THE UNIVERSITY OF BRITISH COLUMBIA

MECH 510

PROGRAMMING ASSIGNMENT 3

Prepared for:

Prof. Carl Ollivier-Gooch

CEME 2050, UBC

Prepared by:

Nicholas Earle

UBC # 21943600

Date:

November 23rd, 2018

INTRODUCTION

The objective of this assignment was to write a program to solve the incompressible, laminar energy equation in two dimensions given a velocity field. This was to be done using a second order centred space discretization for both the convective and diffusive terms. It was then solved using each of an explicit and implicit time advance scheme. The explicit scheme was chosen to be the two stage Runge Kutta (RK2) scheme simply because it is also a second order scheme, so it would produce the same order of accuracy without any extra programming. As for the implicit scheme, the implicit Euler method was used. This was done using approximate factorization of the matrices to obtain a series of tri-diagonal matrices which were then solved using the Thomas algorithm as given. These schemes were then used to solve the overall problem of determining the thermal development of a flow in a channel.

The assignment was broken down into seven parts to make the task more manageable and provide numerous checks along the way. Below the results for each of these parts are described in the following order. Parts 1 and 2, which are concerned with the validation and confirmation of the flux and source terms. Parts 3 and 4 are programming the explicit and implicit time advances, which are combined with parts 5 and 6, which test the stability, accuracy, and efficiency of both schemes. And, finally, part 7 is the real problem of determining the thermal development of the flow in a channel given certain initial conditions and boundary conditions.

FORMULATION

First, we much define the incompressible, laminar energy equation in two dimensions:

$$\frac{\partial T}{\partial t} + \frac{\partial uT}{\partial x} + \frac{\partial vT}{\partial y} = \frac{1}{RePr}\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) + S \tag{1}$$

Where, T is the temperature, u and v are the velocity fields in the x and y directions, respectively, Re is the Reynolds number, Pr is the Prandtl number, and S is the source term as defined by:

$$S = \frac{Ec}{Re}\left(2\left(\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)^2\right) \tag{2}$$

Where, Ec is the Eckert number. Now in order to discretize this function we can integrate of a control volume and apply the Gauss divergence theorem and approximate the derivatives using a second order centred scheme as follows:

$$\frac{\partial uT}{\partial x} \approx \frac{\bar{u}_{i+1,j}\bar{T}_{i+1,j} - \bar{u}_{i-1,j}\bar{T}_{i-1,j}}{2\Delta x}$$
$$\frac{\partial^2 T}{\partial x^2} \approx \frac{\bar{T}_{i+1,j} - 2\bar{T}_{i,j} + \bar{T}_{i-1,j}}{\Delta x^2} \tag{3}$$

Applying these to equation 1, above, we arrive at the semi-discrete form of the energy equation using second order centred differences.

$$\frac{\partial \bar{T}_{i,j}}{\partial t} = -\left(\frac{\bar{u}_{i+1,j}\bar{T}_{i+1,j} - \bar{u}_{i-1,j}\bar{T}_{i-1,j}}{2\Delta x}\right) - \left(\frac{\bar{v}_{i,j+1}\bar{T}_{i,j+1} - \bar{v}_{i,j-1}\bar{T}_{i,j-1}}{2\Delta y}\right) + \frac{1}{RePr}\left[\left(\frac{\bar{T}_{i+1,j} - 2\bar{T}_{i,j} + \bar{T}_{i-1,j}}{\Delta x^2}\right) + \left(\frac{\bar{T}_{i,j+1} - 2\bar{T}_{i,j} + \bar{T}_{i,j-1}}{\Delta y^2}\right)\right] + \bar{S}_{i,j} \tag{4}$$

Where the discretized source term is:

$$S_{i,j} = \frac{Ec}{Re}\left[2\left(\frac{\bar{u}_{i+1,j} - \bar{u}_{i-1,j}}{2\Delta x}\right)^2 + 2\left(\frac{v_{i,j+1} - \bar{v}_{i,j-1}}{2\Delta y}\right)^2 + \left(\frac{\bar{v}_{i+1,j} - \bar{v}_{i-1,j}}{2\Delta x} + \frac{\bar{u}_{i,j+1} - \bar{u}_{i,j-1}}{2\Delta y}\right)^2\right] \tag{5}$$

From now on the combination of each of the equations above will simply be referred by $FI_{i,j}$ as the flux integral at cell *i, j*. We can now look at each of the time advances. Starting with RK2, it is defined as:

$$\begin{aligned} w^{n+1} &= w^n + \lambda \Delta t w^{(1)} \\ w^{(1)} &= w^n + \frac{\lambda}{2}\Delta t w^n \end{aligned} \tag{6}$$

