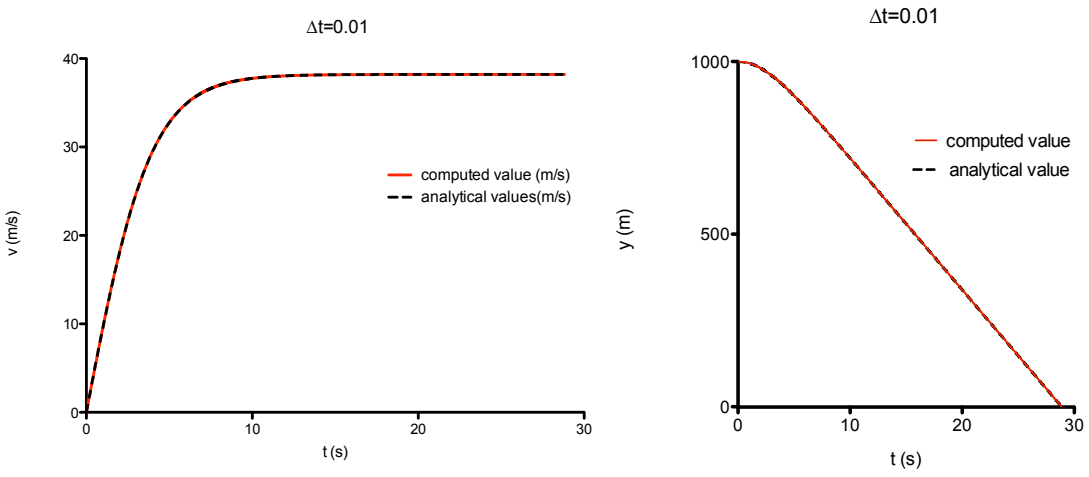
Exercise 3

Problem 1: Free-fall with constant drag factor

→Write a program to model the free fall of a projectile under gravity experiencing a resistive force equal to its velocity squared times a drag factor. Using the Euler method, calculate y(t) and $v_y(t)$.

→My code starts by defining and calculating the drag factor, $k = \frac{C_d \rho_0 A}{2}$, and the mass of the object to be 100kg. Then enters a loop that runs until the projectile reaches the ground. The velocity of the projectile is computed from the previous velocity plus the acceleration (due to the forces) times a small time step Δt. The y coordinate is computed from the previous y coordinate plus the velocity times Δt. The code prints to screen these values every 0.5 s for analysis. The program exits the loop once y becomes negative, therefore there will be a slight overshoot at the end. The time of impact can be found by the final time minus the overshoot dived by the velocity at the last step. The program then calculates the analytical values for comparison.



→

Figure 1.1 – graphs showing v(t) and y(t) both the analytical values and the values computed using a time step of 0.01

The speed of the projectile was found to increase rapidly before levelling off and reaching a roughly constant value as can be seen in figure 1.1. The computed values tracked closely the analytical values for small time steps; the effect of the size of Δt on the difference between computed values and analytical values can be seen in figure 1.2. Unfortunately I could not demonstrate the effect of making Δt too small (with floats as well as doubles) as my code for printing results only every 0.5 seconds failed to work when rounding errors were a factor, and without this code millions of data points were produced which were un-plottable and made my computer crash.

