4A2 Computational Fluid Dynamics - Final Report

CCN: 5690G

1 Introduction

The Lax method used in the scheme developed for the interim report has been further analysed and improvements have been made. These developmental improvements have identified 3 major ways the scheme can be improved upon; the accuracy, the stability and the speed of convergence of the scheme. The effects of all of the improvements have been analysed in detail. The final code which encompassed all of the improvements was then tested using a range of test cases designed to investigate the solution found under a range of conditions.

2 Investigating The Effect Of The CFL Number And The Smoothing Factor

The Courant–Friedrichs–Lewy (CFL) number is the dimensionless ratio defined in Equation 1. It is used in forward time marching numerical schemes, such as the one developed, in order to set the maximum time step for a cell. To maintain stability and accuracy of this kind of scheme, the CFL number should be less than or equal to unity. [1] The smoothing in this scheme allows convergence for CFL numbers above 1.0 but these are avoided for the basic scheme to ensure the accuracy of the solution.

$$CFL = \frac{c\Delta t}{\Delta x} \tag{1}$$

Where, c is the speed of sound, Δt is the time step and Δx is the spatial step.

The Lax method used in the scheme developed is inherently unstable. To achieve stability of the solution, the flow variables at all the grid points are smoothed at every time step. However, this action introduces artificial viscosity in the solution and hence it is desirable to keep the smoothing factor (SF) as small as possible whilst maintaining the stability and speed of convergence.

The effect of varying both the CFL number and the SF on the number of iterations taken to converge was investigated for the basic Lax method code developed using test case 0. The results of this are tabulated in Table 1. Since the computational complexity is independent of the CFL number or SF, the number of iterations to convergence is directly proportional to the run time required to reach convergence.

		Smoothing Factor								
		0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
CFL Number	1.00	NA	NA	NA	NA	1200	1010	1015	1025	1040
	0.90	NA	NA	NA	1620	1230	1115	1125	1140	1155
	0.80	NA	NA	7605	1755	1370	1255	1265	1280	1300
	0.70	NA	NA	2595	1865	1515	1435	1445	1465	1485
	0.60	NA	NA	2855	2020	1680	1670	1690	1710	1730
	0.50	NA	5015	3130	2320	2000	2005	2025	2050	2080
	0.40	NA	5620	3665	2775	2480	1770	2530	2565	2600
	0.30	NA	6785	4710	3540	3300	2345	3375	3415	3470
	0.20	NA	9395	6570	5050	4950	3495	5065	5125	5205
	0.10	NA	NA	NA	NA	9910	6930	NA	NA	NA

Table 1: Number of iterations taken to converge with varying CFL number and SF. *Note: 'NA' shows solution did not converge within 10,000 iterations.*

Figure 1 shows a contour plot of the data in Table 1. The values of CFL number and SF for which the solution did not converge are shown by the white space on the plot. The plot highlights some important features in the rate of convergence as CFL and SF are varied. Firstly, a SF below 0.2 results in instability at all values of CFL. Additionally, a CFL number below 0.2 is unstable except at moderate SF values (between 0.4 - 0.7).

In general, the plot shows that as the CFL number is increased the number of iterations to convergence decreases for a constant smoothing factor. This was expected because a larger CFL number means the time step at each iteration is larger and hence the scheme reaches a steady solution in fewer iterations. However, as the CFL number is increased beyond 0.2, the minimum stable SF also increases. This is because as the time step increases the damping of instabilities is reduced and hence a greater SF is required to maintain stability.

The plot also shows that at all CFL numbers an optimum SF exists at approximately 0.6. This is a particularly interesting result as it shows the rate of convergence can be maximised at all CFL numbers by using a SF of 0.6. Further investigation would be needed to show that this is the case for all test cases. Using a smaller SF will results in a more accurate solution because the artificial viscosity introduced is reduced. Hence, a compromise needs to be made between the rate of convergence and the accuracy of the scheme.

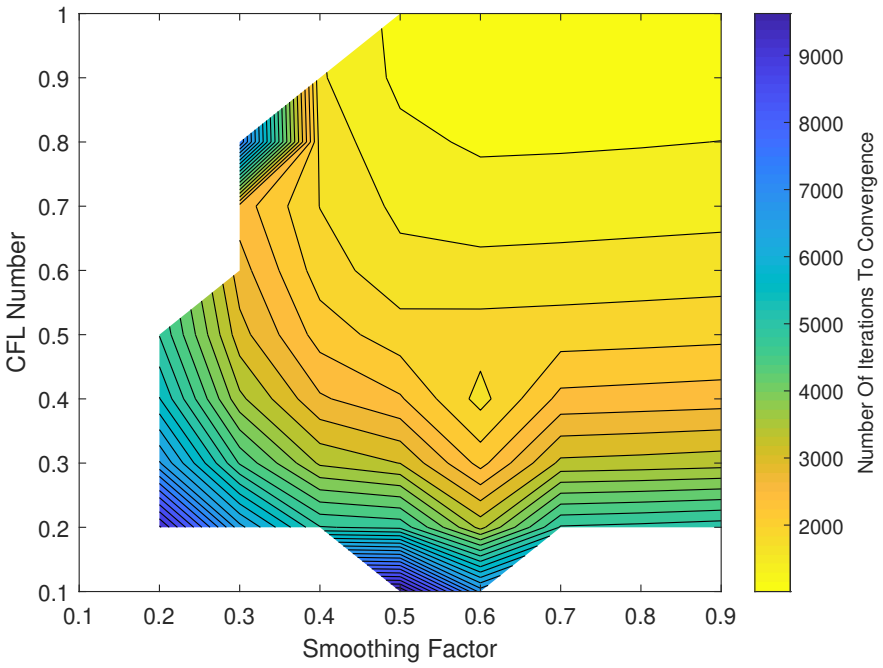


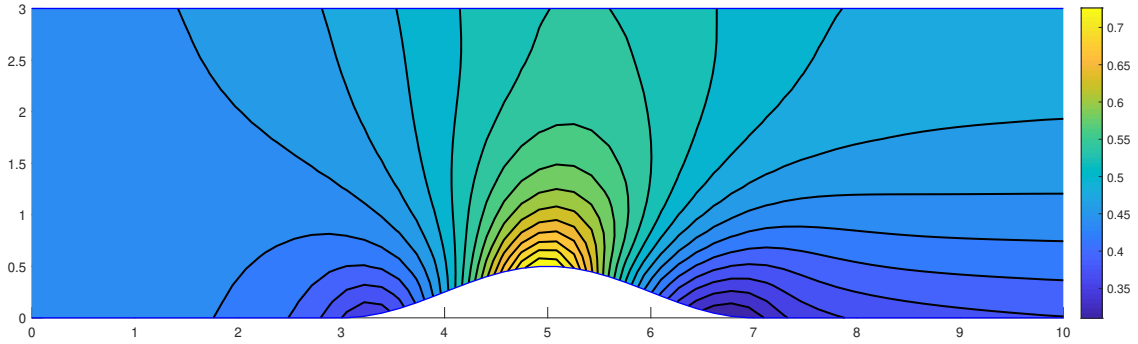
Figure 1: Contour plot of the number of iterations to convergence with varying CFL and SF.

