M3N10-Computational PDES-Project 3

Matthew Cowley, CID 01059624

"The contexts of this report and associated codes are my own work unless otherwise stated" 08/01/2017

Introduction and Method of PDE solver

For the first question we are interested in solving the following equation:

$$u_t = d(u_{xx} + u_{yy}) + Q(x, y, u, t) \qquad \text{with appropriate B.C and} \quad u = u_o(x, y) \quad \text{at} \quad t = 0 \tag{1}$$

The Boundary conditions we are interested in are Neumann Boundary conditions, so we know the value of the derivative at the boundary, e.g solving u on $0 \leq x \leq 1 \quad 0 \leq y \leq L$ then we know u_y on x=0 and x=1 and u_x on y=0 and y=L.

To solve this problem I have used (written) MATLAB code which solves it numerically. It does this by making finite difference approximations and solving it on a discrete grid, solving at each time step using Crank-Nicholson. This then leads to a linear system which we solve iteratively using Gauss-Seidel,improving convergence using geometric multi-grid V-cycles. (If you wish to see more details on this please see projects 1 and 2).

Below is an example of what we are solving using Gauss-Siedel, where Q is evaluated at u_j the old time step or $t + 0.5k$:

$$[1 + dk(\frac{1}{h_x^2} + \frac{1}{h_y^2})]U_{nm}^{j+1} = 0.5dk[\frac{U_{nm+1}^{j+1} + U_{nm-1}^{j+1}}{h_x^2} + \frac{U_{n+1m}^{j+1} + U_{n-1m}^{j+1}}{h_y^2} + \delta^2 U_{nm}^{j}] + kQ_{nm} + U_{nm}^{j} \tag{2}$$

So here k is the time step, h_x is grid space in x direction and δ^2 is finite difference approximation of diffusive terms.

However unlike before now encounter a problem at the boundaries as it is not simply zero. So for U_{nm} on the boundary you know the value of the derivative e.g on y=0 $u_y = \phi(x, y, u, t)$. In the approximation of (2) the only points we don't know are U_{n-1m}^i. However we can make an approximation U_{n-1m}^i in terms of U_{n+1m}^i using ϕ in the following way:

$$\frac{du}{dy} = \frac{U_{n-1m}^i - U_{n+1m}^i}{2h_y} \quad \text{on} \quad y = 0 \tag{3}$$

$$\phi(x(m), 0, U_{nm}^i, t), = \frac{U_{n-1m}^i - U_{n+1m}^i}{2h_y} \quad \text{on} \quad y = 0 \tag{4}$$

We treat ϕ in a similar way to Q in the sense we evaluate it at the old time level if current time level is unknown and at $t + \frac{1}{2}k$ if dependent on time.

$$U_{n-1m}^i = U_{n+1m}^i + 2h_y\phi \quad \text{on} \quad y = 0 \tag{5}$$

We substitute this kind of approximation around all the boundaries being careful which coordinates are used in relation to the sign of ϕ. At the corners we make two of these approximations, so this might mean the solution is particularly inaccurate at the corners.

On a side note when you perform the geometric multi grid you have to be careful using a restrict function. When using a restrict function at the boundaries, you do so by treating it as if a dummy point existed and use approximation (5). In reality this is very impractical, so we just assume $\phi = 0$. So the weighting roughly double the inside points. On a similar note to the above, there will be biggest errors on the corners.

Test of the method

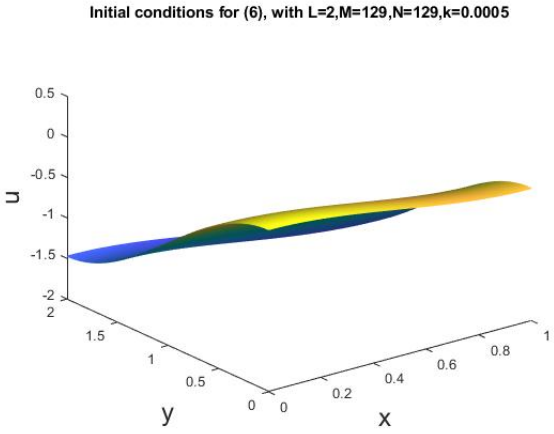
Now let us solve (1) for the following Q and $\phi = 0$ on all boundaries $0 \leq x \leq 1, 0 \leq 2$.

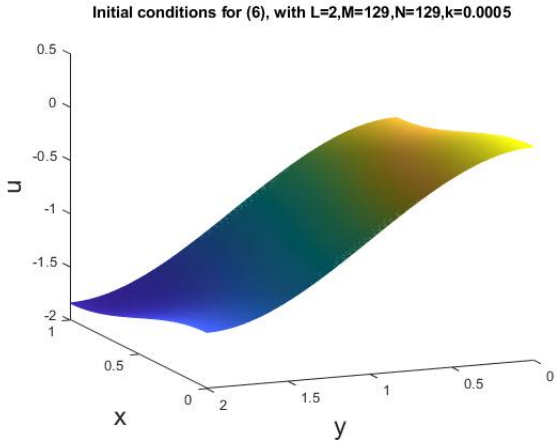
$$Q = (3x^2 - 2x^3 - 12x + 3y^2 - y^3 - 6y + 12)e^{-t}; \quad \text{with I.C } u_o = u(x, y, t = 0) \tag{6}$$

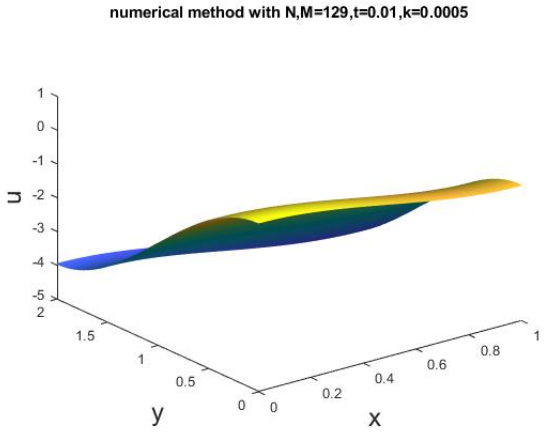
Now this has an exact solution given below, so we can compare the accuracy of the numerical method.

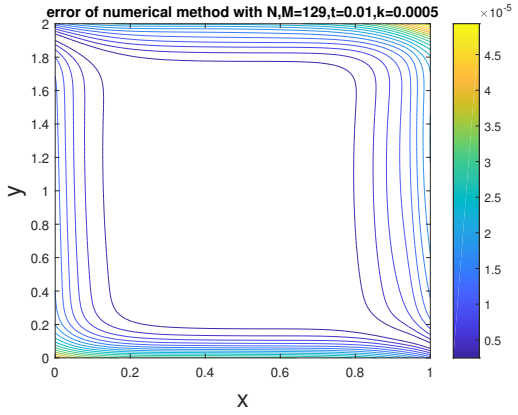
$$u = (2x^3 - 3x^2 + y^3 - 3y^2)e^{-t}$$

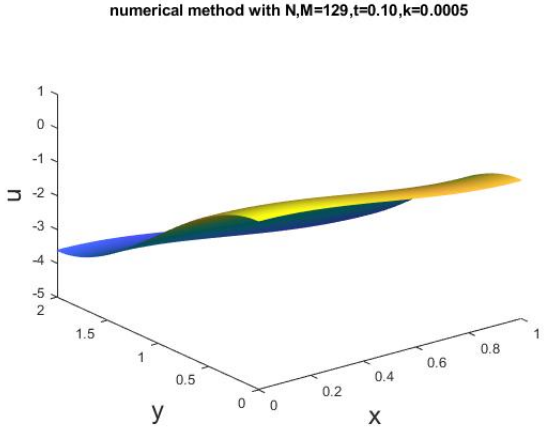
Here are some solutions at various stages of time.

