International Journal of Technical Innovation in Modern Engineering & Science (IJTIMES)
Impact Factor: 5.22 (SJIF-2017), e-ISSN: 2455-2585
Volume 4, Issue 7, July-2018

Code Generation for MacCormack Scheme and its validation for Open Channels

Yatindra Kumar

Assistant Professor, Civil Engineering Department, Punjab Engineering College Chandigarh,

Abstract— The close- form solution is not available for nonlinear partial differential equations except for very simplified cases. Several numerical methods may be used for their integration. Of these methods finite difference methods have been utilized very extensively. For satisfactory results, Several Explicit and Implicit finite difference methods are used for typical hydraulic engineering applications. MacCormack Scheme is one of the explicit finite difference methods. The code of MaCormack Scheme in C language for open channels is currently unavailable in any reference books or materials. The result obtained from developed code is verified from available code in FORTRAN 90-95 language from Yatindra et al. and FORTRAN 77 language code by Dr. M.H. Chaudhry, Open Channel Flow .

Keywords— Conservation and Non conservation form, Explicit finite difference methods, Implicit finite difference methods, Lax-Wendroff Scheme, MacCormack Scheme

I. INTRODUCTION

Basic principles in fluid flow problems are mass, momentum and energy conservation. The Computational fluid dynamics is basically advancing applications of fluid mechanics. If we assume very few assumptions in fluid flow problems, the equations which we generally obtain from analysing the fluid flow situations are non-linear partial differential equations. The analytical solutions of these equations are either very difficult or impossible. To obtain solution of these non-linear partial differential equations, finite difference methods are widely used. The finite difference methods are basically three types forward finite difference, backward finite difference and central finite difference methods. The 1st two are first-order accurate and the last one is second order accurate. The Maccormack Scheme was introduce by Robert W. MacCormack in 1969. It is an explicit scheme.

II. GENERAL EXPRESSION IN FINITE DIFFERENCE METHOD

A finite difference is a mathematical expression of the form $f(x + b) − f(x + a)$. If a finite difference is divided by $b − a$, one gets a difference quotient. Finite difference methods are numerical methods for approximating the solutions to differential equations using finite difference equations to approximate derivatives .In the finite difference method only three forms are commonly considered: forward, backward, and central differences.

Forward difference approximation is an expression of the form:

$$\left.\frac{df}{dx}\right|_{x=x_0} = \frac{f(x_0+\Delta x)-f(x_0)}{\Delta x}$$

Backward difference approximation is an expression of the form:

$$\left.\frac{df}{dx}\right|_{x=x_0} = \frac{f(x_0)-f(x_0-\Delta x)}{\Delta x}$$

Central difference approximation is an expression of the form:

$$\left.\frac{df}{dx}\right|_{x=x_0} = \frac{f(x_0+\Delta x)-f(x_0-\Delta x)}{2\Delta x}$$

III. EXPLICIT FINITE DIFFERENCE METHOD

The spatial partial derivatives replaced in terms of the variables at the known time level are referred to as the explicit finite differences. Referring to equations written below, if ***k*** is the known time and ***k+1*** is the unknown time level, then

Finite difference approximations for the spatial partial derivative, $\partial f / \partial x$, at the grid point (***i,k***) are as follows:

Backward difference

$$\frac{\partial f}{\partial x} = \frac{f_i^k - f_{i-1}^k}{\Delta x}$$

Forward difference

$$\frac{\partial f}{\partial x} = \frac{f_{i+1}^k - f_i^k}{\Delta x}$$

Central difference

$$\frac{\partial f}{\partial x} = \frac{f_{i+1}^k - f_{i-1}^k}{2\Delta x}$$

IV. IMPLICIT FINITE DIFFERENCE METHOD

The spatial partial derivatives replaced in terms of the variables at the unknown time level are referred to as the implicit finite differences. Referring to equations written below, if ***k*** is the known time and ***k+1*** is the unknown time level, then finite difference approximations for the spatial partial derivative, $\partial f / \partial x$,at the grid point (***i,k***) are as follows:

Backward difference

$$\frac{\partial f}{\partial x} = \frac{f_i^{k+1} - f_{i-1}^{k+1}}{\Delta x}$$

Forward difference

$$\frac{\partial f}{\partial x} = \frac{f_{i+1}^{k+1} - f_i^{k+1}}{\Delta x}$$

Central difference

$$\frac{\partial f}{\partial x} = \frac{f_{i+1}^{k+1} - f_{i-1}^{k+1}}{2\Delta x}$$