Where w^{n+1} is the value of the cell at the next time step, w^n is the current value, Δt is the timestep, and $\lambda w^{(n)}$ can be thought of as the evaluation of the flux integral using the values from step (n). The implicit Euler time advance, while even simpler in its formulation:

$$w^{n+1} = w^n + \lambda \Delta t w^{n+1} \tag{7}$$

Is much more involved to solve, because we are evaluating the flux integral at a time step at which we do not yet know the data. If we instead use the δ- form of T, defined as $\delta T_{i,j}^{n+1} = T_{i,j}^{n+1} - T_{i,j}^n$ we can write the full discretization as:

$$\frac{\overline{\delta T}_{i,j}}{\Delta t} + \left(\frac{\bar{u}_{i+1,j}\overline{\delta T}_{i+1,j} - \bar{u}_{i-1,j}\delta \bar{T}_{i-1,j}}{2\Delta x}\right) - \frac{1}{RePr}\left(\frac{\overline{\delta T}_{i+1,j} - 2\overline{\delta T}_{i,j} + \overline{\delta T}_{i-1,j}}{\Delta x^2}\right) + \left(\frac{\bar{v}_{i,j+1}\overline{\delta T}_{i,j+1} - \bar{v}_{i,j-1}\delta \bar{T}_{i,j-1}}{2\Delta y}\right) - \frac{1}{RePr}\left(\frac{\overline{\delta T}_{i,j+1} - 2\overline{\delta T}_{i,j} + \overline{\delta T}_{i,j-1}}{\Delta y^2}\right) = FI_{i,j}^n \tag{8}$$

Where the right hand side is the flux integral evaluated at the current time step as seen above in equation 4. This however, can be represented in a much more compact form:

$$\left([I] + \Delta t[D_x] + \Delta t[D_y]\right)\overrightarrow{\delta T} = -\Delta t\left([D_x]\vec{T}^n + [D_y]\vec{T}^n\right) + \Delta t\vec{S} \tag{9}$$

Where $[D_x]$ and $[D_y]$ are the space discretization of x and y, respectively, in matrix form and $[I]$ is the identity matrix. These matrices must be adjusted accordingly to accommodate any proposed boundary conditions. Now applying approximate factorization to the left hand side we get:

$$([I] + \Delta t[D_x] + \Delta t[D_y])\overrightarrow{\delta T} \approx ([I] + \Delta t[D_x])([I] + \Delta t[D_y])\overrightarrow{\delta T} \tag{10}$$

Which can be shown to be second order accurate, which does not hurt the implicit Euler scheme since it is only first order accurate in time. Subbing this back into equation 9, we get:

$$([I] + \Delta t[D_x])([I] + \Delta t[D_y])\overrightarrow{\delta T} = \Delta t\, FI^n \tag{11}$$

Where is we define $\widetilde{\delta T} = ([I] + \Delta t[D_y])\overrightarrow{\delta T}$ we can split the problem into a system of equations:

$$\begin{gathered}([I] + \Delta t[D_x])\widetilde{\delta T} = \Delta t\, FI^n \\ ([I] + \Delta t[D_y])\overrightarrow{\delta T} = \widetilde{\delta T}\end{gathered} \tag{12}$$

Where each of these equations must be solved for every row and every column to obtain the full timestep. And finally, once we have solved for $\overrightarrow{\delta T}$ we can simply add that to the current time step for the next one:

$$\vec{T}^{n+1} = \vec{T}^n + \overrightarrow{\delta T} \tag{13}$$

Following this as with the explicit scheme, the boundary conditions must be reset. Without going into detail each of the tri-diagonal matrices can be solved using Gauss elimination as performed by the Thomas matrix algorithm.

Results and Discussion

For the problem of defining the thermal development of flow in a channel, a rectangular channel was used with a height to width ratio of 5/1. The velocity fields we given as:

$$\begin{gathered}u(x, y) = 6\bar{u}y(1 - y) \\ v(x, y) = 0\end{gathered} \tag{14}$$

Where $\bar{u} = 3\ m/s$. The inflow is to be set to a fixed temperature, while the upper wall was set to $T = 1$, and the bottom wall $T = 0$, with the outflow condition set to have no flux across the boundary. The flow conditions we such that $Re = 50, Pr = 0.7$, and $Ec = 0.1$.

Parts 1 and 2

Parts 1 and 2 concerned themselves with ensuring that the flux integral and source term were indeed programmed correctly as to produce expected results. This was tested by setting the

temperature and velocity field to values such that the solution to each were known. The domain was set using:

$$T(x,y) = T_0 \cos(\pi x) \sin(\pi y)$$
$$u(x,y) = u_0 y sin(\pi x) \quad (15)$$
$$v(x,y) = v_0 x cos(\pi y)$$

Where if programmed correctly would produce results resembling the exact flux:

$$FI = u_0 T_0 \pi \cos(2\pi x)\, y \sin(\pi y) + v_0 T_0 \pi x \cos(\pi x) \cos(2\pi y) + \frac{2T_0 \pi^2 \cos(\pi x) \sin(\pi y)}{RePr} \quad (16)$$

And the exact source term:

$$S = \frac{Ec}{Re}(2(u_0 \pi\, \mathrm{ycos}(\pi x))^2 + 2(v_0 \pi x \sin(\pi y))^2 + (u_0 \sin(\pi x) + v_0 \cos(\pi y))^2) \quad (17)$$

The results below show the flux integral and source term using these conditions set on a square domain of [0,1] x [0,1]. Figure 1, below, shows images of the error for both the flux and source terms for an 80 x 80 mesh on the domain.