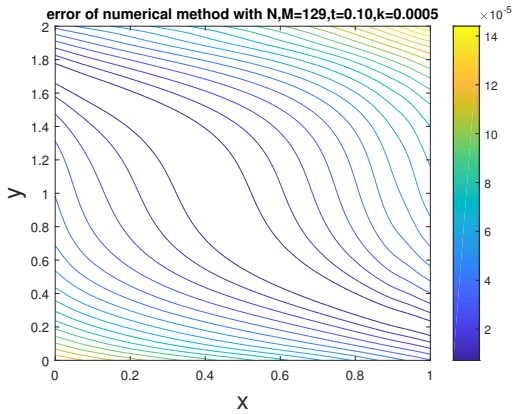


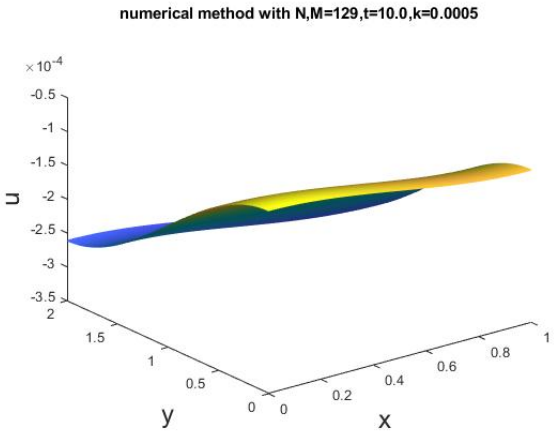


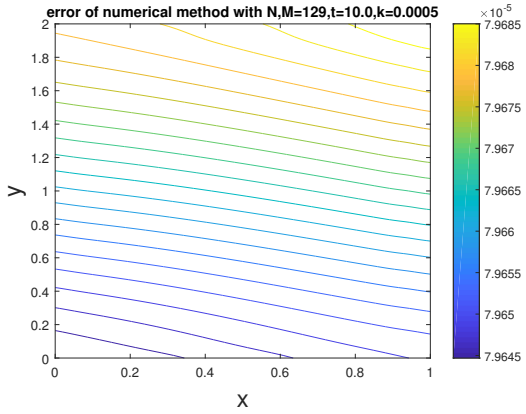












As we can see the error is relative quite good about 10^{-5}, which is about as good as scheme will allow (refer to project 2 error similar). Although as time head towards infinity here the solution becomes very small and error doesn't really change, so for very small small solutions this scheme might not be very good.

Another interesting point is the shape of the error contours, they are largest at the corners then the boundaries. As times goes on they seem to diffuse into the scheme, spreading out. It is also easy to show that as N and M increase so does the quality of solution please refer to project 2 for more information.

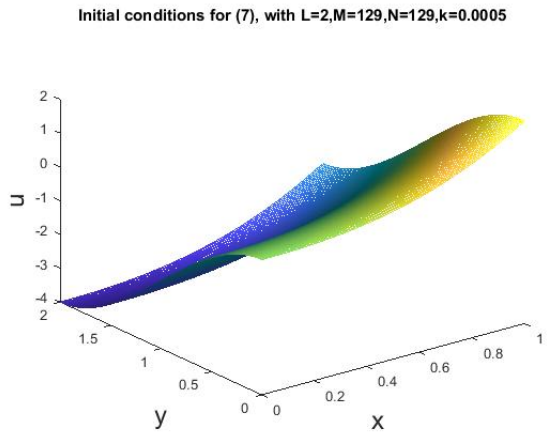
Now let us solve (1) with non-zero boundary conditions. In a similar way we have an exact u and with a corresponding Q given below. An interesting point of note is that when we solve for errors, when using multi grid, we treat all the boundaries as being zero. So in practice it is useful to have a solver for zero boundaries and one for non-zero boundaries. Below are the given Q and U, with Neumann boundary conditions ϕ.

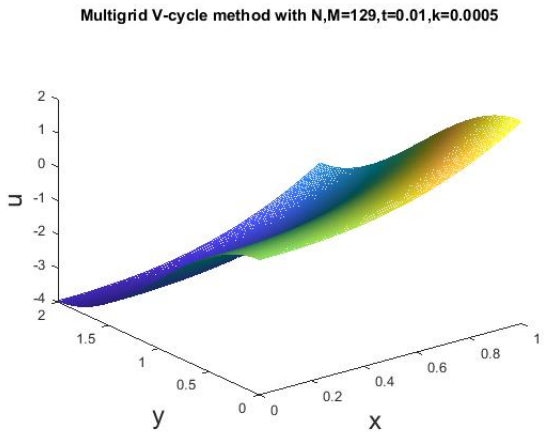
$$Q = (3y^2 - y^3 - x^3 - x^2 - 6y - 6x + 4)e^{-t} \quad \text{with} \quad \phi|_{x=1} = 5e^{-t},\ \phi = 0 \quad \text{everywhere else} \tag{7}$$

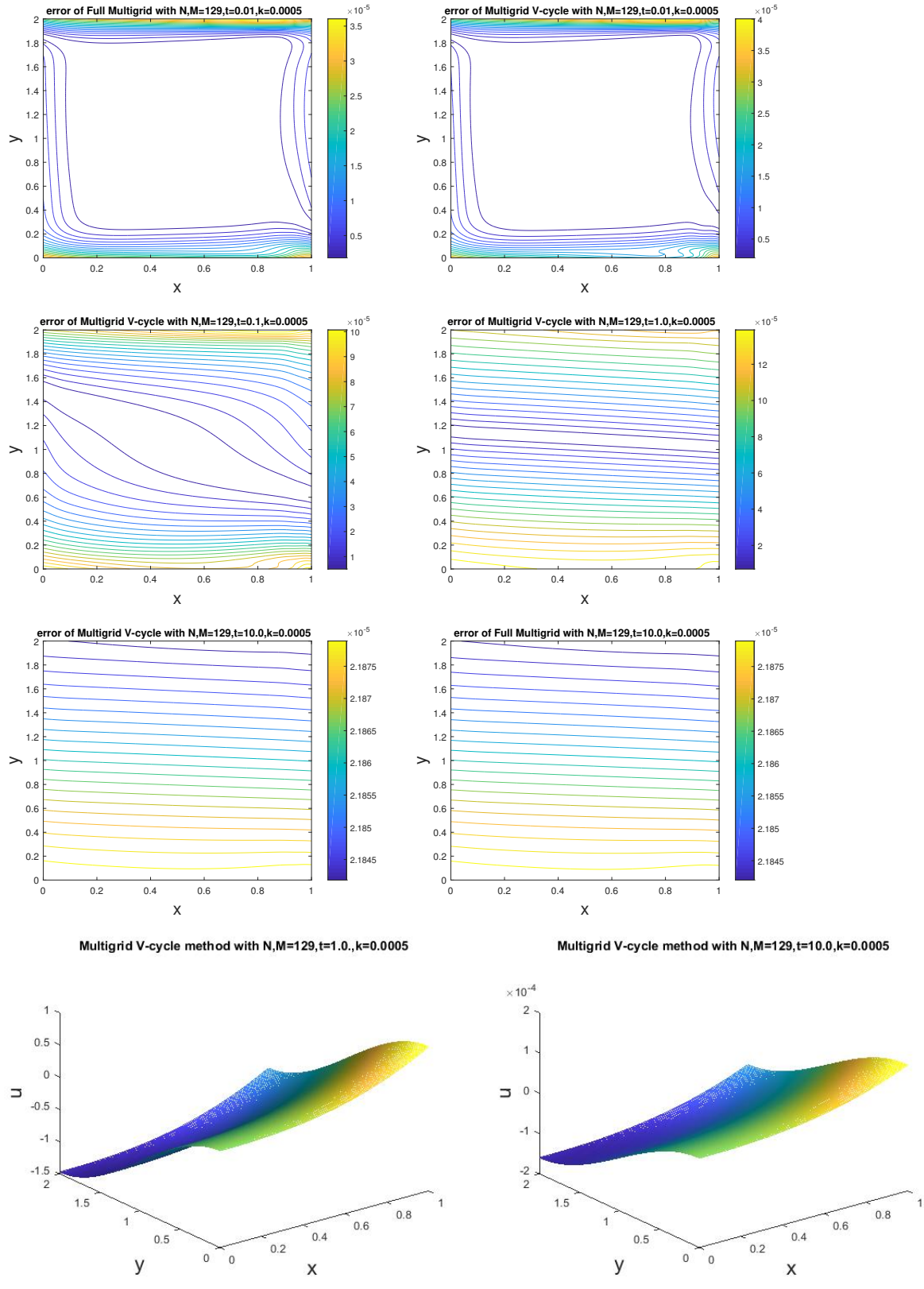
with I.C $u = (y^3 - 3y^2 + x^3 + x^2)$.

$$u = (y^3 - 3y^2 + x^3 + x^2)e^{-t};$$

Here are some numerical solutions, here I have shown the difference between full multi grid (v-cycles solved on many different grids) compared to just v-cycles. As we can see there is very little difference between the two. To save on computing time and practicality will use just V-cycles for the remainder of the project.