V. CONSERVATIVE AND NON-CONSERVATIVE FORMS OF EQUATION

Conservation form refers to an arrangement of an equation or system of equations, that emphasizes that a property represented is conserved, i.e. a type of continuity equation. In Mathematical form both the conservative and Non-Conservative equations are similar type. The difference comes with discretizing the equations. Both the non-conservative and conservative forms come from the conservation equations. For determining conservative or non-conservative forms of equations, we have to discretizing the equations. For a derivative to be conservative, it must form a telescoping series**.**

VI. GOVERNING EQUATIONS FOR ONE DIMENSIONAL UNSTEADY FLOW

Two governing equations used to analysis unsteady, one-dimensional, open channel flow are called the St. Venant equations. They are based on the principles of conservation of mass and momentum.

The conservative form of the continuity equation

$$\frac{\partial A}{\partial t}+\frac{\partial Q}{\partial x}=q_l$$

The conservative form of the momentum equation

$$\frac{\partial Q}{\partial t}+\frac{\partial}{\partial x}(QV+gA\bar{y})=gA(S_0-S_f)+V_x q_l$$

The conservation forms of the St.Venant equations for one-dimensional unsteady flow in vector form are

$$\frac{\partial U}{\partial t}+\frac{\partial F}{\partial x}+S=0$$

Where $U=\begin{pmatrix} A \\ VA \end{pmatrix}$, $F=\begin{pmatrix} VA \\ V^2A+gA\bar{y} \end{pmatrix}$ and $S=\begin{pmatrix} -q_l \\ -gA(S_0-S_F)-V_x q_l \end{pmatrix}$

V=velocity, A=cross-section area, g=acceleration due to gravity, S_0=slope of the channel, $\bar{y}$ =distance from the water surface to the the centroid of the area, S_f =slope of the energy line, q_l =lateral inflow & V_x =component of the velocity of lateral inflow.

VII. LAX WENDROFF DIFFUSIVE METHOD

Based on the explicit central finite difference, the difference method is simple to program and has been used for many hydraulic engineering applications. Solving St. Venant for U_i^+ (A and V) for interior node i at the next time step j+1 yields.

$$U_i^+=\frac{1}{2}(U_{i+1}+U_{i-1})-\frac{1}{2}\frac{\Delta t}{\Delta x}(F_{i+1}-F_{i-1})-\frac{\Delta t}{2}(S_{i-1}+S_{i+1})$$

This equation can be utilized to solve for A and V at interior nodes 2 through N. Boundary conditions & boundary equations are utilised to solve for A & V at boundary nodes 1 & n+1.

VIII. MACCORMACK SCHEME METHOD

MacCormack's method is a two-step predictor-correction version of the Lax-Wendroff scheme without the function computation in the points j+1/2 and j-1/2. The backward difference is utilised for the spatial partial derivative in the predictor step and forward difference is utilised for the special partial derivative in the corrector step. This method is much easier to apply than the Lax-Wendroff method because the Jacobian matrix does not appear in finite-difference equations. The stability requirement and accuracy for the MacCormack's numerical scheme are the same as presented for the Lax-Wendroff scheme.

Mathematical Formulation:

In the predictor step, the partials are defined as.

$$\frac{\partial U}{\partial t} = \frac{U_i^* - U_i}{\Delta t}$$

$$\frac{\partial F}{\partial x} = \frac{F_i - F_{i-1}}{\Delta x}$$

Putting these values in St.Venant equation we get

$$U_i^* = U_i - \frac{\Delta t}{\Delta x}(F_i - F_{i-1}) - \Delta t S_i$$

The computed value U_i^* gives A* & V*, which is used to compute F* and S*.