3 Improvements To The Scheme

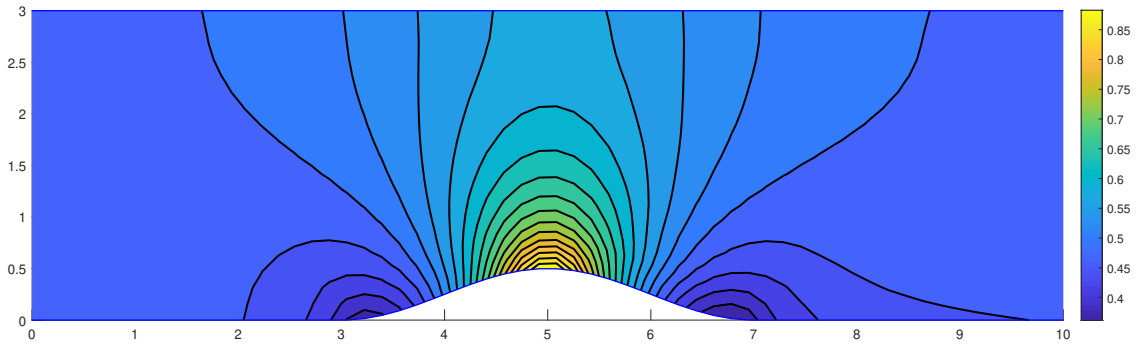
3.1 Developments To The Accuracy

Smoothing is required in the numerical scheme to maintain stability. However, as discussed in the interim report, smoothing results in artificial viscosity being present in the solution. To improve the accuracy of the numerical scheme, deferred correction factors were added. This method uses a gradually introduced correction factor to correct the change in a flow variable for each cell at every time step. This is done to effectively reduce the smoothing factor as the scheme converges without compromising on stability. Therefore, gradually introducing correction factors improves the accuracy for the scheme as less smoothing and hence less artificial viscosity is introduced.

The improved accuracy due to introducing deferred correction factors is shown clearly in Figure 2. Figure 2a shows the contours of Mach number for the original basic code for test case 0 (CFL = 0.5, SF = 0.5) and Figure 2b shows the contours for the code with the deferred correction factors. The analytical solution of this test case for an inviscid flow would be symmetrical either side of the bump. Figure 2 shows that the addition of the deferred correction factors makes the solution more symmetric than for the basic code and hence the accuracy of the scheme has been significantly improved. Additionally, the velocity gradient at the contraction is better preserved because of the reduced smoothing.



(a) Basic code.



(b) With deferred correction factors.

Figure 2: Mach contours for test case 0 (CFL=0.5, SF=0.5)

The improvements to the accuracy of the scheme would not be worthwhile if the additional code results in significantly reduced stability or rate of convergence of the scheme. The effect on stability and rate of convergence is tabulated in Table 2. This shows that the number of iterations to achieve convergence and the run time have both increased (for CFL=0.5 and SF=0.5). The reason for this is because the effective smoothing factor has been reduced by the deferred correction factor method and hence the scheme takes slightly more iterations to reach a stable solution. Additionally, the method adds to the computational complexity of the calculation and hence the run time per iteration is increased. Therefore, other improvements to the scheme are required to ensure that a reasonable run time is maintained.

Table 2 also shows that the minimum stable smoothing factor is marginally increased. The CFL number for evaluating this was chosen to be 0.99 because this is just below the theoretical maximum of 1.0 for the Lax method and hence represents the worst case scenario for stability. The increase in the minimum stable smoothing factor shows the stability of the scheme is reduced by using the deferred correction factor method. This was expected because the method reduces the effective smoothing factor. Overall, the slightly reduced stability and rate of convergence of the scheme is offset by the significantly improved accuracy of the scheme.

	Base Code	Deferred Correction Factors
Iterations to convergence (CFL=0.50, SF=0.50)	2000	2325
Run Time (ms) (CFL=0.50, SF=0.50)	537	625
Run time per iteration (ms) (CFL=0.50, SF=0.50)	0.2685	0.2688
Min Stable SF (CFL=0.99)	0.424	0.433

Table 2: Convergence and run time data comparison between the basic code and code with deferred correction factors.

The suggested correction factor (F_{corr}) in the handout is 0.9. This value was varied (from 0.80 to 0.98) in order evaluate the effect on the stability and rate of convergence and the results are tabulated in Table 3. The results show that as F_{corr} is increased the number of iterations to convergence is also increased. This is expected because an increased correction factor reduces the effective smoothing and hence the scheme needs more iterations to reach a stable solution. The results also show that as the correction factor is increased the stability of the scheme is reduced (shown by the increasing minimum stable SF). This is also explained by the reduced effective smoothing resulting in reduced stability because fluctuations are not damped out as effectively.

F_{corr}	Number of iterations to convergence (CFL=0.5, SF=0.5)	Minimum Stable SF (CFL=0.99)
0.80	2110	0.431
0.82	2295	0.432
0.84	2305	0.432
0.86	2310	0.432
0.88	2320	0.432
0.90	2325	0.433
0.92	2335	0.433
0.94	2345	0.434
0.96	2355	0.435
0.98	2585	0.436

Table 3: Convergence and stability data as the correction factor is varied.

3.2 Developments To The Stability

Two methods were used to improve the stability of the numerical scheme. The first was the use of a 4-step Runge-Kutta scheme. This method breaks down each time step into 4 sub-steps. In each sub-step, the changes in the primary variables are calculated and added to the values at the start of the time step. Breaking down each time step greatly increases the stability of the scheme but at the cost of a longer run-time. This is shown in Table 4 - the minimum stable SF is dramatically reduced by the addition of this scheme but the total run time is more than doubled. Additionally using the Runge-Kutta scheme allows the CFL number to be increased beyond 1 allowing the code to converge in fewer iterations.

