named predicting and correcting estimations founded thru multi-processing Milne's device of order p_1, even though $C^{[1]}_{p_1+4} h^{p_1+4} g^{(p_1+4)}(t_n)$, $C^{[2]}_{p_2+4} h^{p_2+4} g^{(p_2+4)}(t_n)$ and $C^{[3]}_{p_3+4} h^{p_3+4} g^{(p_3+4)}(t_n)$ are each separately called chief local truncation errors. τ_1, τ_2 and τ_3 are bounds of convergency of the multi-processing Milne's device.

Advancing forward, these approximates of the chief local truncation error (Eq. 16) is utilized to make decision on acceptance or rejection thereby iterating with less or smaller varying length. The length is sustain free-based on a try out laid down by Eq. 16 [11,12,16,17,19-22] for more details. The chief local truncation errors (Eq. 16) is the bounds of convergence of the multi-processing Milne's, device denoted differently as multi-processing Milne's device for adjusting to convergence.

Numerical problems: Two problems tested are worked with MPMD. The bounds of convergency considered includes; 10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}, 10^{-11} and 10^{-14}. Find Sunday *et al.*[13], Rufai *et al.*[28] and Sunday *et al.*[29] for more actions. A computer programming codes on MPMD is written utilizing Mathematica 9 kernel. The act of accomplishment is carried out in a multi-processing manner via MPMD (Appendix).

Test problem 1: Consider the nonlinear IVP, g'(t) = -10(g(t)-1)^2, g(0) = 2.

Analytical result: $g(t) = \frac{(2+10t)}{(1+10t)}$.

Test problem 2: Consider Prothero-Robinson periodic vibration ODE, g'(t) = L(g(t)-sin(t))+cos (t), L = -1, g(0) = 0.

Analytical result: g(t) = sin (t).

RESULTS AND DISCUSSION

Under this section, the computational output shows the execution of MPMD for solving first-order ODEs. The final output supplied were obtained with the aid of Mathematica 9 Kernel 64 on Microsoft windows (64 bit) to demonstrate the efficiency and accuracy of the first-order ODEs[13,28,29].

Table 1 demonstrates the numerical results of problems 1 and 2 using MPMD equated with existing methods.

Table 1 presents a summary of the result displayed and items considered. This includes; method utilized, computed max errors and bounds of convergency. Again, shows the comparison with other existing results and justifies MPMD as a preferable proficiency in terms of the computed max errors:

Table 1: Summary of results

$M_{utilized}$	Max_{errors}	B_{cov}
ERR	3.296387e-004	10^{-4}
ERR	2.983380e-004	
ERR	2.819223e-004	
MPMD	1.30546e-004	10^{-4}
MPMD	1.31522e-004	
MPMD	1.32503e-004	
ERS	6.017101e-006	10^{-6}
ERS	5.411308e-006	
ERS	4.880978e-006	
HBM	2.840882e-006	10^{-6}
HBM	2.717126e-006	
HBM	2.588157e-006	
MPMD	1.04804e-006	10^{-6}
MPMD	1.0511e-006	
MPMD	1.06653e-006	
ERR	1.429167e-008	10^{-8}
ERR	1.283029e-008	
ERR	1.159479e-008	
MPMD	9.59886e-009	10^{-8}
MPMD	9.70295e-009	
MPMD	9.80786e-009	
ERA	2.0e-010	10^{-10}
ERA	3.0e-010	
ERA	3.0e-010	
MPMD	1.02131e-010	10^{-10}
MPMD	1.06689e-010	
MPMD	1.31986e-010	
ERS	1.803588e-011	
MPMD	1.04299e-011	10^{-11}
MPMD	1.10816e-011	
MPMD	1.17654e-011	
ERR	7.155093e-014	10^{-14}
ERR	5.921081e-014	
ERR	8.457038e-014	
MPMD	1.70636e-014	10^{-14}
MPMD	1.78416e-014	
MPMD	3.29997e-014	

- The signifiers mentioned on Table 1 are stated below:

MPMD: Computed max errors in MPMD (multi-processing Milne's device) for time-tested problems 1 and 2
$M_{utilized}$: Method utilized
Max_{errors}: Magnitude of computed max errors in MPMD
B_{cov}: Bounds of convergency
1/6 HBM: Computed max errors in one-sixth HBM (1/6 hybrid block method) for time-tested problem 1 [28]
ERR (BI): Computed max errors in ERR (BI) (block integrator) for time-tested tested problem 1 and 2 [13]
ERR: Computed max errors in ERR (quarter-step method of 10^{-4}) for tested problem 1 [29]

Softcodes algorithm rule: A well written algorithmic rule that will execute MMD and assess the computed max errors of MPMD in the family of $P(EC)^j$ or $P(EC)^j$ E style, conditionally, when the style is implemented as many times to ensure convergence. Check out[23]:

Step 1: Take length for h
Step 2: The MPMD of predicting-correcting scheme must have alike range
Step 3: The length of predicting must have higher length than correcting scheme
Step 4: Estimate the chief local truncation errors of the MPMD only when CLTE is reached
Step 5: Fix the bounds of convergency
Step 6: Generate the softcodes of MPMD utilizing Mathematica 9 kernel
Step 7: Adopt single step technique to kick start the procedure if necessary, otherwise avoid 7 and go to step 8
Step 8: Perform the MPMD in the family of $P(EC)^j$ or $P(EC)^j$ E style as j increases
Step 9: When step 8 did not attain convergency, ingeminate the process once again and half the length (h) from step 1 or otherwise, go on to step 10
Step 10: Calculate the computed max errors when convergency is fulfilled
Step 11: Publish computed max errors
Step 12: Use this equation below to devise a new length only when convergency is attained

$$rh = \left|\frac{\tau_1}{2\left(P^{[1]}_{p_1+4} - C^{[1]}_{p_1+4}\right)}\right|^{\frac{10}{40}}$$

CONCLUSION

The computed results displayed MPMD is reached utilizing the bounds of convergency. This bounds of convergency examine the acceptance or rejection of the looping with a smaller length. The mathematical outputs establish the performance of MPMD is remarked to showcase a more acceptable computed max error at all bounds of convergency. This is made possible by seeking a suitable/changing length, determining the bounds of convergency as compare to subsisting schemes implemented without these features. This proficiency for a better result is executed at all examined bounds of convergency such as 10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}, 10^{-11} and 10^{-14}. Thence, it will be concluded that MPMD is worthy for estimating ODEs. Furthermore, MPMD is better and preferred to schemes such as block predictor-corrector methods, block implicit method, block hybrid method because their applications are based on fixed step size, no bounds of convergency and always implemented in predictor-corrector method. Continuous research can be carried out to increase the order of MPMD for examining performance.

SIGNIFICANT STATEMENT

The significant of this study is as follows:

- A new basis function approximation is designed in form of Softcodes for yielding interpolation and collocation estimates
- The scientific community will benefit by using Softcodes in Mathematica format, encrypted for the successful implementation of MPMD
- The accuracy of MPMD is validated on nonlinear IVP and vibration problem
- MPMD advances the utilization of the chief local truncation error outside showing the order
- The MPMD is considered as an option to Backward Differentiation Formula (BDF) on account of some similar advantages it possesses

ACKNOWLEDGMENT

The authors would like to appreciate Covenant University for financing this research work.