In the corrector step, the partials are defined as

$$\frac{\partial U}{\partial t} = \frac{U_i^{**} - U_i}{\Delta t}$$

$$\frac{\partial F}{\partial x} = \frac{F_{i+1}^* - F_i^*}{\Delta x}$$

Substituting these values in St.Venant equations we get

$$U_i^{**} = U_i - \frac{\Delta t}{\Delta x}(F_{i+1}^* - F_i^*) - \Delta t S_i^*$$

The value of the dependent variable at the next time step j+1

$$U_i^+ = \frac{1}{2}(U_i^* + U_i^{**})$$

IX. DEVELOPMENT OF CODE OF MACCORMACK SCHEME FOR OPEN CHANNELS(TRAPEZOIDAL TYPE) IN C LANGUAGE

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float Ar(float D, float B, float S)
{
   return((B+D*S)*D);
}
float HR(float D,float B,float S)
{
   float result;
   result=((B+D*S)*D)/(B+2.0*D*sqrt(1.0+S*S));
   return result;
}
```

```
float TOP(float D, float B, float S)
{
   return(B+2.0*D*S);
}
float CENTR(float D, float B, float S)
{
   float result;
   result=D*D*(B/2.0+D*S/3.0);
   return result;
}
float YBACK(float A, float B, float S)
{
   float result;
   result=(-B+sqrt (B*B+4.0*A*S))/(2.0*S);
   return result;
}
int main()
{

   float g,tlast,iprint,ip,ch1,B,S,cmn,cmn2,SO,C,QO,YO,VO,dx,dt,dtdx,cb,sf2,cn,v,ca,sf,cp,QVI,QVIM1,sfi,sosfi,sfim1;
   float t,yp1,sfp1,QP2,AP2,dtnew,dtnext;
   int i,end,np1,nsec,iflag,j;
   float Q[60],Y[60],YNEW[60],ArNEW[60],QNEW[60],QP1[60],AP1[60],sosfp1[60],factor[60];
   printf("\nInput Accelaration due to gravity(m/sec2)=");
   scanf("%f", &g);
   printf("\nInput No. of channel sections=");
   scanf("%d",& nsec);
   printf("\n Input Time for transient Flow Computations in seconds=");
   scanf("%f",& t last);
   printf("\nInput Counter for printing results=");
   scanf("%f",&iprint);
   printf("\nInput Channel length(m)=");
   scanf("%f",&ch1);
   printf("\nInput channel bottom width(m)=");
   scanf("%f", &B);
   printf("\nInput S channel lateral slope(S Horizontal to l vertical=)");
   scanf("%f", &S);
   printf("\nInput Manning's Coefficient=");
```

```
scanf("%f",&cmn);
printf("\nInput Channel bottom slope=");
scanf("%f", &SO);
printf("\nInput Initial Steady State Discharge(cumecs)=");
scanf("%f", &QO);
printf("\nInput initial steady state flow depth(m)=");
scanf("%f", &YO);
t=0.0;
if(g>10.0)
{
   cmn2=(cmn*cmn)/2.22;

}
else
{
   cmn2=cmn*cmn;
}

/* Steady state conditions */

C=sqrt(g*Ar(YO,B,S)/TOP(YO,B,S));
VO=QO/Ar(YO,B,S);
dx=ch1/(float)nsec;
dtdx=dt/dx;
np1=nsec+1;
for(i=0;i<np1;i++)
{
   Q[i]=QO;
   Y[i]=YO;
}
if(g<=10.0)
{
   printf("\nUnits are in SI\n\n");
}
else
{
   printf("\nUnits are in English\n\n");
}
```

```
ip=iprint;

/*Compute transient conditions*/
do
{
   if (ip>=iprint)
   {
      printf("\nT=%.3f\nY=",t);
      for (i=0;i<np1;i++)
      {
         if((Q[i]/Ar(Y[i],B,S))<0.0)
         {
            printf("%2f",Y[i]);
         }
         else
         {
            printf("%.2f",Y[i]);
         }
      }
      printf("\nV=");
      for(i=0;i<np1;i++)
      {
         printf("%.3f",(Q[i]/Ar(Y[i],B,S)));
      }
      printf("\n\n");
      ip=0;
   }
   ip=ip+1.0;
   t=t+dt;

   /*Upstream End*/
   YNEW [0] =YO;
   ArNEW[0]=Ar(YNEW[0],B,S);
   cb=sqrt(g*TOP(Y[1],B,S)/Ar(Y[1],B,S));
   sf2=cmn2*Q[1]*abs(Q[1])/(Ar(Y[1],B,S)*Ar(Y[1],B,S)*pow(HR(Y[1],B,S),1.333));
   cn=Q[1]/Ar(Y[1],B,S)-cb*Y[1]+g*(SO-sf2)*dt;
   v=cn+cb*YNEW [0];
   QNEW [0] =v*Ar(YNEW[0],B,S);
```