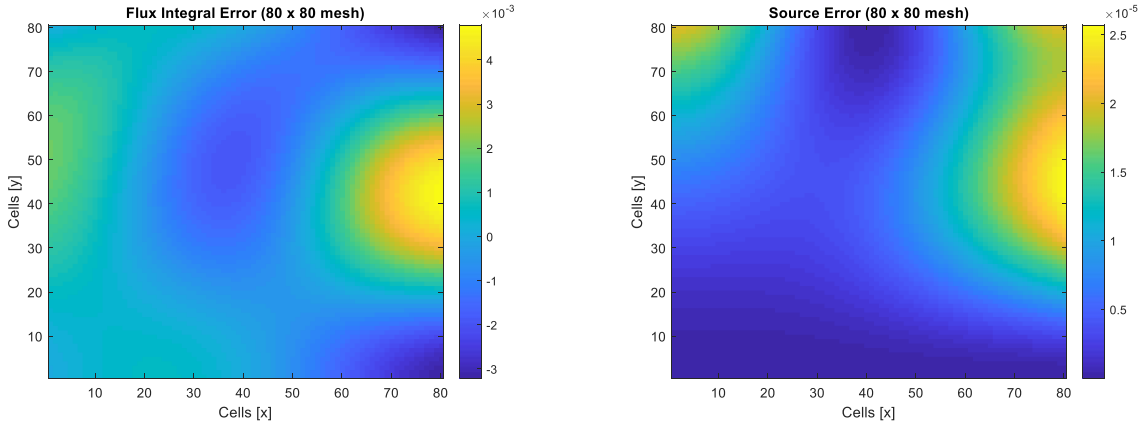


FIGURE 1: FLUX AND SOURCE TERM ERROR

As you can see in each of these images the error is quite small for each term as can be seen more clearly with the L2 norm of each summarized in Table 1, below.

TABLE 1: L2 NORM OF FLUX AND SOURCE TERMS

Mesh Size	Flux Integral		Source Term	
	$L_2\ norm$	Ratio	$L_2\ norm$	Ratio
10 x 10	8.7732e-02	--	5.7723e-04	--
20 x 20	2.2301e-02	3.9340	1.4614e-04	3.9499
40 x 40	5.5984e-03	3.9834	3.6649e-05	3.9874

Mesh Size	Flux Integral		Source Term	
	$L_2\ norm$	Ratio	$L_2\ norm$	Ratio
80 x 80	1.4011e-03	3.9958	9.1696e-06	3.9969

Looking at the error norms in the table above it is very clear that the space discretization scheme used is, for all intents and purposes, second order accurate. That is that as we double the mesh size, the error decreases by a factor of four, or in our case very nearly four.

PARTS 3 – 5

Parts 3 to 5 of the assignment focus on programming an explicit and implicit time advance scheme and subsequently testing them for stability and accuracy. As mentioned above the explicit scheme used was RK2, simply because it is one of the simplest second order schemes, which pairs nicely with the second order space discretization. The implicit scheme as mentioned above is the implicit Euler scheme.

Using the aforementioned velocity fields, on a domain of [0, 5] x [0, 1] using a 25 x 10 mesh, each scheme was given the initial temperature distribution of $T(x, y, 0) = y$, to check whether it indeed converged on the correct solution as given by the inflow boundary condition:

$$T(0, y, t) = y + \frac{3}{4} PrEc\ \bar{u}^2 (1 - (1 - 2y)^4) \tag{18}$$

EXPLICIT TIME ADVANCE

Looking at the explicit scheme first, finding the maximum stable time step is the first step in running the program. To do this a stability analysis was performed, to find app using the second order centred scheme applied to the energy equation for both the x and y discretizations, using the given parameters, to find the eigenvalues. Now the eigenvalues do depend on the velocity, and with u not being constant, the average value of $u = \bar{u} = 3\ [m/s]$ was used, that being said it is difficult to find an exact solution, this will provide us with a good guess to find the actual maximum timestep simply by trial and error. The eigenvalues being:

$$\lambda_x = -\frac{u}{\Delta x}\left(isin(\phi)\right) - \frac{2}{RePr\Delta x^2}(1 - \cos(\phi)) \tag{19}$$

And likewise for y. Plotting these eigenvalues against the stability boundary for RK2, we get the following plots for x and y.

