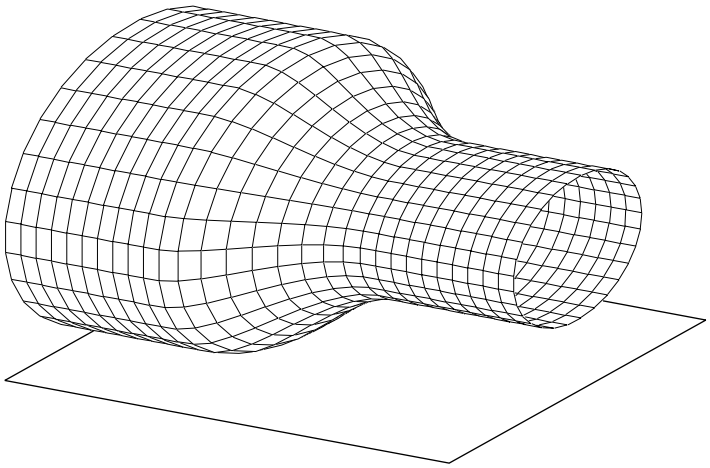
Numerical Solution of One-Dimensional Duct Flow By Method of Characteristics

Nadav Har'El

September 13, 1994



Abstract

This report describes the method of characteristics for numerical computation of flows in a duct of smoothly varying cross-section area, that can be approximated as one dimensional (*quasi-one dimensional approximation*). An inverse-marching method of characteristics is developed for polytropic gas, and results from a program based upon those algorithms are presented, and are compared with results of the GRP conservation laws scheme. A modified interpolation algorithm was employed in order to maintain gradient discontinuities in the flow (e.g., at the head and tail characteristics of a centered rarefaction wave).

Contents

1 Introduction **3**

2 The equations governing quasi one-dimensional flow **4**

3 The difference scheme for a continuous solution **6**

4 Solving the difference scheme **7**
4.1 First guess for characteristics 8
4.2 Find intersections 9
4.3 Interpolation 9
4.4 A new approximation in point 4 12
4.5 Check if we're done 13
4.6 New characteristics slope 13
4.7 Continue 13

5 Results **13**
5.1 Reflecting boundary 13
5.2 Area reduction 15

6 Acknowledgment **18**

List of Figures

1 Cylindrical tube 3
2 Varying area tube 3
3 Difference scheme 6
4 Solving the scheme 8
5 Approximate characteristic 9
6 Algorithm with interpolation only 10
7 Interpolation and extrapolation 10
8 Algorithm with extrapolation only (flow variables P,u,S) 11
9 Algorithm with extrapolation and interpolation (flow variables P,u,S) . . 12
10 Characteristics in reflection example 14
11 P(t) on boundary in reflection example 15
12 P(t) on boundary in reflection example 16
13 $\mathcal{A}(x)$ for area reduction example 17
14 U in area reduction example 17
15 P in area reduction example 18

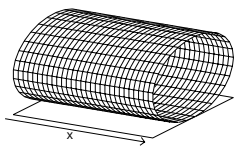


Figure 1: Cylindrical tube

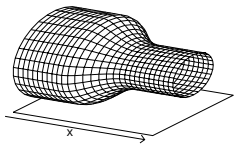


Figure 2: Varying area tube

1 Introduction

When talking about one-dimensional flow, we are talking about flow on a line without width, or equivalently, a flow in a *cylindrical* tube (see figure 1) with a constant radius r (or equivalently, a constant cross-section area $\mathcal{A} = \pi r^2$), *assuming* that the flow is dependent only on x and t, and is independent of the other dimensions, i.e.

$$u(x, y, z, t) = u(x, t)$$

A simple generalization of flow in a constant cross-section tube, is a flow in a rigid tube with varying cross-section area (see figure 2), but still assuming one-dimensional spatial variation, i.e. the flow variables are still dependent only on x and t, and the the flow velocity u is assumed to be a vector in the $\mathbf{x}$ direction. Such an assumption is called a *quasi one-dimensional approximation*. Since the tube is assumed to be rigid, the area $\mathcal{A}$ of the cross-section depends only on x, and not on t. The generalization is called an *approximation*, since obviously the assumption that the flow u depends only on x (and t) can't be true, since the $\mathbf{x}$-direction flow at the oblique tube boundary surface cannot penetrate it. However, for tubes whose cross-section areas don't change too fast, the approximation is believed to be good.

The fact that we assume that $\mathcal{A}$ is constant in time (i.e. the tube is rigid) has two consequences: First, no work is done by the tube on the fluid, so we will expect to have energy conservation. However, unlike the case of the constant area cylinder where the fluid does not exert forces on the tube (since the flow direction is parallel to the tube everywhere), here the fluid exchanges axial momentum with the tube surface, so there is no momentum conservation in the fluid.

2 The equations governing quasi one-dimensional flow

It is a standard result, that the QODA (quasi one-dimensional approximation) flow is governed by the following 3 equations, derived from mass, energy and momentum conservation for an inviscid fluid:

$$\rho_t + \rho u_x + u\rho_x + \lambda\rho u = 0 \quad (1)$$

$$u_t + uu_x + \frac{1}{\rho}P_x = 0 \quad (2)$$

$$e_t + ue_x + \frac{1}{\rho}Pu_x + \frac{\lambda}{\rho}Pu = 0 \quad (3)$$

Where λ is the rate of change of $\log \mathcal{A}$, i.e.

$$\lambda(x) = \frac{\mathcal{A}'(x)}{\mathcal{A}(x)}$$

Note that if we take the cylindrical tube case, where $\mathcal{A}$ is constant, or $\lambda \equiv 0$, we get the usual one-dimensional equations.

