International Journal of Latest Engineering and Management Research (IJLEMR)
ISSN: 2455-4847
www.ijlemr.com || Volume 06 - Issue 09 || September 2021 || PP. 01-13

A Survey on Implementations to a Class of Basic Linear Multistep Methods

Dinh Van Tiep[1],
[1](*Faculty of International Training, Thai Nguyen University of Technology, Vietnam*)

Abstract: This article aims at constructing the implementations in Matlab code for basis linear multistep methods and make the brief conclusion the exactness and efficiency of these implementations and the corresponding methods. This comparision is helpful for further developments of the numerical techniques to solve the ordinary differential equation, especially for the case of stiffness and with the help of today technology, AI driven data.
Keywords: Implementation, Initial Value Problem, Linear Multistep Methods, AI driven data, Stiffness.

I. INTRODUCTION

Basic linear multistep methods to solve the initial value problem (IVP) have advantages on generating exact approximation using more than one data provided from the previous step. This approach seems to be get more care today under the development of cutting edge technology such as AI driven data. After data is updated by the supported from AI technology, the new step of approximation now is construct basing on more exact information provided. Basic linear multistep we are going to work with in this monograph includes the normal (explicit) Adam-Brashforth methods, the implicit Adam-Moulton and backward difference formulas (BDF) initiated with Rung-Katta or extrapolation technique, predictor-corrector methods with predictor constructed basing on Adam-Brashforth and corrector constructed basing on Adam-Moulton or BDF techniques.

II. IMPLEMENTATION CONSTRUCTION

Consider the initial-value problem

$$y' = f(t, y), a \le t \le b, y(a) = \alpha. \qquad (1)$$

The linear k-step methods aims at constructing the approximation w_{i+1} to $y_{i+1} = y(t_{i+1})$ basing on the data provided $w_{i-k+1}, w_{i-k+2}, \dots, w_i$ assuming that the approximations are all exact at the previous $k-1$ steps. A certain k-step method is determined by its difference equation [3]:

$$w_{i+1} = \sum_{j=1}^{k} a_{j-1} w_{i-k+j} + h \sum_{j=1}^{k+1} b_{j-1} f\left(t_{i-k+j}, w_{i-k+j}\right). \qquad (2)$$

An appropriate choice for coefficients a_j, b_j gives the definition for a certain k-step method. Particularly, when $b_k = 0$ we have an explicit method, and $b_k \neq 0$ we have an implicit one.

III. MATLAB CODE

1. Predictor-Corrector Adam-Brashforth 4-step and BDF3-step. This implementation using the pair of predictor-corrector as mentioned which have the same order of convergence, the code follwing implements it.

```
function out=AB_BDF(f,a,b,alpha,N)
syms z(s);
format long;
%---------STEP 1-------------
h=(b-a)/N;
t0=a;
w0=alpha;
t=[t0];
w=[w0];
%-------------------------STEP 2--------Initiate the starting values-----
for i=1:3
k1=h*f(t0,w0);
k2=h*f(t0+0.5*h,w0+0.5*k1);
k3=h*f(t0+0.5*h,w0+0.5*k2);
k4=h*f(t0+h,w0+k3);
w0=w0+(k1+2*k2+2*k3+k4)/6;
```

```
t0=t0+h;
t=[t,t0];
w=[w,w0];
end
%-----------------------------------STEP 3------------
for i=4:N
   t0=t0+h;
   wp=w(i)+h*(55*f(t(i),w(i))-59*f(t(i-1),w(i-1))+37*f(t(i-2),w(i-2))-9*f(t(i-3),w(i-3)))/24;
   w0=18/11*w(i)-9/11*w(i-1)+2/11*w(i-2)+6/11*h*f(t0,wp);
   t=[t,t0];
   w=[w,w0];
end
eqn=diff(z,s)==f(s,z);
inc=z(a)==alpha;
as=dsolve(eqn,inc);
out=[t',w',double(subs(as,s,a:h:b))',abs(w-double(subs(as,s,a:h:b)))'];
end
```