Again it appears to be working well, with the error constantly around 10^{-5}. Until like before the solution exponential decreases as time goes to infinity, the relative order of error then becomes 10^{-1}. So the method is useful for seeing the trend of the solution, but not necessarily good for an exact solution as time goes on. The error contours agree with those of the 1st test(diffusing in from boundaries). The Full Multigrid and V-cycle Multigrid seem to have negligible differences in terms or error, so for practicality I will use V-cycle Multigrid for remainder of project.

Introduction to Turing system (question 2)

In this question we are looking at the following Turing system (paired PDE system) with homogeneous Neumann boundary conditions(so $\phi = o$ on all boundaries):

$$u_t = \gamma(a - u + u^2 v) + u_{xx} + u_{yy} \qquad \text{on} \quad 0 < x < 1,\, 0 < y < 2,\, t > 0 \tag{8}$$

$$v_t = \gamma(b - u^2 v) + d(v_{xx} + v_{yy}) \qquad \text{on} \quad 0 < x < 1,\, 0 < y < 2,\, t > 0 \tag{9}$$

With initial condition a perturbation of the unstable uniform state u_0, v_0. For the remainder of the project a=0.07,b=0.95,d=10. By Turing linear theory and depending on other parameter like gamma, we should be able to obtain spots and strips.

Method to solve

To solve this system with the multi-grid solver, we effectively solve each u and v at each time step and use the previous time step to calculate u and v in the source. I did try using the most readily available u and v, but no significant improvements was seen. So for uniformity of the paired solutions, I decided to use previous time step for both.

To improve the the solver I linearized the source in both by doing the following. Instead of evaluating Q at the the previous time step U^j, we calculate it as follows (in a Crank Nicholson manner):

$$\frac{1}{2}[Q(U^{j+1}_{nm}) + Q(U^j_{nm})]$$

However we do explicitly know U^{j+1} so to get round this we calculate as a truncated power series in one variable.

$$Q(U^{j+1}) = Q(U^j) + \left.\frac{dQ}{du}\right|_{u=U^j} (U^{j+1} - U^j)$$

Obviously in this case Q doesn't solely depend on u it depends on v as well, you could maybe calculate it as two variable power series, but more on that later on. For practicality we treat v as constant calculated from the previous time step. Doing this means we have to make few changes to the Gauss-Seidel routine to cope with the implicit $\left.\frac{dQ}{du}\right|_{u=U^j} U^{j+1}$ term. It now looks as follows:

$$[1 + dk(\frac{1}{h_x^2} + \frac{1}{h_y^2}) - k\frac{1}{2}\left.\frac{dQ}{du}\right|_{\substack{u=U^j \\ v=V^j}}]U^{j+1}_{nm} = A + k[Q_{nm} - \frac{1}{2}\left.\frac{dQ}{du}\right|_{\substack{u=U^j \\ v=V^j}} U^{j+1}] + U^j_{nm} \tag{10}$$

$$\text{where} \quad A = \frac{dk}{2}[\frac{U^{j+1}_{nm+1} + U^{j+1}_{nm-1}}{h_x^2} + \frac{U^{j+1}_{n+1m} + U^{j+1}_{n-1m}}{h_y^2} + \delta^2 U^j_{nm}]$$

A potential improvement on this system could be a two-variable truncated Taylor series which solves U_{j+1} and V_{j+1} at the same time in a Gauss-Seidel like manner. So uses the most readily available U_{j+1} and V_{j+1}, iterating both solutions at the same calculating $Q(U^{j+1})$ like above with the below.

$$Q(U^{j+1}, V^{j+1}) = Q(U^j, v^j) + Q_u(U^j, V^j)(U^{j+1} - U^{j+1}) + Q_v(U^j, V^j)(V^{j+1} - V^j)$$

Some explanation

If we take (8) and (9):

$$v_t = \gamma(b - u^2 v) + d(v_{xx} + v_{yy})$$
$$u_t = \gamma(a - u + u^2 v) + u_{xx} + u_{yy}$$

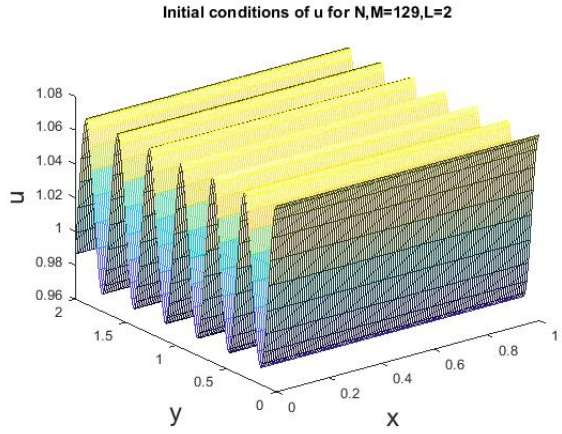
Here $d(v_{xx} + v_{yy})$ represents the diffuse term of molecules, so as d increases, so does the rate of diffusion. γ represents the growing of animal, so larger γ, larger the animal effectively.

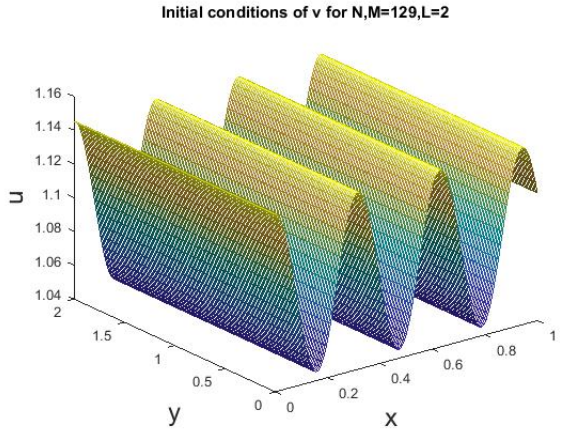
a and b represent a standard increase in the molecule of u and v respectively e.g standard production rate in the skin. $u^2 v$ could represent 2 u molecules reacting with a v molecule to produce a u molecule. With the u term representing u being supplied to the system at increased rate(depending on u). All of this increase as γ increases.

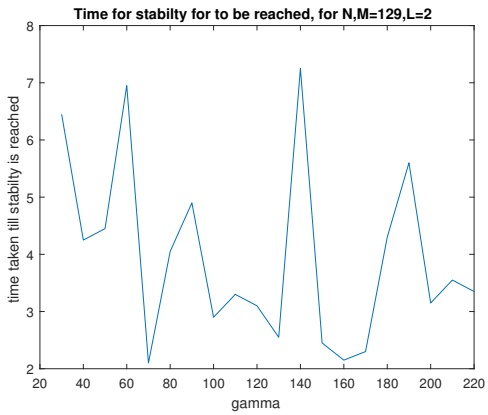
At the end of the day, these u and v will hopefully represent the chemicals in the skin which determine the skin colour or hair colour. So it might be u-v or u or just v which determines this. I am specifically interested in when the system is stable, what patterns occur, this will be stripes,spots, or nothing. I mainly will be focusing on how γ and initial conditions effect this.

When is stability reached

So for a start it would be helpful to know when stability roughly occurs. For example if we look at v, we know t scales with $\frac{1}{\gamma}$ and $\frac{1}{d}$, generally speaking we looking at $\gamma > 10$. It takes longer for the diffusive effects to take effect, but hopefully if stability should occur it is hopefully around t=1 to t=100. The graph below shows roughly when stability occurs for both u and v, for the given initial conditions (IC 1) and γ varying, with time step=0.001.