We shall now transform equations (1)-(3) assuming a polytropic (γ-law) gas, using as dependent variables ρ, u, and S (we shall still use the shortcut c^2, but will rewrite expressions containing P and e in terms of ρ and S). A polytropic gas is an ideal gas (i.e., $P\tau = RT$), for which $e = c_v T$ (with a constant c_v). A polytropic gas has the equation of state $P(\rho, S) = A(S)\rho^\gamma$, where $A(S) = const \cdot e^{S/c_v}$. Let

$$P = P(\rho(x,t), S(x,t))$$

then

$$P_x = \frac{\partial P}{\partial \rho}\rho_x + \frac{\partial P}{\partial S}S_x$$

But by definition,

$$\frac{\partial P}{\partial \rho} = c^2$$

$$\frac{\partial P}{\partial S} = \frac{\partial(A\rho^\gamma)}{\partial S} = \frac{\partial A}{\partial S}\rho^\gamma = \frac{A\rho^\gamma}{c_v} = \frac{c^2\rho}{c_v\gamma}$$

So we get

$$P_x = c^2\rho_x + \frac{c^2\rho}{c_v\gamma}S_x \quad (4)$$

Also, we know that

$$\frac{P}{\rho} = \frac{A\rho^\gamma}{\rho} = A\rho^{\gamma-1} = \frac{c^2}{\gamma} \quad (5)$$

Also, if we look at

$$e = e(\rho(x,t), S(x,t))$$

we get:

$$\begin{aligned} e_t &= \frac{\partial e}{\partial \rho}\rho_t + \frac{\partial e}{\partial S}S_t \\ e_x &= \frac{\partial e}{\partial \rho}\rho_x + \frac{\partial e}{\partial S}S_x \end{aligned}$$

Where by definition (note that $\gamma \neq 1$),

$$\begin{aligned} \frac{\partial e}{\partial \rho} &= \frac{\partial}{\partial \rho}\left(\frac{A}{\gamma-1}\rho^{\gamma-1}\right) = A\rho^{\gamma-2} = \frac{c^2}{\gamma\rho} \\ \frac{\partial e}{\partial S} &= \frac{\partial}{\partial S}\left(\frac{A}{\gamma-1}\rho^{\gamma-1}\right) = \frac{\partial A}{\partial S}\frac{\rho^{\gamma-1}}{\gamma-1} = \frac{A}{c_v}\frac{\rho^{\gamma-1}}{\gamma-1} = \frac{c^2}{c_v\gamma(\gamma-1)} \end{aligned}$$

So we get

$$e_t = \frac{c^2}{\gamma\rho}\rho_t + \frac{c^2}{c_v\gamma(\gamma-1)}S_t \tag{6}$$

$$e_x = \frac{c^2}{\gamma\rho}\rho_x + \frac{c^2}{c_v\gamma(\gamma-1)}S_x \tag{7}$$

Now, we use equations (4), (5), (6) and (7) in equations (1)-(3), to get (after eliminating c^2/γ from the last equation):

$$\rho_t + \rho u_x + u\rho_x + \lambda\rho u = 0 \tag{8}$$

$$u_t + uu_x + \frac{c^2\rho_x}{\rho} + \frac{c^2}{c_v\gamma}S_x = 0 \tag{9}$$

$$\frac{\rho_t}{\rho} + \frac{S_t}{c_v(\gamma-1)} + u\left(\frac{\rho_x}{\rho} + \frac{S_x}{c_v(\gamma-1)}\right) + u_x + \lambda u = 0 \tag{10}$$

Finally, we can use equation (8) to eliminate most terms of equation (10), and get the following basic equations governing QODA flow:

$$\rho_t + \rho u_x + u\rho_x + \lambda\rho u = 0 \tag{11}$$

$$u_t + uu_x + \frac{c^2\rho_x}{\rho} + \frac{c^2}{c_v\gamma}S_x = 0 \tag{12}$$

$$S_t + uS_x = 0 \tag{13}$$

Since we are interested in a method of characteristics, we are interested in converting this system of equations into an equivalent system in the characteristic form, i.e. find three equations which make an equivalent system (i.e. those equations will be linear combinations of the equations (11)-(13)), where in each equation there will be one *characteristic direction*, in which all derivatives in that equation are made. The method of finding those characteristic directions and characteristic equations is straightforward: we take a general linear combination of the three equations: $\epsilon_1(11) + \epsilon_2(12) + \epsilon_3(13) = 0$

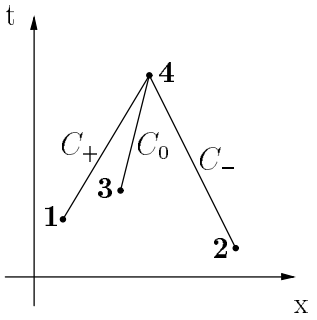


Figure 3: Difference scheme

and then try to find vectors $(\epsilon_1, \epsilon_2, \epsilon_3)$ which make the linear combination have only derivatives in a single direction. This gives us the following equations:

Characteristic directions: (14)

$$
\begin{aligned}
I_+ &: \quad x_\alpha = t_\alpha(u + c) \\
I_- &: \quad x_\beta = t_\beta(u - c) \\
I_0 &: \quad x_\omega = t_\omega u
\end{aligned}
$$

Characteristic equations: (15)

$$
\begin{aligned}
II_+ &: \quad u_\alpha + \frac{2}{\gamma - 1} c_\alpha = \frac{c}{c_v \gamma(\gamma - 1)} S_\alpha - \lambda c u t_\alpha \\
II_- &: \quad u_\beta - \frac{2}{\gamma - 1} c_\beta = -\frac{c}{c_v \gamma(\gamma - 1)} S_\beta + \lambda c u t_\beta \\
II_0 &: \quad S_\omega = 0
\end{aligned}
$$

Note that similar equations (with $\lambda \equiv 0$) appear in [CF48, page 199], but the last line there is completely wrong — it should be $\eta = S/c_v$. I have thoroughly checked that my equations are correct, and those of [CF48] can't be correct - the dimensions are inconsistent.

We usually write $\Gamma_+ = u + \frac{2}{\gamma-1}c$ and $\Gamma_- = u - \frac{2}{\gamma-1}c$ (these are called the *Riemann invariants* since as the characteristic equations show, in the case of planar one-dimensional flow ($\lambda \equiv 0$) and no shocks (S is constant), we get that Γ_+ is invariant along the I_- characteristic, and Γ_- is invariant along the I_+ characteristic).

