Chapter 5

Algorithms for the Model Equations

5.1 Introduction

We begin our presentation of numerical methods with algorithms for the model equations shown in Chapter 1, Section 1.8. Algorithms for the Full Potential, Transonic Small Disturbance, Euler and Navier-Stokes equations follow in later chapters. The algorithms presented here for the model equations are the building blocks for the procedures presented in the following chapters. Algebraic mesh generation for simple geometries is also presented herein.

5.2 The Model Hyperbolic Equation - The Wave Equation

Consider the following *Initial Value Problem*.

$$\text{Solve } \frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \quad \text{for } x_0 \le x \le x_1 \quad \text{where } c \text{ is constant}$$

The initial condition is specified at $t = t_0$ by $u(x,t_0) = g(x)$

The boundary condition is specified at $x = x_0$ if $c > 0$ by $u(x_0,t) = u_0$, or it is specified at $x = x_1$ if $c < 0$ by $u(x_1,t) = u_1$.

The exact solution for $t > t_0$ is $u(x,t) = g(x - c(t - t_0))$

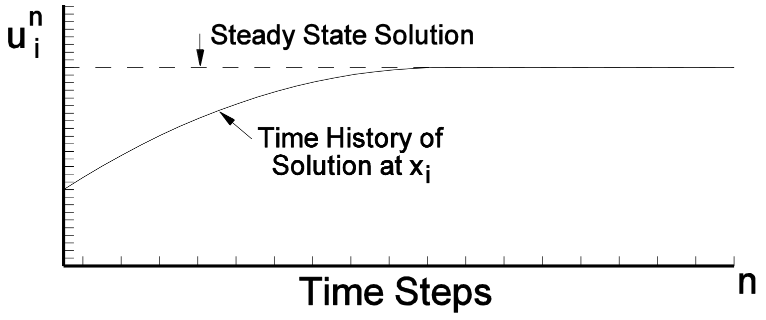


Figure 5.1 Time history of solution at x_i.

Table – Methods for Solving Model Hyperbolic Equation

Method	Difference Equations	Stability Criteria	Order of Accuracy
(1) Backward Explicit	$u_i^{n+1} = u_i^n - c\Delta t \frac{D_- \cdot}{\Delta x} u_i^n$	$c > 0$ $\Delta t \le \frac{\Delta x}{\lvert c \rvert}$	$O(\Delta t, \Delta x)$
(2) Forward	$u_i^{n+1} = u_i^n - c\Delta t \frac{D_+ \cdot}{\Delta x} u_i^n$	$c < 0$	$O(\Delta t, \Delta x)$

Explicit		$\Delta t \le \frac{\Delta x}{\lvert c\rvert}$	
(3) Central Explicit	$u_i^{n+1} = u_i^n - c\Delta t \frac{D_0 \cdot}{\Delta x} u_i^n$	Always Unstable	$O(\Delta t, \Delta x^2)$
(4) Central Implicit	$u_i^{n+1} = u_i^n - c\Delta t \frac{D_0 \cdot}{\Delta x} u_i^{n+1}$	Always stable	$O(\Delta t, \Delta x^2)$
(5) Crank-Nicolson	$u_i^{n+1} = u_i^n - c\Delta t \frac{D_0 \cdot}{\Delta x} \frac{u_i^n + u_i^{n+1}}{2}$	Always stable	$O(\Delta t^2, \Delta x^2)$
(6) Lax	$u_i^{n+1} = \frac{u_{i-1}^n + u_{i+1}^n}{2} - c\Delta t \frac{D_0 \cdot}{\Delta x} u_i^n$	$\Delta t \le \frac{\Delta x}{\lvert c\rvert}$	$O(\Delta t, \frac{\Delta x^2}{\Delta t}, \Delta x^2$
(7) Lax-Wendroff	$u_i^{n+1} = u_i^n - c\Delta t \frac{D_0 \cdot}{\Delta x} u_i^n$ $+ \frac{1}{2} c^2 \Delta t^2 \frac{D_+ \cdot}{\Delta x} \frac{D_- \cdot}{\Delta x} u_i^n$	$\Delta t \le \frac{\Delta x}{\lvert c\rvert}$	$O(\Delta t^2, \Delta x^2)$
(8) MacCormack	$u_i^{\overline{n+1}} = u_i^n - c\Delta t \frac{D_+ \cdot}{\Delta x} u_i^n$ $u_i^{n+1} = \frac{1}{2}\left\{ u_i^n + u_i^{\overline{n+1}} - c\Delta t \frac{D_- \cdot}{\Delta x} u_i^{\overline{n+1}} \right\}$	$\Delta t \le \frac{\Delta x}{\lvert c\rvert}$	$O(\Delta t^2, \Delta x^2)$
(9) Jameson	$u_i^{(0)} = u_i^n$ $u_i^{(k)} = u_i^n - \alpha_k c\Delta t \frac{D_0 \cdot}{\Delta x} u_i^{(k-1)}$ $\alpha_k = \frac{1}{5-k}, k = 1,2,3,4$ $u_i^{n+1} = u_i^{(4)}$	$\Delta t \le \frac{2\sqrt{2}\Delta x}{\lvert c\rvert}$	$O(\Delta t^4, \Delta x^2)$
(10) Warming-Beam	$u_i^{n+1/2} = u_i^n - \frac{c\Delta t}{2} \frac{D_- \cdot}{\Delta x} u_i^n$ $u_i^{n+1} = u_i^n - c\Delta t \frac{D_- \cdot}{\Delta x} \left\{ u_i^{n+1/2} + \frac{\Delta x}{2} \frac{D_- \cdot}{\Delta x} u_i^n \right\}$	$c > 0$ $\Delta t \le \frac{2\Delta x}{\lvert c\rvert}$	$O(\Delta t^2, \Delta x^2)$
(11) Upwind	$f_{i+1/2}^n = c\frac{u_i^n + u_{i+1}^n}{2} - \frac{1}{2}\lvert c\rvert (u_{i+1}^n - u_i^n)$ $u_i^{n+1} = u_i^n - \Delta t \frac{D_- \cdot}{\Delta x} f_{i+1/2}^n$	$\Delta t \le \frac{\Delta x}{\lvert c\rvert}$	$O(\Delta t, \Delta x)$

The above algorithms are the building blocks for the procedures to be presented for solving the Euler equations in Chapter 9.

Notes

1. Not all of the above numerical methods for the simple wave equation are suitable (stable and consistent) for a given value of c.
2. Method (3), centered in space and explicit in time should never be used as is. However, Jameson used it within a Runge-Kutta framework, Method (9), to devise a stable algorithm fourth order accurate in time and second order accurate in space with an explicit CFL limit more than twice as large as standard algorithms.
3. The Lax algorithm, Method (6), is an inconsistent method, because the manner in which the truncation error goes to zero under mesh refinement places a constraint that Δx^2 must go to zero faster than Δt.
4. The MacCormack two step algorithm, Method (8), combines Methods (1) and (2) to form a second order accurate procedure. For solving the linear wave equation it can be written as the single step Lax-Wendroff algorithm, Method (7).
5. The Warming-Beam method can also be written as a single step method for solving the linear wave equation as follows.

$$u_i^{n+1} = u_i^n - c\Delta t \frac{D_- \cdot}{\Delta x} \left\{ u_i^n + \frac{\Delta x}{2}\left(1 - \frac{c\Delta t}{\Delta x}\right) \frac{D_- \cdot}{\Delta x} u_i^n \right\}$$

6. For non-linear sets of equations, i.e., the Euler equations, the above two step procedures will avoid the calculation of Jacobian matrices, here represented for the model hyperbolic equation simply by the wave speed c.
7. Method (11) becomes an "upwind" method, regardless of whether the wind is blowing to the left or right. If $c \geq 0$ it becomes method (1), otherwise it is method (2). This method is the basis of the Steger-Warming and Roe methods to be analyzed later in Chapter 9.

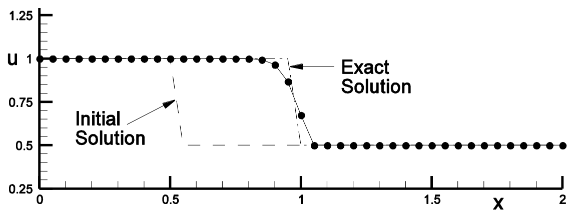
<u>Exercise (1):</u> Solve the initial value problem for the model hyperbolic equation for $c = 1$ using each of the explicit methods above with initial condition given by $u = 1$ for $x \leq 1/2$ and $u = 1/2$ for $x > 1/2$ and boundary condition given by $u = 1$ at $x = 0$. Use 41 points to span the interval $0 \leq x \leq 2$ with mesh points $x_i = (i-1)\Delta x$ and $\Delta x = \dfrac{2}{40}$. Set the CFL Number equal to $\left|\dfrac{c\Delta t}{\Delta x}\right| = 0.9$ and run each method for 10 time steps. Use Method (1) above to calculate the boundary point value at $x = 2$ for each explicit method.