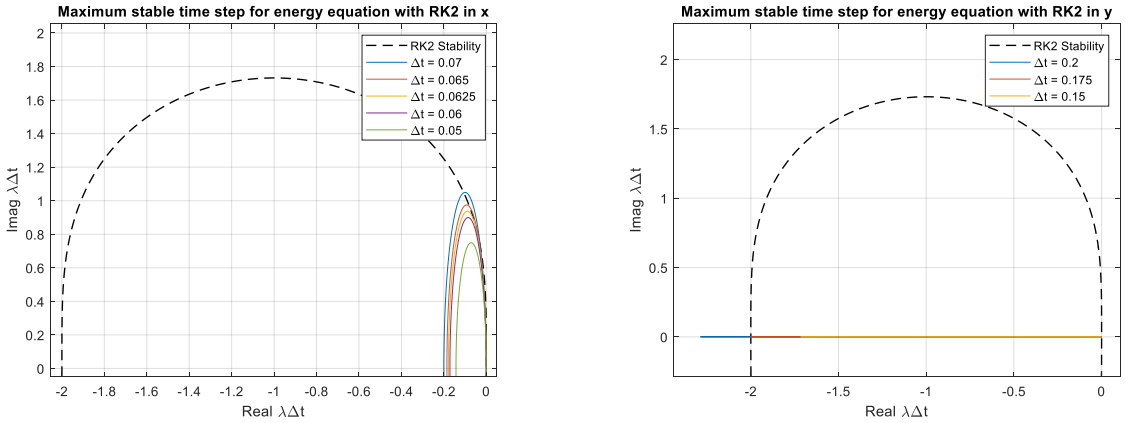


FIGURE 2: STABILITY BOUNDARIES FOR RK2 AND 2ND CENTRED SCHEMES

Looking at the plots, while it may be difficult to see, the maximum stable time step in x is about $\Delta t \approx 0.06$, while in y is $\Delta t \approx 0.175$. So discarding the timestep for y stability we get a good starting point for total stability of about $\Delta t \approx 0.06$. By trial and error, the actual maximum stable timestep, to three decimal points, was found to be $\Delta t_{max} = 0.053$. However, for running the program, it was found that a timestep of about $(\%80 - \%90)\Delta t_{max}$ was the most efficient. The following table shows the results of the explicit scheme for finer and finer meshes. Now, instead of running each iteration for a set number of timesteps where it is unknown whether the solution has converged completely, each will run until the maximum change of any one cell is less than $1e-09$, at which point the loop is broken.

TABLE 2:MAX TIME STEP AND ERROR FOR EXPLICIT SCHEME

Mesh Size	Δt_{max}	Δt_{used}	$L_2\ norm$	Ratio
25 x 10	~0.053	0.04	1.1215e-02	--
50 x 20	~0.030	0.02	2.8815e-03	3.8921
100 x 40	~0.009	0.008	7.2508e-04	3.9740
200 x 80	~0.0022	0.0021	1.1815e-04	6.1369

Onto the accuracy of the scheme, as shown by the results above, as the mesh becomes twice as fine, the error decreases by a factor of about four, except for the final case where it decreases by a factor of six, unexpectedly. But from the results of the previous iterations we can say that the scheme is about second order accurate.

IMPLICIT TIME ADVANCE

For the implicit Euler scheme, unlike explicit schemes, it theoretically doesn't have a maximum stable timestep. Instead however, there is a point at which the solution converges much faster but with a greater error, as shown in the next section. As for the accuracy of the implicit scheme, it looks eerily similar to that of the explicit scheme:

TABLE 3: TIMESTEP AND ERROR FOR IMPLICIT SCHEME

Mesh Size	Δt_{used}	$L_2\ norm$	Ratio
25 x 10	0.2	1.1214e-02	--
50 x 20	0.15	2.8815e-03	3.8921
100 x 40	0.1	7.2508e-04	3.9740
200 x 80	0.05	1.1815e-04	6.1369

So, if we conclude that the explicit scheme is second order accurate, it would be very hard to argue anything for the contrary for the implicit scheme. The implicit scheme does have efficiency on its side compared to the explicit scheme as explained next.

PART 6

Part 6 looks at the efficiency of each scheme, and the effect of differing time step has on each. It should be noted that these results were obtained on Ubuntu 18.04 with the GCC GNU compiler, on a machine running an 8 core Intel Core i7-4790 CPU @ 3.6 GHz, with 15.5 GB of memory. Table 4, below, shows the time taken and number of iterations completed before reaching a tolerance of $1e-09$. The timestep used for the explicit scheme was about %80 of the maximum stable time step as described above, which tended to give the best results, while for the implicit scheme, the timestep used was the timestep that was found to give the best results. This is opposed to what was specified in the assignment as it says to use the maximum timestep the code will allow, however I found that with an ever increasing timestep, it just took much longer to reach the same conclusion as displayed later on.

