Northeastern University
College *of* Engineering

MEMORANDUM

To: Professor O'Connell

From: Michael Brodskiy

Subject: CP3H

Date: Monday, December 7, 2022

Attachments: 7 (3 Figures, 2 Pseudocode, 2 Source code)

Introduction

The purpose of this document is to analyze the transient response curves and maximum weld times for certain mixes of tin/lead in thermocouples, generated by a combination of MATLAB and C++ scripts.

Steady-State Cooling

Thermocouple Response

As shown in 1, the transient response curve is <u>not</u> linear. Logically, this makes sense as the rate at which an object cools down is proportional to the difference between the object and its surrounding environment, as made evident through Newton's Law of Cooling. In this manner, as the thermocouple weld cools down, the rate at which it cools down is decreasing (*i.e.* the function is concave downwards).

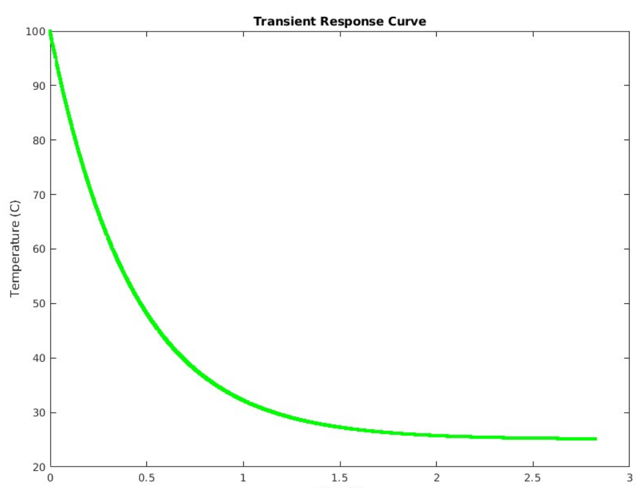

Figure 1: Transient Response Curve for Part 1

Material Combinations

Response Time

Though it is a bit difficult to see, the weld constituted by 43% tin and 57% lead has the fastest response time.

