UNIVERSITY OF MARY
SCHOOL OF ENGINEERING
ENR 338: Final Project
Name: Leo Devick

1 Introduction

We are given the problem to analyze the flow of heat in a 50 by 100 inch steel plate that is being welded. The position of the weld pool changes through time, and our analysis will show the diffusion of heat through the steel plate the welding arc goes along the edge. Later, we investigate welding in the center of the plate as we attempt to repair a hole in the sheet of steel.

2 Analysis

2.1 Material Properties

Stainless Steel 304 is an alloy of iron, carbon, chromium and nickel. It is the most widely used type of stainless steel, from use in mining equipment to commercial kitchen uses. Its properties relevant to this simulation are presented in table (2).

Table 1: Properties of Stainless Steel 304

Property	value
Melting Temperature (C)	1450°
Thermal Conductivity	16.2 W/mK

The welder itself is a TIG Stick welder with an operating temperature of 3500 degrees Celsius. It will create a pool of molten stainless steel $\frac{1}{4}$ of an inch long, so we will use a virtual plate that is 400 by 200 fourths of an inch, since our real plate is 100 by 50 inches.

The welder's speed is customizable, but for our base simulation it moves at a rate of $\frac{1}{4}$ inches/second.

We also find that room temperature is 20 degrees Celsius.

2.2 Numerical Simulation

Using the C Programming Language, I wrote a simulation of our problem.

diffusion.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int usage(char* name){
 printf("Usage: %s <welder speed> <plate
    W> <plate H> <room Temperture> <dt> <T
    Final>\n", name );
 printf("Example: %s 1 50 100 0 0.01 5\n"
    , name );
 return 1;
}

int main(int argc, char** argv){

 if (argc != 7){
  return usage(argv[0]);
 }

 float dWelder = atof(argv[1]);
 int X = atoi(argv[2]);
 int Y = atoi(argv[3]);
 float roomTemp = atof(argv[4]);
 float dt = atof(argv[5]); //Stepsize in
    milliseconds
 float TFinal = atof(argv[6]);

 float D = 16.2; //diffusion coefficent

 float T[Y][X];
 float TNew[Y][X];

 float t;

 int i, j; //indicies

 float xWelder = 1;
 int yWelder = 0;

 float tWelder = 3500.0;

 // Set the whole plate to room
    temperture
 for (j = 0; j < Y; j++){
  for (i = 0; i < X; i++){
   T[j][i] = roomTemp;
  }
 }

 //time step
 for (t = 0; t < TFinal; t += dt) {


  T[0][(int) floor(xWelder)] = tWelder;

  if (xWelder < X){
   xWelder += dWelder * dt;
  }

  //Updating all inner grid points
  for (j = 1; j < Y-1; j++){
   for (i = 1; i < X-1; i++){

    TNew[j][i] = T[j][i] + D*(T[j+1][i] +
     T[j-1][i] + T[j][i+1] + T[j][i-1] -
    4.0*T[j][i])*dt;

   }
  }


  //Updating T to be TNew
  for (j = 1; j < Y-1; j++){
   for(i = 1; i < X-1; i++){
    T[j][i] = TNew[j][i];
```

```
    }
   }


  }


  // printing out final state of the plate
  for (j = 0; j < Y; j++){
   for (i = 0; i < X; i++){
    printf("%f ", T[j][i] );
   }
   printf("\n");
  }


  return 0;
}
```

I also customized the provided gnuplot script to get an up close view of the steel.

closeup.gnuplot

```
set term x11

set title "Map of the temperature of a
    steel plate"
set xlabel "Grid X"
set ylabel "Grid Y"
set cblabel "Temperature"
unset key

set yrange [0:39] # NY-1
set xrange [0:39] # NX-1

set palette rgbformulae 33,13,10 # A
    heatmap palette
set pm3d map # Use pm3d to create a 2D
    color map

splot 'out.dat' matrix with image
```

2.2.1 Quirks of the simulation

Programmers and those accustomed to C might notice we use the convention `T[j][i]`, where `j`, being the y axis, comes before the `i` coordinate. This is intentional. Observe a basic 2D matrix written in a C array.