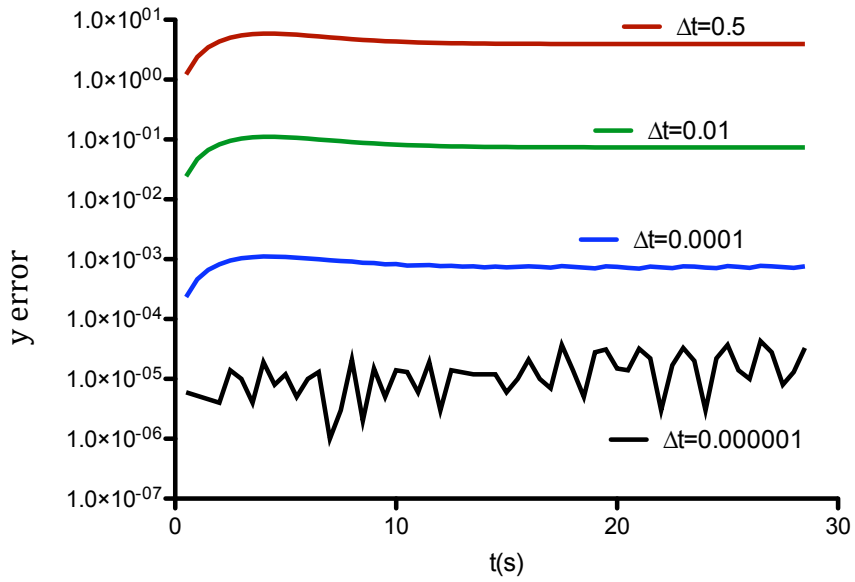


Figure 1.2 shows the precision of y for different Δt sizes.

The effect of having Δt too large can be seen in figure 1.3. It seems that the error in v is largest where the change in gradient of v is largest.

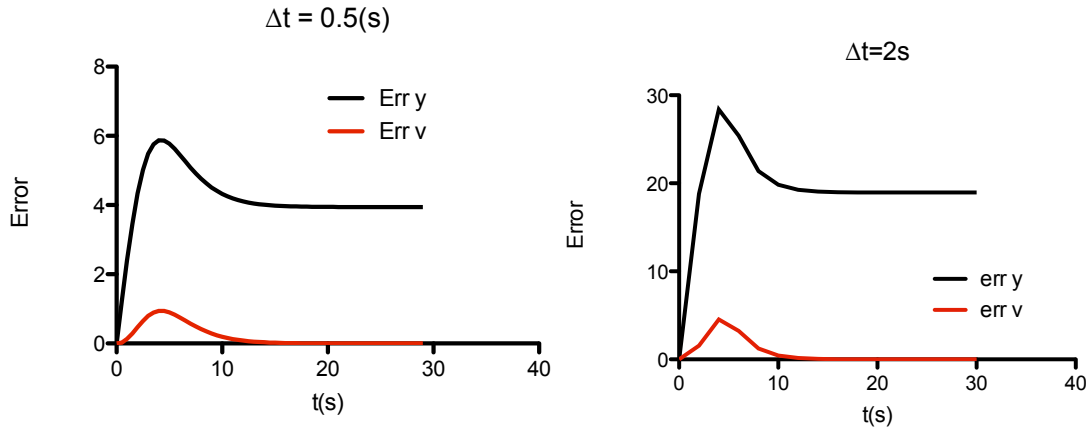


Figure 1.3 shows the y and v errors when Δt is large.

The effect of the size of the drag factor can be seen in figure 1.4. For large k the particle reaches terminal velocity very quickly, as can be seen with the blue line not appearing to bend at all, for small k the particle never reaches terminal velocity before it hits the ground as is seen in the black line. The