TABLE 4: EFFICIENCY OF EXPLICIT AND IMPLICIT SCHEMES

Mesh size	Explicit Scheme			Implicit Scheme		
	Δt	$time\ (s)$	Iterations	Δt	$time\ (s)$	Iterations
25 x 10	0.04	0.03056	245	0.2	0.01417	43
50 x 20	0.02	0.06019	173	0.16	0.01879	39
100 x 40	0.008	0.42764	394	0.09	0.02430	51
200 x 80	0.0021	6.11973	1415	0.05	0.4411	92
400 x 160	00005	94.7845	5546	0.03	3.0052	162

Looking at these results it is clear that, for optimal timesteps, the implicit scheme is much faster than the explicit scheme requiring only a fraction of the iterations to obtain the same accuracy. However, as with the explicit scheme, the implicit scheme doesn't perform as well for timesteps that are much too small or much too large. Table 5, below, shows snippet of an experiment showing the error, time, and iterations for the 25 x 10 mesh running with timesteps varying from $1e-07$ to $1e+07$, the full results can be seen in the Appendix. Looking at these results there is no clear "maximum time step" (unless it is greater than $1e+07$, which I didn't test) but there is a point at which the program converges on a different solution, with much greater error. That is at a timestep

of about $\Delta t = 27458$, where the L2 norm goes from 0.01121 to 0.07287, which is a relatively huge jump. Furthermore, when doing the same experiment for the 50 x 20 mesh, we see the same behaviour, as shown in the Appendix. For this case however, the "maximum" timestep is much smaller and the L2 norm changes from 0.00288, which is of course second order accurate with the 25 x 10 mesh, to 0.03142, which is only about half of that of the previous case, leading to the hypothesis that the scheme becomes first order accurate. This may be because by using such a giant timestep, a local truncation error is introduced that the scheme simply cannot overcome but does its best to produce an acceptable result.

TABLE 5: IMPLICIT SCHEME FOR DIFFERING TIMESTEP

Mesh Size: 25 x 10			
Δt	$L_2\ norm$	time (s)	Iterations
0.0000001	0.01121	370.686	3818889
0.000001		261.891	2707750
0.001		0.45362	4598
0.01		0.057257	488
0.1		0.018143	68
0.2		0.014165	43
0.4		0.014496	47
0.8		0.018322	79
1		0.014437	96
10		0.087152	788
1000		2.63056	27579
10000		20.1965	210781
25000		47.8102	501020
27457		52.7434	550259
27458		52.8759	550279
27459	0.07287	0.461225	4680
27460		0.457621	4680
50000		0.837793	8521
100000		1.64398	17041
1000000		16.5132	170263
10000000		162.003	1688566

PART 7

Part 7 gets to the real problem of the assignment that is determining the fully-developed value of the temperature gradient at the bottom wall and the thermal development length. This was done using an inflow boundary condition of $T(0, y, t) = y$. When plotting the gradient at the bottom wall against x, the fully-developed value was taken to be the value to which the solution converges at steady state, that is once it has stopped changing. This is of course when the gradient of the gradient reaches zero and the point at which this happens is then the development length. For this experiment the value of the gradient of the gradient was considered zero once it was less than $1e -$

04 because the outputted values only carried four digits of significance, and values afterwards were less reliable, oscillating before finding zero. So, the steady state point was approximated by the first instance where this is true. Figure 3, below, shows the solution for a mesh size of $400\ x\ 80$ with the channel on the domain $[0,200]\ x\ [0,1]$.

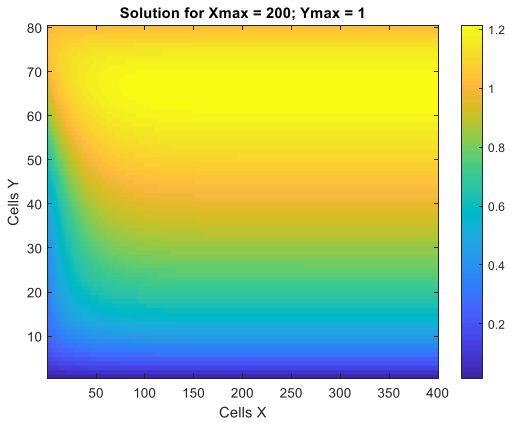


FIGURE 3: SOLUTION ON [0, 200] X [0,1]

Looking at the image it looks as though it reaches a steady state near or after cell 100 in the x direction. To see more accurately where we do in fact reach the steady state solution Figure 4, below, shows the gradient along the bottom wall and the gradient of that gradient.