2. Predictor-Corrector Adam-Brashforth 4-step and BDF 4-step.

```
function out=AB_BDF4(f,a,b,alpha,N)
syms z(s);
format long;
%---------STEP 1-------------
h=(b-a)/N;
t0=a;
w0=alpha;
t=[t0];
w=[w0];
%---------------------------STEP 2--------Initiate the starting values-----
for i=1:3
k1=h*f(t0,w0);
k2=h*f(t0+0.5*h,w0+0.5*k1);
k3=h*f(t0+0.5*h,w0+0.5*k2);
k4=h*f(t0+h,w0+k3);
w0=w0+(k1+2*k2+2*k3+k4)/6;
t0=t0+h;
t=[t,t0];
w=[w,w0];
end
%-----------------------------------STEP 3------------
for i=4:N
   t0=t0+h;
   wp=w(i)+h*(55*f(t(i),w(i))-59*f(t(i-1),w(i-1))+37*f(t(i-2),w(i-2))-9*f(t(i-3),w(i-3)))/24;
   w0=48/25*w(i)-36/25*w(i-1)+16/25*w(i-2)-3/25*w(i-3)+12/25*h*f(t0,wp);
   t=[t,t0];
   w=[w,w0];
end
eqn=diff(z,s)==f(s,z);
inc=z(a)==alpha;
as=dsolve(eqn,inc);
out=[t',w',double(subs(as,s,a:h:b))',abs(w-double(subs(as,s,a:h:b)))'];
end
```

3. Predictor-Corrector Adam-Brashforth 4-step and Adam-Moulton 3-step

```
function out=AB_AM(f,a,b,alpha,N)
syms z(s);
format long;
%---------STEP 1-------------
```

```
h=(b-a)/N;
t0=a;
w0=alpha;
t=[t0];
w=[w0];
%--------------------------STEP 2--------Initiate the starting values-----
for i=1:3
k1=h*f(t0,w0);
k2=h*f(t0+0.5*h,w0+0.5*k1);
k3=h*f(t0+0.5*h,w0+0.5*k2);
k4=h*f(t0+h,w0+k3);
w0=w0+(k1+2*k2+2*k3+k4)/6;
t0=t0+h;
t=[t,t0];
w=[w,w0];
end
%----------------------------------STEP 3------------
for i=4:N
   t0=t0+h;
   wp=w(i)+h*(55*f(t(i),w(i))-59*f(t(i-1),w(i-1))+37*f(t(i-2),w(i-2))-9*f(t(i-3),w(i-3)))/24;
   w0=w(i)+h*(9*f(t0,wp)+19*f(t(i),w(i))-5*f(t(i-1),w(i-1))+f(t(i-2),w(i-2)))/24;
   t=[t,t0];
   w=[w,w0];
end
eqn=diff(z,s)==f(s,z);
inc=z(a)==alpha;
as=dsolve(eqn,inc);
out=[t',w',double(subs(as,s,a:h:b))',abs(w-double(subs(as,s,a:h:b)))'];
end%-------------------END FUNCTION-----------------
```

4. Predictor-Corrector Adam-Brashforth 4-step and Adam-Moulton 4-step.

```
function out=AB_AM4(f,a,b,alpha,N)
syms z(s);
format long;
%---------STEP 1-------------
h=(b-a)/N;
t0=a;
w0=alpha;
t=[t0];
w=[w0];
%--------------------------STEP 2--------Initiate the starting values-----
for i=1:3
k1=h*f(t0,w0);
k2=h*f(t0+0.5*h,w0+0.5*k1);
k3=h*f(t0+0.5*h,w0+0.5*k2);
k4=h*f(t0+h,w0+k3);
w0=w0+(k1+2*k2+2*k3+k4)/6;
t0=t0+h;
t=[t,t0];
w=[w,w0];
end
%----------------------------------STEP 3------------
for i=4:N
   t0=t0+h;
   wp=w(i)+h*(55*f(t(i),w(i))-59*f(t(i-1),w(i-1))+37*f(t(i-2),w(i-2))-9*f(t(i-3),w(i-3)))/24;
   w0=w(i)+h*(251*f(t0,wp)+646*f(t(i),w(i))-264*f(t(i-1),w(i-1))+106*f(t(i-2),w(i-2))-19*f(t(i-3),w(i-3)))/720;
   t=[t,t0];
   w=[w,w0];
```

```
end
eqn=diff(z,s)==f(s,z);
inc=z(a)==alpha;
as=dsolve(eqn,inc);
out=[t',w',double(subs(as,s,a:h:b))',abs(w-double(subs(as,s,a:h:b)))'];
end
```