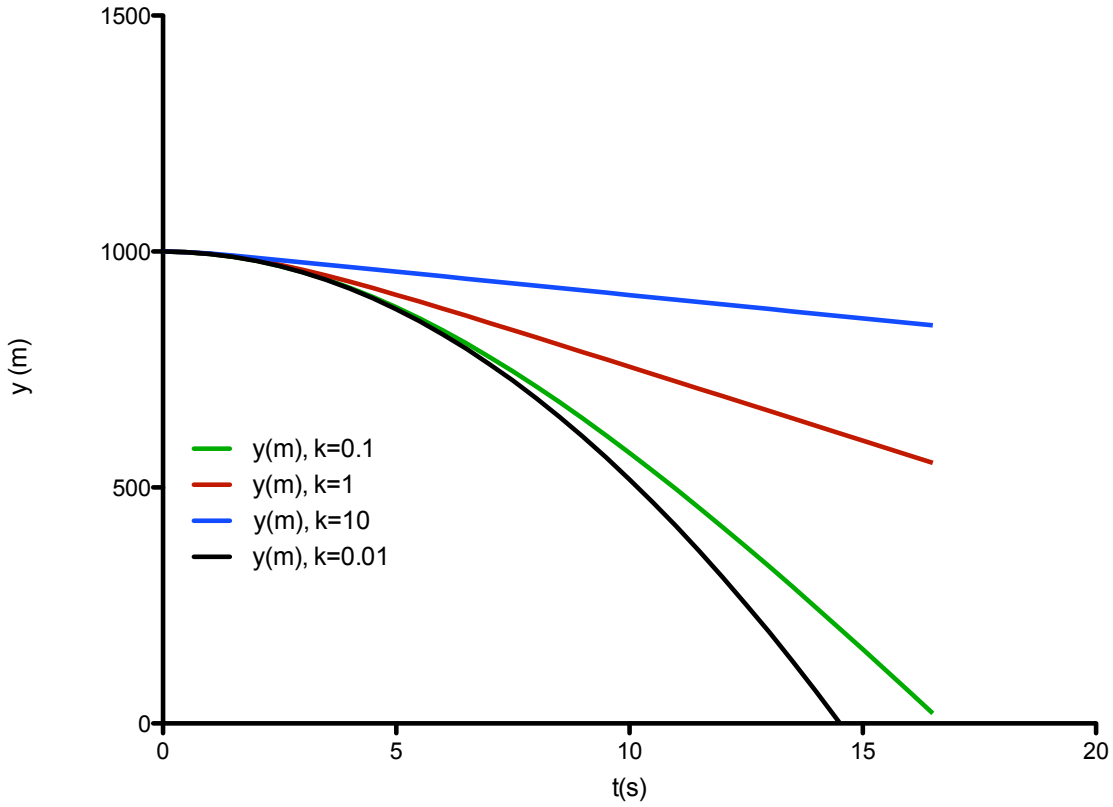


Figure 1.4 – y(t) for various magnitudes of drag factor. The impact time for each: 14.517599, 16.734311, 34.140611, and 101.663603.

Problem 2: Free-fall with varying drag factor

→Alter the program from problem 1 to model Felix Baumgartner's free fall from 39045m.

→Setting the variables for drag factor to approximately that for this jump. The mass of his suit was 119kg(according to http://newsfeed.time.com/2012/10/08/felix-baumgartners-23-mile-freefall-attempt-by-the-numbers/) he probably weighs about 70kg so the total mass M = 190kg. Drag coefficient = 1, roughly the same as a skydiver. Cross sectional area = $1.2m^2$, an estimate. Since the density of the air is much lower at 39045 and increases as he falls I have replaced the constant air density with a function $\rho(y) = \rho_0 \exp(-y/h)$ where h is the scale height of the atmosphere.

→With these parameters my program calculates his maximum speed to have been 325.98 m/s, achieved at a height of 25960m, the v(t) graph is shown in figure 2.1. At this height the speed of sound is 316m/s. Lower than at the earths surface due mainly to the decreased temperature at this altitude (figure 2.2).

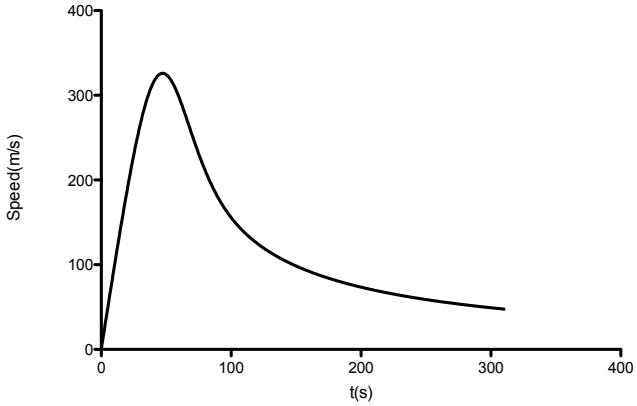


Figure 2.1 – Speed – time graph for modelled Baumgartner's descent.