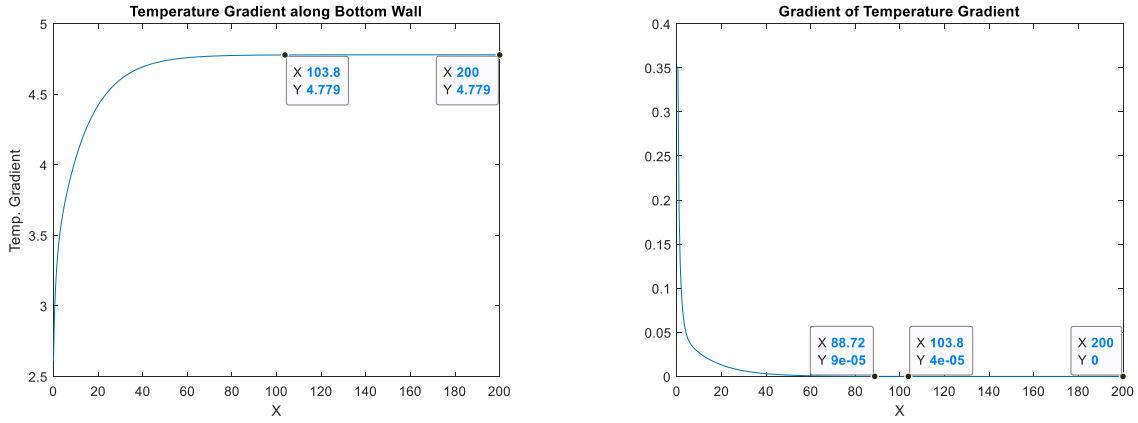


FIGURE 4: TEMPERATURE GRADIENTS

Looking at these plots we can see more accurately that the solution reaches a steady state at about $x = 104$, to three decimal places. So, we can now conclude that the fully-developed value for the temperature gradient given the initial conditions mentioned above is 4.779. And staying consistent with the stopping criteria above, the thermal development length is about 100 channel widths.

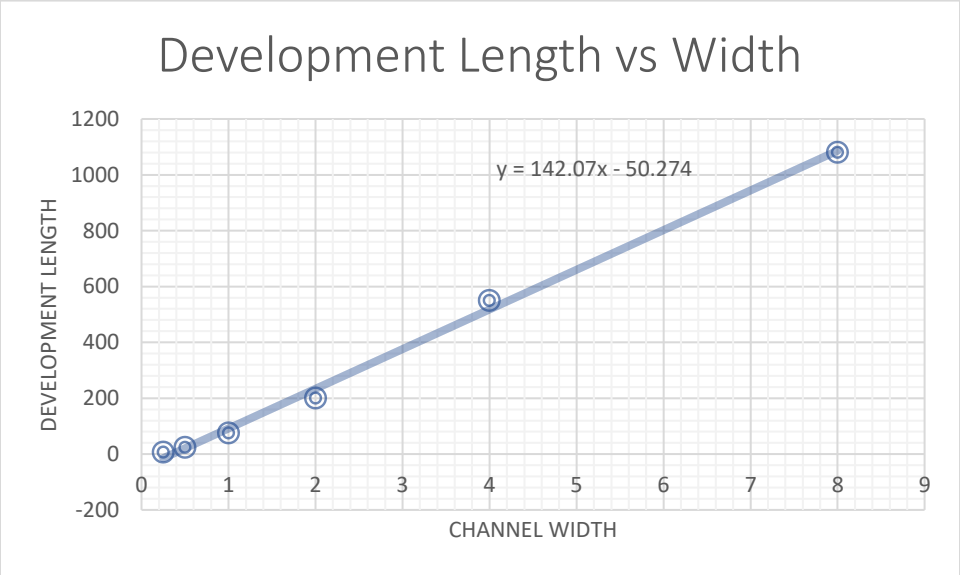
We will now try to define both the gradient value and development length more generally for the initial condition above. First, we will look at the effects of changing the width of the channel. For this experiment a constant 500 cells per grid point was used to keep the resolution of each solution

consistent. That is with $x_{max} = 100,500$ cells were used, and from there scaled accordingly. As we will see, because the gradient is decreases as the channel gets wider the stopping criteria must also adjust, so for each width the stopping criteria will also be divided by the new ratio. Table 6 below, summarizes the results of the experiment while all the plots of the gradient can be found in the Appendix.

TABLE 6: DEVELOPMENT LENGTH AND GRADIENT ALONG BOTTOM WALL

Width	Dev. Length	Gradient
0.25	~7	19.08
0.5	~24	9.541
1	~75	4.771
2	~200	2.385
4	~550	1.188
8	~1080	0.9365

Looking at the data gathered we can say that for small widths the development length approximately triples for a double in width while for larger widths the development length tends double for a doubling of width. Figure 5, below, shows the development length plotted against the width. Looking at this plot, all the points very nearly fit on a linear interpolant.