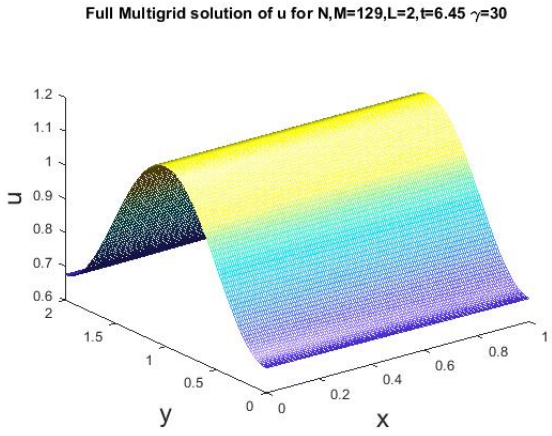


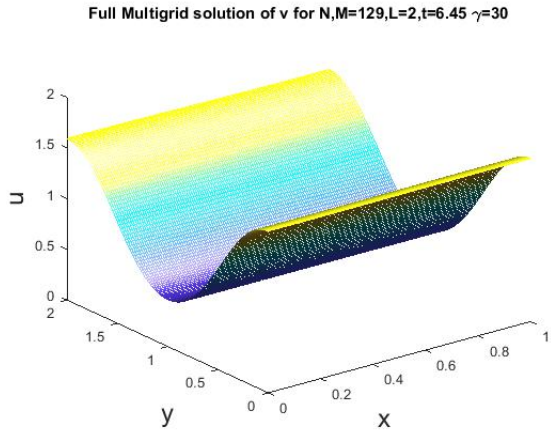


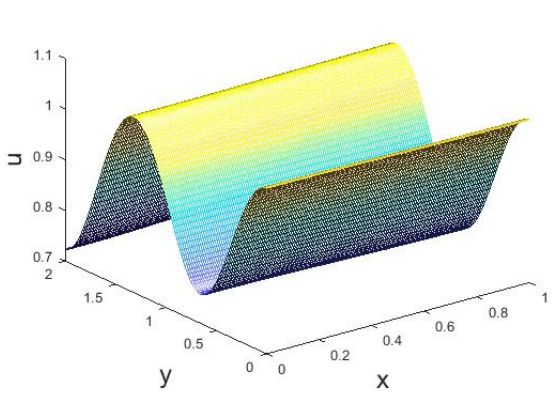
As we can see stability occurs round this 1 to 10 time bracket. With stability occurring at random times, this will be useful when we look at the patterns.

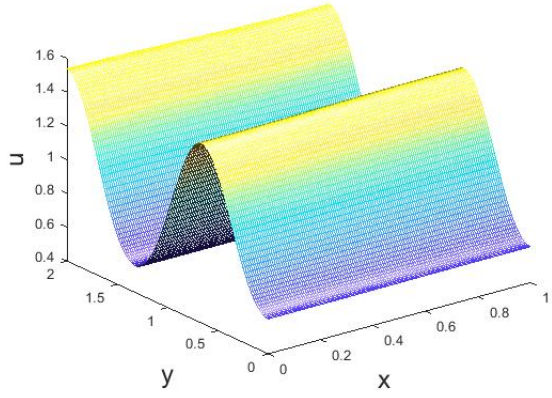
Steady State Patterns

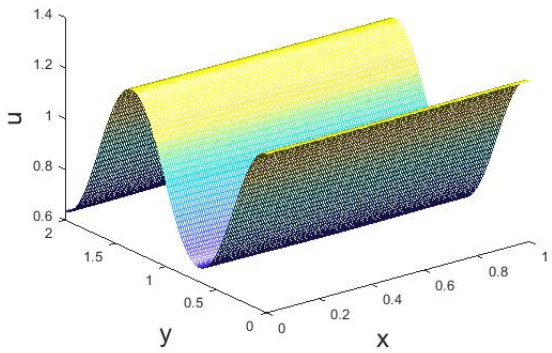
Let us now take a look at the steady states achieved for some the of the times above. For γ less then 20, it does not appear to converge to a steady state for $t < 10$. So we will look at $\gamma > 20$.

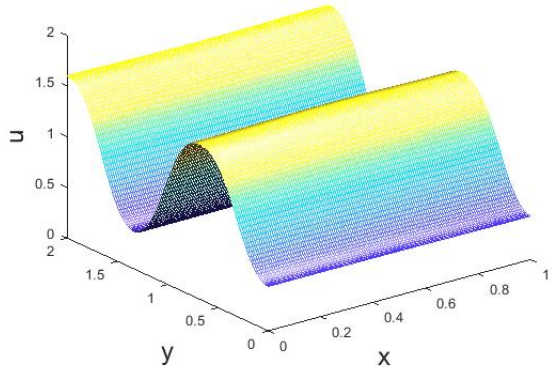


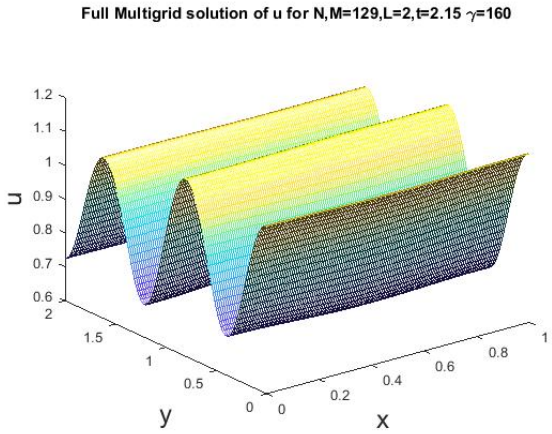


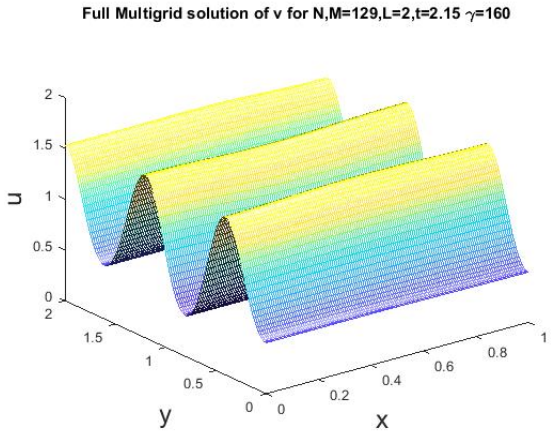




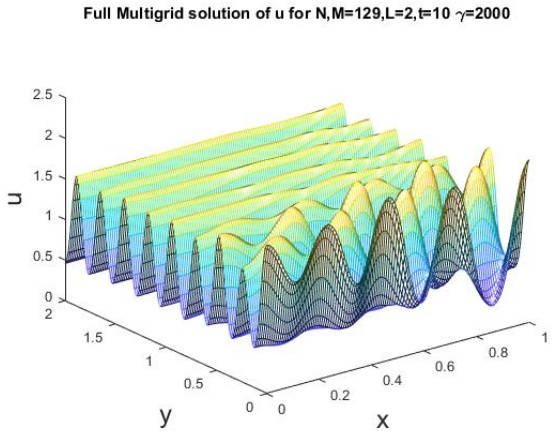


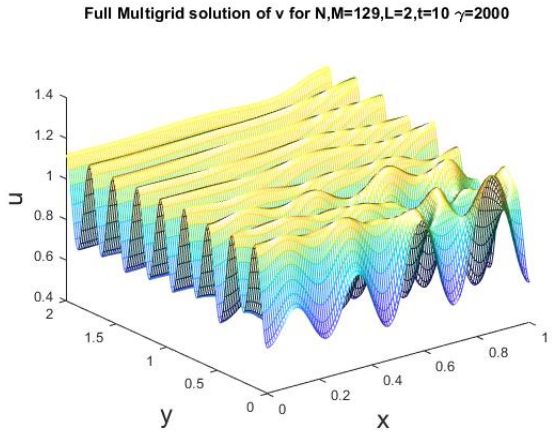






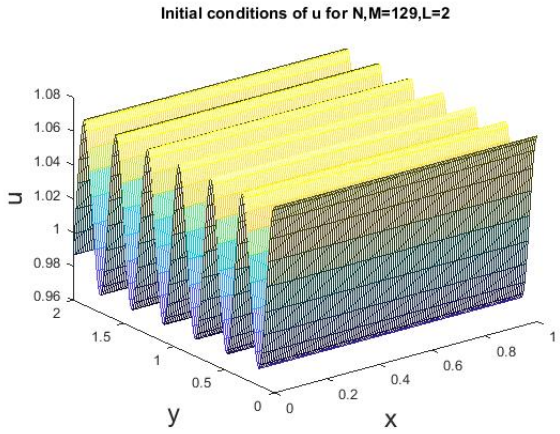
As we can see we are clearly getting stripes, with more stripes as γ increases. I selected the first multiple of ten of γ for which more stripes are seen. For later comparison I have also included a very high value of γ, for a fixed time.