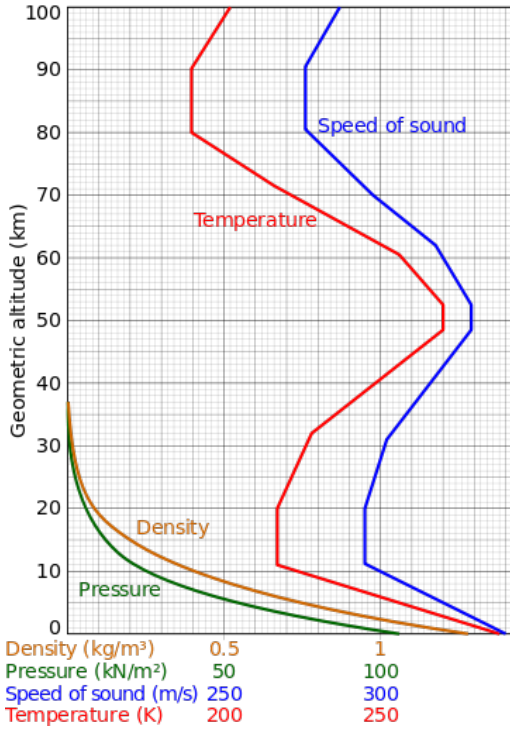


Figure 2.2 – a graph showing how the speed of sound varies with altitude.

Therefore my program predicts the he did indeed brake the sound barrier.

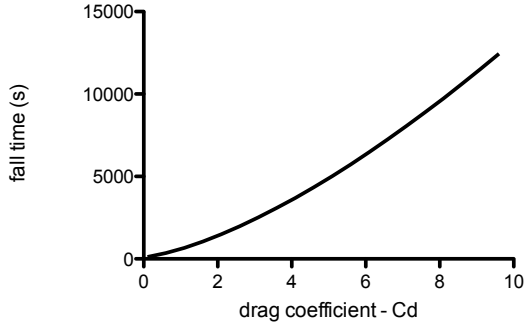


Figure 2.3 - a graph showing the relationship between the total fall time and the drag coefficient

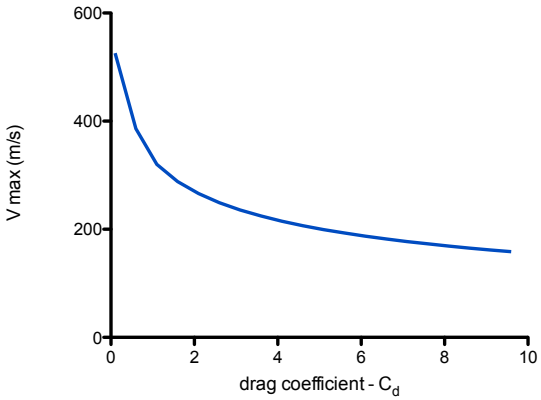


Figure 2.4 – shows the relationship between the top speeds reached in the jump and the drag coefficient.

Looking at how the jump is affected by the drag coefficient/ the cross sectional area, we can see that the total fall time is increased roughly linearly, the top speed achieved in the jump decreases sharply at first with an increased C_d then begins to level out. This is because at the start of the jump the air density is close to zero anyway so he is experiencing very little drag, allowing him to reach a fairly high speed before drag slows him down. Increasing the mass increases the max speed considerably as shown in in fig 2 .5. Increasing the jump height had the same affect.