Appendix: The softcodes of problem 1 and 2 implemented via MPMD is given below

Softcodes of multi-processing predictor method

```
clear[g];
soln=DSolve[{g'[t]==-10(g[t]-1)^2,g[0]==2},g[t],t];
Simplify[soln]
g[t_]=(2+10t)/(1+10t)
h=given values
x[n]=given values
t=given values
g[1]=g[0]+h(g'[0])+(h^2/2)g''[0]+(h^3/6)g'''[0]+(h^4/24)g''''[0]
g[2]=g[1]+h(g'[x[n]])+(h^2/2)g''[x[n]]+(h^3/6)g'''[x[n]]+(h^4/24)g''''[x[n]]
g[3]=g[2]+h(g'[x[n]+h])+(h^2/2)g''[x[n]+h]+(h^3/6)g'''[x[n]+h]+(h^4/24)g''''[x
[n]+h]
g[4]=g[3]+h(g'[x[n]+2h])+(h^2/2)g''[x[n]+2h]+(h^3/6)g'''[x[n]+2h]+(h^4/24)g
''''[x[n]+2h]

t=x[n]+h
g[3]=g[1]+h((1/3)g'[t-x[n]-h]-(2/3)g'[t-x[n]]+(7/3)g'[t])
t=x[n]+3h
g[5]=g[2]+h((9/4)g'[t-x[n]-h]-(6)g'[t-x[n]]+(27/4)g'[t])
t=x[n]+5h
g[7]=g[3]+h((20/3)g'[t-x[n]-h]-(52/3)g'[t-x[n]]+(44/3)g'[t])

t=x[n]+4h
g[6]=g[4]+h((1/3)g'[t-x[n]-h]-(2/3)g'[t-x[n]]+(7/3)g'[t])
t=x[n]+6h
g[8]=g[5]+h((9/4)g'[t-x[n]-h]-(6)g'[t-x[n]]+(27/4)g'[t])
t=x[n]+8h
g[10]=g[6]+h((20/3)g'[t-x[n]-h]-(52/3)g'[t-x[n]]+(44/3)g'[t])

t=x[n]+7h
g[9]=g[7]+h((1/3)g'[t-x[n]-h]-(2/3)g'[t-x[n]]+(7/3)g'[t])
t=x[n]+9h
g[11]=g[8]+h((9/4)g'[t-x[n]-h]-(6)g'[t-x[n]]+(27/4)g'[t])
t=x[n]+11h
g[13]=g[9]+h((20/3)g'[t-x[n]-h]-(52/3)g'[t-x[n]]+(44/3)g'[t])

t=x[n]+10h
g[12]=g[10]+h((1/3)g'[t-x[n]-h]-(2/3)g'[t-x[n]]+(7/3)g'[t])
t=x[n]+12h
g[14]=g[11]+h((9/4)g'[t-x[n]-h]-(6)g'[t-x[n]]+(27/4)g'[t])
t=x[n]+14h
g[16]=g[12]+h((20/3)g'[t-x[n]-h]-(52/3)g'[t-x[n]]+(44/3)g'[t])
clear[y];
soln=DSolve[{y'[u]==-10(y[u]-1)^2,y[0]==2},y[u],u];
Simplify[soln]
```