The second method used to improve the stability of the scheme was the addition of a second order term in the time derivative. Using the recommended value for the second order factor (facsec) of 0.5, the scheme becomes second order accurate in time. Table 4 shows that this significantly reduces the minimum stable SF and hence has improved the schemes stability. The table also shows that this addition allows the scheme to converge in fewer iterations however this improvement is offset by the increased numerical complexity which results in a slightly longer run time compared to the basic code. In summary, the addition of a second order term improves the stability of the method at the cost of a small increase in the run time.

	Basic Code	Runge-Kutta	2nd Order Time
Iterations to convergence (CFL=0.5, SF=0.5)	2000	1985	1985
Run time (ms) (CFL=0.5, SF=0.5)	537	1313	546
Run time per iterations (ms) (CFL=0.5, SF=0.5)	0.2685	0.6615	0.2751
Minimum stable SF (CFL=0.99)	0.424	0.007	0.108
Maximum stable CFL (SF=0.50)	1.07	1.90	1.66

Table 4: Effect of code additions on rate of convergence and stability.

3.2.1 Runge-Kutta Scheme

The effect of varying the number of sub-steps for the Runge-Kutta scheme was further investigated and the results are tabulated in Table 5. These results show that increasing the number of sub-steps increases the stability of the scheme (shown by the reduced minimum stable smoothing factor). However, increasing the number of sub-steps also increases the run time of the scheme because the computational complexity increases. Diminishing gains in stability are seen as the number of sub-steps increases. Therefore, using more than 3 or 4 sub-steps is not worthwhile because the increased run time outweighs the minimal stability gains. In the final version of the code (Section 4) a Runge-Kutta scheme with 4 sub-steps is used as this shows a good compromise between stability and run time.

Runge-Kutta steps	Iterations to convergence (CFL=0.99, SF=0.5)	Run time (ms) (CFL=0.99, SF=0.5)	Minimum Stable SF (CFL=0.99)
1	1170	380	0.424
2	1005	528	0.031
3	1005	697	0.007
4	1005	863	0.007
5	1005	1020	0.006
6	1005	1218	0.006
7	1005	1344	0.005

Table 5: The effect of varying the number of Runge-Kutta sub-steps on the rate of convergence and stability.

3.2.2 Second Order Term In Time Derivative

Increasing the second order factor (facsec) beyond 0.5 adds more of the second order term than is needed to achieve second order accuracy. However, doing this can further improve the stability of the scheme. The effect of increasing the second order factor was investigated and the results are shown in Figure 3 and Table 6.

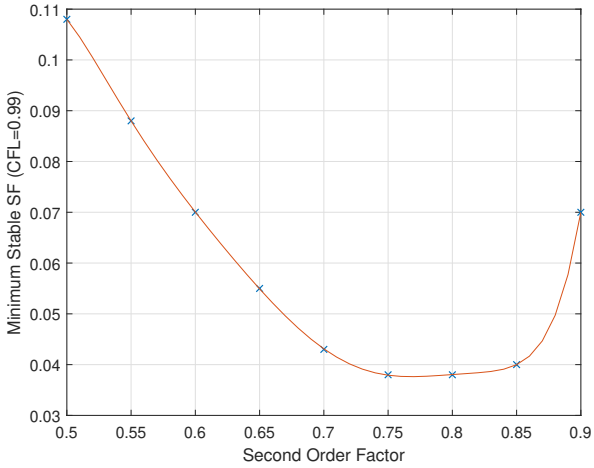


Figure 3: Variation of the minimum stable SF with varying second order factor (facsec).

Second Order Factor (facsec)	Min Stable SF (CFL=0.99)
0.50	0.108
0.55	0.088
0.60	0.070
0.65	0.055
0.70	0.043
0.75	0.038
0.80	0.038
0.85	0.040
0.90	0.070

Table 6: Variation of the minimum stable SF with varying second order factor (facsec).

Figure 3 shows that the minimum stable SF decreases as the second order factor increases up until the second order factor exceeds 0.75. Beyond this value the minimum stable SF increases as the second order factor increases. This shows that the stability of the scheme using this test case and numerical parameters, can be improved by using a greater second order factor. For the combined scheme discussed in Section 4, the recommended second order factor of 0.5 was used. However, the data here suggests that there are possible stability gains that can be achieved from using a greater second order factor. Further work would need to be carried out using different flow parameters and test cases to confirm this and to find the optimum second order factor for all test cases.

The effect of varying the second order factor on the maximum stable CFL number was also investigated and the results are shown in Figure 4 and Table 7. These results show that the maximum stable CFL number decreases linearly as the second order factor is increased above 0.5 for a constant SF. A larger CFL number allows the scheme to converge in fewer iterations. Therefore, further investigation is needed in order to find the optimum second order factor that optimises the rate of convergence and the stability of the scheme.

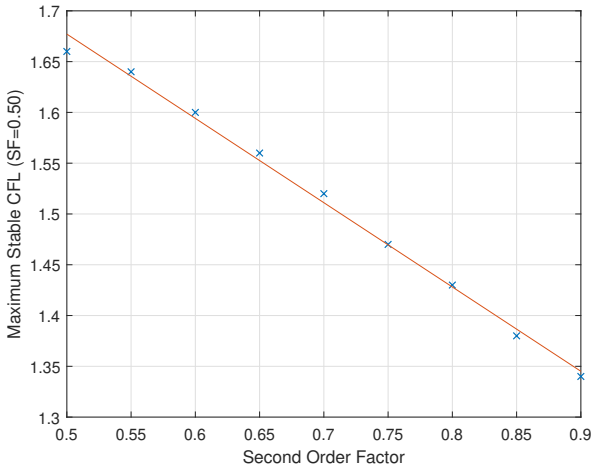


Figure 4: Variation of the maximum stable CFL with varying second order factor (facsec).

Second Order Factor (facsec)	Max Stable CFL (SF=0.50)
0.50	1.66
0.55	1.64
0.60	1.60
0.65	1.56
0.70	1.52
0.75	1.47
0.80	1.43
0.85	1.38
0.90	1.34

Table 7: Variation of the maximum stable CFL with varying second order factor (facsec).