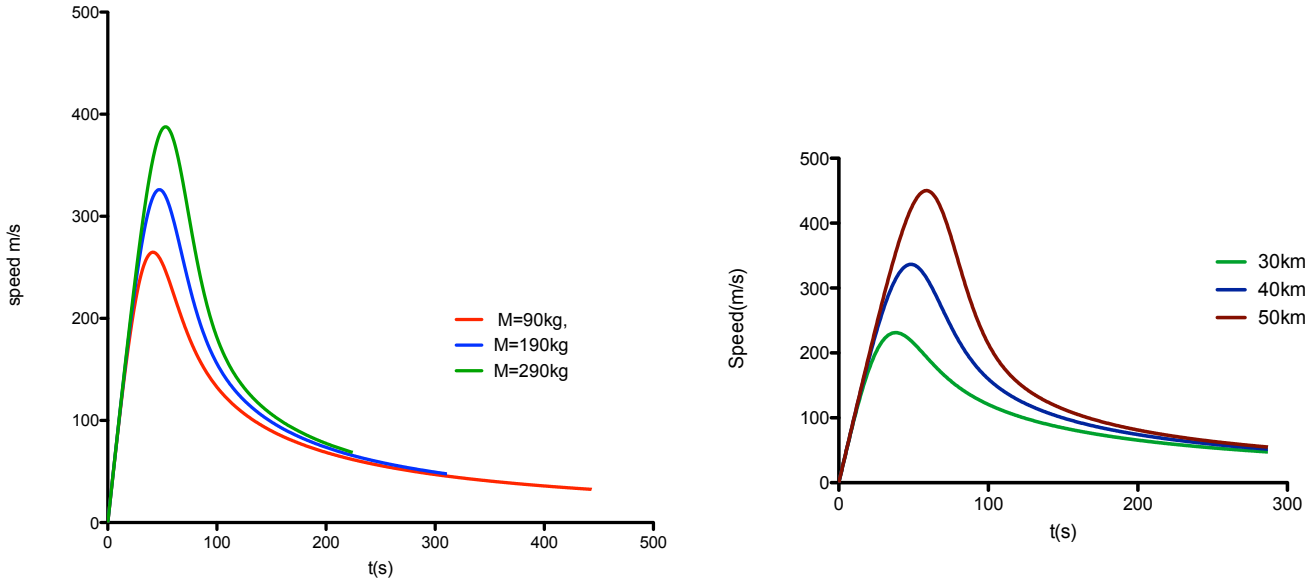


Figure 2.5 – Speed time graphs for various mass projectiles, and for different jump heights.

Problem 3: Electric Fields and Potentials

→This program applies a relaxation method to determine the electrostatic potential around the two conductors shown in figure 1. The region is split into points; the

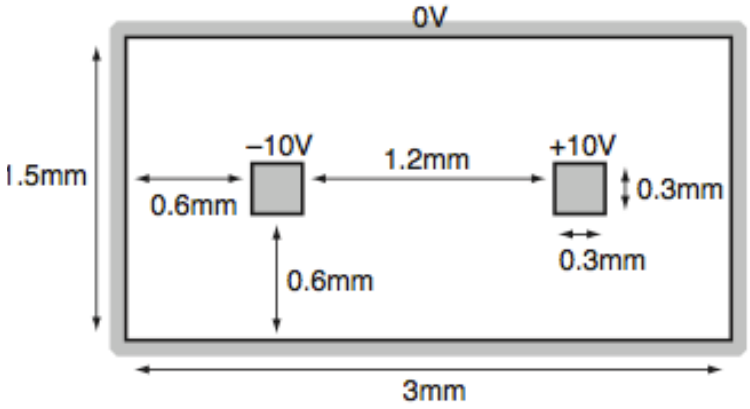


Figure 2 – Diagram of the region being modelled.

potential at any point can be calculated from the average of the potentials of the points neighbouring it on four sides. The electric field is E=-gradV, the x and y components of electric field have been calculated for each point.

→My program splits the region up into 0.1mm blocks there are 30 in the x direction and 15 in the y. Two (double) arrays are declared; these hold the potential values at each point. The potential of the conductors is set to +10V and -10V, the rest of the points are set to zero. The potential at each point is then calculated by averaging the points on four sides. The code runs through the points that are not on the borders, then through each of the borders and corners separately since these latter require only 3 and 2 points respectively in the sum since the boundary is fixed at zero potential. At each iteration the new potentials are calculated and stored in one array, Vn, from the same values in another array, Vo. The largest difference between elements in these two arrays is found and this is used as a convergence test.
The x and y electric field components are calculated for each point and stored in 2 separate arrays. They have been calculated by taking the V values of the points above and below (for y component) and to the left and right (x component) finding the difference and dividing by the distance, 0.2mm.