```
{ {1,2,3},
  {4,5,6},
{ 7,8,9 } }
```

If we imagine that the coordinate $(0, 0)$ has a value of 1, we would access the element programatically using `array[0][0])` and all is well. However, if we wanted to access the element directly to the right of 1, 2, using Cartesian coordinates we would write $(1, 0)$ and in C we would type `T[0][1]`. This comes from the nature of arrays in C. The first set of square brackets refers to the "outer" array, and in the visual example that is our y coordinate. As we increase the first index, we go "down" in arrays along the y axis. Likewise, the second index refers to the x coordinate of the "inner" array. Elements are on the same horizontal axis when they have the same x (or `i`) value.

I also decided to only output the result of the simulation once the simulation had run for `TFinal` seconds. I wrote a driver script to output the results of the simulation through time, and this effectively "animated" the results.

driver.sh

```
#!/bin/bash

gcc diffusion.c -lm -o diff

for (( i=1; i<10; i++ ))
do

 ./diff 1 400 200 32 0.01 $i > out.dat

 gnuplot -p closeup-png.gnuplot > plot_$i
    .png
done
```

2.3 Analysis

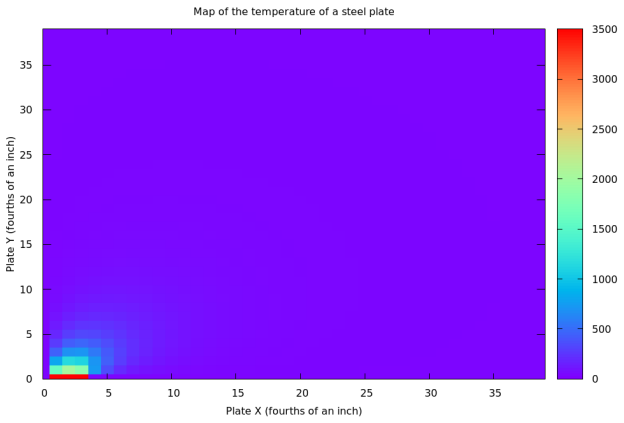

Figure 1: Closeup of the weld point beginning to move through the plate at T=10

We find a few interesting things about welding the steel plate. First, the hottest point on the plate is the tip of the welder, and heat radiates outward in a parabola like structure. As the welder begins its journey across the long edge of the plate, the heat is more bunched up and does not spread out as much (Figure 1). Once the arc

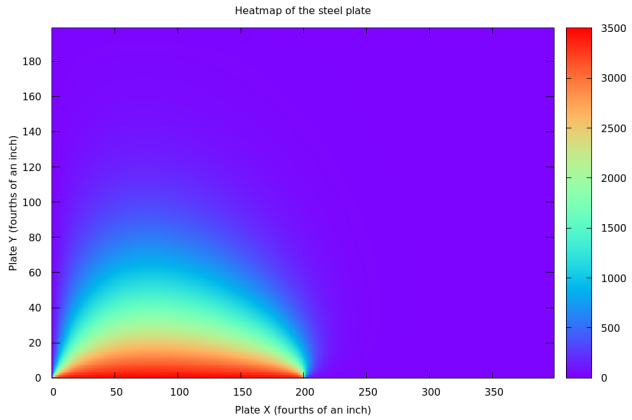

Figure 2: Heat of the plate at t_{half}

is away from the short edge, we see the heat diffuse out more (Figure 2).

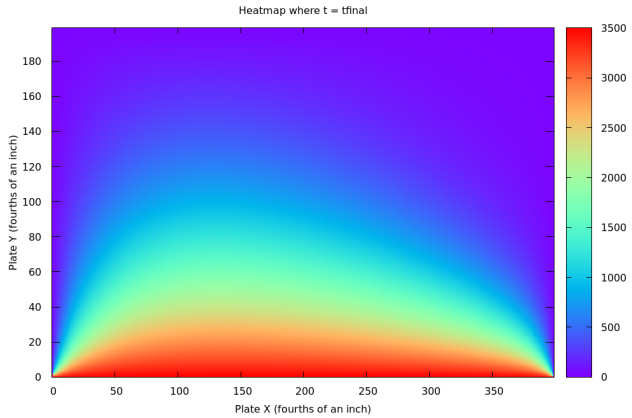

Figure 3: Heat of the plate at t_{final}

Finally, we can see the heat spreads halfway across the plate at $t = t_{final}$ (Figure 3).

3 How we can improve our model

Our model is a good start, but for industries like space or deep ocean exploration, we would need to add some additional details to make sure we're getting as close of a physical approximation as possible. One issue is that our sheet of metal doesn't have depth. As we run the welder across the sheet, we are not sure how the heat dissipates throughout the thickness of our steel. From what we've learned from our 2D analysis, I hypothesize that the middle of the sheet would be warmer than the outer layers.

Another observation is that room temperature might fluctuate during the welding process. A welder welding on a windy outdoor structure will have a different room temperature than a welder in a little welding booth. The windy welder might experience drastic temperature swings, while the little welding booth might heat up during the process.

Finally, we would want to simulate the edges as well as the center of the sheet. As the welder moves across the bottom, the previous points will change temperature because the heat source isn't being applied directly to them.

4 Edge Case - I'll Patch It!