```
/*Downstream End*/
QNEW [np1-1] =0.0;
ca=sqrt(g*TOP(Y[nsec-1],B,S)/Ar(Y[nsec-1],B,S));
sf=Q[nsec-1]*abs(Q[nsec-1])*cmn2/(Ar(Y[nsec-1],B,S)*Ar(Y[nsec-1],B,S)*pow(HR(Y[nsec-1],B,S),1.333));
cp=ca*Y[nsec-1]+g*(SO-sf)*dt+Q[nsec-1]/Ar(Y[nsec-1],B,S);
YNEW [np1-1]=cp/ca;
ArNEW[np1-1]=Ar(YNEW[np1-1],B,S);

/*Interior Nodes*/
/*Predictor Step*/
i=1;
do
{
   QVI=Q[i]*Q[i]/Ar(Y[i], B, S) +g*CENTR(Y[i], B, S);
   QVIM1=Q[i-1]*Q[i-1]/Ar(Y[i-1],B,S)+g*CENTR(Y[i-1],B,S);
   sfi=cmn2*Q[i]*abs(Q[i])/(Ar(Y[i],B,S)*Ar(Y[i],B,S)*pow(HR(Y[i],B,S),1.333));
   sosfi=g*Ar(Y[i],B,S)*(SO-sfi);
   sfim1=cmn2*Q[i-1]*abs(Q[i-1])/(Ar(Y[i-1],B,S)*Ar(Y[i-1],B,S)*pow(HR(Y[i-1],B,S),1333));
   QP1 [i]=Q[i]-dtdx*(QVI-QVIM1)+sosfi*dt;
   AP1 [i]=Ar(Y[i],B,S)-dtdx*(Q[i]-Q[i-1]);
   if(S<=0.00)
   {
      yp1=AP1 [i]/B;
   }
   else
   {
      yp1=YBACK (AP1[i],B,S);
   }
   sfp1=cmn2*QP1[i]*abs(QP1[i])/(AP1[i]*AP1[i]*pow(HR(yp1,B,S),1.333));
   sosfp1 [i] =g*AP1 [i]*(SO-sfp1);
   factor[i]=QP1[i]*QP1[i]/Ar(yp1,B,S)+g*CENTR(yp1,B,S);
   i=i+1;
} while (i<nsec);
QP1 [np1-1] =0.0;
factor [np1-1]=g*CENTR(YNEW[np1-1],B,S);

/*Corrector Step*/
i=1;
```

```
do
{
   QP2=Q[i]-dtdx*(factor [i+1]-factor[i]) +sosfp1 [i]*dt;
   AP2=Ar(Y[i], B, S)-dtdx*(QP1 [i+1]-QP1 [i]);
   QNEW[i]=(QP1[i]+QP2)*0.5;
   ArNEW[i]=(AP1[i]+AP2)*0.5;
   if(S<=0.0)
   {
      YNEW[i]=ArNEW[i]/B;
   }
   else
   {
      YNEW[i]=YBACK(ArNEW[i],B,S);
   }
   i=i+1;
}while(i<nsec);

/*Check for stability*/
iflag=0;
j=0;
do
{
   C=sqrt(g*ArNEW[j]/TOP(YNEW[j],B,S));
   v=QNEW[j]/ArNEW[j];
   dtnew=dx/(abs(v)+C);
   if(dt<(0.75*dtnew))
   {
      dtnext=1.5*dt;
   }
   if(dtnew>=(0.75*dt))
   {
      dtnext=dt;
   }
   j=j+1;
} while ((j<=(np1-1))&&(iflag==0));