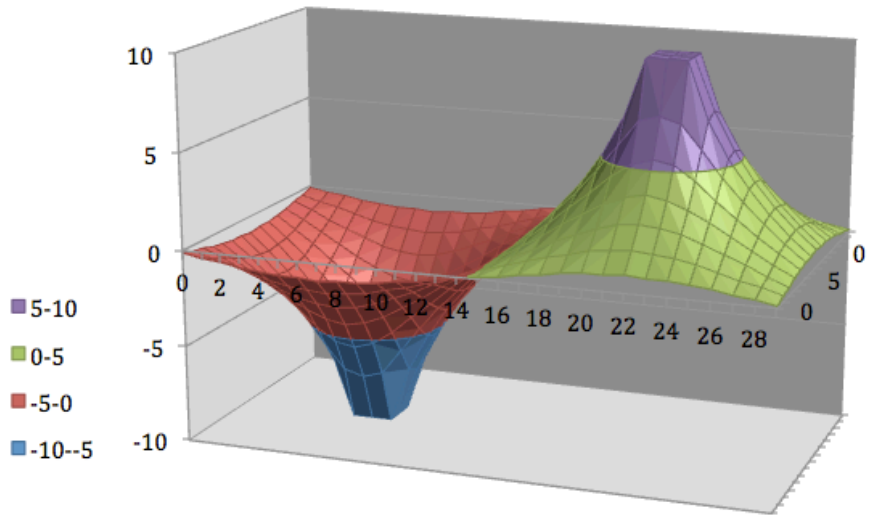


Figure 3 - The surface plot of the converged potential of the region.

→Figure 2 shows the converged (to 0.001V) potential. With the points all initially at zero (except the conductors) the relationship between the required convergence and the number of iterations require to achieve this is shown in figure 3.

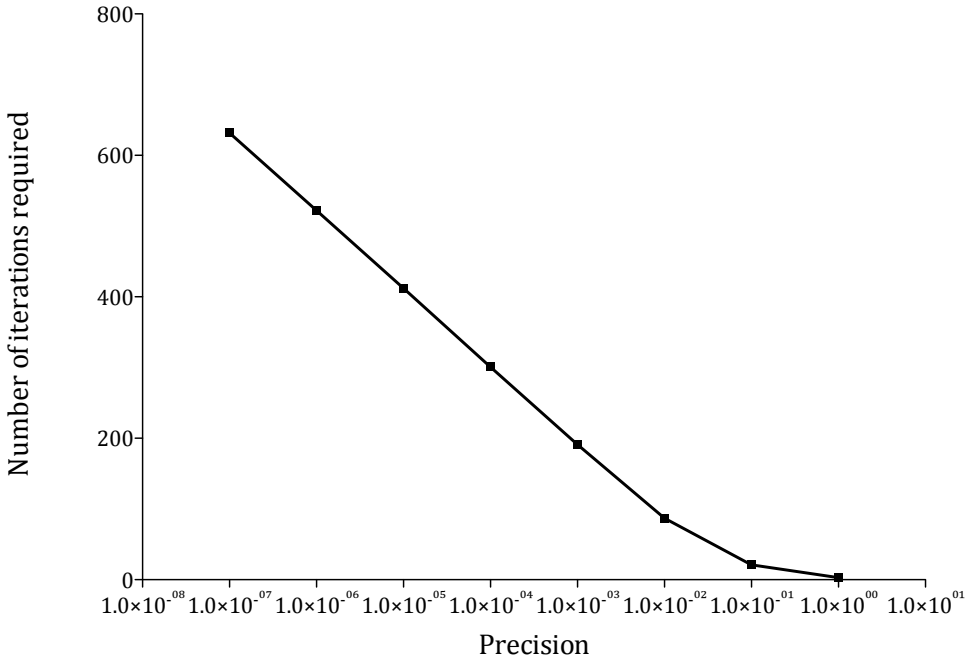


Figure 4 Shows the number of iterations required to achieve a precision. The x is a logarithmic scale.

The x components of the electric field are shown in figure 4, and the y components in figure 5.