3 The difference scheme for a continuous solution

In this section we will use the characteristic equations (14) and (15) to develop a difference scheme for numerical solution of the initial value QODA flow problem. We will currently assume the flow is continuous, i.e. there is no shock wave in the initial data, and no shock wave appears at a later time. It is possible to extend the present scheme to treat the tracking or formation of shock, but at the present version we chose not to do so.

Let's take a look at a given point in space-time **4** (see figure 3), and the three characteristics emerging from it backward in time (but note that the characteristics do not have to be straight lines, as they are schematically depicted), and one other point on

each of those characteristics: **1**,**2**, and **3**. We shall now write the three characteristic equations (15) in a finite difference form, by writing each equation along the respective characteristic, and multiplying II_+ by $d\alpha$ and so on. We get

$$\begin{aligned} C_+ : \quad & \Delta\Gamma_+ = \left[\frac{c}{c_v\gamma(\gamma-1)}\right]\Delta S - [\lambda cu]\,\Delta t \\ C_- : \quad & \Delta\Gamma_- = \left[-\frac{c}{c_v\gamma(\gamma-1)}\right]\Delta S + [\lambda cu]\,\Delta t \\ C_0 : \quad & \Delta S = 0 \end{aligned} \tag{16}$$

Or, more precisely,

$$\begin{aligned} C_+ : \quad & \Gamma_4^+ - \Gamma_1^+ = \left[\frac{c}{c_v\gamma(\gamma-1)}\right]_{4,1}(S_4 - S_1) - [\lambda cu]_{4,1}(t_4 - t_1) \\ C_- : \quad & \Gamma_4^- - \Gamma_2^- = \left[-\frac{c}{c_v\gamma(\gamma-1)}\right]_{4,2}(S_4 - S_2) + [\lambda cu]_{4,2}(t_4 - t_2) \\ C_0 : \quad & S_4 = S_3 \end{aligned} \tag{17}$$

Where $[\cdots]_{i,j}$ denotes some kind of *average* of the expression over the appropriate characteristic segment. We take this average, and not just the value at the other point, to make the scheme more accurate. Indeed, assuming monotonic change of the characteristic slope, the finite difference relations (17) become exact for some (unknown) convex combination of endpoint values. The arithmetic average $[\cdots]_{i,j}$ is an approximation to that unknown mean value.

4 Solving the difference scheme

The algorithm that will be shown here for solving the difference scheme is called *inverse marching*. Unlike other, more direct algorithms for solving the characteristic difference equations (such as following characteristics lines continuously), that algorithm has the benefit of employing a predetermined space-time lattice, i.e. the algorithm takes an array of flow variable values at one time, and calculates the new values at those points at the new time. That has two benefits: first it allows for easier computer implementation, since simple arrays can be used for storing values, and second, the grid points on which the solution is calculated do not get sparser, even if the characteristics diverge.

Let's assume the solution of the flow on the space grid at time t_n is known, and we want to find the flow variables at the next time t_{n+1} in the grid point **4** (see figure 4). We'll assume the characteristics going back in time intersect with the time t_n at the trace points **1**, **2** and **3**, which are all within the cell to the left of **4′** or to the right (this is important for stability, and as we shall later see, for interpolation). This condition can be easily satisfied if Δt is chosen sufficiently small.

At first glance, it might appear that we can just extend the three characteristics from **4**, then using the known values at the trace points **1**, **3**, and **2** (which can be found by somehow interpolating the values already known on the grid at time t_n) in the difference

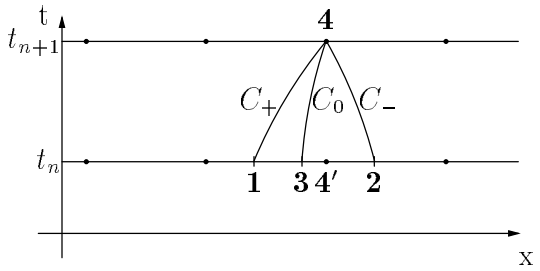


Figure 4: Solving the scheme

scheme (17) we readily calculate S_4, Γ_4^-, Γ_4^+, and from it the other flow variable values in **4**. However, there is one flaw in this scheme: how do we know what the characteristics look like, if we don't know the flow variable values at **4**? It appears we have a problem with an implicit situation, and the solution that comes to mind is to use an iterative technique for solving it.

We suggest the following iterative algorithm for finding the flow variables at **4**. First we shall *shortly* outline the algorithm, then explain in detail each step:

1. First guess for slopes $u \pm c$ and u of characteristics: As first guess we assume flow variables have not changed much since time t_n, and we'll take the values of $u \pm c$ and u in **4′** as the first guess.

2. By the last guess for the characteristics' slopes calculate an approximation for the location of **1**, **2** and **3**.

3. Interpolate the values of the flow variables in points **1**, **2** and **3**, by using the known values on the grid at time t_n.

4. Solve the difference scheme (17) to get a new approximation for Γ_+, Γ_-, and S at point **4**. Then from those values calculate the other flow variables.

5. If the current approximation is close enough to the previous approximation, then use it as the solution in point **4**, and stop the iterations.

6. Calculate new characteristics' slopes, by averaging current approximation of $u \pm c$ and u at the point **4** and at the respective trace points.

7. Go to step 2

The detailed algorithm follows:

4.1 First guess for characteristics

The characteristics emerging from **4** are not necessarily straight lines. However, if Δt is not too large, they can be approximated by straight lines, emerging from **4** with a certain slope (see figure 5). The appropriate slope is (by Lagrange's theorem) the same as the slope of the characteristic in a certain, unknown, midpoint. Since we don't know that point, we'll further approximate this slope by the average slope of the characteristic at the two endpoints **4** and **1**, or $\frac{1}{2}\left((u+c)_4 + (u+c)_1\right)$.