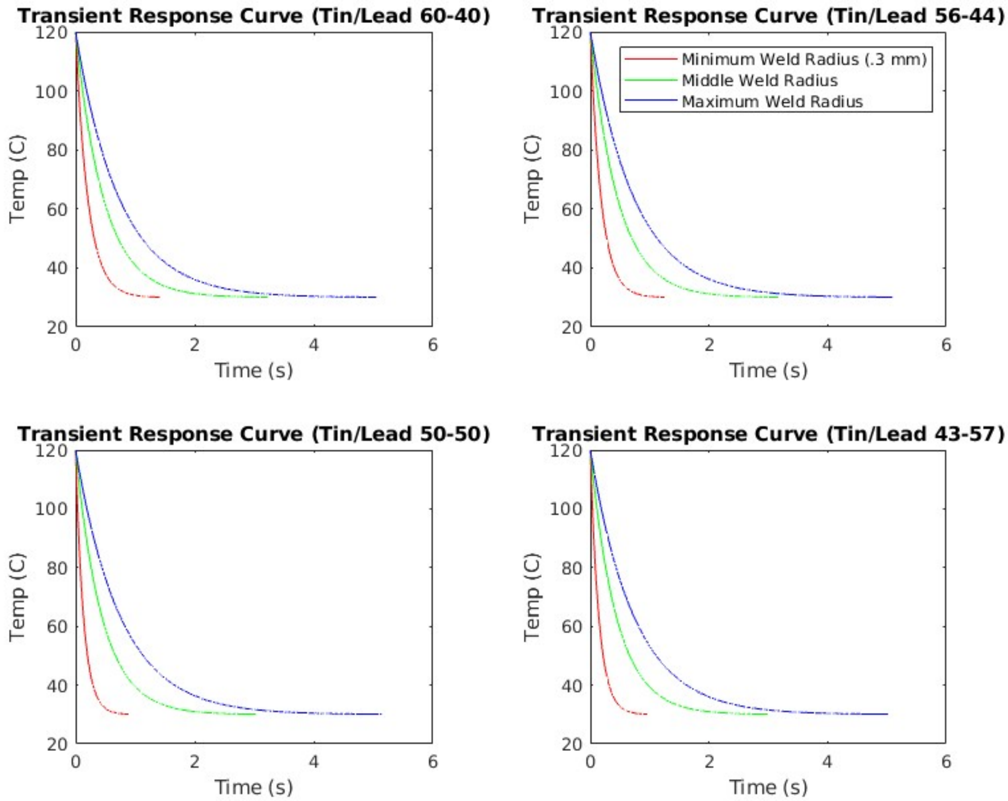

Figure 2: The Transient Response Curves of the Four Tin/Lead Combinations for Minimum, Maximum, and Average Radius Values

Material Safety

Given that a higher tin content increases the strength and reliability, in tandem with the fact that these will be used in engine safety monitoring systems, it would be best to use the weld with the highest tin content. As such, a weld with 60% tin and 40% lead would be the safest option.

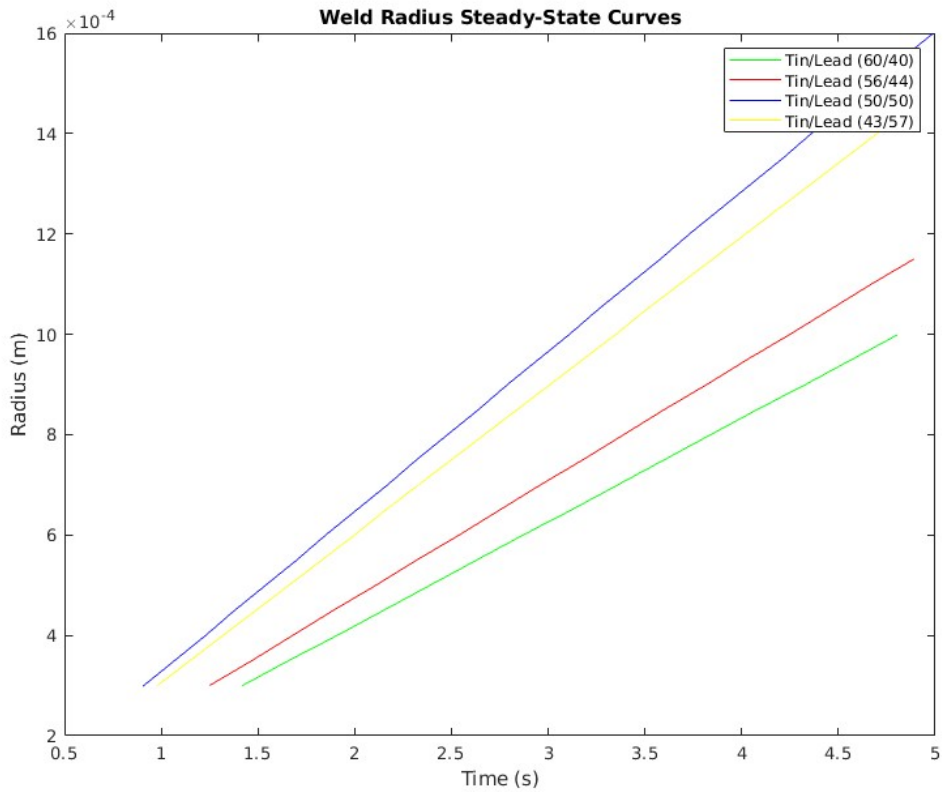

Figure 3: The Maximum Weld Radii of Various Tin/Lead Combinations

Algorithm 1 CP3H1

```
procedure THERMOCOUPLE BENCHMARK TESTER
    Ask user for inputs
    do
        Add time step to time
        Calculate new sphere temperature using RATE_OF_CHANGE()
        Output values to file
    while Sphere temperature minus liquid temperature is greater than .1°C
    Print steady-state time
    procedure RATE_OF_CHANGE
        Return (3· heat transfer coefficient · sphere and liquid temperature difference) /(radius
· density · specific heat)
```