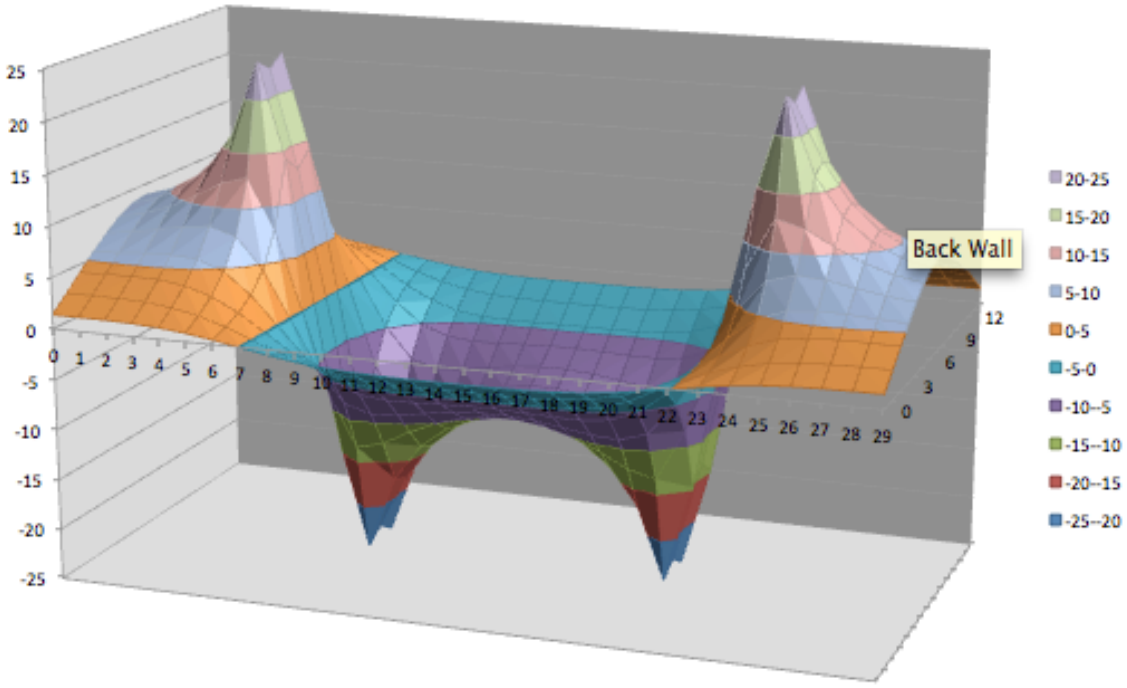


Figure 5 – A surface plot of the E_x component. Positive values indicate vectors to the right, negative to the left.

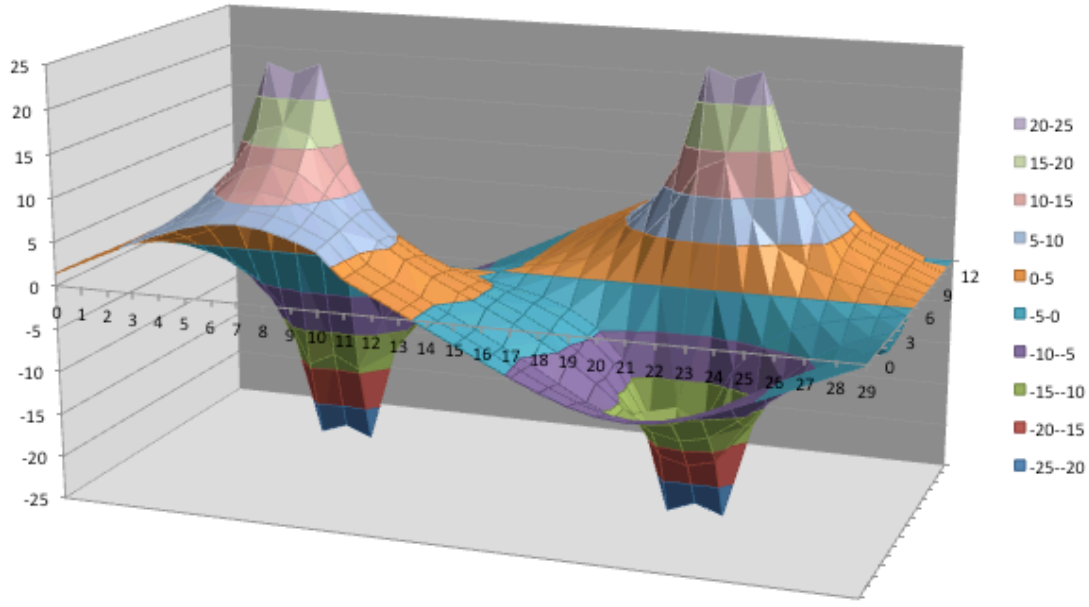


Figure 6 – A surface plot of the E_y component.

These plots show that the electric field is strongest close to the conductors, roughly zero at the middle of the region where the two fields meet.