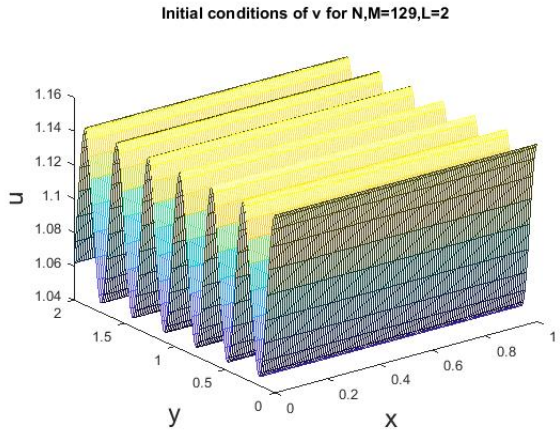


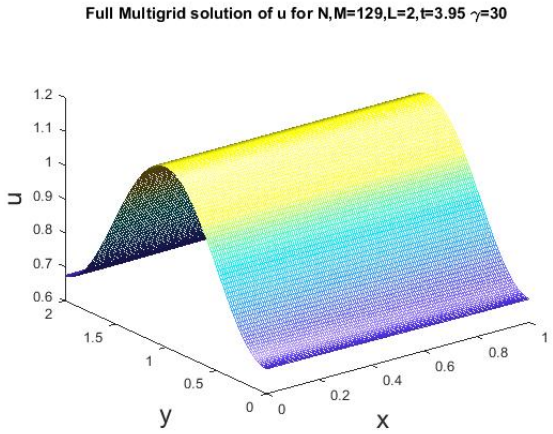


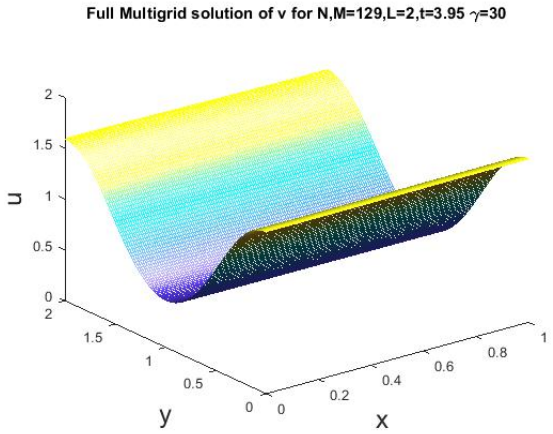
As we can see it does not yet seem like it has fully converged to stripes, even though time is quite large relative to before. This could be to do with the fact that the high value of γ means that the system oscillates a lot and so the diffusive terms take longer to take effect.

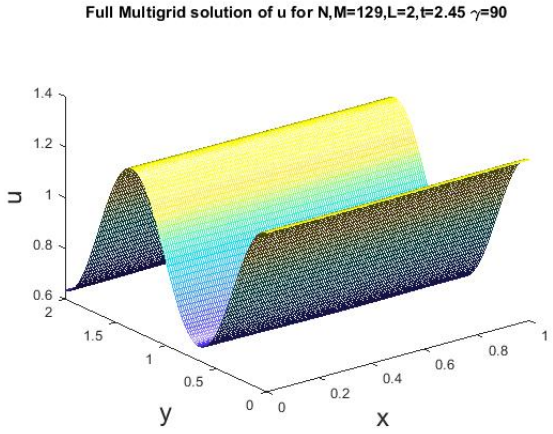
Now let us alter the initial conditions slightly to the following (I.C 2).

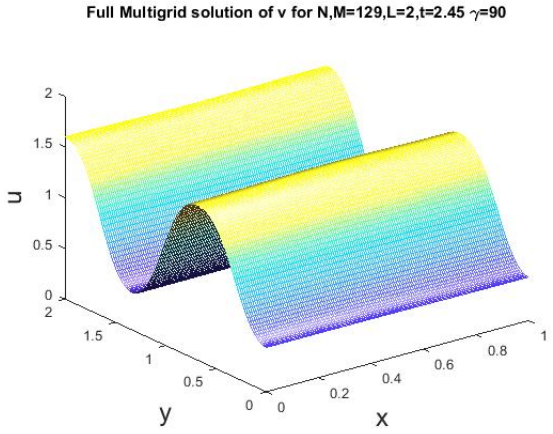




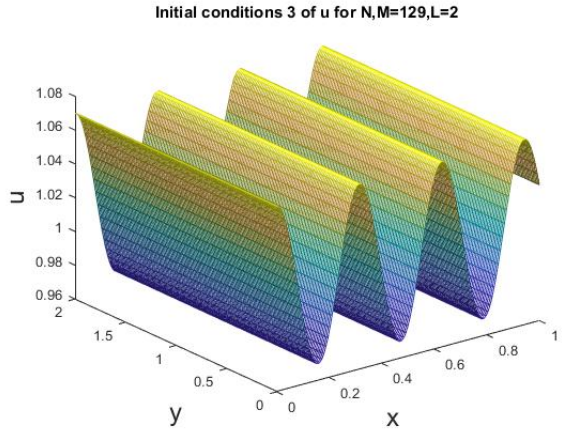


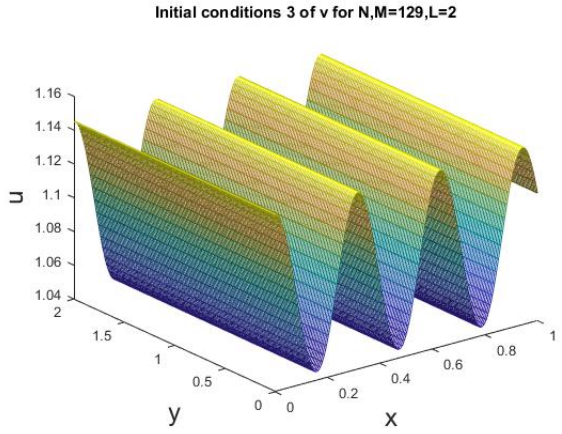


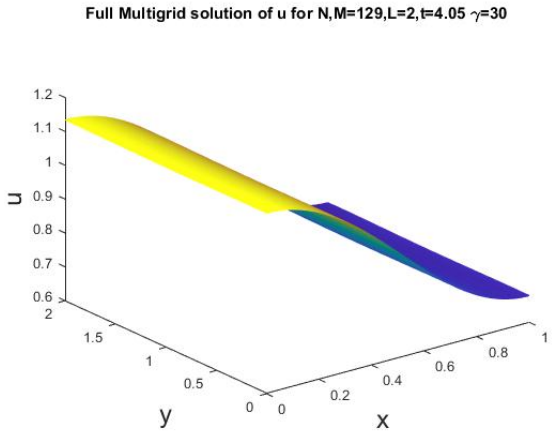


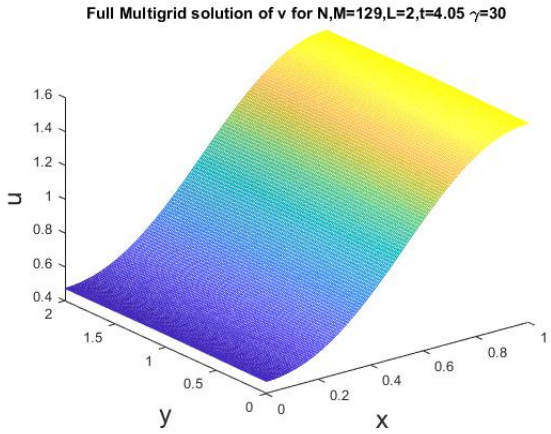


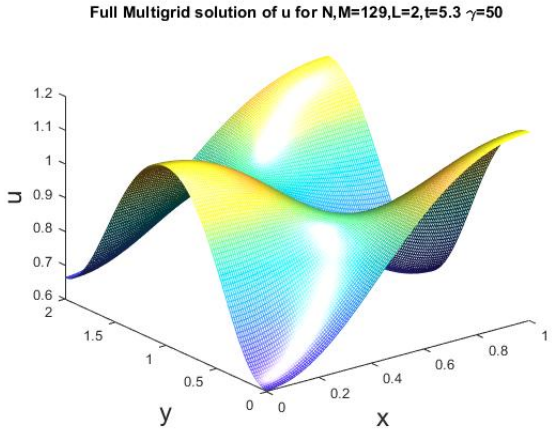
As we can see we are getting virtually the same results as before. Now let us alter the initial conditions again to the following (I.C 3).