/*Reduce dt for stability and reluctance*/
if(iflag==1)
```

```
    {
      t=t-dt;
      dt=0.9*dtnew;
      ip=ip-1;
    }
    else
    {
      dt=dtnext;
      ip=ip+1.0;
      i=0;
      do
      {
        Q[i]=QNEW[i];
        Y[i]=YNEW[i];
        i=i+1;
      }while(i<=(np1-1));
    }
    dtdx = dt/dx;
  } while (t<=tlast);
  scanf("%d",&end);
  return 0;
}
/*Steady-State Conditions*/
```

X. INPUT DATA FOR C LANGUAGE CODE

The data utilised in case of transients for validations of C Language code is same as used for obtaining results FORTON language codes as given in "Open Channel flow" authored by Dr. M.H. Chaudhry, and Yatindra et al (Analysis in Transients in open channels, IIT Kharagpur 2012) for FORTRAN 77 and FORTRAN 90-95 respectively.

Type of channel: Trapezoidal Earth Channel

Manning's roughness coefficient =0.01

Channel bottom width= 10 meter

Channel side slope= 2 H: 1V

Length of channel= 1000 meter

Bottom slope of channel= 0.0001

No. of sections in which channel is divided= 11

Uniform flow depth: 3 meter

Initial steady state discharge= 77.46 m^3/s

No. of sections in which channel is divided= 11

Duration of transient computations = 350 seconds

XI. RESULTS OBTAINED FROM C LANGUAGE CODE

The Result obtained from C language code and FORTRON language codes are similar. The detailed results are tabulated below.

Table-1(Velocity variation along longitudinal direction)

Time, T (Sec.)	Velocity, m/s										
	Node-1	Node-2	Node-3	Node-4	Node-5	Node-6	Node-7	Node-8	Node-9	Node-10	Node-11
0	1.614	1.614	1.614	1.614	1.614	1.614	1.614	1.614	1.614	1.614	1.614
16.025	1.614	1.614	1.614	1.614	1.614	1.614	1.614	1.614	1.614	0.987	0
32.05	1.614	1.614	1.614	1.614	1.614	1.614	1.614	1.614	1.313	0.599	0
48.076	1.614	1.614	1.614	1.614	1.614	1.614	1.614	1.496	0.86	0.318	0
64.101	1.614	1.614	1.614	1.614	1.614	1.614	1.57	1.168	0.461	0.144	0
80.126	1.614	1.614	1.614	1.614	1.614	1.598	1.393	0.709	0.208	0.067	0
96.151	1.614	1.614	1.614	1.614	1.608	1.517	1.007	0.331	0.079	0.039	0
112.177	1.614	1.614	1.614	1.612	1.574	1.27	0.537	0.123	0.037	0.021	0
128.202	1.614	1.614	1.613	1.598	1.444	0.818	0.211	0.038	0.029	0.015	0
144.227	1.614	1.613	1.608	1.538	1.111	0.369	0.059	0.026	0.023	0.015	0
160.252	16.13	1.611	1.581	1.341	0.614	0.116	0.017	0.031	0.023	0.01	0
176.278	1.609	1.6	1.482	0.913	0.226	0.02	0.03	0.03	0.021	0.01	0
192.303	1.587	1.552	1.194	0.415	0.05	0.017	0.038	0.031	0.017	0.011	0
208.328	1.499	1.383	0.687	0.119	0.006	0.041	0.039	0.028	0.018	0.009	0
224.353	1.197	0.943	0.238	0.011	0.031	0.047	0.04	0.025	0.017	0.01	0
240.379	0.452	0.276	0.005	0.004	0.052	0.049	0.034	0.026	0.016	0.01	0
256.404	-0.588	-0.35	-0.146	0.015	0.05	0.045	0.033	0.025	0.017	0.009	0
272.429	-1.317	-0.814	-0.337	-0.056	0.029	0.035	0.033	0.025	0.017	0.009	0
288.454	-1.54	-1.128	-0.6	-0.234	-0.046	0.018	0.028	0.025	0.017	0.009	0
304.48	-1.519	-1.319	-0.885	-0.485	-0.196	-0.038	0.015	0.021	0.017	0.009	0
320.505	-1.463	-1.425	-1.13	-0.746	-0.404	-0.16	-0.031	0.01	0.014	0.009	0
336.53	-1.445	-1.473	-1.306	-0.979	-0.632	-0.337	-0.133	-0.028	0.004	0.007	0

Table-2 (Depth variation along longitudinal direction)

Time,T (Sec.)	Depth, meter										
	Node-1	Node-2	Node-3	Node-4	Node-5	Node-6	Node-7	Node-8	Node-9	Node-10	Node-11
0	3	3	3	3	3	3	3	3	3	3	3