<u>5.2.1 Algorithm for explicit methods</u>

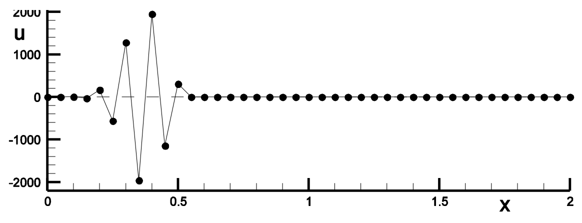
All eleven methods listed above are explicit methods, except Methods (4) and (5). They are solved using at least two levels of machine storage, one for the present solution u_i^n for all i and the other to place the new solution u_i^{n+1} for all i. Otherwise, the present solution may be overwritten inadvertently on sweeping through the mesh. The Jameson method requires three levels of storage, one each for u_i^n, $u_i^{(k-1)}$ and $u_i^{(k)}$. As an example, consider the following program for solving the initial value problem given in Exercise (1) using Method (1), the backward explicit method.

```
1) Set up storage for $x_i$, $u_i^n$ and $u_i^{n+1}$ for all $i$, $1 \le i \le 41$
2) Set up mesh, $\Delta x = 2/40$, $x_i = (i-1)\Delta x$ for $i = 1, 2, 3, \cdots, 41$
3) Set initial conditions, $u_i^n = 1$ if $x \le 1/2$ and $u_i^n = 1/2$ if $x > 1/2$.
4) Set $n = 0$
5) Determine the time step size, $\Delta t = 0.9 \times \min_{i=2,3,\cdots,41} \left\{ \frac{x_i - x_{i-1}}{|c|} \right\}$
6) Calculate $u_i^{n+1}$, $u_i^{n+1} = u_i^n - c\Delta t \frac{u_i^n - u_{i-1}^n}{x_i - x_{i-1}}$ for $i = 2, 3, \cdots, 40$
7) Apply the boundary condition at $x = 2$, $u_i^{n+1} = u_i^n - c\Delta t \frac{u_i^n - u_{i-1}^n}{x_i - x_{i-1}}$, for $i = 41$
8) Reset $u_i^n$, $u_i^n \leftarrow u_i^{n+1}$ for $i = 2, 3, \cdots, 41$.
9) Set $n \leftarrow n + 1$
10) If $n < 10$ go to step 5
11) End
```

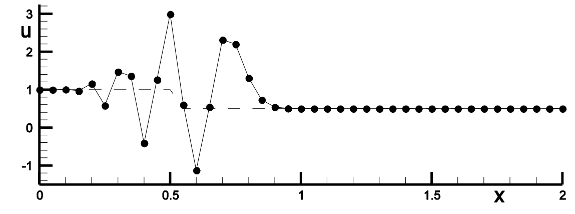
Exercise (1) Results:



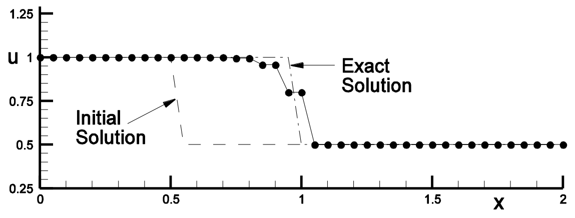
Method (1) Explicit Backward



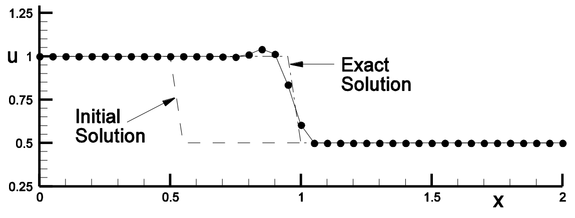
Method (2) Explicit Forward



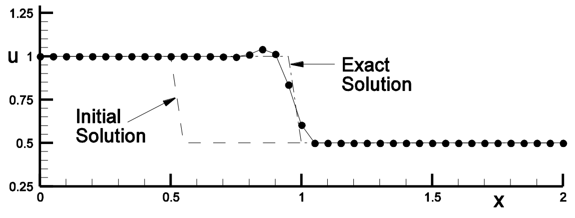
Method (3) Explicit Central



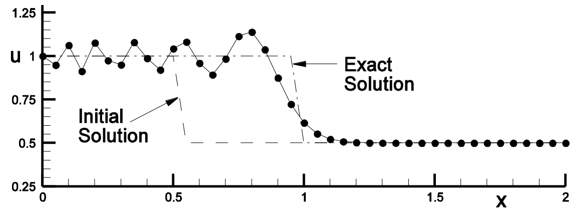
Method (6) Lax



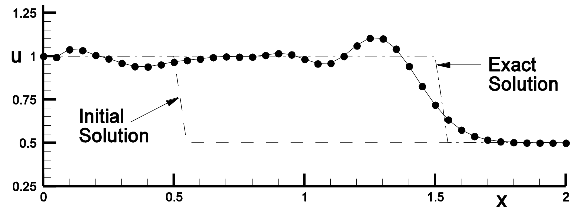
Method (7) Lax-Wendroff



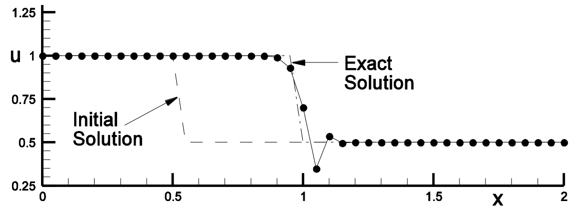
Method (8) MacCormack



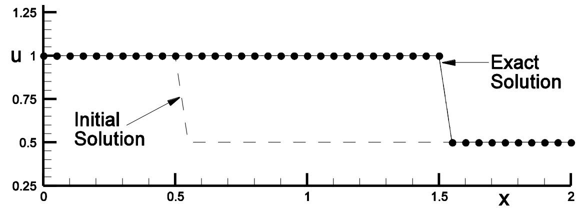
Method (9) Jameson



Method (9) Jameson, CFL=2.0



Method (10) Warming-Beam



Method (10) Warming-Beam, CFL=2.0

Figure 5.2 Results for explicit methods

The explicit method results are given in the Figure 5.2 for CFL=0.9, unless otherwise noted in the captions. Note that for $c > 0$ Method (2), Explicit Forward, is unstable as is Method (3), Explicit Central, for any c. Method (6), the Lax method, exhibits a characteristic stair case appearance. The Lax-Wendroff and MacCormack methods give identical results for the linear model hyperbolic equation and yield steeper discontinuity fronts than the first order methods, but also with attendant overshoots. Jameson's method can be used at CFL numbers more than twice as large as the other explicit methods. The Warming-Beam method has an under shoot ahead of the discontinuity for CFL=0.9 but gives an exact solution for CFL=2.0. The under and over shoots can be eliminated using a TVD (Harten's Total Variation Diminishing) strategy that will be discussed later in Chapter 10.

Exercise (2): Solve the initial value problem for the model hyperbolic equation again using Methods (1), (6), (7) and (8) with the CFL Number set equal to $\left|\frac{c\Delta t}{\Delta x}\right| = 1.0$. Notice that these algorithms have **perfect shift**, that is the solution moves the wave exactly one mesh point per time step in exact agreement with the exact solution.

[Note that in Section 3.5.3 it was shown that if Δt is chosen so that $\Delta t = \frac{\Delta x}{c}$, all the truncation error terms vanish for Method (1), thus enabling perfect shift. The same is true for Methods (6), (7), (8) and (10). Method (10), remarkably, has perfect shift for $\Delta t = 2\frac{\Delta x}{c}$, see Section 4.4.8 and Figure 5.2]

5.2.2 Algorithm for implicit methods

Implicit Methods (4) and (5) may be combined as shown below

$$u_i^{n+1} = u_i^n - (1-\alpha)c\Delta t \frac{D_0 \cdot}{\Delta x} u_i^n - \alpha c \Delta t \frac{D_0 \cdot}{\Delta x} u_i^{n+1}$$

$$\alpha = \begin{cases} 1/2, & Crank - Nicolson \\ or & \\ 1, & Fully\ Implicit \end{cases}$$

The new solution at (x_i, t^{n+1}) depends *implicitly* on itself at other points at time t^{n+1}. Thus, the new solution must in general be solved for simultaneously at all points (x_i, t^{n+1}), by inverting a matrix.

To solve, we first write it as follows

$$\alpha \frac{c\Delta t}{2\Delta x} u_{i+1}^{n+1} + u_i^{n+1} - \alpha \frac{c\Delta t}{2\Delta x} u_{i-1}^{n+1} = u_i^n - (1-\alpha)c\Delta t \frac{D_0 \cdot}{\Delta x} u_i^n$$

By defining $a_i = 1$, $b_i = -c_i = \alpha \frac{c\Delta t}{2\Delta x}$, and $f_i = u_i^n - (1-\alpha)c\Delta t \frac{D_0 \cdot}{\Delta x} u_i^n$, we can express the above equation in tridiagonal matrix form

$$\begin{bmatrix} a_I & c_I & 0 & 0 & 0 & 0 & 0 \\ b_{I-1} & a_{I-1} & c_{I-1} & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & b_i & a_i & c_i & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & b_2 & a_2 & c_2 \\ 0 & 0 & 0 & 0 & 0 & b_1 & a_1 \end{bmatrix} \begin{bmatrix} u_I^{n+1} \\ \vdots \\ \vdots \\ u_i^{n+1} \\ \vdots \\ \vdots \\ u_1^{n+1} \end{bmatrix} = \begin{bmatrix} f_I - b_I u_{I+1}^{n+1} \\ \vdots \\ \vdots \\ f_i \\ \vdots \\ \vdots \\ f_1 - c_1 u_0^{n+1} \end{bmatrix}$$

with $u_1^{n+1} = u(0, t^{n+1}) = u_0$, $u_I^{n+1} = u(2, t^{n+1}) = u_1$ supplied by boundary conditions at time $t^{n+1} = t_0 + (n+1)\Delta t$. Actually, from characteristics theory (see example (2) in Section 2.7.1 and Section 2.9 on boundary conditions) we are only allowed to specify a boundary condition at one end point for the model hyperbolic equation and not as shown here at both. However, we still need to place a value at the non-specified boundary location to complete the above matrix. Let's assume that $c > 0$, which indicates that information is being carried by the equation from left to right along a characteristic path with speed c. A boundary condition compatible with this theory and the governing equation is given below.