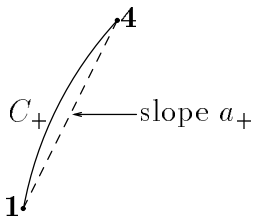


Figure 5: Approximate characteristic

However, at this stage we have no idea at all where the intersection points **1**, **2**, and **3** might be, so we'll try to further approximate the characteristics' slopes by using only the characteristics' slopes at point **4**, which are $(u \pm c)_4$ and u_4. However, although we know *where* the point **4** is, we have no idea about the values of u and c there. So we'll assume, as our first guess, that those values haven't changed much since the time t_n (which makes sense if the flow is smooth and Δt is not too large). So, finally, we can make our first guess for the values of u and c at **4**:

$$\begin{aligned} u_4 &:= u_{4'} \\ c_4 &:= c_{4'} \end{aligned}$$

And for the characteristic's slope:

$$\begin{aligned} a_+ &:= (u + c)_{4'} \\ a_- &:= (u - c)_{4'} \\ a_0 &:= u_{4'} \end{aligned}$$

Where the assignment ":=" means assignment as in a computer program, i.e. the value of the right hand side is assigned as the new approximation for the left hand side. On such assignments, variables in the right hand side mean the previous approximations of those variables, or known values (just like variables in computer languages).

4.2 Find intersections

If we assume the characteristics are approximately straight lines with the slopes $a_\pm$, a_0 (which were assigned before), then it is straightforward to find an approximation for the intersections of the characteristics at time t_n, i.e. the trace points **1**, **2** and **3** (see figure 4): if we write the x coordinate of trace point **i** as x_i, we assign

$$\begin{aligned} x_1 &:= x_4 - a_+(t_{n+1} - t_n) \\ x_2 &:= x_4 - a_-(t_{n+1} - t_n) \\ x_3 &:= x_4 - a_0(t_{n+1} - t_n) \end{aligned}$$

4.3 Interpolation

Now that we have a guess for the location of the points **1**, **2** and **3**, we want to interpolate the flow variables there, or actually just S and $\Gamma_\pm$ from which the other flow variables

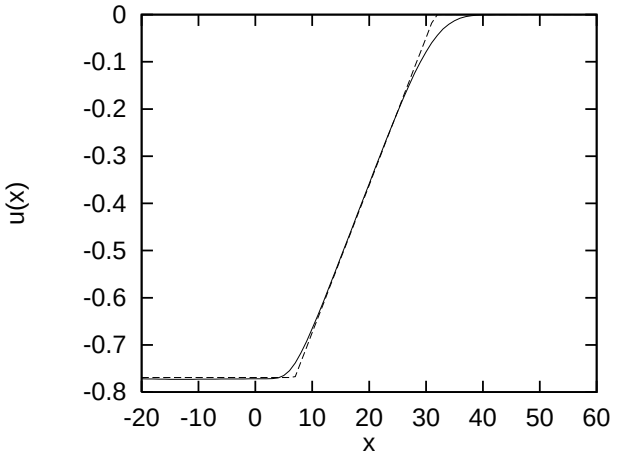


Figure 6: Algorithm with interpolation only

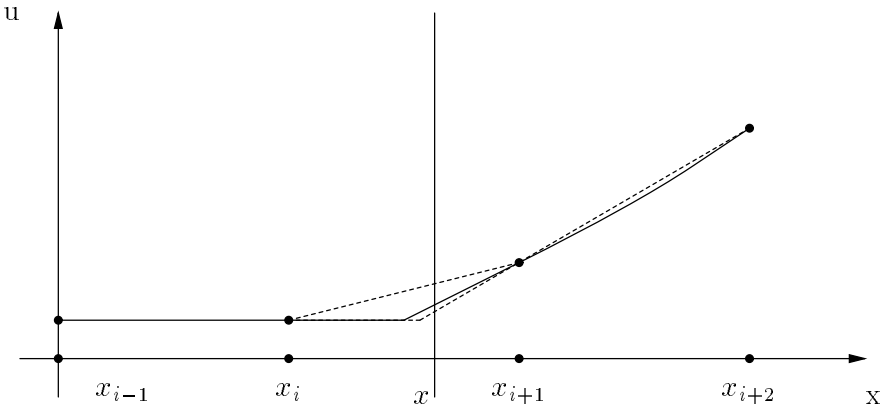


Figure 7: Interpolation and extrapolation

can be calculated. This seems very straightforward, but as it turns out, it is probably the most difficult point of the algorithm. At first, I tried using simple linear interpolation (using the two adjacent grid points). However, the results were poor. It turns out that if the flow has a jump in spatial gradient, as in the simple case of rarefaction fan (where all flow variables are constant ahead and behind the fan and vary smoothly through the fan) then the interpolation has very bad accuracy near the points of gradient jump — where the line between the two grid points is a bad approximation of the function itself. See figure 6 for an example of what happened when starting with a centered rarefaction wave some time after its creation, when it spanned 10 grid points (we couldn't take the time of the creation as initial data, since it is not continuous) and running 50 cycles – using linear interpolation as the only means of interpolation. In this sample case, the initial discontinuity giving rise to the centered rarefaction wave is at $(x,t) = (0,0)$, the grid step is $\Delta x = 1$, and the rarefaction wave is such that $(u+c)_{left} = 0.22(u+c)_{right}$. As can easily be seen, the result (solid line) is a severely smoothed out version of the accurate analytic solution (dashed), which has a sharp gradient discontinuity at two points.