16.025	3	3	3	3	3	3	3	3	3	3.28	3.76
32.05	3	3	3	3	3	3	3	3	3.12	3.5	3.76
48.076	3	3	3	3	3	3	3	3.05	3.33	3.65	3.81
64.101	3	3	3	3	3	3	3.02	3.19	3.55	3.73	3.82
80.126	3	3	3	3	3	3.01	3.1	3.41	3.7	3.77	3.81
96.151	3	3	3	3	3	3.04	3.27	3.63	3.76	3.8	3.82
112.177	3	3	3	3	3.02	3.15	3.5	3.75	3.79	3.81	3.83
128.202	3	3	3	3.01	3.08	3.36	3.69	3.79	3.8	3.81	3.83
144.227	3	3	3	3.03	3.22	3.6	3.78	3.8	3.81	3.82	3.83
160.252	3	3	3.01	3.12	3.46	3.75	3.8	3.8	3.81	3.82	3.84
176.278	3	3.01	3.06	3.31	3.68	3.8	3.8	3.8	3.82	3.83	3.84
192.303	3	3.03	3.18	3.57	3.79	3.8	3.8	3.81	3.82	3.83	3.84
208.328	3	3.09	3.42	3.75	3.8	3.8	3.8	3.81	3.82	3.83	3.84
224.353	3	3.24	3.66	3.8	3.8	3.79	3.81	3.82	3.83	3.84	3.85
240.379	3	3.44	3.76	3.8	3.79	3.8	3.81	3.82	3.83	3.84	3.85
256.404	3	3.49	3.73	3.77	3.79	3.8	3.81	3.82	3.83	3.84	3.85
272.429	3	3.37	3.6	3.72	3.78	3.8	3.82	3.83	3.84	3.85	3.86
288.454	3	3.2	3.44	3.63	3.75	3.8	3.82	3.83	3.84	3.85	3.86
304.48	3	3.08	3.29	3.51	3.68	3.77	3.82	3.83	3.85	3.86	3.87
320.505	3	3.02	3.18	3.39	3.58	3.71	3.8	3.83	3.85	3.86	3.87
336.53	3	3.01	3.1	3.28	3.46	3.63	3.75	3.81	3.85	3.86	3.87

XII. CONCLUSIONS

The Maccormack Scheme is useful in solving nonlinear partial differentials equations whose analytical solution is either difficult are impossible. The method is advantageous over lax wendroff diffusive method. The coding is very important for Civil Engineers and it helps for in easier in calculations, time saving and for more precise results. Nowadays C is being preferred for Civil Engineering students and Engineers.

.

REFERENCES

[1] Dean Miller, Bradley L. Jones and Peter Aitken, "Sams Teach Yourself - C Programming in One Hour a Day", Pearson Education, Indiana, USA, 2014

[2] E Balagurusamy, "Programming In Ansi C", TMH Publications, New Delhi, India, 2004

[3] Joe D. Hoffman and Steven Frankel, "Numerical Methods for Engineers and Scientists", McGraw-Hill, New York,USA, 2001

[4] John A. Trangenstein, "Numerical Solution of Hyperbolic Partial Differential Equations", Cambridge University Press, Cambridge, U.K., 2007

[5] Justyna Machalińska-Murawska and Michał Szydłowski, "Lax-Wendroff and McCormack Schemes for Numerical Simulation of Unsteady Gradually and Rapidly Varied Open Flow Channel", *Archives of Hydro-Engineering and Environmental Mechanics*, vol.60, no.1-4, pp.51-62, 2013

[6] M. Hanif Chaudhry, "Applied Hydraulic Transients", Springer Science, New York, USA, 2014

[7] M. Hanif Chaudhry, "Open Channel Flow", Springer Science, New York, USA, 2008

[8] Nopparat Pochai, "Numerical Treatment of a Modified MacCormack Scheme in a Nondimensional Form of the Water Quality Models in a Nonuniform Flow Stream," *Journal of Applied Mathematics*, vol. 2014, Article ID 274263, 8 pages, 2014. https://doi.org/10.1155/2014/274263

[9] Robert S. Bernard, "A MacCormack scheme for incompressible flow", *Computers and Mathematics with Applications*, vol.24, no.5-6, pp.151-168, 1992

[10] Ronald Fedkiw, Byung Moon Kim, Yingjie and Jarek Rossignac, "An Unconditionally Stable MacCormack Method", *Journal of Scientific Computing*, vol. 35, no.2-3, pp.350-371, 2008

[11] S.J. Ying and V.C. Liu, "An Extension of Maccormack's Method for Flows with Higher-Order Equations and In Different Configurations ", *Computers and Fluids*, vol.6, pp.173-184, 1978