Let's suppose we're welding a patch over a hole in our steel sheet, and the patch is made from titanium. How does the heat flow where these two metals touch?

I put a 30 x 30 sheet of titanium in the middle of our sheet of stainless steel, and changed the simulation to use the heat coefficient of Titanium for points that were inside the titanium sheet.

I also changed the position of the welder so that it would move along the perimeter of the patch.

Table 2: Properties of Commercial Titanium

Property	**value**
Melting Temperature (C)	1668°
Thermal Conductivity at 20C	180 W/mK

4.1 Updated Simulation Code

patch.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int usage(char* name){
 printf("Usage: %s <welder speed> <plate
    W> <plate H> <room Temperture> <dt> <T
     Final>\n", name );
 printf("Example: %s 1 50 100 0 0.01 5\n"
    , name );
 return 1;
}

int main(int argc, char** argv){

 if (argc != 7){
  return usage(argv[0]);
 }

 float dWelder = atof(argv[1]);
 int X = atoi(argv[2]);
 int Y = atoi(argv[3]);
 float roomTemp = atof(argv[4]); //
```

```
   degrees celcius
float dt = atof(argv[5]); //Stepsize in
   milliseconds
float TFinal = atof(argv[6]);

float D1 = 0.162; //diffusion coefficent
    Steel Divided by 100 for cM
float D2 = 1.8; //diffusion coefficent
   Titanium Divided by 100 for cM
float D;

float T[Y][X];
float TNew[Y][X];

float t;

int i, j; //indicies

float xWelder = 10;
float yWelder = 10;

int welderVelIndex = 0;
int welderVel[4][2] = {
            {1,0},
            {0,1},
            {-1,0},
            {0,-1}
           };

float tWelder = 1450.0; // degrees
   celcius

int patchXStart = 10;
int patchXEnd = 40;
int patchYStart = 10;
int patchYEnd = 40;

// Set the whole plate to room
   temperture
for (j = 0; j < Y; j++){
 for (i = 0; i < X; i++){
  T[j][i] = roomTemp;
 }
}

//time step
for (t = 0; t < TFinal; t += dt) {


 T[(int) floor(yWelder)][(int) floor(
   xWelder)] = tWelder;

 xWelder += welderVel[welderVelIndex][0]
    * dWelder * dt;
 yWelder += welderVel[welderVelIndex][1]
    * dWelder * dt;

 fprintf(stderr, "vX %d vY %d\n",
   welderVel[welderVelIndex][0],
   welderVel[welderVelIndex][1]);
 fprintf(stderr, "X %f Y %f\n", xWelder,
    yWelder);

 if (floor(xWelder) > patchXEnd &&
   welderVelIndex == 0){
  welderVelIndex++;
  xWelder = patchXEnd;
 }

 if (floor(yWelder) > patchYEnd &&
   welderVelIndex == 1){
  welderVelIndex++;
  yWelder = patchYEnd;
 }

 if (floor(xWelder) < patchXStart &&
   welderVelIndex == 2){
  welderVelIndex++;
  xWelder = patchXStart;
 }



 //Updating all inner grid points
 for (j = 1; j < Y-1; j++){
  for (i = 1; i < X-1; i++){


   if (i > patchXStart && i < patchXEnd
   && j > patchYStart && j < patchYEnd){
    D = D2;
   } else {
    D = D1;
   }


   TNew[j][i] = T[j][i] + D*(T[j+1][i] +
    T[j-1][i] + T[j][i+1] + T[j][i-1] -
   4.0*T[j][i])*dt;

  }
 }


 //Updating T to be TNew
 for (j = 1; j < Y-1; j++){
  for(i = 1; i < X-1; i++){
   T[j][i] = TNew[j][i];
  }
 }

}


// printing out final state of the plate
for (j = 0; j < Y; j++){
 for (i = 0; i < X; i++){
  printf("%f ", T[j][i] );
 }
 printf("\n");
}


return 0;
}
```