3.3 Developments To The Speed

To improve the speed of convergence of the numerical scheme, a step to smooth the residuals was added at each iteration. By smoothing out fluctuations in the residuals any instabilities are damped out more rapidly without any detriment to the accuracy of the scheme. The effect on the rate of convergence and stability of implementing this scheme are tabulated in Table 8. The residual smoothing factor (SF_{res}) is introduced. Applying smoothing to the residuals once is compared to the semi-implicit method of applying smoothing 5 times with an equivalent overall residual smoothing factor.

	Basic Code	Residual smoothing called once $SF_{Res} = 0.5$	Residual smoothing called 5 times $SF_{Res} = 0.1$
Iterations to convergence (CFL=0.5, SF=0.5)	2000	2005	2000
Run time (ms) (CFL=0.5, SF=0.5)	537	578	778
Run time per iteration (ms) (CFL=0.5, SF=0.5)	0.2685	0.2883	0.3890
Min stable SF (CFL=0.99)	0.424	0.441	0.450

Table 8: Effect of residual averaging on the rate of convergence and stability.

Table 8 shows that the residual smoothing method actually reduces the rate of convergence compared to the basic code. This is because additional computational complexity is introduced with this method and hence the run time increases. Interestingly, the stability of the scheme is also slightly decreased - shown by the increased minimum stable smoothing factor. The reason for this could be that during the initial iterations, smoothing the residuals introduces small instabilities. This is a relatively small reduction in the stability of the scheme and hence is not a concern.

As noted by the handout, the benefits of adding residual averaging varies for different methods. It was suggested that the method offers little benefit for the Lax method used in the basic code but could offer greater benefits with the Runge-Kutta method. This was investigated using test case 0 and the results are tabulated in Table 9.

	Runge-Kutta	Residual smoothing called once $SF_{Res} = 0.5$	Residual smoothing called 5 times $SF_{Res} = 0.1$
Iterations to convergence (CFL=0.5, SF=0.5)	1985	1990	1990
Run time (ms) (CFL=0.5, SF=0.5)	1313	1498	2015
Run time per iteration (ms) (CFL=0.5, SF=0.5)	0.6615	0.7528	1.0126
Min stable SF (CFL=0.99)	0.007	0.008	0.007

Table 9: Effect of residual averaging on the rate of convergence and stability for Runge-Kutta scheme.

Table 9 shows similar results to Table 8. The rate of convergence is reduced by the addition of residual averaging. The stability is also slightly decreased for the case where residual averaging is carried out once per time step. These results are slightly concerning because they show that this method has not improved the speed of the convergence. To further investigate this, all of the other additions discussed previously were implemented and the effect of residual averaging analysed. The results of this are shown in Table 10.

The results show that the residual averaging step increases the maximum stable CFL number for a constant SF. This means the code converges in fewer iterations. When residual smoothing is called once per time step, this results in a shorter run time to convergence showing that residual smoothing has improved the speed of the method. When the residual smoothing is called 5 times per time step, the iterations required to

reach convergence are reduced compared to without the residual averaging step (using the maximum stable CFL number). However, the increased computational complexity results in a longer overall run time. On balance, the residual averaging does improve the rate of convergence of the scheme and is a worthwhile addition when carried out once per time step.

	Code with all other additions	Residual smoothing called once $SF_{Res} = 0.5$	Residual smoothing called 5 times $SF_{Res} = 0.1$
Iterations to convergence (CFL=0.5, SF=0.5)	2565	2665	1660
Run time (ms) (CFL=0.5, SF=0.5)	1843	2031	2857
Run time per iteration (ms) (CFL=0.5, SF=0.5)	0.7185	0.7621	1.7211
Min stable SF (CFL=0.99)	0.007	0.009	0.006
Max stable CFL (SF=0.2)	1.78	2.23	2.20
Iterations to converge (CFL=max, SF=0.2)	925	760	750
Run time (ms) (CFL=max, SF=0.2)	877	859	1084

Table 10: Effect of residual smoothing on code with all other additions implemented.

The effect of varying the residual smoothing factor on the maximum stable CFL number and the rate of convergence was investigated. The results of this are tabulated in Table 11. The results show that as the residual smoothing factor is increased, the maximum stable CFL number increases. A greater CFL number means that the time step at each iteration is greater and hence the solution should converge in fewer iterations and in a shorter amount of time. This is supported by the results obtained. When solving the test cases described in Section 4, the recommended residual smoothing factor of 0.5 (from handout) was used. However, these results suggest that by optimising this variable, the rate of convergence of the method could be significantly improved.

SF_{Res}	Max stable CFL number (SF=0.20)	Iterations to convergence (CFL=max, SF=0.20)	Run time (ms) (CFL=max, SF=0.20)
0.1	1.88	875	930
0.2	1.97	835	928
0.3	2.06	800	892
0.4	2.17	785	880
0.5	2.23	760	872
0.6	2.28	745	850
0.7	2.4	725	830
0.8	2.46	710	805
0.9	2.53	710	795

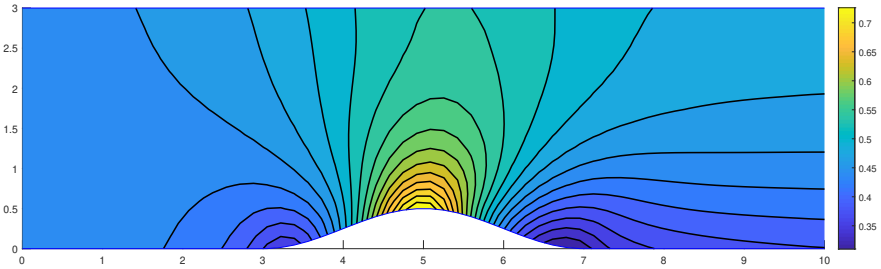
Table 11: Effect of residual smoothing factor on the maximum stable CFL number and the rate of convergence.

4 Test Cases

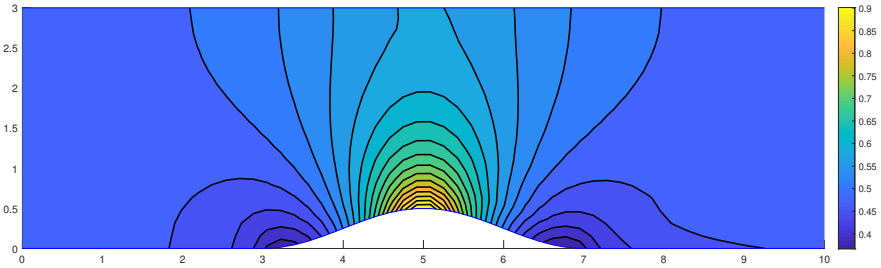
All the code additions described in Section 3 were implemented and the scheme was tested on a variety of test cases to evaluate its effectiveness.