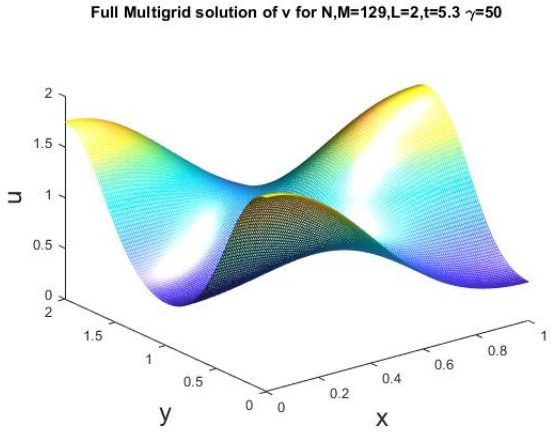


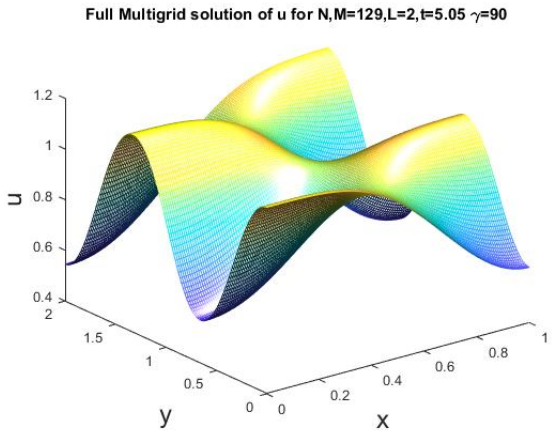


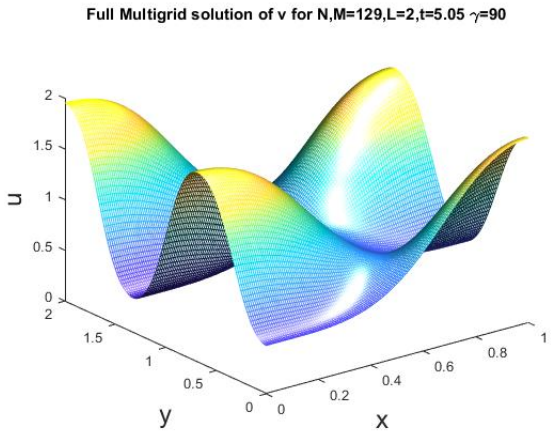






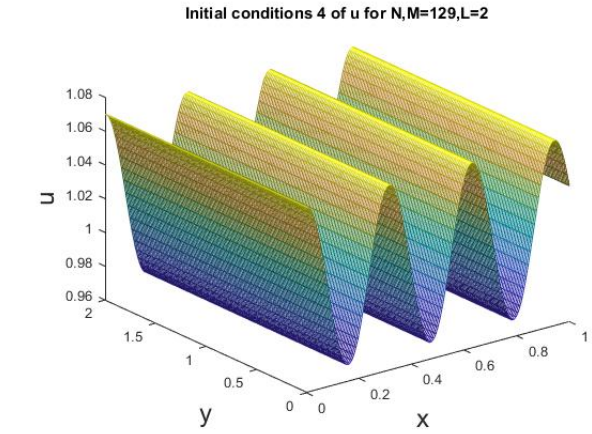


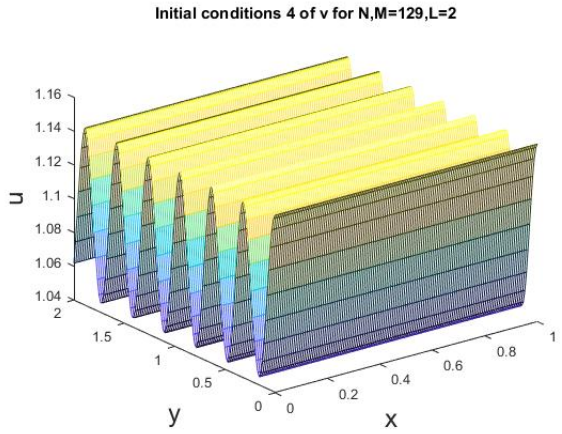


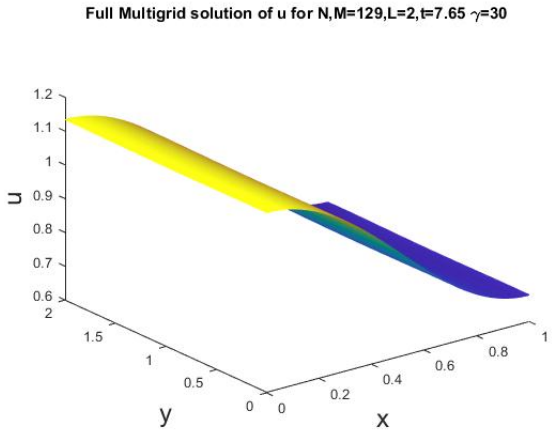


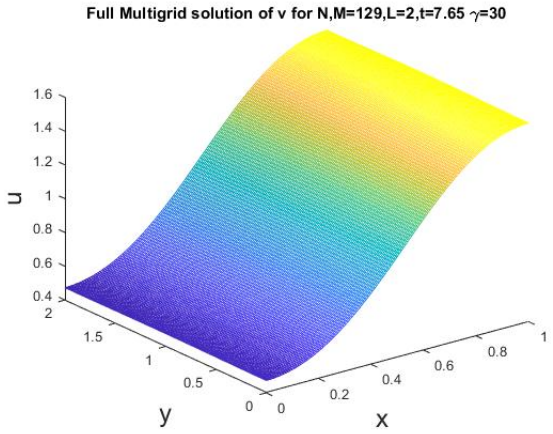
Now this is very interesting as we are no longer getting just stripes we getting spots as well. After some investigation it appears that we keep alternating between stripes and spots depending on γ but the larger the γ the more spots or stripes we have. If for example we have 2 stripes for initial conditions 1 and 2, then we appear to get 4 individual spots. Sometimes the solution appears to be between forming spots and stripes, but these take a long time to converge and often do no give a nice pattern at the end, I have omitted these as they take too long to converge and thus produce a graph for.

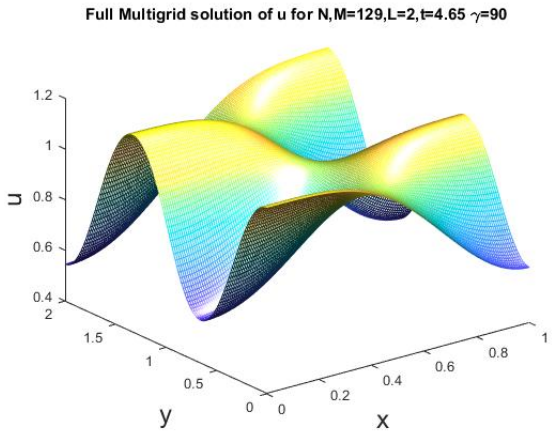
Again let us use the following initial conditions (IC 4). It appears that we get get the same as for IC 3.

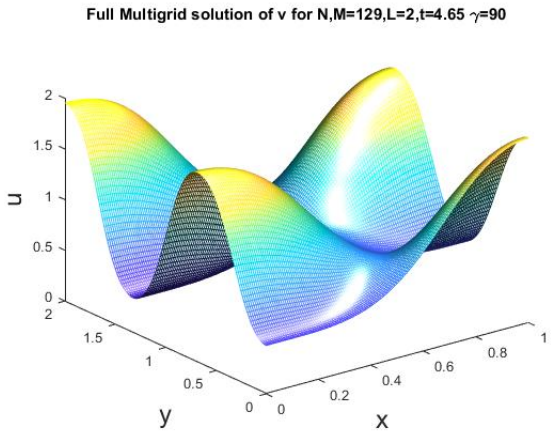






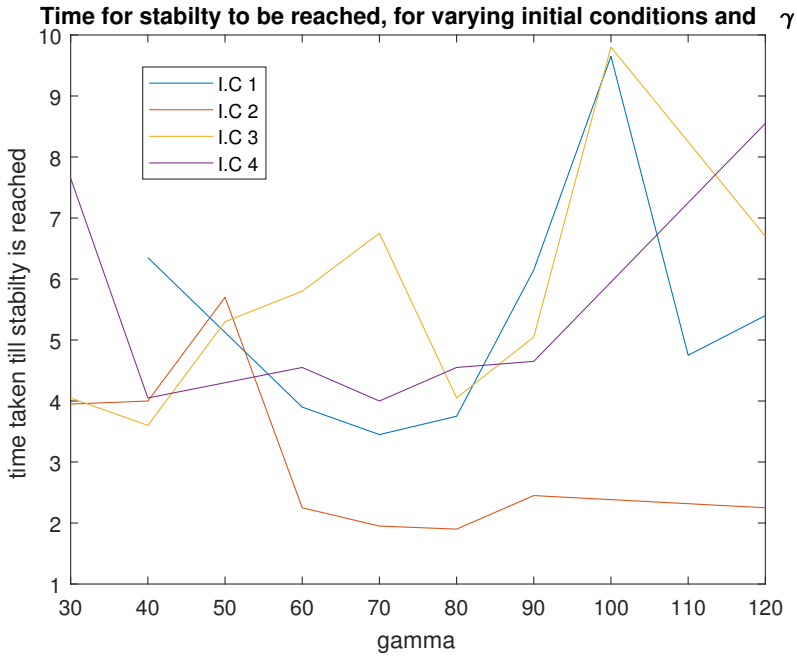






So after this I think we can maybe conclude that if the u initial condition is perturbed enough in the x direction (in a certain way), you can achieve spots for the correct γ.