$$u_I^{n+1} = u_I^n - c\Delta t \frac{D_- \cdot}{\Delta x} u_I^{n+1} \quad or \quad \left(1 + \frac{c\Delta t}{\Delta x}\right) u_I^{n+1} - \frac{c\Delta t}{\Delta x} u_{I-1}^{n+1} = u_I^n$$

Therefore, for the top matrix row, the above equation becomes for $c > 0$

$$a_I = 1 + \frac{c\Delta t}{\Delta x}, \ c_I = -\frac{c\Delta t}{\Delta x}, \ b_I = 0 \text{ and } \ f_I = u_I^n$$

And at the left boundary, $x = 0$, we specify the boundary condition in the matrix equation as

$$a_1 = 1, \ b_1 = 0, \ c_1 = 0 \ \text{ and } \ f_1 = u_0$$

5.2.3 Solution by tridiagonal inversion

The above matrix equation is solved by first decomposing the tridiagonal matrix into the product of a lower bidiagonal matrix and an upper bidiagonal matrix, an $L \cdot U$ decomposition, as shown below

$$\begin{bmatrix} \alpha_I & 0 & 0 & 0 & 0 & 0 & 0 \\ b_{I-1} & \alpha_{I-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & b_i & \alpha_i & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & b_2 & \alpha_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & b_1 & \alpha_1 \end{bmatrix} \begin{bmatrix} 1 & \gamma_I & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \gamma_{I-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \gamma_i & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \gamma_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_I^{n+1} \\ \vdots \\ \vdots \\ u_i^{n+1} \\ \vdots \\ \vdots \\ u_1^{n+1} \end{bmatrix} = \begin{bmatrix} f_I - b_I u_{I+1}^{n+1} \\ \vdots \\ \vdots \\ f_i \\ \vdots \\ \vdots \\ f_1 - c_1 u_0^{n+1} \end{bmatrix}$$

$$\text{or } \ L \cdot U\{u \ \} = \{ f_i \}, \quad \text{where } \ \alpha_I = a_I, \ \gamma_I = \frac{c_I}{\alpha_I} \ \text{ and}$$

$$\text{in general } \ \alpha_i = a_i - b_i\gamma_{i+1} \ \text{ and } \ \gamma_i = \frac{c_i}{\alpha_i} \ \text{ for } \ i = I-1, \cdots, 2, 1.$$

The solution of the above $L \cdot U$ decomposed matrix equation is obtained by a forward elimination down through the L matrix followed by and a backward substitution up through the U matrix.

The forward elimination solves the equation $L\{v_i\} = \{ f_i\}$ as follows

$$v_I = \frac{f_I - b_I u_{I+1}^{n+1}}{\alpha_I} \ \text{ then } \ v_i = \frac{f_i - b_i v_{i+1}}{\alpha_i}, \text{ for } \ i = I-1, \cdots, 3, 2 \ \text{ and } \ v_1 = \frac{f_1 - c_1 u_0^{n+1} - b_1 v_2}{\alpha_1}.$$

The backward substitution solves the equation $U\{u_i^{n+1}\} = \{v_i\}$ as follows

$$u_1^{n+1} = v_1 \ \text{ and then } \ u_i^{n+1} = v_i - \gamma_i u_{i-1}^{n+1}, \text{ for } \ i = 2, \cdots, \mathrm{I}-1, \mathrm{I}.$$

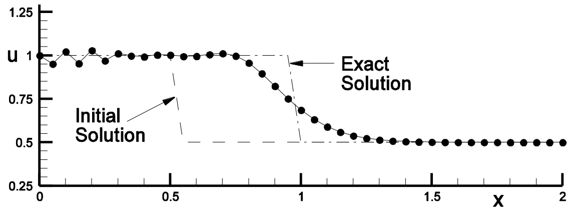
A simple "two loop" program for solving the tridiagonal matrix equation is as follows.

```
1) Set $v_{I+1}$, $u_{I+1}^{n+1}$ and $\gamma_{I+1}$ equal to $0$, and set $f_1$ equal to $f_1 - c_1 u_0^{n+1}$
2) First Loop: For $i = I, I-1, \cdots, 3, 2, 1$
    a) Calculate $a_i, b_i$ and $c_i$.
    b) Calculate $\alpha_i$. $\alpha_i = a_i - b_i \gamma_{i+1}$
    c) Calculate and store $\gamma_i$. $\gamma_i = \frac{c_i}{\alpha_i}$
    d) Calculate and store $v_i$. $v_i = \frac{f_i - b_i v_{i+1}}{\alpha_i}$
3) Set $u_1^{n+1} = v_1$
4) Second Loop: For $i = 2, \cdots, \mathrm{I}-1, \mathrm{I}$
    a) Calculate and store $u_i^{n+1} = v_i - \gamma_i u_{i-1}^{n+1}$
```

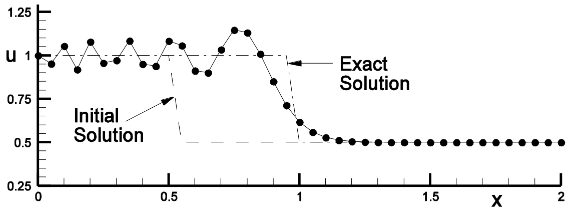
Note that the variables a_i, b_i, c_i *and* α_i need not be stored for further use, but γ_i and v_i do.

<u>Exercise (3):</u> Use the implicit procedure discussed above to solve the initial value problem for the model hyperbolic equation for implicit Methods (4) and (5). Use the same conditions as in the preceding exercise for the explicit methods. Use CFL numbers 0.9 and 2.0.

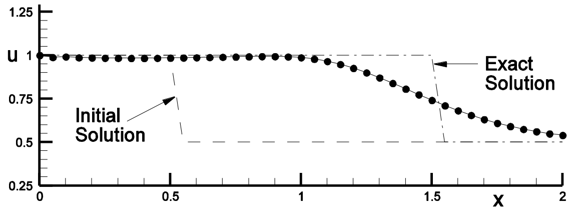
<u>Exercise (3) Results:</u>



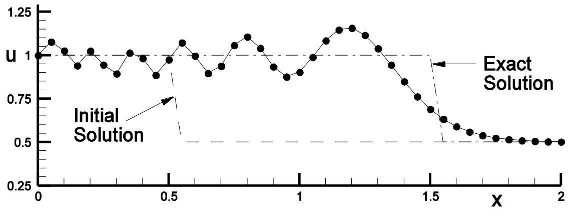
Method (4) Central Implicit, CFL=0.9



Method (5) Crank-Nicolson, CFL=0.9



Method (4) Central Implicit, CFL=2.0



Method (5) Crank-Nicolson, CFL=2.0

Figure 5.3 Results for implicit methods

Note that the higher order Crank-Nicolson method yields steeper discontinuity fronts, with the attendant over and undershoots, than the first order accurate in time Central Implicit method. The solution deterioration shown here is particularly severe for the model hyperbolic equation. This linear equation has no self steepening mechanism as is found for shock waves in the non-linear Euler equations and wave fronts smear in time with the square root of the number of time steps traveled. A simple non-linear equation is given below.

<u>5.2.3 Non-Linear Model Hyperbolic Equation</u>

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0, \quad or \quad \frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = 0 \quad with \quad f = \frac{1}{2}u^2$$

Method (1), Backward Explicit, applied to this equation is

$$u_i^{n+1} = u_i^n - \Delta t \frac{D_- \cdot}{\Delta x} f_i^n$$

and Method (4), Central Implicit becomes

$$u_i^{n+1} = u_i^n - \Delta t \frac{D_0 \cdot}{\Delta x} f_i^{n+1}$$

However, the implicit equation above requires a linearization of f_i^{n+1} (expansion of f_i^{n+1} by Taylor Series up to the linear term) before a matrix solution is possible.

$$\begin{aligned} f_i^{n+1} &= f_i^n + \left.\frac{\partial f}{\partial u}\right|_i^n (u_i^{n+1} - u_i^n) \\ &= \frac{(u_i^n)^2}{2} + \frac{2u_i^n}{2}(u_i^{n+1} - u_i^n) = -u_i^n u_i^{n+1} - f_i^n \end{aligned}$$

Method (4), Central Implicit, applied to the non-linear equation then becomes

$$u_i^{n+1} = u_i^n - \Delta t \frac{D_0 \cdot}{\Delta x} u_i^n u_i^{n+1} + \Delta t \frac{D_0 \cdot}{\Delta x} f_i^n$$

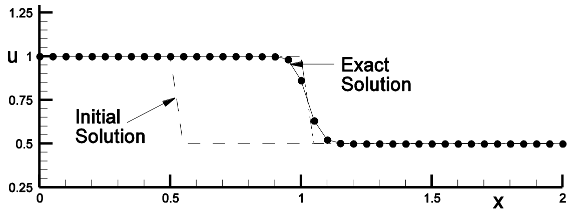
or

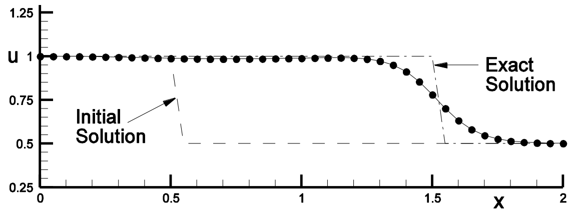
$$\left(\frac{\Delta t}{2\Delta x} u_{i+1}^n\right) u_{i+1}^{n+1} + u_i^{n+1} - \left(\frac{\Delta t}{2\Delta x} u_{i-1}^n\right) u_{i-1}^{n+1} = u_i^n + \Delta t \frac{D_0 \cdot}{\Delta x} f_i^n$$

For $u > 0$, a boundary condition can be specified at the left side, i.e., $u_1^{n+1} = 1$, but u_I^{n+1} should be calculated using backward spatial difference approximations at the right boundary, as shown below.

$$\left(1+\frac{\Delta t}{\Delta x}u_I^n\right)u_I^{n+1}-\left(\frac{\Delta t}{\Delta x}u_{I-1}^n\right)u_{I-1}^{n+1}=u_I^n+\Delta t\frac{D_-\cdot}{\Delta x}f_I^n$$

The solutions for these two methods are shown below, after 15 time steps.