4.1 Test Case 0

Test case 0 is the most basic case and was used to evaluate the effect of the code enhancements in Section 3. This test simulates a subsonic flow through a contraction in a straight pipe. Figure 5 shows the Mach contour plots of test case 0 for (a) the original basic code and (b) the improved code. The analytical solution for an inviscid flow is symmetric about the contraction. These two plots show that the improved code is significantly closer to the analytical solution and hence more accurate. Additionally, the velocity gradient at the contraction point is far better preserved in the improved code. Interestingly the solution found by the improved code is slightly less accurate than for the basic code with the addition of deferred correction factors (shown in Figure 2). Small inaccuracies are clearly introduced by some of the other additions to the code. However, overall the accuracy of the solution is still significantly improved.



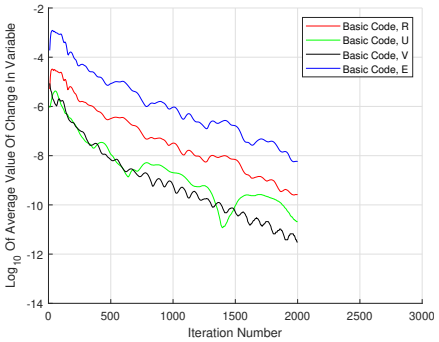
(a) Basic code.



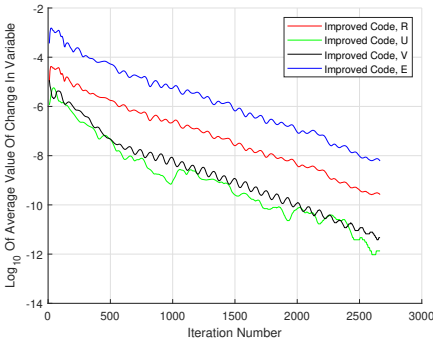
(b) Improved code.

Figure 5: Mach number contour plots of test case 0.

Figure 6 shows the convergence of the flow variable for (a) the basic code and (b) the improved code (CFL=0.5, SF=0.5). This shows that although the improved code requires more iterations for this test case and input parameters, the convergence of all the variables is more linear and hence more stable. This improved stability is what allows the improved code to converge using higher CFL numbers and lower SF and hence converge on a more accurate solution more rapidly.



(a) Basic code.



(b) Improved code.

Figure 6: Convergence of the flow variables (CFL=0.5, SF=0.5). *R is the density; U is the velocity in the x-direction; V is the velocity in the y-direction and E is the enthalpy.*

4.2 Test Case 1

Test case 1 simulates a subsonic flow around a bend. The pressure gradient across the bend results in lower temperatures on the inside of the bend (via ideal gas equation). This means that the Mach number on the inside of the bend is greater than on the outside of the bend. As mentioned in the interim report, the stagnation pressure loss provides a good visualisation of where the solution deviates from the isentropic analytical solution. The contour plots of stagnation pressure for the basic and improved code are shown in Figure 8. This shows that the improved code has significantly reduced the losses due to the addition of deferred correction factors.

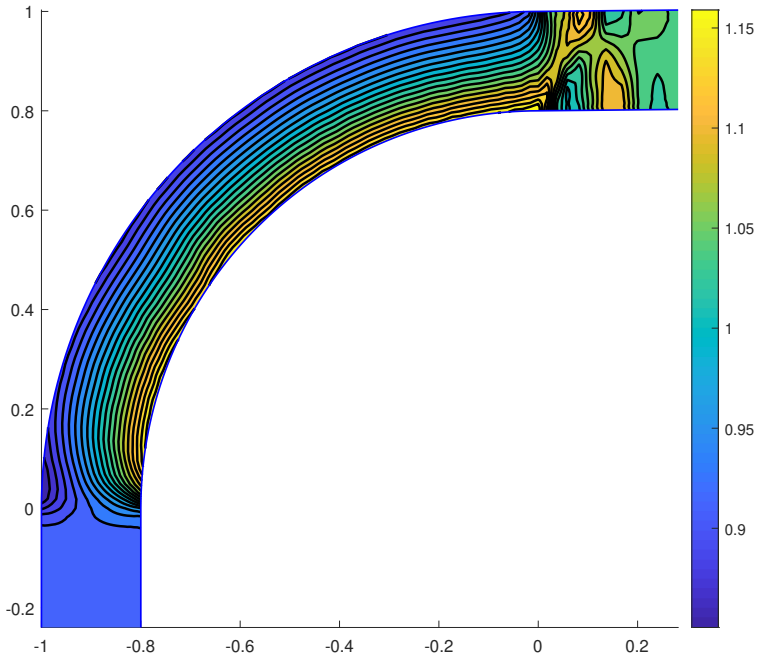
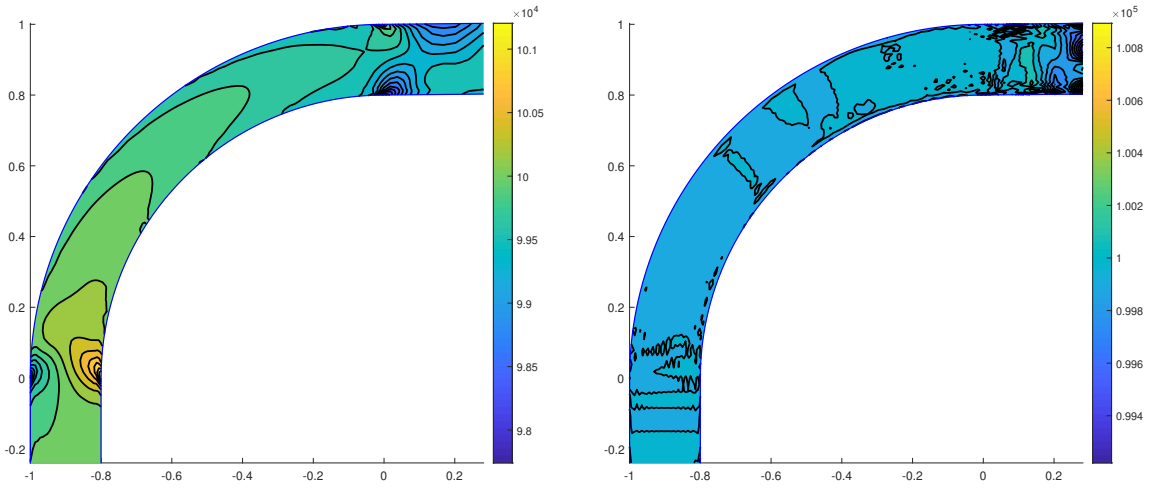


Figure 7: Mach number contour plot of test case 1.