Algorithm 2 CP3H2

procedure WELD RADIUS TESTER

procedure USER_INPUTS
    Ask user for global variable values

procedure RATE_OF_CHANGE
    Return (3· heat transfer coefficient · sphere and liquid temperature difference) /(radius · density · specific heat)

procedure GEN_TIME_FILE
    Use passed output object
    Use passed radius value
    do
        Add time step to time
        Calculate new sphere temperature using RATE_OF_CHANGE()
        Output values to file
    while Sphere temperature minus liquid temperature is greater than .1°C

procedure TIME_TO_STEADY_STATE
    Use passed radius value
    do
        Add time step to time
        Calculate new sphere temperature using RATE_OF_CHANGE()
    while Sphere temperature minus liquid temperature is greater than .1°C
    Update time value
    Return time value

procedure GEN_RAD_FILE
    Use passed output object
    while TIME_TO_STEADY_STATE(new radius value) is less than 5 **do**
        Print TIME_TO_STEADY_STATE(new radius value) and new radius value to file
        Update radius value by radius step

procedure MAX_WELD_SIZE
    while TIME_TO_STEADY_STATE(new radius value) is less than 5 **do**
        Update radius value by radius step State Return radius value

Listing 1: CP3H1 Script

```
/*
 * =============================================================
 *
 *      Filename:  CP3H1_MBROD.cpp
 *      Assignment:  C++ Lab #3 Homework 1
 *      Title:  Thermocouple Benchmark Tests
 *
 *    Description:  Takes in a plethora of values and calculates time
    for sphere temperature to equal liquid temperature of a
    thermocouple
 *
 *       Version:  1.0
 *       Created:  12/06/2022
```

```
 *        Revision:  N/A
 *        Compiler:  GCC
 *
 *          Author:  M. Brodskiy
 *
 * ===================================================================
 */

// -- Libraries & Directives --
#include <iostream> // Needed for normal cin & cout
#include <fstream> // Needed to read or write files on disk
#include <iomanip> // For Formatting Communications
using namespace std;

  const double pi = 3.1415926535; // Define pi

double rate_of_change(int htc, double rad, double T, int Tl, int dens,
    int specH) {

    return((3 * htc * (Tl - T)) / (rad * dens * specH));

}

// PROGRAM CP3H1
int main() {

  // -- Declare Variables --
  double radius, timeStep, tempS, time = 0;
  int c, rho, h, tempL; // Define necessary variables
  ofstream outfile; // An output stream object to output data

  //  Ask user for necessary inputs
  cout << "Enter initial temperature of thermocouple junction (sphere)
     (C): ";
  cin >> tempS;
  cout << "Enter liquid temperature (C): ";
  cin >> tempL;
  cout << "Enter heat transfer coefficient (W/m2K): ";
  cin >> c;
  cout << "Enter sphere density (kg/m3): ";
  cin >> rho;
  cout << "Enter sphere specific heat (J/kgK): ";
  cin >> h;
  cout << "Enter sphere radius (m): ";
  cin >> radius;
  cout << "Enter desired time step for temperature history (s): ";
  cin >> timeStep;

```

```
  // Open text file as an output stream
  outfile.open("test.txt");
  outfile << fixed << setprecision(4); // Set output file number
     formatting

  // Output initial values to text file
  outfile << "Time (s)" << "\t" << "Temp (C)" << endl;
  outfile << time << "\t\t" << tempS << endl;

  do { // Loop through values until temperature difference is less than
      1C

      time = time + timeStep;
      tempS = tempS + timeStep * rate_of_change(c, radius, tempS, tempL
         , rho, h);
      outfile << time << "\t\t" << tempS  << endl;

  } while (tempS - tempL > .1);

  outfile.close(); // Write to output file

  // Print steady-state time
  cout << "Time to steady-state temperature: " << fixed << setprecision
     (4) << time << " Seconds" << endl;

  //  Check the output file was created

  if (!outfile) {

      cout << "Unable to write to file" << endl;
      cout << "No output generated...\n";
      return(1);

    }

}
```