5. Predictor-Corrector Adam-Brashforth 4-step and Adam-Moulton 3-step which take form of $(PE)^M P$ with the process in correction stage is repeated M times.

```
function out=ABM_Modified(f,a,b,alpha,N,tol,M)
syms z(s);
format long;
%---------STEP 1-------------
h=(b-a)/N;
t0=a;
w0=alpha;
t=[t0];
w=[w0];
%--------------------------STEP 2--------Initiate the starting values-----
for i=1:3
k1=h*f(t0,w0);
k2=h*f(t0+0.5*h,w0+0.5*k1);
k3=h*f(t0+0.5*h,w0+0.5*k2);
k4=h*f(t0+h,w0+k3);
w0=w0+(k1+2*k2+2*k3+k4)/6;
t0=t0+h;
t=[t,t0];
w=[w,w0];
end
%---------------------------------STEP 3------------
for i=4:N
   t0=t0+h;
   wp=w(i)+h*(55*f(t(i),w(i))-59*f(t(i-1),w(i-1))+37*f(t(i-2),w(i-2))-9*f(t(i-3),w(i-3)))/24;
   Tiep=0; count=1;
while (Tiep==0)&(count<M)
   w0=w(i)+h*(9*f(t0,wp)+19*f(t(i),w(i))-5*f(t(i-1),w(i-1))+f(t(i-2),w(i-2)))/24;
if abs(w0-wp)<tol
        Tiep=1;
break
else wp=w0;
        count=count+1;
end
end
   t=[t,t0];
   w=[w,w0];
end
eqn=diff(z,s)==f(s,z);
inc=z(a)==alpha;
as=dsolve(eqn,inc);
out=[t',w',double(subs(as,s,a:h:b))',abs(w-double(subs(as,s,a:h:b)))'];
end
```