(a) Basic code.

(b) Improved code.

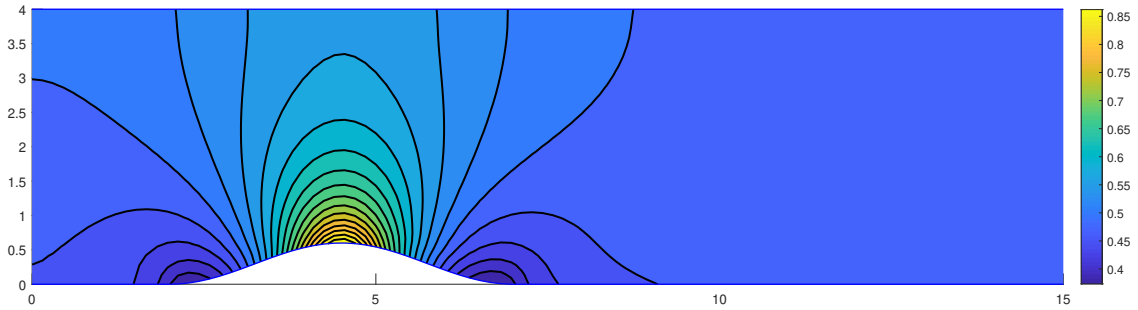
Figure 8: Stagnation pressure ($\times 10^5$Pa) plots of test case 1. (CFL=0.5, SF=0.5)

4.3 Test Case 2

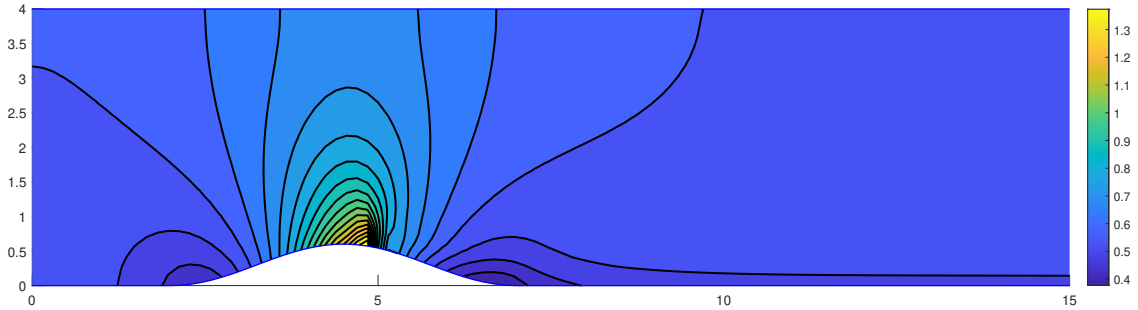
Test case 2 is similar to test case 0 but the downstream region has been extended. Figure 9 shows contour plots of the Mach number of the solution with different exit pressures. Using a downstream pressure of 85000Pa (Figure 9a), the solution is subsonic throughout. Similar to test case 0, the analytical solution for an isentropic flow would by symmetrical about the contraction in the pipe. This is well achieved in the computational solution.

Figure 9b shows the solution found when the downstream pressure is set to 80000Pa. The contour plot shows that the flow turns supersonic at the contraction but returns to subsonic further downstream. There is a high density of contours a short distance downstream of the throat of the contraction. This appears to be irregular and could indicate the initial formation of a shock wave in the flow.

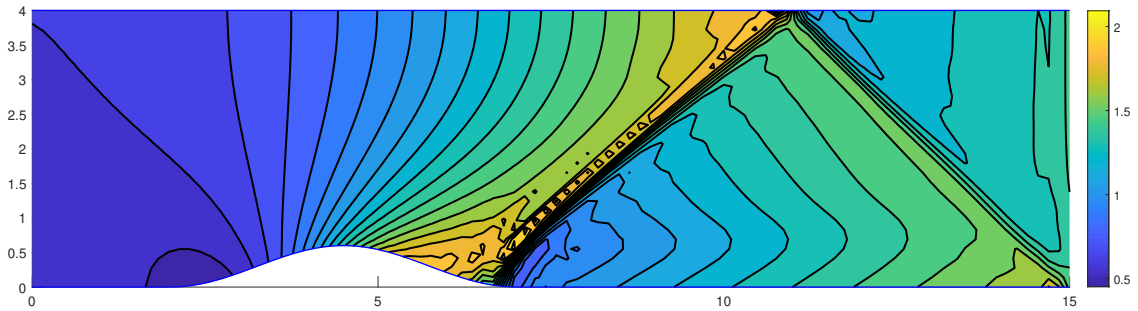
When the downstream pressure is set to 27200Pa (Figure 9c), the solution shows the flow is supersonic downstream of the contraction. An oblique shock wave is formed at the downstream end of the contraction and this is reflected when it reaches the upper wall. The solution shows the shocks and the reflection clearly although they are slightly blurred due to the smoothing in the scheme. At the exit the flow readjusts to meet the uniform exit conditions specified in the solver. This is why the contours at the exit look unusual. This is a limitation of the way in which the exit conditions are specified in the scheme.



(a) Mach number contour plot of test case 2 with $P_{down} = 85000$Pa.



(b) Mach number contour plot of test case 2 with $P_{down} = 80000$Pa.



(c) Mach number contour plot of test case 2 with $P_{down} = 27200$Pa.

Figure 9: Mach number contour plots of test case 2 with various exit pressures.

4.4 Test Case 3

Test case 3 simulates a 180 degree bend with a contraction near the upstream end of the bend of a similar geometry to test case 0. The downstream pressure was set to 27200Pa such that the flow is supersonic downstream of the contraction - similar to test case 2 (Figure 9c). Figure 10 shows the Mach number contour plot of the solution found. The solution shows that an oblique shock is formed at the downstream end of the contraction and this is reflected down the pipe. This agrees with the solution found for test case 2. Additionally, the solution aligns with test case 1 where the Mach number on the inside of the bend is greater than the Mach number on the outside of the bend.