Softcodes of multi-processing corrector method

```
y[u_]=(2+10u)/(1+10u)
h=given value
x[n]=given value
u=given value
y[1]=y[0]+h(y'[0])+(h^2/2)y''[0]+(h^3/6)y'''[0]+(h^4/24)y''''[0]
y[2]=y[1]+h(y'[x[n]])+(h^2/2)y''[x[n]]+(h^3/6)y'''[x[n]]+(h^4/24)y''''[x[n]]
y[3]=y[2]+h(y'[x[n]+h])+(h^2/2)y''[x[n]+h]+(h^3/6)y'''[x[n]+h]+(h^4/24)y''''[x[
n]+h]
y[4]=y[3]+h(y'[x[n]+2h])+(h^2/2)y''[x[n]+2h]+(h^3/6)y'''[x[n]+2h]+(h^4/24)y'''
'[x[n]+2h]

u=x[n]+h
y[3]=y[1]+h((19/3)y'[u+x[n]]-(20/3)y'[u+x[n]+h]+(7/3)
y'[u+x[n]+2h])
u=x[n]+3h
y[5]=y[2]+h((27/4)y'[u+x[n]]-(6)y'[u+x[n]+h]+(9/4)
y'[u+x[n]+2h])
u=x[n]+5h
y[7]=y[3]+h((20/3)y'[u+x[n]]-(16/3)y'[u+x[n]+h]+(8/3)
y'[u+x[n]+2h])

u=x[n]+4h
y[6]=y[4]+h((19/3)y'[u+x[n]]-(20/3)y'[u+x[n]+h]+(7/3)
y'[u+x[n]+2h])
u=x[n]+6h
y[8]=y[5]+h((27/4)y'[u+x[n]]-(6)y'[u+x[n]+h]+(9/4)
y'[u+x[n]+2h])
u=x[n]+8h
y[10]=y[6]+h((20/3)y'[u+x[n]]-(16/3)y'[u+x[n]+h]+(8/3)
y'[u+x[n]+2h])

u=x[n]+7h
y[9]=y[7]+h((19/3)y'[u+x[n]]-(20/3)y'[u+x[n]+h]+(7/3)
y'[u+x[n]+2h])
u=x[n]+9h
y[11]=y[8]+h((27/4)y'[u+x[n]]-(6)y'[u+x[n]+h]+(9/4)
y'[u+x[n]+2h])
u=x[n]+11h
y[13]=y[9]+h((20/3)y'[u+x[n]]-(16/3)y'[u+x[n]+h]+(8/3)
y'[u+x[n]+2h])

u=x[n]+10h
y[12]=y[10]+h((19/3)y'[u+x[n]]-(20/3)y'[u+x[n]+h]+(7/3)
y'[u+x[n]+2h])
u=x[n]+12h
y[14]=y[11]+h((27/4)y'[u+x[n]]-(6)y'[u+x[n]+h]+(9/4)
y'[u+x[n]+2h])
u=x[n]+14h
y[16]=y[12]+h((20/3)y'[u+x[n]]-(16/3)y'[u+x[n]+h]+(8/3)
y'[u+x[n]+2h])
```

REFERENCES

1. Abell, M.L. and J.P. Braselton, 2009. Mathematica By Example. 4th Edn., Elsevier, USA., ISBN: 9780080921693, Pages: 576.
2. Akinfenwa, O., S. Jator and N. Yoa, 2011. An eighth order backward differentiation formula with continuous coefficients for stiff ordinary differential equations. Int. J. Math. Comput. Sci., 7: 160-165.
3. Ken, Y.L., I.F. Ismail and M. Suleiman, 2011. Block Methods for Special Second Order ODEs. Lambert Academic Publishing, Germany, ISBN-13: 978-3844384130, Pages: 196.
4. Ngwane, F.F. and S.N. Jator, 2013. Solving oscillatory problems using a block hybrid trigonometrically fitted method with two off-step points. Electron. J. Differ. Equations, 20: 119-132.
5. Bakoji, A.M., A.M. Bukar and M.I. Bello, 2014. Formulation of predictor-corrector methods from 2-step hybrid Adams methods for the solution of initial value problems of ordinary differential equations. Int. J. Eng. Applied Sci., 5: 9-13.