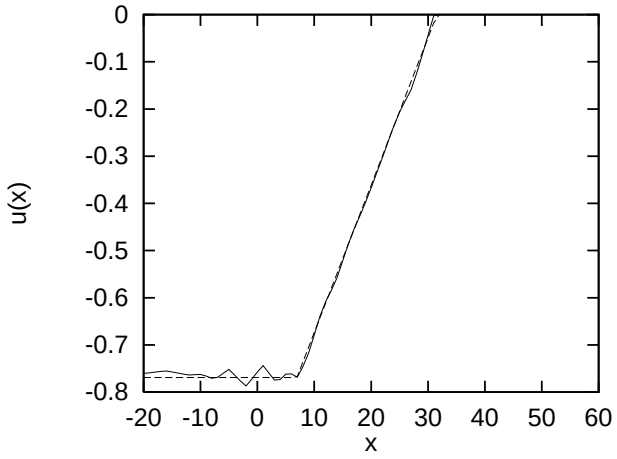


Figure 8: Algorithm with extrapolation only (flow variables P,u,S)

Another method of approximating the value of a function at a point not on the known grid is by *extrapolation* (see figure 7): instead of approximating the function at point x between x_i and x_{i+1} by a linear function as in interpolation, we approximate the function by linearly extrapolating the function values at x_{i+1} and x_{i+2} on the right, and the function values at x_{i-1} and x_i on the left, therefore the approximating function is not smooth but rather contains a gradient discontinuity at some point. By looking at figure 7, we see that in the case a gradient discontinuity in the interval (x_i, x_{i+1}) containing x, the extrapolating function (lower, non-smooth dashed line) is much closer to the real function (solid line) than the interpolating function (higher dashed line). Because it looks so promising, we may now try the algorithm with extrapolation as the sole means of interpolating values.

When trying the algorithm with extrapolation on the rarefaction fan example, the results were perfect. However, this is not a proof that extrapolation is good — on the contrary: as we said, we want to extrapolate S and $\Gamma_{\pm}$, which are all piecewise linear functions in this example. Obviously, for such functions extrapolation is indeed perfect, but such functions do not occur in more general cases (e.g., non-planar flow). To simulate a realistic situation, where the interpolated functions are not linear, but still use our example (to which an accurate analytic solution is known) we simply change the scheme to interpolate S, P (which is not linear), and u, and then calculate the other flow variables from them. Now we had the result we feared: although extrapolation is very good near the gradient discontinuities, it is less accurate than interpolation where the function is smooth. Moreover, the errors started growing and traveling in the characteristic direction, and the result was a poor, wavy, graph (see figure 8).

We have seen that interpolation does well in segments where the function is smooth, while extrapolation does well in the other segments, where the function derivative is discontinuous. The obvious choice now would be to use extrapolation only in those intervals, and interpolation everywhere else. However, this is not straightforward: we do not want to track or somehow calculate the location of the gradient discontinuity, since

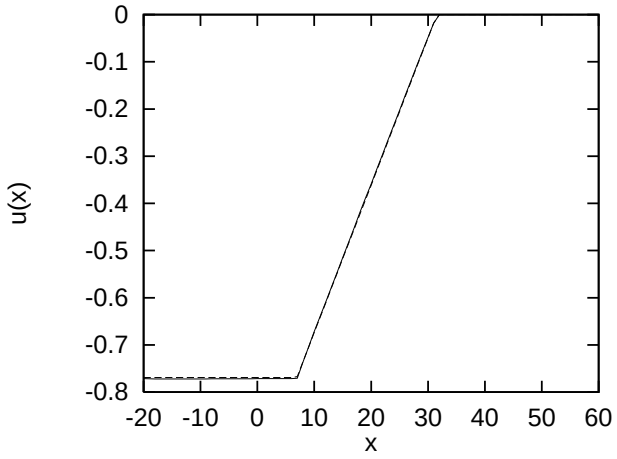


Figure 9: Algorithm with extrapolation and interpolation (flow variables P,u,S)

that will be possible only in special cases such as the rarefaction wave, where we know where the gradient discontinuity started, and how it moves. The only way we can try to guess where the gradient discontinuity is, is by using the fact that the absolute value of the numeric second derivative in such a point is a local maximum. Not all such maxima, however, are gradient discontinuities.

It is important for the algorithm choosing between interpolation and extrapolation to have two properties: first, extrapolation must be used, almost always, in intervals of real gradient discontinuities. Second, extrapolation should be used in as few as possible other intervals. After some experimentation, we have decided upon using the following algorithm: extrapolation is used in an interval if the numeric second derivative in it is maximum in the five adjacent points (five on the left and five on the right). Such an algorithm gave us very nice results for the rarefaction wave example (again with non-linear P being interpolated, otherwise the results will simply be totally accurate). See figure 9 for the graph of the result.

4.4 A new approximation in point 4

We want to use the difference scheme (17) to find a new approximation for S_4, Γ_4^+, Γ_4^-. We already have a current approximation for the terms $t_4 - t_1$, $t_4 - t_2$, S_3, Γ_1^+, Γ_2^-, S_2, and S_3, which we will use. We also have to consider the $[\lambda cu]_{4,1}$ and similar expressions: if we hadn't been using an iterative method, we would have had to use only the value at **1**. However in an iterative method we can improve accuracy by using the average of the previous approximation at **4** and the approximated value at **1**. We can now rewrite equation (17) as (note that we use $R = c_v(\gamma - 1)$):

$$\begin{aligned} S_4 &:= S_3 \\ \Gamma_4^+ &:= \Gamma_1^+ + \frac{c_4 + c_1}{2R\gamma}(S_3 - S_1) - \tfrac{1}{2}(\lambda_4 c_4 u_4 + \lambda_1 c_1 u_1)(t_{n+1} - t_n) \end{aligned}$$

$$\Gamma_4^- \quad := \quad \Gamma_1^- - \frac{c_4 + c_2}{2R\gamma}(S_3 - S_2) + \tfrac{1}{2}(\lambda_4 c_4 u_4 + \lambda_2 c_2 u_2)(t_{n+1} - t_n)$$

From S_4, Γ_4^+ and Γ_4^-, we can calculate the other flow variables' values at point **4**.

4.5 Check if we're done

If the current approximation is close enough to the previous one, we are done and the current approximation of flow variables in **4** are the values we wanted, i.e. the solution to the implicit difference scheme (17). There are many possibilities for checking if the approximation is close enough. What I did in my program, for example, is to define an error measure

$$E := \frac{|P_4 - P_{4p}|}{P_{4p}} + \frac{|S_4 - S_{4p}|}{S_{4p}} + \frac{|u_4 - u_{4p}|}{c_{4p}}$$

(where P_{4p} is the previous approximation for P_4), and stop the iterations if $E < \varepsilon$, where ε is some small number, such as $\varepsilon = 10^{-14}$ in my program.