4.2 Analysis

Since the titanium has a higher thermal conductivity, we can see that heat dissipates more on the inside of the outline (the patch) and less on the outside.

I also tried setting the thermal coefficient of the patch to 0, and we can see that the patch has no temperature

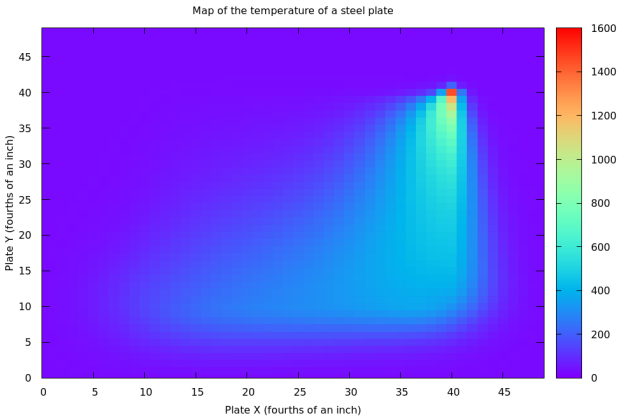

Figure 4: State of the sheet halfway into the simulation

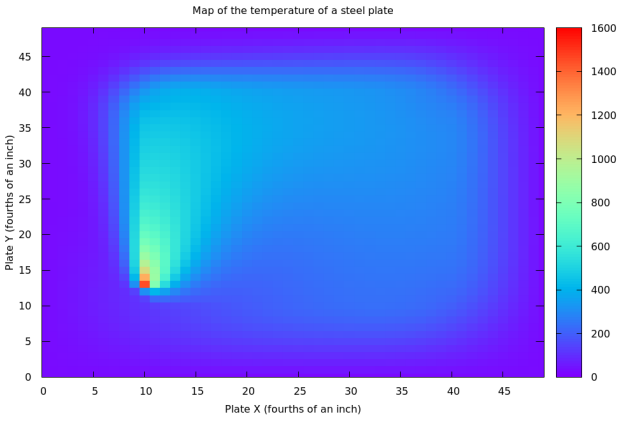

Figure 5: Sheet after the completed simulation

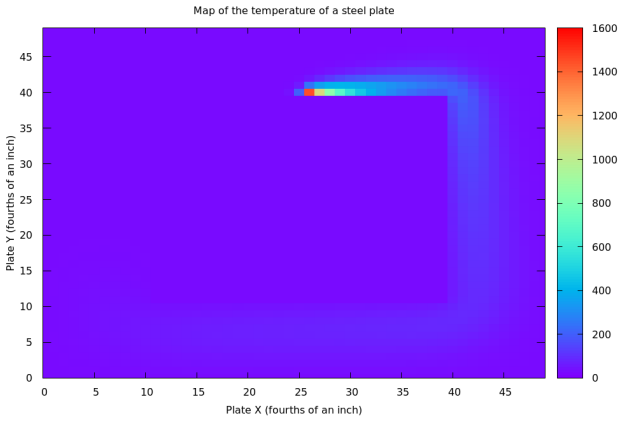

Figure 6: Case where heat diffusion of patch is 0

change as the arc traces the outline.

5 Conclusion

Numerical Analysis is a powerful tool in the engineering toolbox. It allows us to solve previously unsolvable problems in engineering and science, and to predict how materials will react to heat in a dynamic environment.

We saw how heat transfers through a plate when the heat source moves, and considered how to expand upon our simulation to make it more realistic.

The skills I have learned in Advanced Engineering Mathematics are highly transferable (diffusible? That's a little Heat Equation joke) to different career paths I may follow, and I am glad for the opportunity to learn more about the incredible universe that God has created.