6. Adejumo, G., M.T. Abioye and T.A. Anake, 2014. Adoption of computer assisted language learning software among Nigerian secondary school. Proceedings of the International Conference on Education and New Learning Technologies, July 7-9, 2014, Barcelona, Spain, pp: 950-955.
7. Akinfenwa, O.A., S.N. Jator and N.M. Yao, 2013. Continuous block backward differentiation formula for solving stiff ordinary differential equations. Comput. Math. Applic., 65: 996-1005.
8. Anake, T.A., D.O. Awoyemi and A.O. Adesanya, 2012. One-step implicit hybrid block method for the direct solution of general second order ordinary differential equations. IAENG Int. J. Applied Math., 42: 224-228.
9. Anake, T.A. and L.O. Adoghe, 2013. A four point block integration method for the solutions of IVP in ODE. Aust. J. Basic Applied Sci., 7: 467-473.
10. Jain, M.K., S.R.K. Iyengar and R.K. Jain, 2007. Numerical Methods for Scientific and Engineering Computation. 5th Edn., New Age International (Pvt.) Ltd., New Delhi, India, ISBN-13: 978-8122420012, Pages: 816.
11. Lambert, J.D., 1973. Computational Methods in Ordinary Differential Equations. John Wiley and Sons, New York, USA., ISBN: 9780471511946, pp: 87-88.
12. Lambert, J.D., 1991. Numerical Methods for Ordinary Differential Systems: The Initial Value Problem. 1st Edn., John Wiley and Sons, New York, USA., ISBN: 9780471929901, pp: 103-105.
13. Sunday, J., M.R. Odekunle, A.A. James and A.O. Adesanya, 2014. Numerical solution of stiff and oscillatory differential equations using a block integrator. Br. J. Math. Comput. Sci., 4: 2471-2481.
14. Xie, L. and H. Tian, 2014. Continuous parallel block methods and their applications. Applied Math. Comput., 241: 356-370.
15. Adesanya, A.O., A.A. James and J. Sunday, 2014. Hybrid block predictor-hybrid block corrector for the solution of first-order ordinary differential equations. Eng. Math. Lett., 13: 1-12.
16. Dormand, J.R., 1996. Numerical Methods for Differential Equations: A Computational Approach. CRC Press, Boca Raton, FL., ISBN: 9780849394331, pp: 10, 142-143, 231.
17. Faires, J.D. and R.L. Burden, 2012. Initial-value problems for ODEs, variable step-size multistep methods. Dublin City University, Dublin, Republic of Ireland, pp: 2-32.
18. Majid, Z.A. and M. Suleiman, 2011. Predictor-corrector block iteration method for solving ordinary differential equations. Sains Malays., 40: 659-664.
19. Oghonyon, J.G., S.A. Okunuga and O.O. Agboola, 2015. K-step block predictor-corrector methods for solving first order ordinary differential equations. Res. J. Applied Sci., 10: 779-785.
20. Oghonyon, J.G., S.A. Okunuga and S.A. Iyase, 2016. Milne's implementation on block predictor-corrector methods. J. Applied Sci., 16: 236-241.
21. Oghonyon, J.G., S.A. Okunuga and S.A. Bishop, 2017. A variable-step-size block predictor-corrector method for ordinary differential equations. Asian J. Applied Sci., 10: 96-101.
22. Ascher, U.M. and L.R. Petzold, 1998. Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations. SIAM, Philadelphia, USA., ISBN-13: 9780898714128, Pages: 314.
23. Ngwane, F.F. and S.N. Jator, 2014. Trigonometrically-fitted second derivative method for oscillatory problems. SpringerPlus, Vol. 3. 10.1186/2193-1801-3-304
24. Ngwane, F.F. and S.N. Jator, 2015. Solving the telegraph and oscillatory differential equations by a block hybrid trigonometrically fitted algorithm. Int. J. Differ. Equations, Vol. 2015. 10.1155/2015/347864
25. Ngwane, F.F. and S.N. Jator, 2017. A Trigonometrically fitted block method for solving oscillatory second-order initial value problems and hamiltonian systems. Int. J. Differ. Equations, Vol. 2017. 10.1155/2017/9293530
26. Ibrahim, Z.B., K. Othman and M. Suleiman, 2007. Variable step block backward differentiation formula for solving first-order stiff ODEs. Proceedings of the World Congress on Engineering, July 2-4, 2007, London, UK., pp: 2-6.
27. Ngwane, F.F. and S.N. Jator, 2013. Block hybrid method using trigonometric basis for initial value problems with oscillating solutions. Numer. Algorithm, 63: 713-725.
28. Rufai, M.A., M.K. Duromola and A.A. Ganiyu, 2016. Derivation of one-sixth hybrid block method for solving general first order ordinary differential equations. IOSR J. Math., 12: 20-27.
29. Sunday, J., D. Yusuf and J.N. Andest, 2016. Integration of first-order modeled differential equations using a quarter-step method. Adv. Res., 7: 1-8.