4.6 New characteristics slope

As I explained in step 1, we approximate each characteristic by a line emerging from **4** at a certain slope, which we are trying to approximate now. That slope will be approximated (as explained there) by the average of the slopes of the characteristics at its two end points, **4**, and the other appropriate point. Therefore we have a new approximation for the slopes, by using the new approximations of u and c at **4**:

$$\begin{aligned} a_+ &:= \tfrac{1}{2}\left((u+c)_4 + (u+c)_1\right) \\ a_- &:= \tfrac{1}{2}\left((u-c)_4 + (u-c)_2\right) \\ a_0 &:= \tfrac{1}{2}\left(u_4 + u_3\right) \end{aligned}$$

4.7 Continue

Go to step 2. The new approximation for characteristics slope will be used (just as the first guess was used in the first iteration) to make another approximation.

5 Results

I have written a `fortran` program based on the aforementioned algorithm. After we were pleased with its solution of the simple planar rarefaction wave (see figure 9), giving a totally accurate solution (all significant digits match the analytic solution), we tried other, more interesting, problems.

5.1 Reflecting boundary

In this section we will make the centered rarefaction wave a little less trivial, by putting a reflecting boundary condition (rigid wall, i.e., $u = 0$) on the right boundary, causing

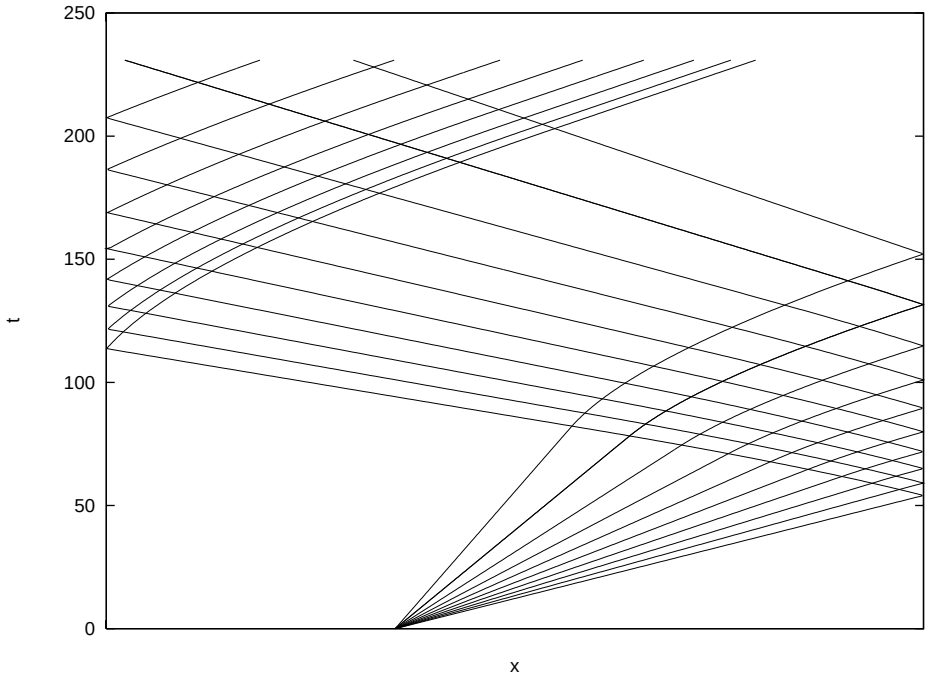


Figure 10: Characteristics in reflection example

the rarefaction wave that arrives at that boundary to be reflected, and the reflected wave interacts with the original wave — after which the wave is no longer a simple wave. See figure 10 for a sketch of the characteristics created by the program, to better understand the example. Note that the first reflection seen in the figure is a reflection of C_+ characteristics to C_- characteristics by the reflecting boundary. However the second (left) reflection seen in the figure is not a reflection (the left boundary is not reflecting) but rather new C_+ characteristics that enter from that non-reflecting boundary (these characteristics are parallel far enough to the left outside the picture, but are bent by the C_- characteristics. These characteristics are not meaningful for understanding the reflection at reflecting boundary).

One of the many interesting things to check for assessing the accuracy of the scheme is the pressure history $P(t)$ on the right boundary (it should remain constant until the wave arrives at that boundary, then vary for some time, and when the wave passes, remain a new constant, which can readily be calculated analytically, so it is easy to compare with our results). We ran the problem both on GRP and on our inverse-marching QODA algorithm. The results can be seen in figure 11. The limit $\lim_{t\to\infty} P(t)$ was found to be the same in both numerical schemes, and turned out to be equal in all significant digits to the analytic limit that was calculated. However, although the GRP (dashed) and QODA (solid) lines are very close, they are not exactly equal. We wondered why there is such a difference, and does that difference disappear as both methods' number of grid points tends to infinity (i.e. $\Delta x \to 0$). In figure 12 we see that indeed both methods seem to converge to the same limit as the number of grid points is increased (the solid lines are QODA's results, with $N = 100$, $N = 200$, $N = 400$ from left to right, and the dashed