The shock waves don't appear as clearly as in test case 2 and are blurred because of the smoothing in the scheme. The shocks appear to maintain strength upon reflection. The last shock wave is not reflected off the wall because the flow is turned parallel to the pipe. The flow downstream of the bend shows some non-uniformities but overall the solution found matches the analytical solution and the previous test cases relatively well.

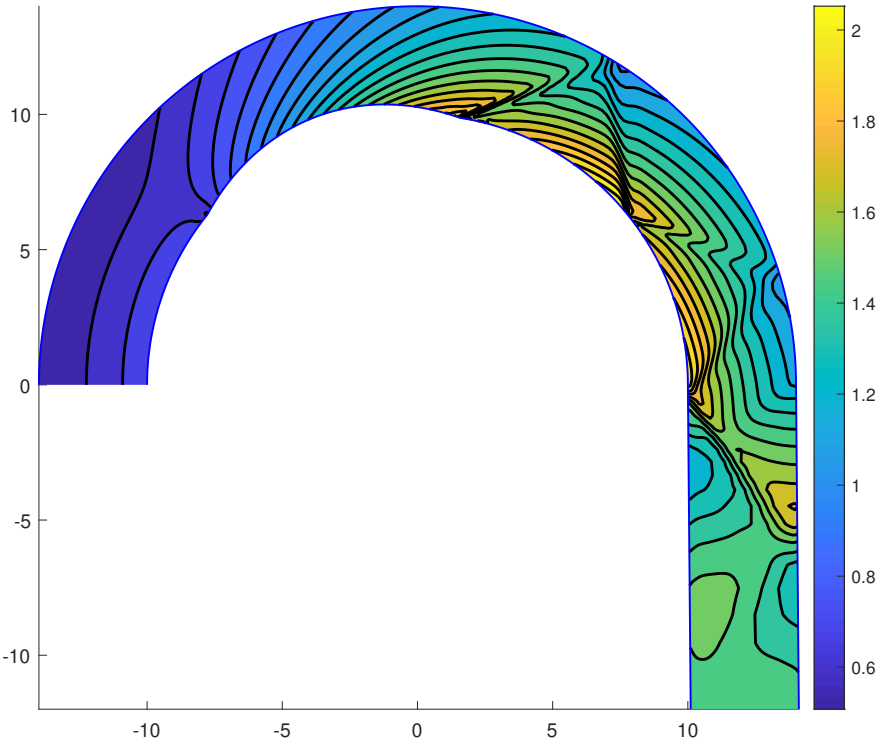


Figure 10: Mach number contour plot of test case 3 with $P_{down} = 27200$Pa

4.5 Test Case 4

Test case 4 simulates a wedge with supersonic flow throughout the solution. The geometry of this test case is designed based on an exact analytical solution for an inlet Mach number of 1.6 and wedge angle of 5.727^o. The analytical solution for this case shows that an oblique shock is formed at the upstream end of the wedge and is reflected once on the upper surface. The reflected shock should end at the downstream end of the wedge according to the analytical solution because the flow is turned parallel to the walls. As is shown in Figure 11, the numerical solution matches the analytical solution relatively well. The only major deviation is the Mach contour downstream of the wedge which appears as if it is a reflection from the shock. This suggests the flow downstream of the wedge is not exactly parallel to the walls and hence the solution found doesn't perfectly match the analytical solution. However, the bulk of the downstream flow is uniform and hence this error appears to be relatively small.

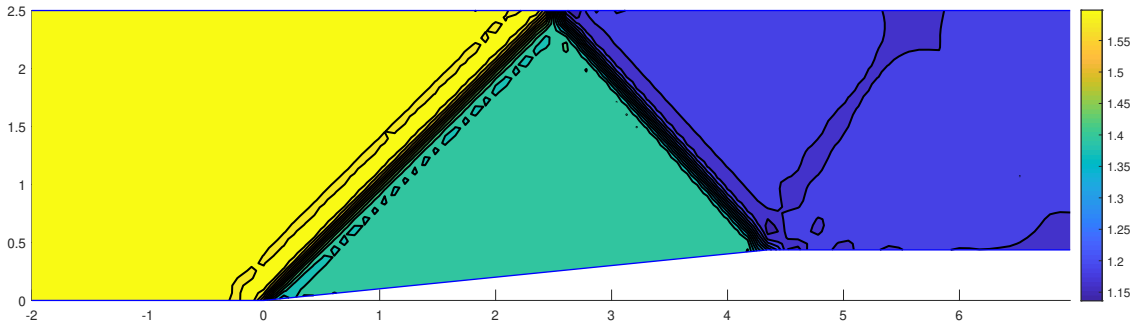
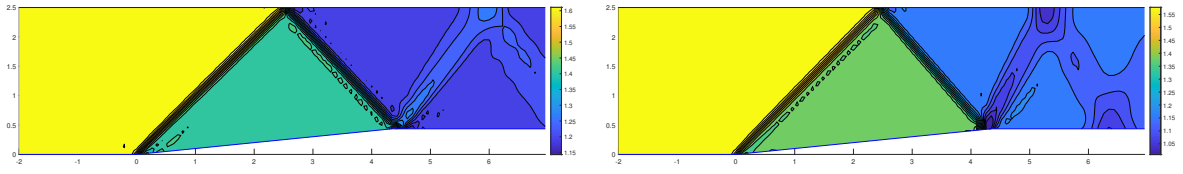


Figure 11: Mach number contour plot of test case 4.

The upstream Mach number can be increased by decreasing the downstream pressure and vice versa. Figures 12a and 12b show the effect of increasing and decreasing the upstream Mach number respectively. This is achieved by setting the downstream pressures to 23000Pa and 24000Pa. For reference, the downstream pressure in Figure 11 is set to 23527Pa. Analytically, if the upstream Mach number is increased, the angle of the oblique shock to the flow direction should decrease. Therefore, the reflected shock will hit the lower wall downstream of the wedge and hence will be reflected again and expansion waves will turn the flow around the downstream corner of the wedge. Similarly, if the upstream Mach number is decreased, the reflected shock wave will hit the lower wall upstream of the downstream corner of the wedge and hence the shock is reflected again.