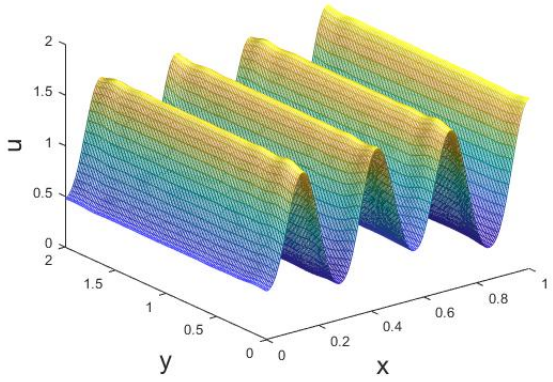
Here is also a graph of showing how long each system took to become steady, one which took longer then t=10 I have omitted.



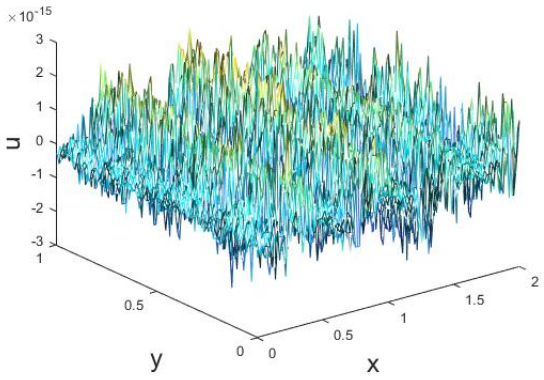
High values of γ

An interesting area is for high γ and t large, seeing how they change for small t. I will show graphs just for u , the v graph is always very similar to the u graph in the sense the gradient of u at a point is equal to the negative gradient of v. The following have been created using I.C 1 and a time step 0.00025.

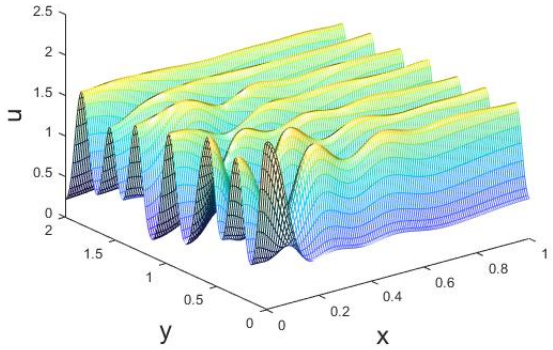
Full Multigrid solution of u for N,M=129,L=2,t=3 γ=1400



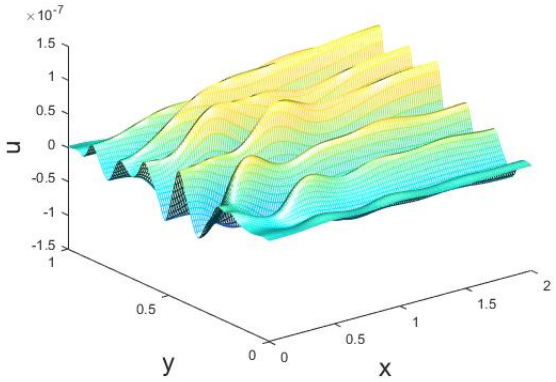
difference for u between t=3.0000 and t=2.9975 for N,M=129,L=2, γ=1400



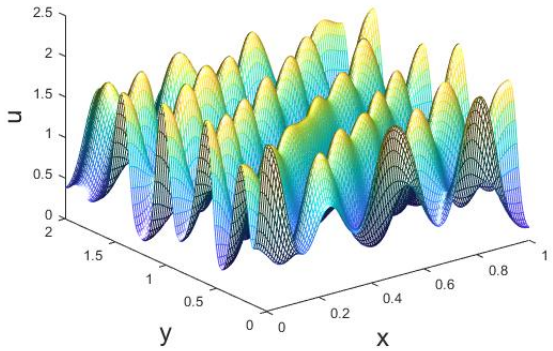
Full Multigrid solution of u for N,M=129,L=2,t=3 γ=1450



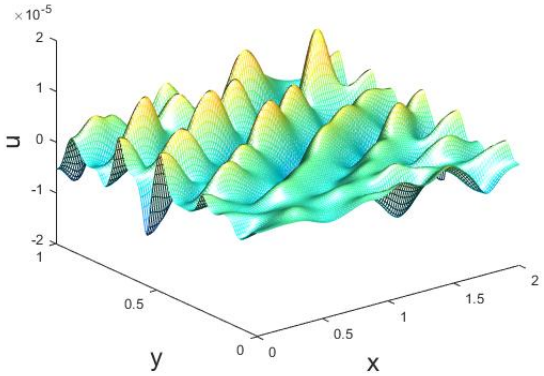
difference for u between t=3.0000 and t=2.9975 for N,M=129,L=2, γ=1450

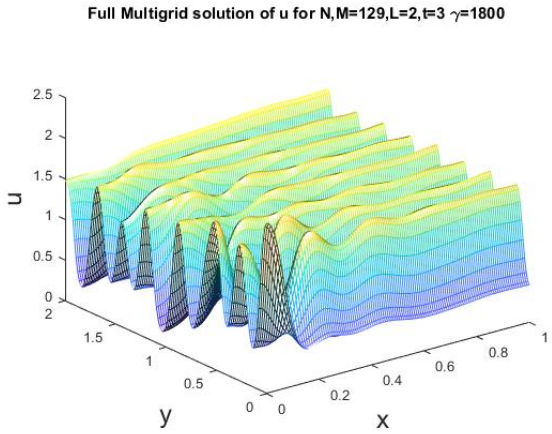


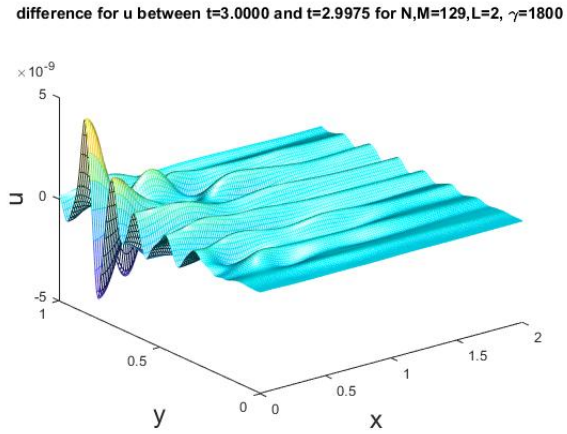
Full Multigrid solution of u for N,M=129,L=2,t=3 γ=1650