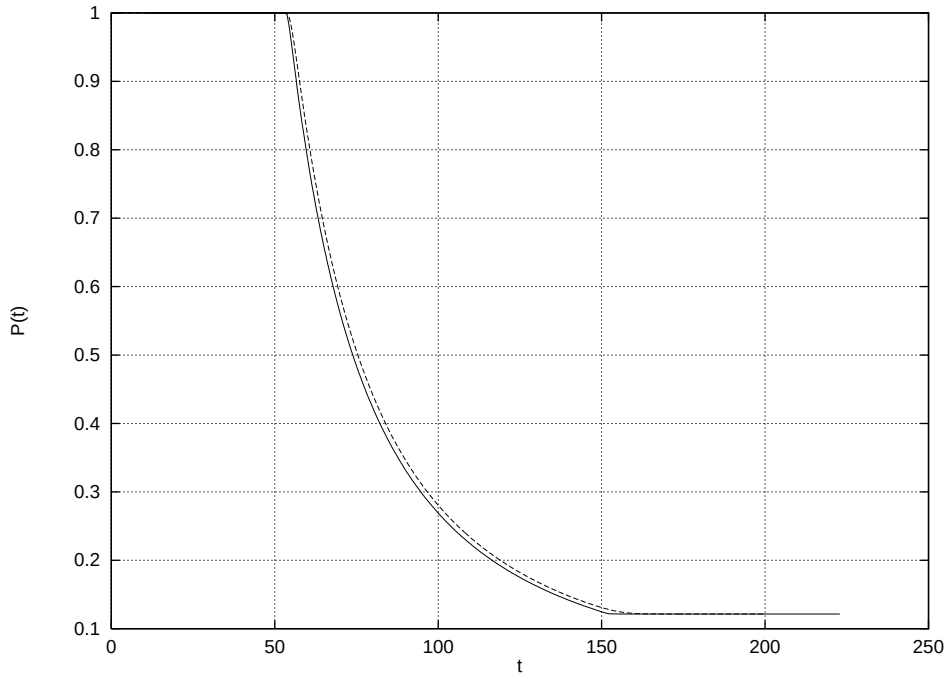


Figure 11: P(t) on boundary in reflection example

lines are GRP's results, with $N = 100$, $N = 200$, $N = 400$, $N = 800$, this time from right to left). We can also see from figure 12 that QODA's results tend to be much shaper (i.e. preserve derivative discontinuities) than GRP's, but not necessary more accurate, since after all the difference scheme provides only an approximation.

5.2 Area reduction

In this section we try to replicate the result in [IF86, figure 4], where we initially have a centered rarefaction wave, in a tube with area reduction of $\mathcal{A}_1/\mathcal{A}_0 = 0.25$. More precisely, we have again a centered rarefaction wave, but this time with a varying cross-section duct, with variance as shown on the title page (for the exact function, see figure 13). For a more precise formulation of the parameters used, see table 1 (these parameters were chosen to give results similar to those in [IF86, figure 4]).

We ran the program on that problem, but since a centered rarefaction wave is initially discontinuous, we instead gave as initial condition the continuous wave, at the time the rarefaction wave spanned 10 cells. That initial condition is equivalent to a centered rarefaction wave, centered at $t = 0$ and $x = 0$. We plotted the pressure and velocity graphs at increasing times (see figures 14 and 15), to get a plot which gives an impression of traveling waves. The graphs turn out to be very similar to those in [IF86, figure 4], however they seem less noisy (the GRP method used there, while being a much better method than the RCM method also used there, still gives output which is a little wavy and smeared, unlike the inverse marching QODA method's almost straight lines and derivative discontinuities).

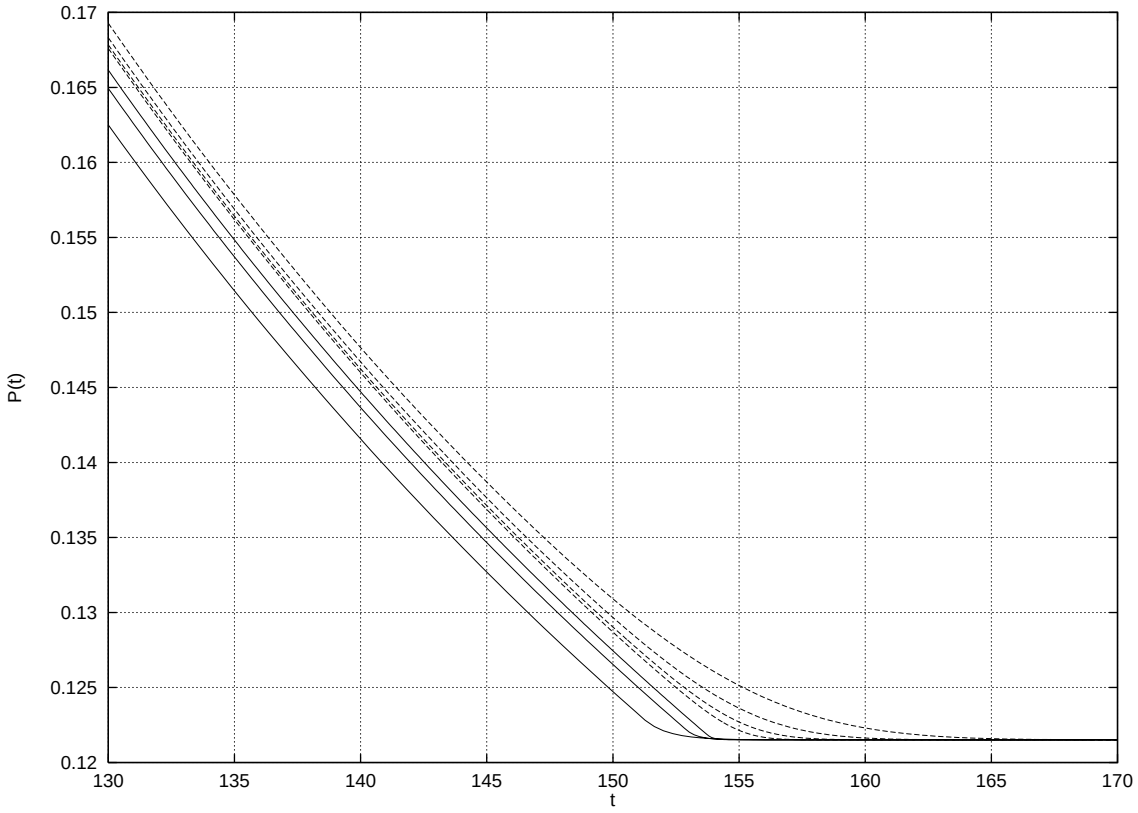
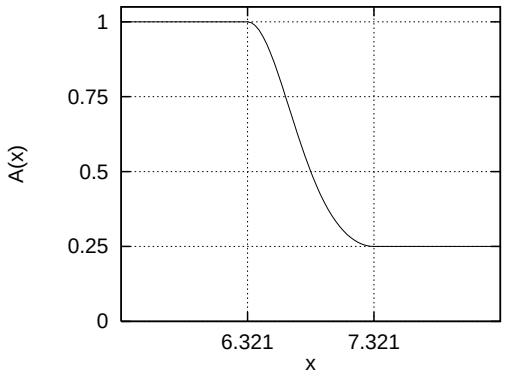