Listing 2: CP3H2 Script

```
/*
 * ===========================================================
 *
 *        Filename:  CP3H2_MBROD.cpp
 *        Assignment: C++ Lab #3 Homework 2
 *        Title: Thermocouple Benchmark Tests part 2
 *
 *    Description:  Takes in a plethora of values and calculates time
    for sphere temperature to equal liquid temperature of a
```

```
    thermocouple for a variety of radii, outputting these values to a
    file
 *
 *        Version:   1.0
 *        Created:   12/06/2022
 *       Revision:   N/A
 *       Compiler:   GCC
 *
 *         Author:   M.Brodskiy
 *
 * ============================================================
 */

// -- Libraries & Directives --
#include <iostream> // Needed for normal cin & cout
#include <fstream> // Needed to read or write files on disk
#include <iomanip> // For Formatting Communications
#include <string> // To manipulate time-file names
using namespace std;

  // -- Declare Variables --
  const double timeStep = .01;
  const double minRad = .0003;
  const double radStep = .00005;
  const double pi = 3.1415926535; // Define pi
  const int tempS = 120;
  const int tempL = 30;
  const int c = 1200;
  int rho, h; // Define necessary variables

  string tinLeadRatio = ("");

void user_inputs() {

  //  Ask user for necessary inputs
  cout << "Please input the necessary material parameters:" << endl;
  cout << "Enter sphere density (kg/m3): ";
  cin >> rho;
  cout << "Enter sphere specific heat (J/kgK): ";
  cin >> h;
  cout << "Enter tin/lead ratio (x%-y%): ";
  cin >> tinLeadRatio;

}

double rate_of_change(double T, double rad) {

    return((3 * c * (tempL - T)) / (rad * rho * h));
```

```cpp

}

void gen_time_file(ofstream &file, double rad) { // Generates a file calculating temperature changes until steady-state

    double tempChange = tempS;
    double time = 0; // Define changing variables

    // Output initial values to text file
    file << "Time (s)" << "\t" << "Temp (C)" << endl;
    file << time << "\t\t" << tempChange << endl;

    do { // Loop through values until temperature difference is less than 1C

      time = time + timeStep;
      tempChange = tempChange + timeStep * rate_of_change(tempChange, rad);
      file << time << "\t\t" << tempChange << endl;

    } while (tempChange - tempL > .1);

}

double time_to_steady_state(double rad) { // Returns time to steady-state for certain values

    double tempChange = tempS;
    double time = 0; // Changing variables

    do { // Loop through values until temperature difference is less than 1C

      time = time + timeStep;
      tempChange = tempChange + timeStep * rate_of_change(tempChange, rad);

    } while (tempChange - tempL > .1);

    return time;

}

void gen_rad_file(ofstream &file) { // Generates a file calculating times to steady-state depending on various radii

    double radChange = minRad;
```

```
    // Output initial values to text file
    file << "Time (s)" << "\t" << "Radius (m)" << endl;

    while (time_to_steady_state(radChange) < 5) {

        file << time_to_steady_state(radChange) << "\t\t" << radChange
           << endl;
        radChange = radChange + radStep;

    }

}

double max_weld_size() { // Returns the maximum weld size

    double radChange = minRad;

    while (time_to_steady_state(radChange) < 5) {

        radChange = radChange + radStep;

    } return radChange;

}

// PROGRAM CP3H2
int main() {

  user_inputs(); // Receive inputs from user

  ofstream outfile; // Create output file object

  outfile << fixed << setprecision(5); // Manipulate output

  for (int i = 0; i <= 2; i = i + 1) { // Generate three transient
      response curves

    if ( i == 0 ) {

        outfile.open(tinLeadRatio + "minRad.txt");
        gen_time_file(outfile, minRad);

    } else if ( i == 1 ) {

        outfile.open(tinLeadRatio + "midRad.txt");
        gen_time_file(outfile, ((minRad + max_weld_size()) / 2));

```

```
        } else {

            outfile.open(tinLeadRatio + "maxRad.txt");
            gen_time_file(outfile, max_weld_size());

        }

        outfile.close(); // Write to each file

    }

    // Open text file as an output stream
    outfile.open(tinLeadRatio + "RAD.txt"); // Generate material maximum
       weld size in time file
    gen_rad_file(outfile);
    outfile.close(); // Write to output file

    //  Check the output file was created

    if (!outfile) {

        cout << "Unable to write to file" << endl;
        cout << "No output generated...\n";
        return(1);

      }

}
```