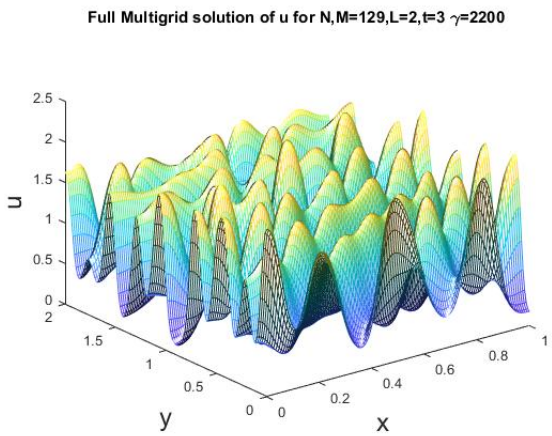


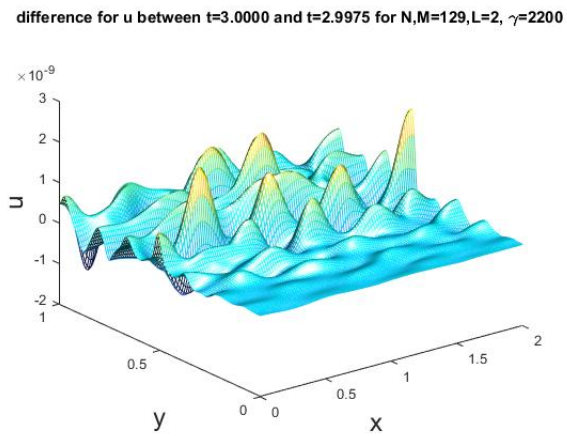
difference for u between t=3.0000 and t=2.9975 for N,M=129,L=2, γ=1650

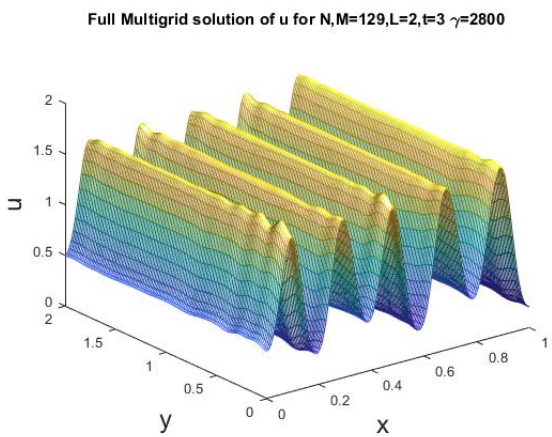


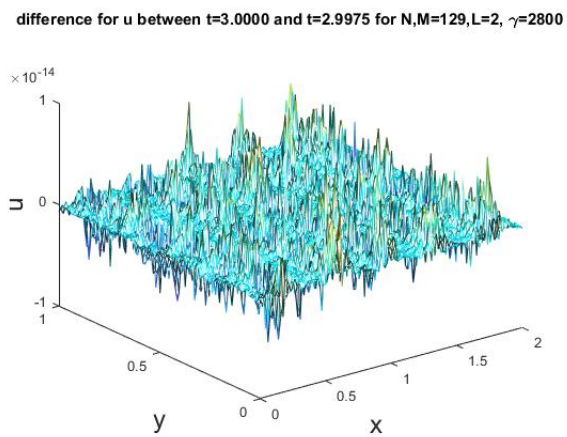












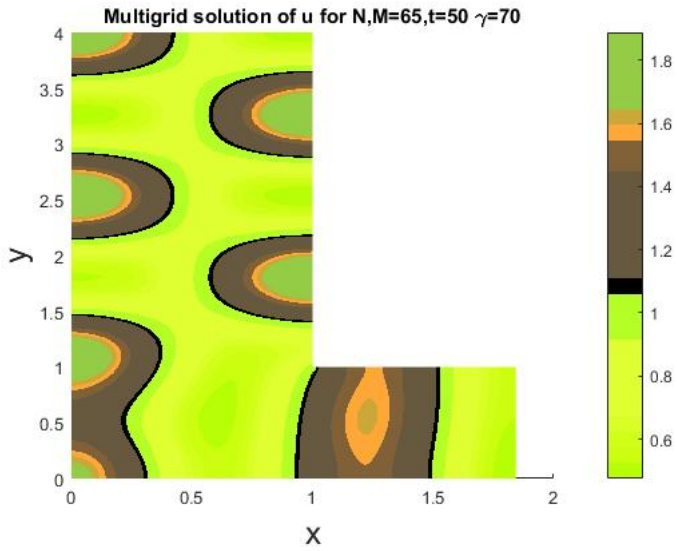
As we can see we keep getting that cyclic structure of γ with it going from stripes to spots, which aren't seeming to converge or sometimes in between. This is unexpected as for low γ we did not get spots just stripes for these initial conditions. This might be due to the our scheme might not be working properly, as shown in test 2 as time becomes large, so does the error. Especially when the change over a period of time is less then 10^{-8}.

Domain decomposition

Now we wish to use our scheme solve not just in a rectangle but say 2 overlapping rectangles joined together to form an L. To do this we solve in one rectangle and then use the data to calculate the boundary conditions for the second, which we then solve. We then solve the first rectangle with altered boundary conditions calculated from second solution. We then iterate over this until the solution is stable (important to note when we solve we still have the same initial condition just altered boundary conditions). The idea of this very simple, but implementing it can be very fiddly due to how you have combine the data. I have written some MATLAB code which implements this.

Hopefully this will allow us to model a limb or tail attached to a body say. In some instances in nature these occurs and you get spots on the body and stripes on the tail. We want to be able to produce this.

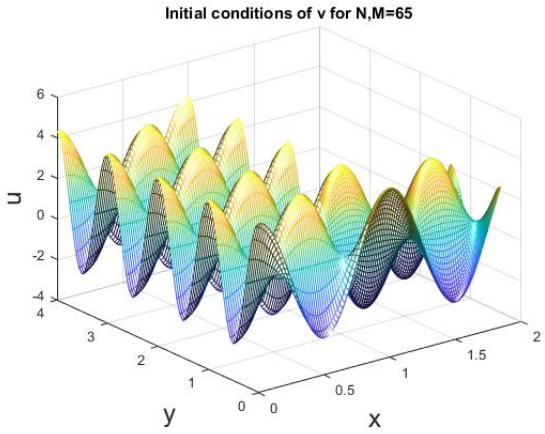
For example shown below is the Northern leopard frog (scientific name: Lithobates Pipens). reference (http://www.naturalhistoryonthenet.com/Amphibians/NorthernLeopardFrog.htm). We can model it using this method domain decomposition, along with a custom color map.

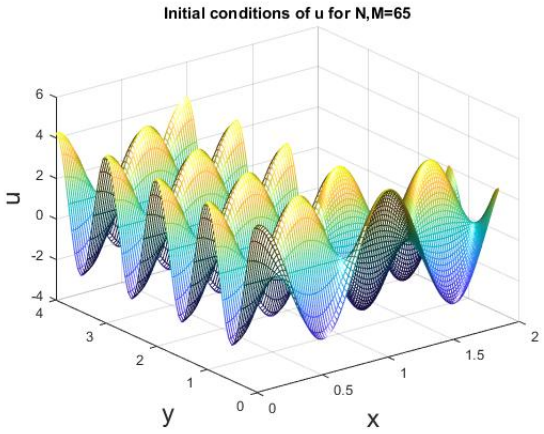


Here N,M refer to the number of points over which each rectangle was solved.

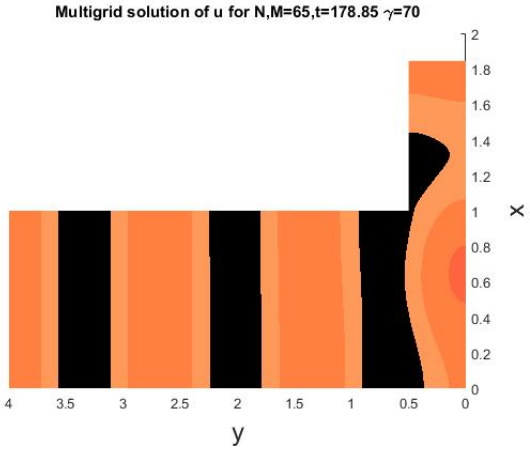
This solution, as we can see could potentially be modeling the body and leg of the frog. The system has stabilized, but it took a lot longer then normal, this might be due to the fact it solved on a bigger area and so the diffusive terms term took longer to take effect. As well as the modeled limb solution diffusing across the rest of the system.

Also the initial conditions used to create the above are given below (which are perturbed in the x-direction for u).





I written the code in such a way that I can alter the width of the modeled limb, if we half it for the same γ we appear to get stripes, which could model as perhaps the below a tiger. This is maybe because of the way the limb interacts with the solution.



So for larger γ we would perhaps expect more stripes or spots. One of the impracticalities of this methods is that for higher γ we struggle to calculate the solution, as we have to reduce the time step to cope, along with all the other factors like having to iterate at every time step. means in reality it takes too long to calculate.