Figure 12: P(t) on boundary in reflection example

parameter	*value*
Number of cells	220
Cell Δx	0.1
x range	-5...17
γ	1.4
P_1/P_0	0.8
u_0	0
ρ_0	1
$\mathcal{A}(x)$	*see figure 13*
boundary	non-reflecting boundary condition

Table 1: Parameters for area reduction example



$$\mathcal{A}(x) = \begin{cases} 1 & \text{if } x < 6.321 \\ 0.25 & \text{if } x > 7.321 \\ \exp\left(\frac{1}{2}\log(\frac{1}{4})(1-\cos\pi(x-6.321))\right) & \text{otherwise} \end{cases}$$

Figure 13: $\mathcal{A}(x)$ for area reduction example

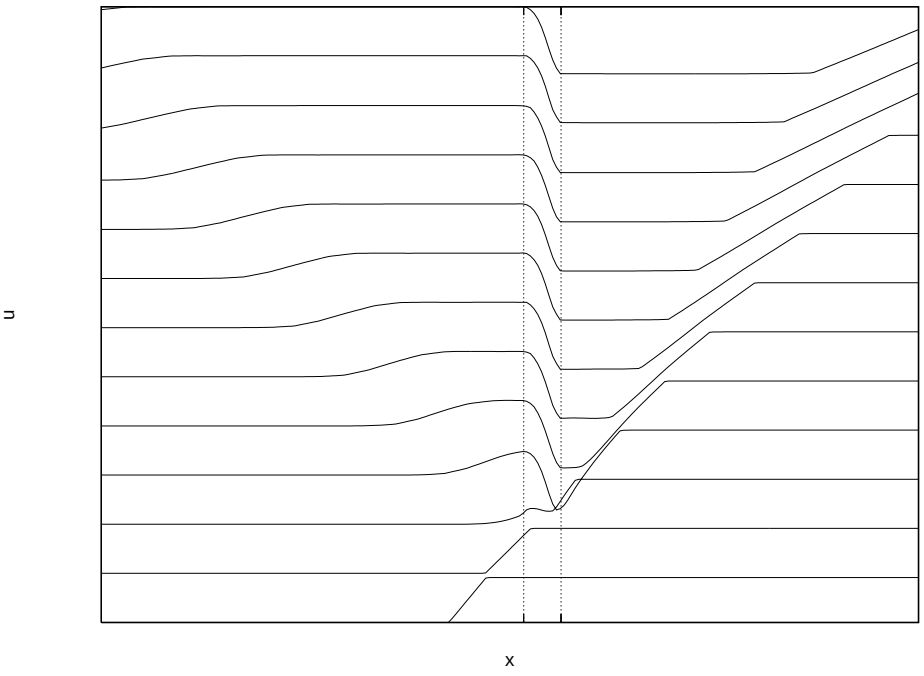


Figure 14: U in area reduction example

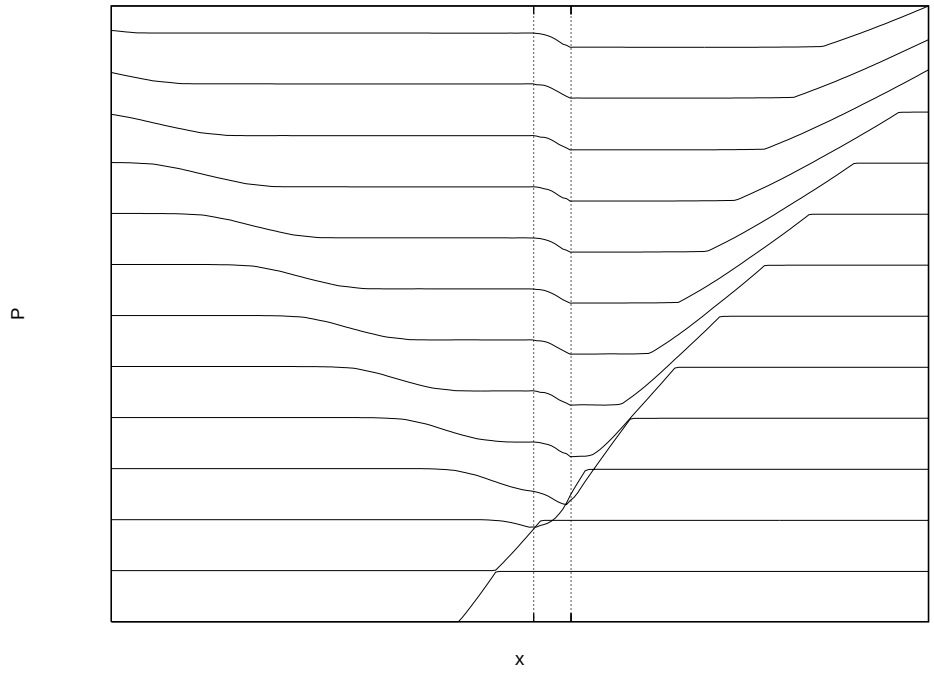


Figure 15: P in area reduction example

6 Acknowledgment

None of the above algorithms and results would have been possible, without the help of Prof. J. Falcovitz, who guided me throughout the project.

Illustrations for this report have been prepared with Collin Kelley and Thomas Williams' `gnuplot 3.5`, and by `gpic`, James Clark's implementation of Brian W. Kernighan's `pic`. `Gnuplot` was also a great help for visualizing data while writing the program.

This report was typeset with Donald Knuth's TEX 3.1415, with Leslie Lamport's LATEX macro package.

References

[CF48] R. Courant and K. O. Friedrichs. *Supersonic Flow and Shock Waves*, volume 1 of *Pure And Applied Mathematics*. Interscience Publishers, New York, London, August 1948.

[IF86] Ozer Igra and Joseph Falcovitz. Numerical solution to rarefaction or shock wave/duct area-change interaction. *AIAA Journal*, 24(8):1390, August 1986.