As for the gradient, it almost perfectly scales with the inverse of the width. So, a doubling in width results in halving the gradient. To further explore how these parameters change, the Reynolds number and Prandtl number were differed as above.

TABLE 7: DEVELOPMENT LENGTH AND GRADIENT ALONG BOTTOM WALL

$Re \cdot Pr$	Dev. Length	Gradient
35	~100	4.771
70	~200	4.771
140	~400	4.771

These results were much easier to interpret as it is clear that by doubling the Reynolds number or Prandtl number the development length also doubles and has no effect on the gradient. It was also found that by doubling the temperature everywhere, by doubling the upper wall boundary condition and the Eckert number, this had no effect on the development length but doubled the value of the gradient at the bottom wall, as would be expected. From all the summarized results we can conclude that the gradient along the bottom wall and development length for the initial conditions given looks of the order:

$$\Delta T_0 \approx \frac{4.771(T_u - T_0)}{W} \qquad (20)$$

$$L_{dev} \approx (4.06W - 1.44)RePr$$

Now, obviously this is only valid for widths larger than 0.3547, which certainly doesn't even apply to each of the cases tested above. That being said however, it does give a decent simple approximation to the development length for all but the narrowest of channels. Clearly, we could determine a function that fits each point, but given each of these lengths are estimations themselves that seems troublesome.

CONCLUSION

To conclude, while I feel this assignment may have been more a test of patience while debugging than anything else, it was interesting to explore the characteristics of both explicit and implicit methods. I would like to see the performance of other explicit methods compared to the implicit Euler method, both in terms of duration and accuracy, as many methods can be much more accurate but take much longer, and to see where the cut-off for one or the other might lie.

Looking at the results of the actual problem, it was interesting to see how much of a role the resolution of the mesh plays. While trying to determine to development length it was difficult to settle on one number because by changing the mesh size the length could change not insignificantly. A mesh of 5 cells per grid point was used simply because that was a nice even trade off between accuracy and time needed to run the code.

APPENDIX

TABLE 8: EFFICIENCY RESULTS

Mesh	Δt	$L_2\ norm$	time (s)	Iterations
25 x10	0.0000001	0.01121	370.686	3818889
	0.000001		261.891	2707750
	0.00001		31.9382	332201
	0.0001		4.21625	43578
	0.001		0.45362	4598
	0.01		0.057257	488
	0.1		0.018143	68
	0.2		0.014165	43
	0.4		0.014496	47
	0.8		0.018322	79
	1		0.014437	96
	10		0.087152	788
	1000		2.63056	27579
	10000		20.1965	210781
	20000		38.1666	400817
	25000		47.8102	501020
	26250		50.9847	526070
	26875		51.7797	538596
	27200		52.2057	545109
	27350		53.1083	548115
	27425		53.3168	549618
	27440		52.6586	549919
	27450		52.7947	550119
	27455		52.8345	550219
	27457		52.7434	550259
	27458		52.8759	550279
	27459	0.07287	0.461225	4680
	27460		0.457621	4680
	27500		0.453407	4687
	30000		0.504438	5113
	50000		0.837793	8521
	100000		1.64398	17041
	1000000		16.5132	170263
	10000000		162.003	1688566
50 x 20	0.1	0.00288149	0.024301	51
	0.15		0.019776	41
	0.16		0.018789	39
	0.2		0.029272	56

	0.4		0.042014	104
	0.8		0.07105	193
	1		0.083972	232
	1000		9.53208	29484
	10000		1.20986	3661
	100000	0.0314212	11.9689	36568
	1000000		119.528	363178
	0.09		0.069731	51
	0.1		0.069492	53
	0.2		0.08574	67
100 x 40	0.4	0.00072508	0.200536	162
	1		0.473619	400
	100		9.53842	8142
	1000		92.6391	79328
	5000		357.24	309990
	0.03		0.583916	123
	0.04		0.438977	92
	0.05		0.441127	92
	0.1		0.552037	117
200 x 80	0.2	0.00018155	0.923047	198
	0.4		1.66082	356
	1		3.69493	789
	10		26.3183	5754
400 x 160	0.03		3.00516	162