Figures 12a and 12b show clearly that additional shock reflections are present downstream of the wedge. However, these shocks are relatively blurred out due to the smoothing of the scheme. Additionally, the flow at the downstream corner of the wedge is visibly more complex than in Figure 11. This is because of the additional reflections and interaction between the expansion waves and shock wave in the case where the upstream Mach number is increased. Overall, these solutions match the analytical solution relatively well but the blurring of shock waves due to smoothing does lead to noticeable blurring of flow features.



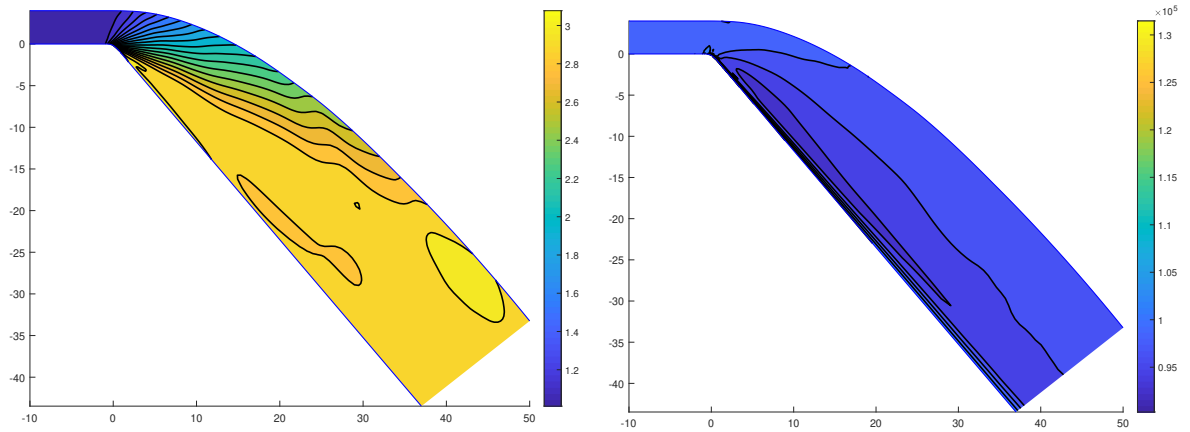
(a) $P_{down} = 23000$Pa.

(b) $P_{down} = 24000$Pa.

Figure 12: Mach number contour plots of test case 4 with different downstream pressures.

4.6 Test Case 5

The geometry for this test case has been designed using the method of characteristics to increase the flow Mach number from 1 to 3 isentropically and without any shock waves. Figure 13a shows the Mach number contour plot for the solution found. The contour lines show the expansion waves as the flow turns the corner clearly. Near the right hand wall these contour lines are not straight because the expansion waves are reflected off the wall. In the analytical solution there are no reflections off the wall and the expansions waves are straight. Therefore, this highlights a slight error in the solver. This error could arise from the discretisation of the wall surface - ideally this would be continuous to match the geometry of the analytical solution. After the final expansion wave the flow should be uniform, however, elliptical Mach contour lines are visible showing that this is not the case. Despite this, the Mach number at exit is uniform across the whole area and hence the bend appears to achieve a uniform flow at Mach 3 relatively well.



(a) Mach contour plot.

(b) Stagnation pressure ($\times 10^5$Pa) plot.

Figure 13: Plots of test case 5.

Another way to visualise the errors in the solution is to look at the stagnation pressure loss. Figure 13b shows the stagnation pressure contours. This shows that there is relatively little stagnation pressure losses in the solution and hence the errors in the solution are small. The losses are largest near the left hand wall and corner. This plot also shows that no shock waves are present in the solution. Overall, this test case highlights the accuracy of the improved code and shows that the errors that arise are small.

4.7 Test Case 6

This test case was designed to investigate whether the scheme can converge when two shock waves collide. The geometry used is similar to that of test case 4 but there is a wedge on both the upper and lower surfaces. The downstream section has also been extended to observe the reflected shock waves. Figure 14 shows the Mach contour plot of the solution found. This shows that the scheme handles the collision of the two shocks relatively well. The collision point is slightly blurred but strength is maintained and the shocks turn slightly as expected. Interestingly, the start of the shock at the upstream corner of the wedge is not as clean as in test case 4.

The expansion waves from the ends of the wedges are seen clearly. The shocks are also well reflected in the downstream passage. Overall, the scheme handles the additional complexity well and the flow structures are well preserved.

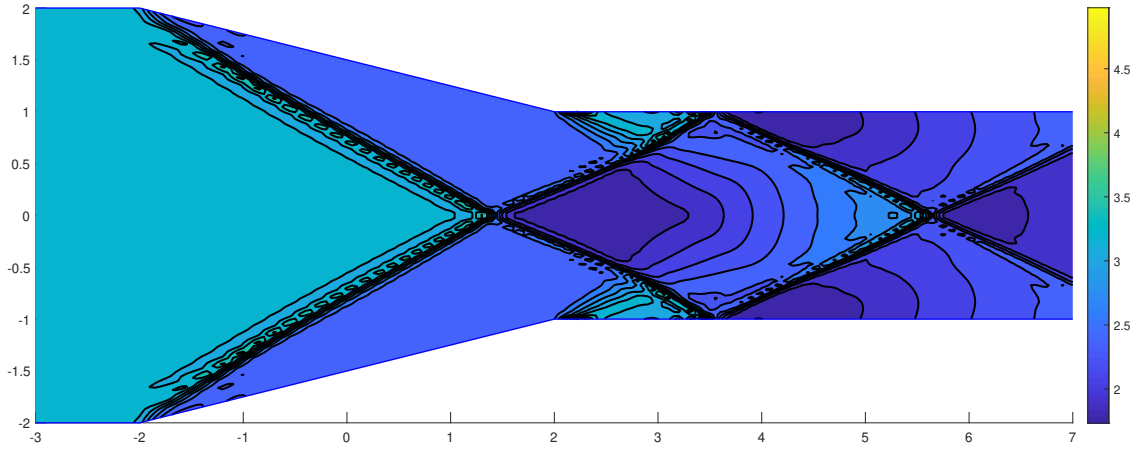


Figure 14: Mach contour plot of test case 6.

5 Conclusion

The accuracy, stability and speed of the simple Lax method from the interim report has been improved significantly. The scheme has been analysed under a variety of conditions and produces respectable solutions for both subsonic and supersonic inviscid flows. The next steps to develop this scheme further would be to make it three dimensional and implement a turbulence model and viscosity.

References

[1] *Computational fluid dynamics : principles and applications.* Butterworth-Heinemann, 2015.