6. BDF 5-step with initial stage using the extrapolation technique. The initial stage is used to generate 5 initial value to process in the next step. This alternative choice other than using Runge-Kutta technique has more advantages in providing better approximation at the early stage.

```
function Ext=BDF_Extra(f,a,b,alpha,N,tol,hmin,nmax)
syms su;
syms z(r);
```

```
format long;
%---------------------------------PHASE 1--------------------
%----------------STEP 1---------------
q=[2,4,6,8,12,16,24,32];
t0=a;
w0=alpha;
hmax=(b-a)/N;
h=hmax;
FLAG=1; % This varable is used to exit While-loop in STEP 2
%----------------STEP 2-----------------
m=1;% Use this to count the number of starting-values
t=[t0];
w=[w0];
while (FLAG==1)&(h>1.0e-15)
   k=1;
   NFLAG=0;
%------------------STEP 3------------
while (NFLAG==0)&(k<=8)% DO STEPS 4-6
%---------------STEP 4-----------
       HK=h/q(k);
       T=t0;
       W2=w0;
       W3=W2+HK*f(T,W2);
       T=t0+HK;
for j=1:1:(q(k)-1)
          W1=W2;
          W2=W3;
          W3=W1+2*HK*f(T,W2);%Midpoint METHOD
          T=t0+(j+1)*HK;
end
       y(k,1)=0.5*(W3+W2+HK*f(T,W3)); % Endpoint correction to generate the 1st element in the row k
%-------------------------STEP 5------------
if k>=2
for j=2:1:k
                y(k,j)=y(k,j-1)+(y(k,j-1)-y(k-1,j-1))*(q(k-j+1))^2/((q(k))^2-(q(k-j+1))^2);
end
%----------------------STEP 6------Check the condition of TOLERANCE
if (abs(y(k,k)-y(k-1,k-1))<=tol)
           NFLAG=1;
end
end
%-------------------END--STEP 5-------------------
       k=k+1;
end
%----------------------END STEP 3-------------------------
   k=k-1; %To return the value of k by which the procedure in STEP 3 stopped
%-------------------------STEP 7----------------------------
if NFLAG==0 % Do STEPS 8-9
%---------------STEP 8-------------
        h=h/2;%Reduce h because the extrapolation does not procedure a desired accurate results after 8 rows
%---------------STEP 9-------------
if h<hmin
            fprintf('hmin was exceeded.The procedure fails.');
            FLAG=0;
end
else% Do STEPS 10-11
%----------------STEP 10-------------
        w0=y(k,k);
```

```
        t0=t0+h;
        t=[t,t0];
        w=[w,w0];
%----------------STEP 11------------
if t0>=b
           FLAG=0;
elseif (t0+h>b)
           h=b-t0;
elseif (k<=3)&(h<0.5*hmax)
           h=2*h;% INCREASE h if it is possible
end
end
if FLAG==1 %-----------------TIEP------------
        m=m+1;
if m>4
break%to Break while-loop in STEP 2
end
end
end%END While-loop in STEP 2
%---------------------------------PHASE 2-------------------------
h=(b-t0)/(N-4);
%-----------------------------------------------
for i=5:N
   t0=t0+h;
   w0=w(i);
   j=1;
   FLAG=0;
while FLAG==0
      w1=w0-(w0-300/137*w(i)+300/137*w(i-1)-200/137*w(i-2)+75/137*w(i-3)-12/137*w(i-4)-
60/137*h*f(t0,w0))/(1-60/137*h*double(subs(diff(f(s,u),u),{s,u},{t0,w0})));
%-------------STEP 6----------
if abs(w1-w0)<tol
          FLAG=1;
else
          j=j+1;
          w0=w1;
if j>nmax
             fprintf('Number iteration exceeded!');
break
end
end
%-------------------END STEP 6--------
end
   w=[w,w1];
   t=[t,t0];
%-------To Break for loop------
if j>nmax
break
end
%---------------------------
end
%-----------------------------------------------------------
eqn=diff(z,r)==f(r,z);
icd=z(a)==alpha;
as=dsolve(eqn,icd);
xi=double(subs(as,r,a:(b-a)/N:b));
ti=a:(b-a)/N:b;
Ext=[t',w',ti',xi',abs(w-xi)'];
```

```
end%------------------END FUNCTION-----------------
```

7. Predictor-Corrector Adam-Brashforth 4-step and Adam-Moulton 3-step with the modication in correction step using Newton iteration technique. This implementation once is presented in [4].

```
function out=ABM_Modification2(f,a,b,alpha,N,tol,M)
% N is the number of meshpoints
% alpha is the initial value
% tol is the tolerance used to control the iteration
% M is the maximum number of iteration permitted
syms z(s);
format long;
%----------------------------STEP 1----------------------------------------
h=(b-a)/N;
t0=a;
w0=alpha;
t=[t0];
w=[w0];
%----------------------------STEP 2--------Initiate the starting values-----
for i=1:3
k1=h*f(t0,w0);
k2=h*f(t0+0.5*h,w0+0.5*k1);
k3=h*f(t0+0.5*h,w0+0.5*k2);
k4=h*f(t0+h,w0+k3);
w0=w0+(k1+2*k2+2*k3+k4)/6;
t0=t0+h;
t=[t,t0];
w=[w,w0];
end
%----------------------------------STEP 3---------------------------------
for i=4:N
   t0=t0+h;
   wp=w(i)+h*(55*f(t(i),w(i))-59*f(t(i-1),w(i-1))+37*f(t(i-2),w(i-2))-9*f(t(i-3),w(i-3)))/24;
   w0=w(i)+h*(9*f(t0,wp)+19*f(t(i),w(i))-5*f(t(i-1),w(i-1))+f(t(i-2),w(i-2)))/24;
   FLAG=0;
   count=1;
   w1=w0;
while (FLAG==0)&(count<M)
      w0=w0-(w0-w(i)-h*(9*f(t0,w0)+19*f(t(i),w(i))-5*f(t(i-1),w(i-1))+f(t(i-2),w(i-2)))/24)*(w0-wp)/(w0-wp-
3/8*h*(f(t0,w0)-f(t0,wp)));
if abs(w0-w1)/abs(w0)<tol
         FLAG=1;
break
else wp=w1;
          w1=w0;
         count=count+1;
end
end
   t=[t,t0];
   w=[w,w0];
end
eqn=diff(z,s)==f(s,z);
inc=z(a)==alpha;
as=dsolve(eqn,inc);
out=[t',w',double(subs(as,s,a:h:b))',abs(w-double(subs(as,s,a:h:b)))'];
end
```