Method (1) Backward Explicit, CFL=0.9 Method (4) Central Implicit, CFL=1.8

Figure 5.4 Results for implicit methods applied to nonlinear equation

Note the improvement in the appearance of the discontinuity front, particularly for the Central Implicit method, from that given earlier for the linear model hyperbolic equation. Note also that the discontinuity moves with speed $w=\frac{u_1+u_2}{2}$ (see Section 2.4.3.2) and is self steepening because the characteristic paths converge toward the discontinuity, as shown below.

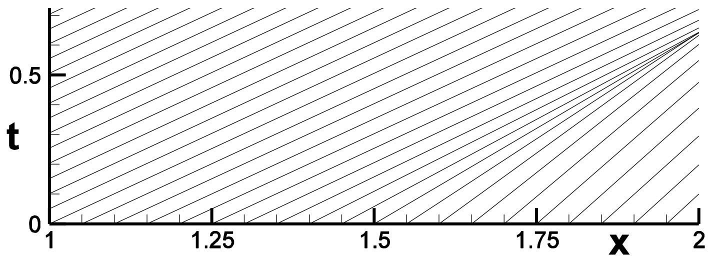


Figure 5.5 Characteristic paths converging toward a nonlinear discontinuity

However, numerical dissipation counteracts this self steepening mechanism, yielding the observed rounded “shock” profile moving through the mesh. Numerical solutions to the linear model hyperbolic equation continue to degrade, in general, with passage through the mesh.

5.2.4 Generic Form of the Algorithm for the Model Hyperbolic Equation

Consider the following algorithmic form for solving the model hyperbolic equation.

$$u_i^{n+1}=u_i^n-(1-\alpha)c\Delta t\frac{u_{i+1/2}^n-u_{i-1/2}^n}{\Delta x}-\alpha c\Delta t\frac{u_{i+1/2}^{n+1}-u_{i-1/2}^{n+1}}{\Delta x}$$

$$\text{where}\quad \alpha=\begin{cases}0, & \text{explicit}\\ 1/2, & \text{semi-implicit}\\ 1, & \text{fully implicit}\end{cases}$$

$$\text{and} \quad u_{i+1/2} = \begin{cases} u_i, & \text{backward differemce} \\ \dfrac{u_i + u_{i+1}}{2}, & \text{central differemce} \\ u_{i+1,} & \text{forward differemce} \end{cases}$$

The above form of the algorithm can be rearranged into a "delta' form

$$\left\{1 + \alpha c \Delta t \frac{D_* \cdot}{\Delta x}\right\} \delta u_i^{n+1} = -c\Delta t \frac{u_{i+1/2}^n - u_{i-1/2}^n}{\Delta x} = \Delta u_i^n$$

where $\frac{D_* \cdot}{\Delta x}$ represents the algorithm's choice of backward, central or forward difference operator and $\delta u_i^{n+1} = u_i^{n+1} - u_i^n$.

Consider a general equation of form $\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0$. We can define a generic form for the algorithm for solving this equation as follows.

$$u_i^{n+1} = u_i^n - \alpha \Delta t \frac{f_{i+1/2}^{n+1} - f_{i-1/2}^{n+1}}{\Delta x} - \Delta t(1-\alpha) \frac{f_{i+1/2}^n - f_{i-1/2}^n}{\Delta x}$$

Expanding $f_{i\pm 1/2}^{n+1}$ in time

$$f_{i\pm1/2}^{n+1} = f_{i\pm1/2}^n + \left.\frac{\partial f}{\partial t}\right|_{i\pm1/2}^n \Delta t + \ldots = f_{i\pm1/2}^n + \left.\frac{\partial f}{\partial u}\right|_{i\pm1/2}^n \left.\frac{\partial u}{\partial t}\right|_{i\pm1/2}^n \Delta t + \ldots. = f_{i\pm1/2}^n + \left.\frac{\partial f}{\partial u}\right|_{i\pm1/2}^n (u_{i\pm1/2}^{n+1} - u_{i\pm1/2}^n) + \ldots.$$

Retaining the first two terms, rearranging and using $\delta u_i^{n+1} = u_i^{n+1} - u_i^n$, we obtain finally

$$\left\{1 + \alpha \Delta t \frac{D_* \cdot}{\Delta x} \frac{\partial f}{\partial u}\right\} \delta u_i^{n+1} = -\Delta t \frac{f_{i+1/2}^n - f_{i-1/2}^n}{\Delta x} = \Delta u_i^n$$

The asterisk on the difference operator indicates that it could be a forward, backward or central difference operator and the dot indicates that it operates on all factors to the right of it. The script "delta" on δu_i^{n+1} indicates herein that it is the change in the solution calculated implicitly and the "triangle delta" on Δu_i^n indicates the solution change if calculated explicitly, for example if $\alpha = 0$, or it can be used to indicate the residual left in the solution if relaxing to steady state. After solving for δu_i^{n+1} at all interior points, the new solution becomes

$$u_i^{n+1} = u_i^n + \delta u_i^{n+1}$$

<u>Example:</u> Write the implicit central difference algorithm of the last section in *delta law form*.

In this case $f = \frac{1}{2}u^2$, $\frac{\partial f}{\partial u} = u$, $\alpha = 1$ and $\left\{1 + \Delta t \frac{D_0 \cdot}{\Delta x}\frac{\partial f}{\partial u}\right\}\delta u_i^{n+1} = -\Delta t \frac{D_0 \cdot}{\Delta x} f_i^n = \Delta u_i^n$

or

$$\left(\frac{\Delta t}{2\Delta x}u_{i+1}^n\right)\delta u_{i+1}^{n+1} + \delta u_i^{n+1} - \left(\frac{\Delta t}{2\Delta x}u_{i-1}^n\right)\delta u_{i-1}^{n+1} = -\Delta t \frac{D_0 \cdot}{\Delta x} f_i^n = \Delta u_i^n$$

Again assuming $u > 0$, the boundary condition specified at the left side is now $\delta u_1^{n+1} = 0$ and the backward spatial difference approximation at the right boundary becomes

$$\left(1 + \frac{\Delta t}{\Delta x}u_I^n\right)\delta u_I^{n+1} - \left(\frac{\Delta t}{\Delta x}u_{I-1}^n\right)\delta u_{I-1}^{n+1} = -\Delta t \frac{D_- \cdot}{\Delta x} f_I^n = \Delta u_I^n$$

Finally, the matrix equation to be solved now has the form

$$\begin{bmatrix} a_I & c_I & 0 & 0 & 0 & 0 & 0 \\ b_{I-1} & a_{I-1} & c_{I-1} & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & b_i & a_i & c_i & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & b_2 & a_2 & c_2 \\ 0 & 0 & 0 & 0 & 0 & b_1 & a_1 \end{bmatrix} \begin{bmatrix} \delta u_I^{n+1} \\ \vdots \\ \vdots \\ \delta u_i^{n+1} \\ \vdots \\ \vdots \\ \delta u_1^{n+1} \end{bmatrix} = \begin{bmatrix} \Delta u_I^n \\ \vdots \\ \vdots \\ \Delta u_i^n \\ \vdots \\ \vdots \\ \Delta u_1^n \end{bmatrix}$$

In the present case $a_1 = 1$, $b_1 = 0$ and $\Delta u_1^n = 0$, etc.

5.3 The Model Elliptic Equation

$$A\phi_{xx} + \phi_{yy} = 0, \quad A \geq 0$$

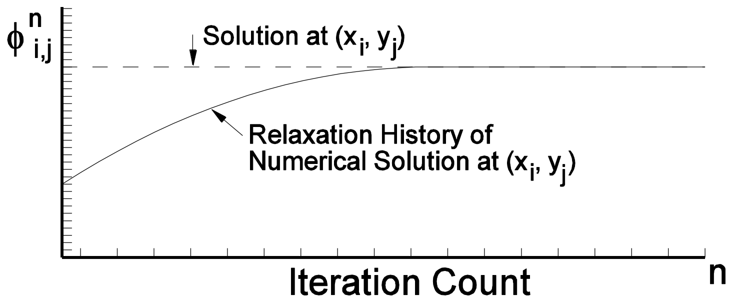


Figure 5.6 Relaxation history of solution at (x_i, y_j).

This elliptic equation is time independent and is solved by *relaxation* using the time-like iteration index n, as shown in Figure 5.6. Each method given below for solving the model elliptic equation is of $O(\Delta x^2, \Delta y^2)$.