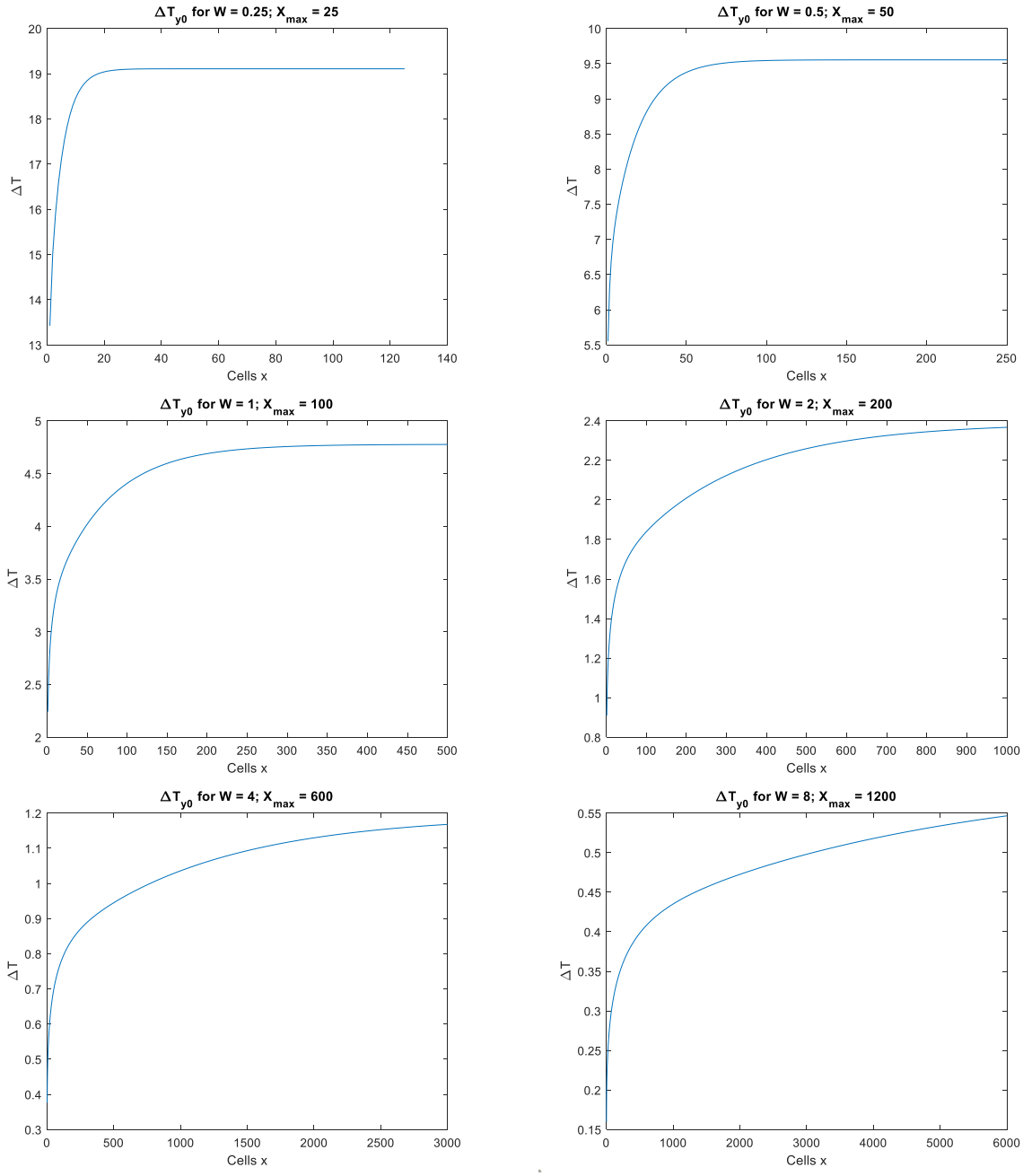


FIGURE 5: GRADIENT SOLUTIONS FOR VARIOUS CHANNEL WIDTHS

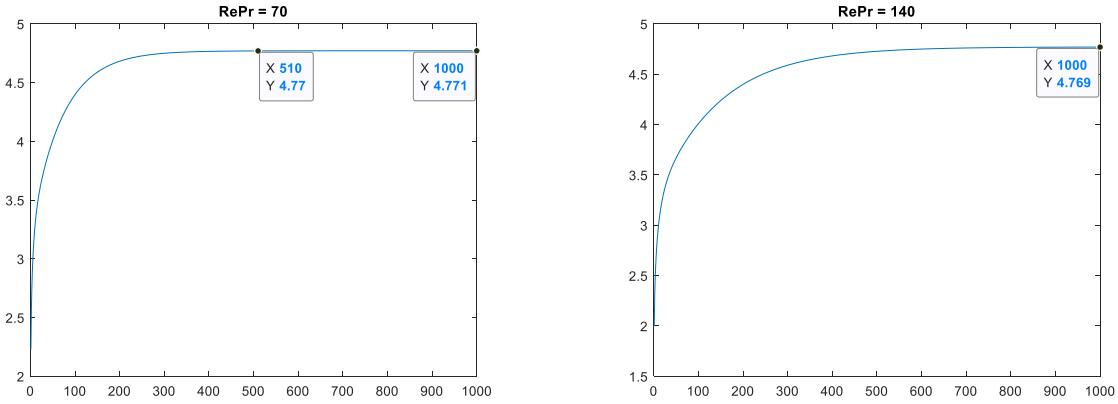


FIGURE 6: GRADIENT SOLUTION FOR VARIOUS RE PR