8. BDF 4-step method with the use of Newton's iteration to approximate the solution of the difference equation

```
function BDFout=BDFiv(f,a,b,alpha,N,tol,nmax)
```

```
syms sy;
syms z(r);
format long;
%---------STEP 1-------------
h=(b-a)/N;
t0=a;
w0=alpha;
t=[t0];
w=[w0];
%---------------------------STEP 2--------Initiate the starting values-----
for i=1:3
k1=h*f(t0,w0);
k2=h*f(t0+0.5*h,w0+0.5*k1);
k3=h*f(t0+0.5*h,w0+0.5*k2);
k4=h*f(t0+h,w0+k3);
w0=w0+(k1+2*k2+2*k3+k4)/6;
t0=t0+h;
t=[t,t0];
w=[w,w0];
end
%-----------------------------------STEP 3------------
for i=4:N
   t0=t0+h;
   w0=w(i);
   j=1;
   FLAG=0;
while FLAG==0
      w1=w0-(w0-48/25*w(i)+36/25*w(i-1)-16/25*w(i-2)+3/25*w(i-3)-12/25*h*f(t0,w0))/(1-
12/25*h*double(subs(diff(f(s,y),y),{s,y},{t0,w0})));
%-------------STEP 6----------
if abs(w1-w0)<tol
          FLAG=1;
else
          j=j+1;
          w0=w1;
if j>nmax
             fprintf('Number iteration exceeded!');
break
end
end
%-------------------END STEP 6--------
end
   w=[w,w1];
   t=[t,t0];
%-------To Break for loop------
if j>nmax
break
end
%----------------------------
end
eqn=diff(z,r)==f(r,z);
icd=z(a)==alpha;
as=dsolve(eqn,icd);
xi=double(subs(as,r,a:(b-a)/N:b));
ti=a:(b-a)/N:b;
BDFout=[t',w',xi',abs(w-xi)'];
end
```

9. BDF 5-step method with the use of Newton's iteration technique to approximate the solution of the implicit difference equation

```
function BDFout=BDFv(f,a,b,alpha,N,tol,nmax)
syms sy;
syms z(r);
format long;
%---------STEP 1-------------
h=(b-a)/N;
t0=a;
w0=alpha;
t=[t0];
w=[w0];
%--------------------------STEP 2--------Initiate the starting values-----
for i=1:4
k1=h*f(t0,w0);
k2=h*f(t0+0.5*h,w0+0.5*k1);
k3=h*f(t0+0.5*h,w0+0.5*k2);
k4=h*f(t0+h,w0+k3);
w0=w0+(k1+2*k2+2*k3+k4)/6;
t0=t0+h;
t=[t,t0];
w=[w,w0];
end
%---------------------------------STEP 3------------
for i=5:N
   t0=t0+h;
   w0=w(i);
   j=1;
   FLAG=0;
while FLAG==0
      w1=w0-(w0-300/137*w(i)+300/137*w(i-1)-200/137*w(i-2)+75/137*w(i-3)-12/137*w(i-4)-
60/137*h*f(t0,w0))/(1-60/137*h*double(subs(diff(f(s,y),y),{s,y},{t0,w0})));
%-------------STEP 6----------
if abs(w1-w0)<tol
          FLAG=1;
else
          j=j+1;
          w0=w1;
if j>nmax
             fprintf('Number iteration exceeded!');
break
end
end
%-------------------END STEP 6--------
end
   w=[w,w1];
   t=[t,t0];
%-------To Break for loop------
if j>nmax
break
end
%----------------------------
end
eqn=diff(z,r)==f(r,z);
icd=z(a)==alpha;
as=dsolve(eqn,icd);
xi=double(subs(as,r,a:(b-a)/N:b));
ti=a:(b-a)/N:b;
```

```
BDFout=[t',w',xi',abs(w-xi)'];
end
```