Method	Difference Equations
(1) Point Jacobi	$\phi_{i,j}^{n+1} = \frac{1}{2(A/\Delta x^2 + 1/\Delta y^2)}\left(A\frac{\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n}}{\Delta x^2} + \frac{\phi_{i,j+1}^{n} + \phi_{i,j-1}^{n}}{\Delta y^2} \right)$
(2) Point Gauss-Seidel	$\phi_{i,j}^{n+1} = \frac{1}{2(A/\Delta x^2 + 1/\Delta y^2)}\left(A\frac{\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n+1}}{\Delta x^2} + \frac{\phi_{i,j+1}^{n} + \phi_{i,j-1}^{n+1}}{\Delta y^2} \right)$ $for\ \ i = 2,3,4,\cdots I-1\ \ and\ \ j = 2,3,4,\cdots J-1$
(3) Line Jacobi	$\phi_{i,j}^{n+1} = \frac{\Delta x^2}{2A}\left(A\frac{\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n}}{\Delta x^2} + \frac{D_+ \cdot}{\Delta y}\frac{D_- \cdot}{\Delta y}\phi_{i,j}^{n+1} \right)$
(4) Gauss-Seidel Line Relaxation	$\phi_{i,j}^{n+1} = \frac{\Delta x^2}{2A}\left(A\frac{\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n+1}}{\Delta x^2} + \frac{D_+ \cdot}{\Delta y}\frac{D_- \cdot}{\Delta y}\phi_{i,j}^{n+1} \right)$ $for\ \ i = 2,3,4,\cdots I-1$
(5) Alternating Direction Implicit (ADI) Two Steps	(1) $\phi_{i,j}^{n+1} = \frac{\Delta x^2}{2A}\left(A\frac{\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n+1}}{\Delta x^2} + \frac{D_+ \cdot}{\Delta y}\frac{D_- \cdot}{\Delta y}\phi_{i,j}^{n+1} \right),$ $for\ \ i = 2,3,4,\cdots I-1$ (2) $\phi_{i,j}^{n+1} = \frac{\Delta y^2}{2}\left(A\frac{D_+ \cdot}{\Delta x}\frac{D_- \cdot}{\Delta x}\phi_{i,j}^{n+1} + \frac{\phi_{i,j+1}^{n} + \phi_{i,j-1}^{n+1}}{\Delta y^2} \right),$ $for\ \ j = 2,3,4,\cdots J-1$

The Point Jacobi method is explicit and requires, for convenience, two levels of storage, one for $\phi_{i,j}^{n}$ and one for $\phi_{i,j}^{n+1}$, to prevent overwriting the solution. However, the Point Gauss-Seidel method, also explicit, requires only one level of storage, with the new value $\phi_{i,j}^{n+1}$ overwriting the previous value $\phi_{i,j}^{n}$ at the same machine storage location in computer memory. Besides memory storage reduction, Point Gauss-Seidel will usually converge to a relaxed solution faster than Point Jacobi. The superscripts $n+1$ above indicate that the algorithm is sweeping through the mesh in the increasing i and j directions. Therefore, when mesh point (i, j) is being updated the algorithm can make use of the newest data at points $(i-1, j)$ and $(i, j-1)$, because they have been previously updated during the current sweep through the mesh.

The Line Jacobi method updates $\phi_{i,j}^{n+1}$ at all the mesh points along a vertical line, (i,j) *for all* $j = 2,3,J-1$, simultaneously. The line is swept through the mesh of size $I \times J$ for each $i = 2,3,4,\cdots,I-1$. Line Jacobi also requires two levels of storage and is solved by inverting a tridiagonal matrix equation, a line of which is shown below.

$$-\frac{1}{\Delta y^2}\phi_{i,j+1}^{n+1} + \left(\frac{2A}{\Delta x^2} + \frac{2}{\Delta y^2}\right)\phi_{i,j}^{n+1} - \frac{1}{\Delta y^2}\phi_{i,j-1}^{n+1} = A\frac{\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n}}{\Delta x^2}$$

The Gauss-Seidel Line Relaxation method is similar to Line Jacobi with the exception that it uses the latest available updated data from previous line solutions during the same sweep, thus requiring only one level of storage. A line from its tri-diagonal matrix equation is shown below.

$$-\frac{1}{\Delta y^2}\phi_{i,j+1}^{n+1} + \left(\frac{2A}{\Delta x^2} + \frac{2}{\Delta y^2}\right)\phi_{i,j}^{n+1} - \frac{1}{\Delta y^2}\phi_{i,j-1}^{n+1} = A\frac{\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n+1}}{\Delta x^2}$$

The Alternating Direction Implicit (ADI) method alternates from vertical simultaneous line solutions for all lines for $i = 2,3,4,\cdots,I-1$ to horizontal simultaneous line solutions for all lines for $j = 2,3,4,\cdots,J-1$.

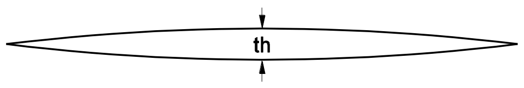


Figure 5.7 Symmetric circular arc airfoil.

Exercise (4): Solve the model elliptic equation for $A = 1 - M_\infty^2$ and $M_\infty = 0.5$ for flow past a 6% thick circular arc airfoil of chord length $c = 1$, $th = 0.06$, using each of the five methods above on a 51×51 mesh, equi-spaced over the airfoil and stretched to far field boundaries 50 chord lengths away.

Solution Approach: Assume that we normalize pressure and density so that $p_\infty = 1$ and $\rho_\infty = 1$. We can use $\phi = V_\infty x$, with $V_\infty = M_\infty a_\infty$ and $a_\infty = \sqrt{\frac{\gamma p_\infty}{\rho_\infty}}$, as an initial condition. Because of the symmetry of the flow, only the top or bottom half of the flow field needs to be calculated. We can place the chord line of the airfoil along the lower boundary given by $y = 0$ and the top, left and right far field boundaries 50 chord lengths away. Along the far field boundaries ϕ is held constant and along the lower boundary the following boundary condition on the derivative of ϕ is used.

$$\frac{\partial \phi}{\partial y} = V_\infty \left.\frac{dy(x)}{dx}\right|_{body} \quad \textit{for} \quad 0 \le x \le c \quad \textit{and} \quad \frac{\partial \phi}{\partial y} = 0 \quad \textit{otherwise.}$$

The airfoil is thin enough to use "thin airfoil" boundary conditions that apply the above condition directly on the line $y = 0$, including the chord line, and not on the actual airfoil surface itself. Therefore there is no need for a body fitted mesh and a simple mesh aligned with the Cartesian coordinate system will suffice. The broken lines in the figure below indicate extension to 50 chord lengths away from airfoil.

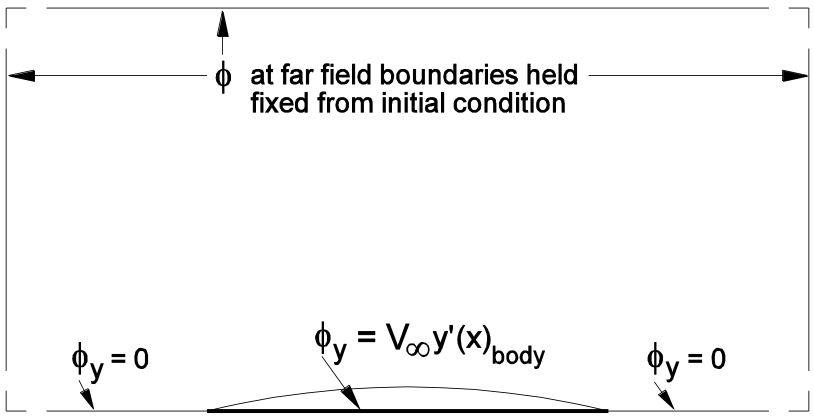

ϕ at far field boundaries held fixed from initial condition
$\phi_y = V_\infty y'(x)_{body}$
$\phi_y = 0$
$\phi_y = 0$

Figure 5.8 Flow volume about symmetric circular arc airfoil.

Let's use 21 mesh points equally spaced to span the chord line, $0 \le x \le c$, and then stretch the mesh from the airfoil leading edge, starting with $\Delta x = c/20$, to the left far field boundary and also from the trailing edge to the downstream right far field boundary, using 15 additional mesh points in each direction. From the chord line, starting with $\Delta y = th/10$, $th = 0.06c$, stretch the grid to the top far field boundary. Because the mesh is stretched, the difference approximations given in Sec 3.3 for a non-equally spaced mesh need to be used.

5.3.1 A Simple Stretching Function

A simple exponential mesh stretching formula (suggested to me by John Rakich of NASA Ames) is illustrated below for the y direction stretching, starting at $y = y_1$, ending at $y = y_{JL} = D$ and using JL mesh points. We also want the mesh point spacing to start with $y_2 - y_1 = \Delta y_{\min}$. Thus, $y_1,\ \Delta y_{\min}, D\ and\ JL$ are given, but κ is not.

$$y_j = y_1 + D\frac{e^{\kappa\frac{j-1}{JL-1}} - 1}{e^{\kappa} - 1}$$

The parameter κ is determined by Newton's method for finding the value for which $f(\kappa) = \Delta y_{\min} - (y_2 - y_1) = 0$, where $y_2 = y_1 + D\dfrac{e^{\kappa\frac{1}{JL-1}} - 1}{e^{\kappa} - 1}$. In our application, $D = 50c$, $JL = 51$ and $\Delta y_{\min} = th/10$.

Newton's method for finding κ such that $f(\kappa) = 0$ is

$$\kappa_1 = 1$$
$$\kappa_{k+1} = \kappa_k - \frac{f(\kappa_k)}{f'(\kappa_k)} \quad for \quad k = 1, \cdots, k_{\max}$$

where $k_{\max} = 10$ is usually more than sufficient to converge $\kappa_k \to \kappa$.