10. Continuous block BDF 3-step, which was presented in [5].

```
function blc=block(f,a,b,alpha,N,tol,M)
t0=a;
y0=alpha;
h=(b-a)/N;
%-----------
syms p(r);
format long;
eqn=diff(p,r)==f(r,p);
icd=p(a)==alpha;
as=dsolve(eqn,icd);
xi=double(subs(as,r,a:(b-a)/N:b));
ti=a:(b-a)/N:b;
%--------
A=[1;1;1];
B=[23/12,-4/3,5/12;7/3,-2/3,1/3;9/4,0,3/4];
syms s0suzs1s2s3;
K=diag([subs(diff(f(s0,z),z),[s0,z],[s+h,s1]),subs(diff(f(s0,z),z),[s0,z],[s+2*h,s2]),subs(diff(f(s0,z),z),[s0,z],[s+3
*h,s3])]);
Jg=h*B*K-eye(3);
G=u*A+h*B*[f(s+h,s1);f(s+2*h,s2);f(s+3*h,s3)]-[s1;s2;s3];
t=[t0];
w=[y0];
Y=[0;0;y0];
i=0;
FLAG=0;K=[0];
while (i<N)&(FLAG==0)
   k=1;
while k<=M
   u1=Y(2);
   u2=Y(3);
   J=double(subs(Jg,[s,s1,s2,s3],[t0,Y(1),Y(2),Y(3)]));
   G1=double(subs(G,[u,s,s1,s2,s3],[y0,t0,Y(1),Y(2),Y(3)]));
   X=linsolve(J,-G1);
   Y=Y+X;
if (norm(X)<tol)
if i==0
         K=[K;k;k;k];
           w=[w,Y'];
           t=[t,t0+h,t0+2*h,t0+3*h];
           t0=t0+3*h;
           y0=Y(3);
else
         K=[K;k];
       w=[w,Y(1)];
       t=[t,t0+h];
       t0=t0+h;
       y0=Y(1);
       Y=[u1;u2;y0];
end
break
elseif k==M
if i==0
      K=[K;M;M;M];
else
```

```
        K=[K;M];
end
    FLAG=1;
end
  k=k+1;
end
if i==0
  i=i+3;
else
  i=i+1;
end
end
if FLAG==1
  fprintf('fail');
else
blc=[t',w',xi',abs(w-xi)',K];
end
end
```

IV. NUMERICAL EXPERIMENT AND COMPARISON

Example 1. [4]Consider the following IVP:

$$y' = y - t^2 + 1, y(0) = \frac{1}{2}, 0 \le t \le 2.$$

The exact solution of the equation is $y = t^2 + 2t + 1 - \frac{e^t}{2}$. To make the comparison between above implementations, we use the tolerance $tol = 0.001$ and other parameters are given in the Table 1. The error of each result shown in Table 1 is the absolute error attained at the last step $t_N = 2$. Name of the corresponding method is identical to its index list above.

Method 1: N=6, 10, 15	**Method 2**:N = 6, 10, 15	**Method 3**: N=6, 10, 15	**Method 4**: N= 6, 10, 15	**Method 5:** N=6, 10, 15; M =10; tol =0.001
0.0152, 0.00634, 0.00253	0.00254, 0.000726, 0.000213	0.000491, 0.000101, 0.000286	0.0000922, 0.0000398, 0.821×10^{-5}	0.00151, 0.000101, 2.86×10^{-5}
Method 6: N=6, 10, 15; hmin=0.01; nmax=8; tol=0.001.	**Method 7**: N=6, 10, 15; M= 10; tol=0.001	**Method 8**: N= 6, 10, 15; nmax=10; tol =0.001	**Method 9:** N= 6, 10, 15; nmax=10; tol =0.001	**Method 10**: N=6, 10, 15; M= 10, tol=0.001.
0.000616, 0.000162, 3.07×10^{-5}	0.00165, 0.000262, 5.59×10^{-5}	0.00528, 0.001194, 0.00301	0.00143, 0.000237, 4.27×10^{-5}	0.12577, 0.02524, 0.00718.

Table 1. The absolute error provided by the corresponding method at the last mesh point t_N, numbered from 1 to 10.

Example 2. [4]The IVP $y' = 5e^{5t}(y - t)^2 + 1, y(0) = -1, t \in [0,1]$,
has solution $y(t) = t - e^{-5t}$. Table 2 shows the absolute error of approximation with respect to each method.

Method 1: N=6, 10, 15	**Method 2**:N = 6, 10, 15	**Method 3**: N=6, 10, 15	**Method 4**: N= 6, 10, 15	**Method 5:** N=6, 10, 15; M =10; tol =0.001
394.675, 0.0004129, 0.0002571	594.29, 0.00514, 0.00032	2×10^{-6}, 5×10^{-7}, 1.4×10^{-7}	7.45×10^{-7}, 1.34×10^{-7}, 0.2827×10^{-7}	0.0008572, 0.0001352, 1.98×10^{-5}
Method 6: N=6, 10, 15; hmin=0.01; nmax=8; tol=0.001.	**Method 7**: N=6, 10, 15; M= 10; tol=0.001	**Method 8**: N= 6, 10, 15; nmax=10; tol =0.001	**Method 9:** N= 6, 10, 15; nmax=10; tol =0.001	**Method 10**: N=6, 10, 15; M= 10, tol=0.001.
0.002366, 0.0001185, 1.1144×10^{-5}	0.0003818, 1.6581×10^{-5}, 2.7449×10^{-6}	0.003947, 0.0001926, 2.2835×10^{-5}	0.0018343, 9.684×10^{-5}, 1.2752×10^{-5}	0.0003928, 0.00013755, 6.153×10^{-5}.

Table 2. Absolute error to the approximation of the IVP in Example 2 at the last mesh point t_N produced by the corresponding method.

Example 3 [4] Consider the IVP $y' = -20y + 20\cos t - \sin t, y(0) = 0, t \in [0,2]$.
The exact solution to the IVP is $y(t) = \cos t - e^{-20t}$. The approximations produced by the study methods have the absolute errors at the last mesh point are shown in Table 3.

Method 1: N=6, 10, 15	**Method 2**:N = 6, 10, 15	**Method 3**: N=6, 10, 15,40	**Method 4**: N= 6, 10, 15,40	**Method 5:** N=6, 10, 15, 40; M =10; tol =0.001
1.7543×10^{10}, 7.241×10^{10}, 1.1767×10^{10}	1.22541×10^{10}, 3.421×10^{10}, 4.1765×10^{9}	3.393×10^{9}, 3.7982×10^{8}, 0.6368, 7.321×10^{-6}	2.4289×10^{9}, 1.2122×10^{8}, 21.4459, 4.846×10^{-8}	1.47×10^{19}, 1.265×10^{19}, 0.6368, 2.765×10^{-7}
Method 6: N=6, 10, 15; hmin=0.01; nmax=8; tol=0.001.	**Method 7**: N=6, 10, 15, 40; M= 10; tol=0.001	**Method 8**: N= 6, 10, 15, 40; nmax=10; tol =0.001	**Method 9:** N= 6, 10, 15, 40; nmax=10; tol =0.001	**Method 10**: N=6, 10, 15; M= 10, tol=0.001.
Fail	3.486×10^{5} 3.088×10^{2}, 0.09887, 7.764×10^{-9}	7.783×10^{3}, 2.215, 4.206×10^{-4} 6×10^{-8}	1.48×10^{6}, 66.2723, 4.235×10^{-3}, 3.9×10^{-7}	0.0001126, 2.4748×10^{-5}, 8.02×10^{-6}

Table 3. Absolute error at the last mesh point for the approximation to the IVP in Example 3.

Example 4 [4] The IVP $y' = -20(y - t^2) + 2t, y(0) = \frac{1}{3}, t \in [0,1]$, has the solution

$$y(t) = t^2 + \frac{e^{-20t}}{3}.$$

The approximations with the corresponding absolute errors at the last mesh point t_N produced by the survey methods are shown in Table 4.