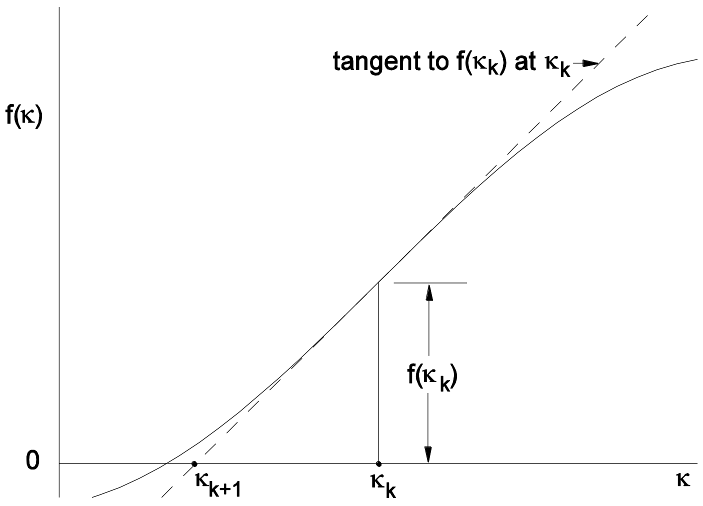


Figure 5.9 Schematic of Newton's method for finding a root of $f(\kappa) = 0$

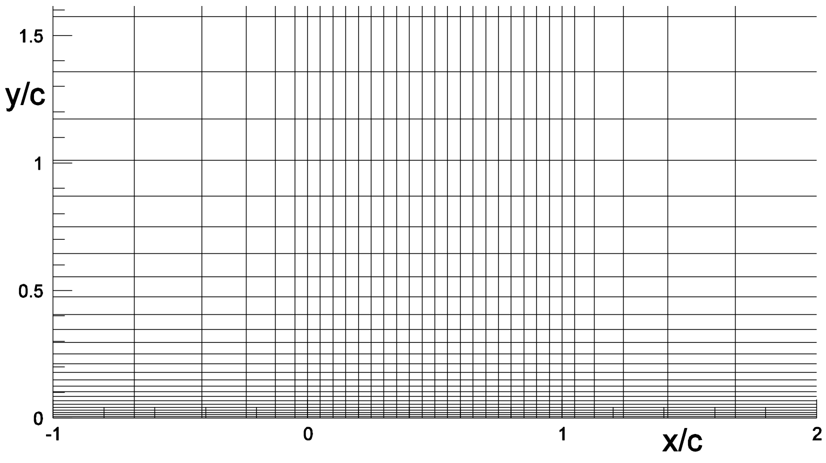


Figure 5.10 Mesh near airfoil

5.3.2 Matrix Equation for the Implicit Methods

As an example, the equation for the "i"th line for the Gauss-Seidel Line Relaxation method is (shown for simplicity for an equally spaced mesh – not as in the present exercise)

$$-\frac{1}{\Delta y^2}\phi_{i,j+1}^{n+1} + \left(\frac{2A}{\Delta x^2} + \frac{2}{\Delta y^2}\right)\phi_{i,j}^{n+1} - \frac{1}{\Delta y^2}\phi_{i,j-1}^{n+1} = A\frac{\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n+1}}{\Delta x^2},$$

which we can write as

$$b_j\phi^{n+1}_{i,j+1} + a_j\phi^{n+1}_{i,j} + c_j\phi^{n+1}_{i,j-1} = f_j = A\frac{\phi^{n}_{i+1,j} + \phi^{n+1}_{i-1,j}}{\Delta x^2}, \text{ with } b_j = c_j = -\frac{1}{\Delta y^2} \text{ and } a_j = \frac{2A}{\Delta x^2} + \frac{2}{\Delta y^2}$$

We can express the above equation in tridiagonal matrix form as

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_{JL-1} & a_{JL-1} & c_{JL-1} & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & b_j & a_j & c_j & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & b_2 & a_2 & c_2 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} \phi^{n+1}_{i,JL} \\ \vdots \\ \vdots \\ \phi^{n+1}_{i,j} \\ \vdots \\ \vdots \\ \phi^{n+1}_{i,1} \end{bmatrix} = \begin{bmatrix} \phi_\infty \\ \vdots \\ \vdots \\ f_j \\ \vdots \\ \vdots \\ -\Delta y_{\min} V_\infty \left.\dfrac{dy(x)}{dx}\right|_{body} \end{bmatrix}$$

where the top and bottom rows express the boundary conditions. This matrix equation can be solved in a similar manner to that shown in Section 5.2.2. The line is swept through the mesh, of dimension $IL \times JL$, usually in the flow direction, starting at $i = 2$ and ending at $i = IL - 1$, during each iteration $n = 1, 2, 3, \cdots$.

<u>Exercise (4) Results:</u> The mesh near the airfoil surface is shown above and the pressure coefficient c_p, plotted with the negative c_p up, is shown below.

$$c_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty V_\infty^2},\ p = p_\infty\left[1 - \frac{\gamma - 1}{2} M_\infty^2 \left(\frac{u^2 + v^2}{V_\infty^2} - 1\right)\right]^{\frac{\gamma}{\gamma-1}},\ u = \frac{\partial\phi}{\partial x} = \phi_x \text{ and } v = \frac{\partial\phi}{\partial y} = \phi_y$$

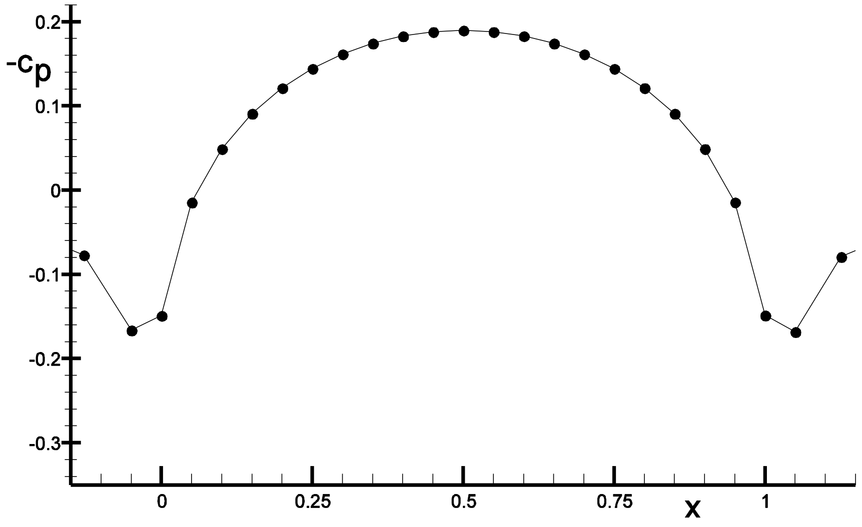


Figure 5.11 c_p vs. x for subsonic flow with $M_\infty = 0.5$

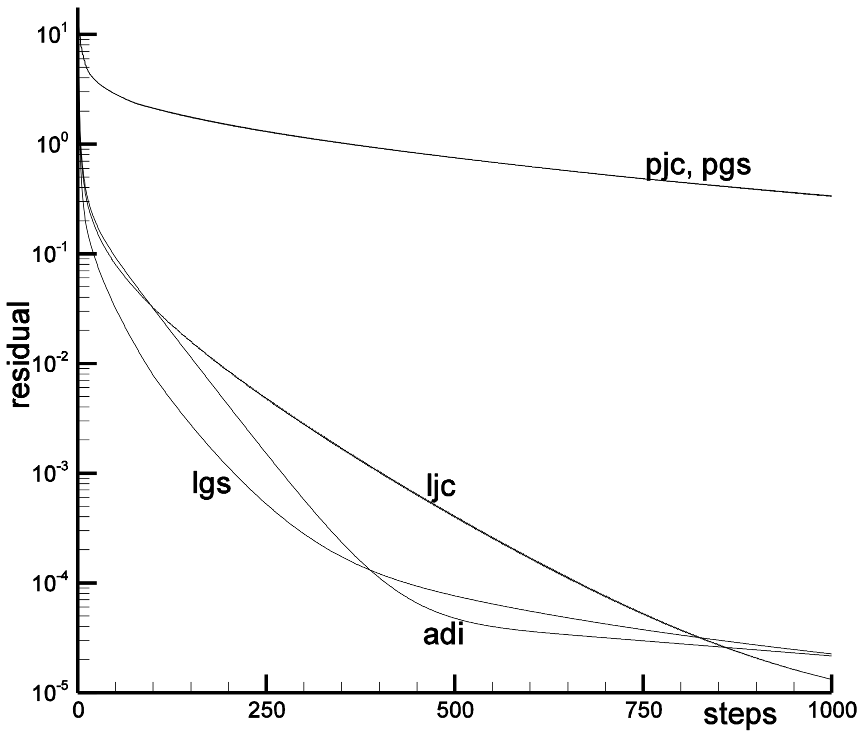


Figure 5.12 Residual vs. iteration step for several methods

The residual versus iteration step is shown in Figure 5.12 for each of the five methods. The residual is defined at each step by

$$\text{residual at } n = \left\{ \left| A \frac{\phi^n_{i+1,j} - 2\phi^n_{i,j} + \phi^n_{i-1,j}}{\Delta x^2} + \frac{\phi^n_{i,j+1} - 2\phi^n_{i,j} + \phi^n_{i,j-1}}{\Delta y^2} \right| \right\}_{\substack{\text{maximum} \\ \text{over mesh}}}$$

The point implicit methods require the most iteration steps to converge. However, the operation count per time step is higher for the implicit methods.