Method 1: N=6, 10, 15, 40	**Method 2**:N = 6, 10, 15, 40	**Method 3**: N=6, 10, 15,40	**Method 4**: N= 6, 10, 15,40	**Method 5:** N=6, 10, 15, 40; M =10; tol =0.001
4.7164×10^{3}, 3.5824×10^{2}, 43.53683, 1.3×10^{-9}	3.5048×10^{3}, 2.8249×10^{2}, 53.541, 2.1×10^{-9}	2.94×10^{2}, 0.272144, 0.0258, 1.68×10^{-10}	156.076, 0.2041, 3.728×10^{-3}, 1.078×10^{-10}	1.5196×10^{5}, 3.98×10^{-4}, 1.65×10^{-4}, 1.601×10^{-10}
Method 6: N=6, 10, 15; hmin=0.01; nmax=8; tol=0.001.	**Method 7**: N=6, 10, 15, 40; M= 10; tol=0.001	**Method 8**: N= 6, 10, 15, 40; nmax=10; tol =0.001	**Method 9:** N= 6, 10, 15, 40; nmax=10; tol =0.001	**Method 10**: N=6, 10, 15; M= 10, tol=0.001.
0.00907, 0.001467, 0.0002303	2.843, 4.41228×10^{-4}, 2.97×10^{-7}, 0.304×10^{-10}	0.4164, 2.51×10^{-4}, 2.357×10^{-5}, 3.34×10^{-10}	1.1276, 1.26768×10^{-4}, 1.589×10^{-4}, 2.98×10^{-9}	0.000013224, 1.08×10^{-7}, 8.098×10^{-9}

Table 4. Absolute errors of the approximations at the last mesh point produced by the corresponding methods to the IVP in Example 4.

Since the results shown in Tables 1-4, we can see that for the non stiff problems, the most efficient methods are methods 4, 3, 5, and 6. They are most suitable for such class of IVP. Even they do not have quite large absolute stability region [1-4], they produce more accurate approximation if the stepsize is suitable.

For the IVP whose the stiffness is not too great, such as in Example 2 and Example 4, methods 10, 8 and 6, as well as 4, 3 and 6 are the most suitable to approximate the solution. They have an upper hand of high order of convergence.

For the IVP which has a great stiffness, such as the IVP in Example 3, methods 1, 2 and especially 6 are not suitable for the use even with very small stepsize. Method 8 and especially method 10, however, provide a great benefit in computation and the accuracy. These are very appropriate for the high stiff IVP.

V. CONCLUSION

The monograph contributes to the implementations of basic linear multistep methods. These methods are needed to take more care with the development of nowadays technology, such as AI driven data, which seems to make a revolution in the future for this area of numerical analysis. The conclusion can also be extracted from this paper is that the predictor-corrector Adam-Brashforth and Adam-Moulton are the most suitable for dialing with the non-stiff problem and they still have such advantages for the problems whose the stiffness are not too high or moderate. One can alse see the fact that even take a complicated steps to get started and a large number of calculation, the continuous block BDF techniques (method 10) seem to be the most suitable for stiff IVP, especially for ones whose the stiffness are much great.

VI. Acknowledgements

This work is a part of the scientific project numbered T2020-B35 supported by Thai Nguyen University of Technology. I am thankful for this financial aide.

REFERENCES

[1]. J. C. Butcher, *Numerical Method for Ordinary Differential Equations*(2nd edition, John-Wiley & Sons).
[2]. E. Süli.;D. Mayers; and Barsky, *An introduction to Numerical Analysis*(Cambridge University Press, 2003).
[3]. Richard L. Burden, J. Douglas Faires, *Numerical Analysis*(9th Edition, Brooks/Cole, 2010).
[4]. Dinh V. T., Pham. T. T. H., On The Stability of Predictor-Corrector Methods between Adam and Backward Difference Formula, *Journal of Science and Technology, Thai Nguyen University, Vol. 225 (13), 2020, 24-30.*
[5]. Dinh V. T., Pham T. T. H., Constructing The Implementation to The Continuous Block BDF Methods, *Journal of Science and Technology, Thai Nguyen University, Vol. 225 (0.6), 2020, 424-431*.