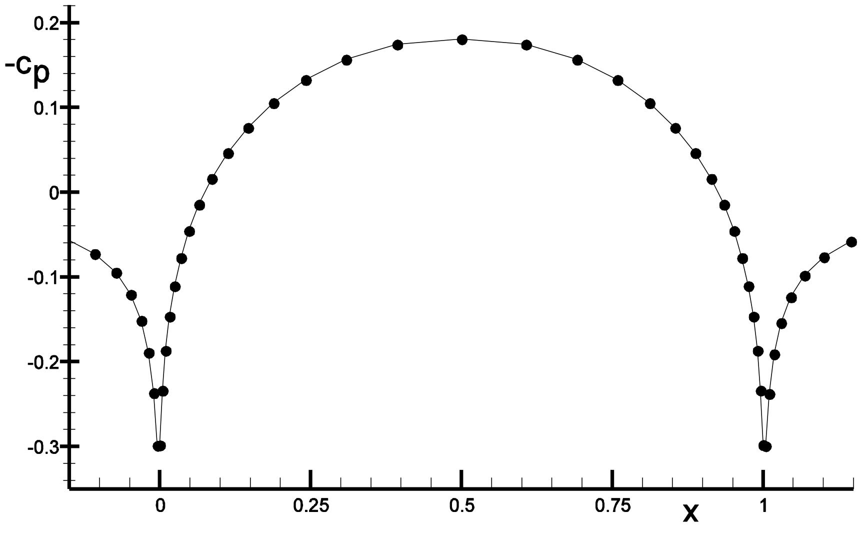


Figure 5.13 c_p vs. x for subsonic flow with $M_\infty = 0.5$ on finer mesh

The quality of the solution obtained is not sufficiently accurate near the leading and trailing edges of the airfoil. It was repeated on an 81×51 mesh, with 41 mesh points spanning the chord line and stretched to refine the leading and trailing edge regions of the flow. The result for pressure coefficient is shown in Figure 5.13.

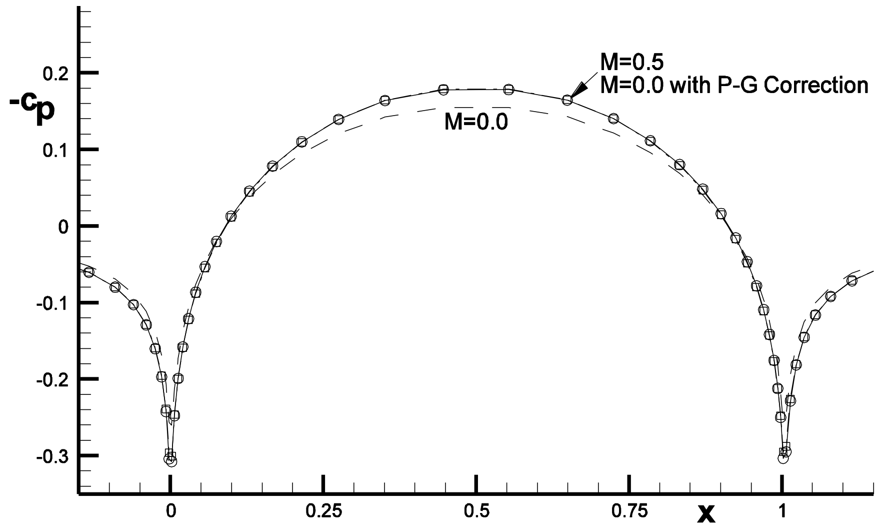


Figure 5.14 c_p comparison using Prandtl-Glauert compressibility correction factor

Figure 5.14 below shows the cp distribution for $M_\infty = 0.0$ flow, c_{p_0} (dashed curve), and two cp distributions for $M_\infty = 0.5$, one calculated from c_{p_0} using the Prantdl-Glauert compressibility correction factor $c_{p_{M_\infty=0.5}} = \dfrac{c_{p_0}}{\sqrt{1-M_\infty^2}}$ (square symbols), and the other calculated directly as in the above figure (circle symbols). The agreement between the latter two is very good, thus validating Prandtl-Glauert theory.

5.4 The Panel Method for Solving the Linear Potential Flow Equation

$$(1-\mathrm{M}_\infty^2)\phi_{xx} + \phi_{yy} + \phi_{zz} = 0$$

In the preceding we used the linear potential flow equation above as a model equation to illustrate algorithm applications. These algorithms are meant to be used for solving non-linear equations, for example, the Navier-Stokes equations to be discussed later. This equation, however, has had a central part in aerodynamics to date. It has provided needed flow solutions to nearly all air craft in flight today. Sophisticated methods have been developed, called Panel Methods, for its solution in three dimensions for both subsonic and supersonic flow as long as flow remains linear (i.e., small disturbances to the freestream away from Mach 1). Because the focus of our study is on solving the non-linear equations governing compressible flow, only a very brief description of panel methods is presented here.

We can transform this equation into Laplace's equation as follows

$$\left.\begin{array}{l} x' = \dfrac{x}{(1-M_\infty^2)^{1/2}} \\ y' = y \;\; and \;\; z' = z \end{array}\right\} \quad \Rightarrow \quad \begin{array}{c} \phi_{x'x'} + \phi_{y'y'} + \phi_{z'z'} = 0 \\ or \\ \nabla^2 \phi = 0 \end{array}$$

The velocity potential is obtained by integration over the body surface.

$$\phi(x_p, y_p) = \int_{S_{body}} \left(\sigma K_\sigma + \mu K_\mu + \gamma K_\gamma\right) ds$$

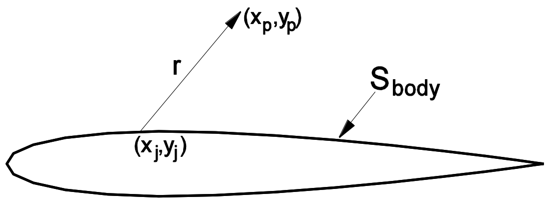
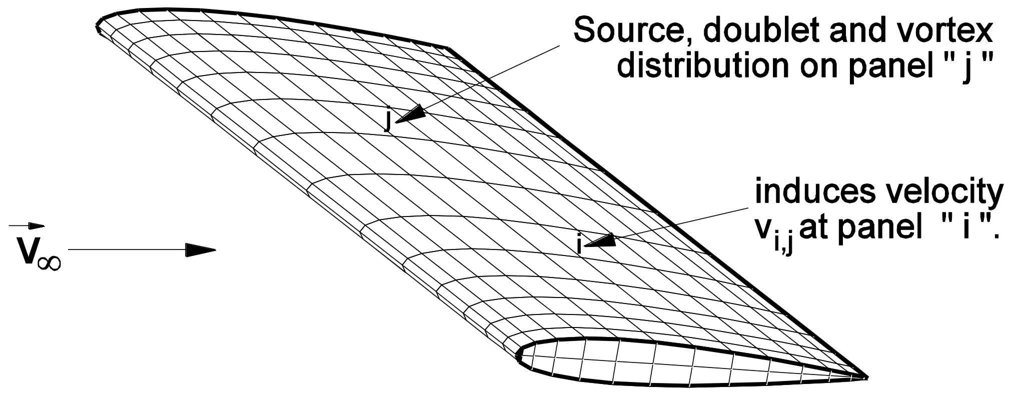


Figure 5.15 Panel Method Geometry.

where K_σ, K_μ and K_γ are known functions of body surface geometry and M_∞ and σ, μ and γ represent unknown source, doublet and vortex strengths distributed along the body surface to enforce flow tangency, The body surface is then discretized into a set of panels, as shown below.



Source, doublet and vortex distribution on panel " j "
j
induces velocity $v_{i,j}$ at panel " i ".
i
$\vec{V}_\infty$

Figure 5.16 Panels distributed along wing surface.

The strengths of σ, μ and γ are determined by satisfying the boundary condition for an impermeable surface at each of the N panels, $\left(\vec{v}_\infty + \sum_{j=1}^{N} \vec{v}_{i,j}\right) \cdot \vec{n}_i = 0$, where $\vec{n}_i$ is the unit normal to panel i, $\vec{v}_{i,j} = \vec{\nabla}\phi_j$ is the velocity induced at the control point of panel i by panel j and $\phi_j = \int_{S_j} \left(\sigma K_\sigma + \mu K_\mu + \gamma K_\gamma \right) ds$ is the velocity potential contributed by panel j. The above boundary condition requires the solution of a matrix of the order of $N \times N$. Panel elements numbering more than 1500 are common.

The velocity field about a complete aircraft configuration can be obtained with a single surface integration. Density and pressure can then be obtained using the isentropic relations given earlier (see Section 1.5). This linear numerical technique has contributed to the design of virtually all aircraft in flight today. It represents a mature well described technology and will not be discussed further herein.

5.5 Numerical Method for The Model Parabolic Equation

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2}, \quad \nu > 0$$

Consider the following numerical methods for solving the model parabolic equation

$$u_i^{n+1} = u_i^n + \nu \Delta t \frac{D_+ \cdot}{\Delta x} \frac{D_- \cdot}{\Delta x} \left((1-\alpha) u_i^n + \alpha u_i^{n+1} \right)$$

$$\alpha = \begin{cases} 0, & \textit{fully explicit}, \qquad \textit{stable if} \quad \Delta t \le \Delta x^2/(2\nu) \\ 1/2, & \left.\begin{matrix} \textit{Crank-Nicolson} \\ \textit{fully implicit} \end{matrix}\right\}, \ \textit{stable for all} \quad \alpha \ge 1/2 \\ 1, & \end{cases} , \quad O((1/2-\alpha)\Delta t, \Delta t^2, \Delta x^2)$$

Parabolic Initial Value Problem:

Initial condition: at $t = t_0 \;\; u(x,t_0) = u_0, \quad for \quad x_0 \le x < x_1$

Boundary conditions: at $x = x_0 \;\; u(x_0,t) = u_0$ and at $x = x_1 \;\; u(x_1,t) = u_1$

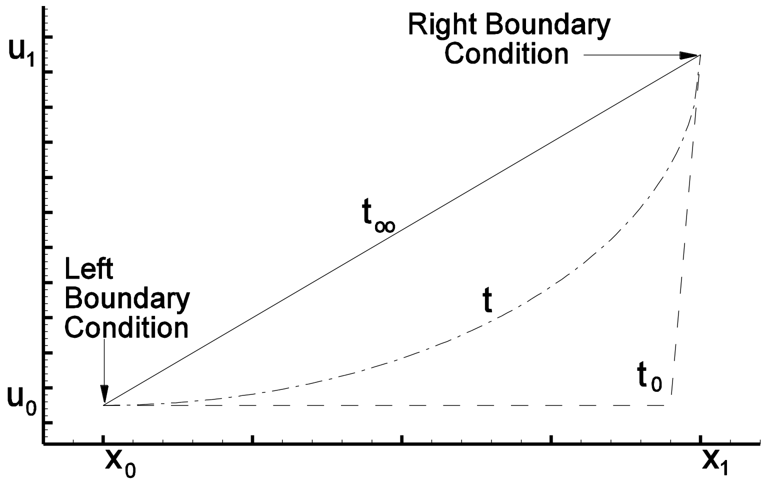


Figure 5.17 Initial value problem for the model parabolic equation

Exercise (5): Solve the initial value problem for the model parabolic equation for $\nu = 1$, $x_0 = 0$, $x_1 = 1$, $u_0 = 0$ and $u_1 = 1$ using the above algorithm with $\alpha = 0,\ 1/2$ *and* 1. Use $I = 41$ points to span the interval $0 \le x \le 1$, with mesh points $x_i = (i-1)\Delta x$ and $\Delta x = \dfrac{1}{40}$. Set the time step for the fully explicit method to the maximum allowed (see Sec.4.4.5) and run your program for 100 time steps. For the implicit methods, use $\Delta t = 1/8$ for 10 time steps with $\alpha = 1/2$ and use $\Delta t = 10^9$ for 1 time step for $\alpha = 1$.

Matrix Equation to be Solved

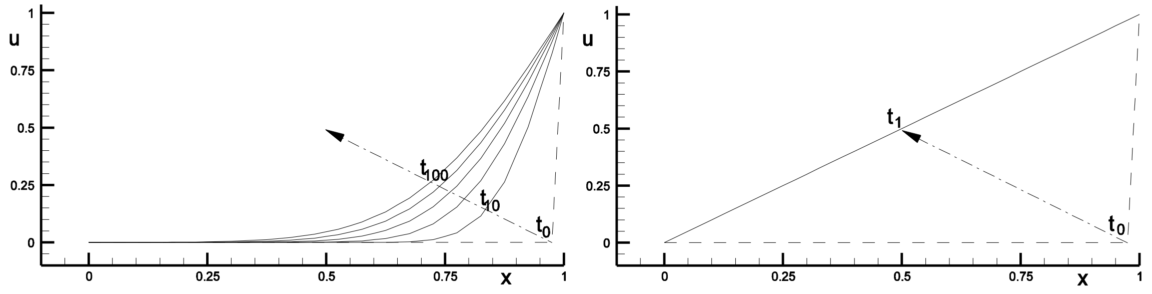
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_{I-1} & a_{I-1} & c_{I-1} & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & b_i & a_i & c_i & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & b_2 & a_2 & c_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_I^{n+1} \\ \vdots \\ \vdots \\ u_i^{n+1} \\ \vdots \\ \vdots \\ u_1^{n+1} \end{bmatrix} = \begin{bmatrix} 1 \\ f_{I-1} \\ \vdots \\ f_i \\ \vdots \\ f_2 \\ 0 \end{bmatrix}$$

$$\text{with,} \qquad u_1^{n+1} = u(0, t^{n+1}) = 0, \qquad u_I^{n+1} = u(1, t^{n+1}) = 1,$$

$$a_i = 1 + 2\alpha\nu \frac{\Delta t}{\Delta x^2}, \quad b_i = c_i = -\alpha\nu \frac{\Delta t}{\Delta x^2} \quad \text{and}$$

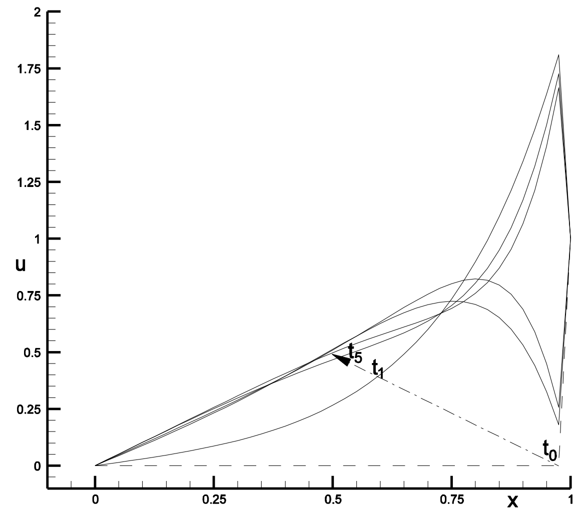
$$f_i = u_i^n + \nu\Delta t(1-\alpha) \frac{D_+ \cdot}{\Delta x} \frac{D_- \cdot}{\Delta x} u_i^n = u_i^n + \nu\Delta t(1-\alpha) \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}$$

Exercise(5) Results



Explicit, 100 steps, $\Delta t = \dfrac{\Delta x^2}{2\nu}$ Fully Implicit, 1 step, $\Delta t = 10^9$

Figure 5.18 Results for explicit and fully implicit methods applied to model parabolic equation.



Crank-Nicolson, 10 steps, $\Delta t = 1/8$

Figure 5.19 Results for the Crank-Nicolson method applied to model parabolic equation.

Note that the explicit method slowly converges toward the steady state solution and that the fully implicit method reaches it with only one time step. The second order accurate Crank-Nicolson method oscillates severely as it converges toward a steady state. The amplification factors for these three methods can be shown to be

$$G_j = \frac{1-(1-\alpha)\dfrac{2\nu\Delta t}{\Delta x^2}\left(1-\cos(k_j\Delta x)\right)}{1+\alpha\dfrac{2\nu\Delta t}{\Delta x^2}\left(1-\cos(k_j\Delta x)\right)}$$

which for $\alpha = 1\ \ G_j \to 0\ \ as\ \ \Delta t \to \infty$ and for $\alpha = 1/2\ \ G_j \to -1\ \ as\ \ \Delta t \to \infty$. This helps explain why the fully implicit method can converge in a single time step for the linear model parabolic equation and why the Crank-Nicolson method will oscillate with a "kink" in the solution from one side to the other about the steady state solution for large Δt.

An alternative look at the performance of the implicit methods can be observed from the difference equations in the limit of large Δt. The fully implicit method becomes $u_{i+1}^{n+1} - 2u_i^{n+1} - u_{i-1}^{n+1} = 0$, an equation that can only be satisfied on the equi-spaced mesh by a straight line through the boundary point values. For the Crank-Nicolson method the difference equation becomes in the limit $u_{i+1}^{n+1} - 2u_i^{n+1} - u_{i-1}^{n+1} = -\left(u_{i+1}^{n} - 2u_i^{n} - u_{i-1}^{n}\right)$, which implies that any section of the solution along a straight line at time $t = t_0 + n\Delta t$ will also be a straight line at $t = t_0 + (n+1)\Delta t$ and that points for which $u_{i+1}^{n} - 2u_i^{n} - u_{i-1}^{n}$ is non-zero $u_{i+1}^{n+1} - 2u_i^{n+1} - u_{i-1}^{n+1}$ will also be non-zero with opposite sign. Therefore, starting with our initial solution with a "kink" at the right boundary, the solution will reflect the shape of the "kink" across the steady state solution on alternate time steps, as shown below where $\Delta t = 10^9$ was used.

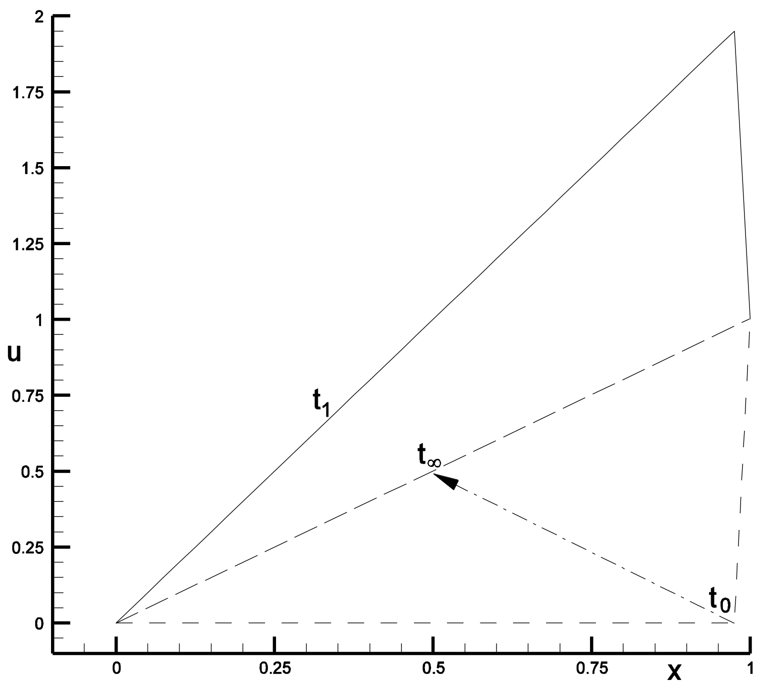


Figure 5.20 Crank-Nicolson, 1 step, $\Delta t = 10^9$

The time step used for the Crank-Nicolson method of $\Delta t = 1/8$ is far less than that used for the fully implicit case, but is sufficiently greater than that required by the explicit method of $\Delta t = \frac{\Delta x^2}{2\nu} = \frac{1}{3200}$. This choice shows qualitatively solution oscillation during convergence. There is an optimum value of Δt for convergence of this method, smaller than that used here, which will be much faster than the explicit method, but will also be much slower than the fully implicit method.