Program 1 Report

Kelson A. Petersen, A02346768

October 4, 2025

Contents

1 Problem 1

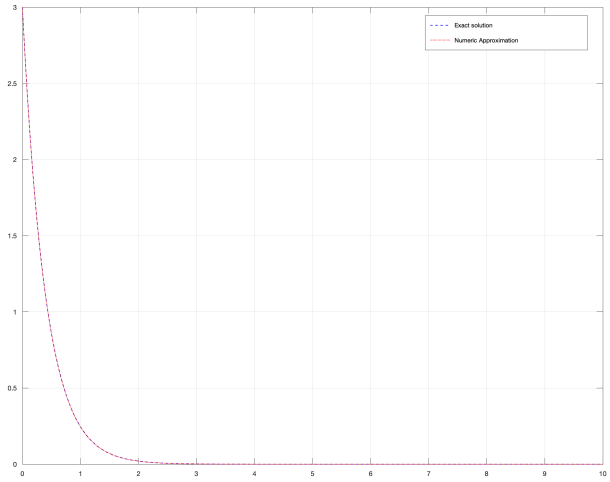


Figure 1: The Exact solution compared to the numeric approximation of Problem 1.

Using figure 1 above we see the C program approximate solution and the exact solution follow the same curve and we have a good approximation of the first order ODE given by $(D + 2.5)y_0(t) = 0$ with the initial condition $y_0(0) = 3$. Scratch work is included at the end of the document.

The following code is the loop responsible for solving the ODE that was given above. The entirety of the code is included later in this document.

```c
// Problem 1 Code
int problem_1(void) {
  FILE* fout = fopen("./prog_out/prog_sol_1.txt", "w");
  if (fout == NULL) {
    perror("output file failed");
    return EXIT_FAILURE;
  }
  // Declare and Initialize variables
  double y       = 3.0;
  double delta_t = 0.001;
  double time    = 0;
  double a       = -2.5;
  // This line doesn't change in the loop
  a = (1 + a * delta_t);
  // Do the math iteratively in a loop
  for (time = 0.0; time < 10.0; time += delta_t) {
    // Print the t, y(t) coordinate
    fprintf(fout, "%0.3lf\t%0.10lf\n", time, y);
    // Do the calculation
    y = a * y;
  }
  return EXIT_SUCCESS;
}
```

2 Problem 2

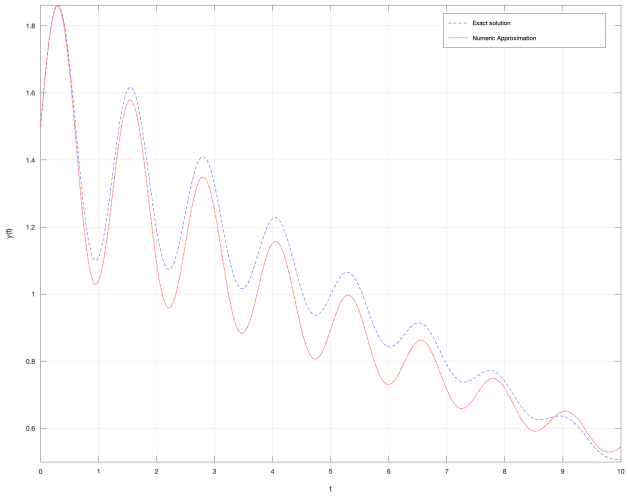


Figure 2: The Exact solution compared to the numeric approximation in Problem 2.

Using figure 2 above we compare the exact solution that is the blue dashed line and the approximate solution in the C code below follow a similar path; however, the approximation is less accurate than the exact solution in this case. The oscillilatory nature of this function causes the 'average' to be less accurate. Below is the figure that has the roots of the exact solution.

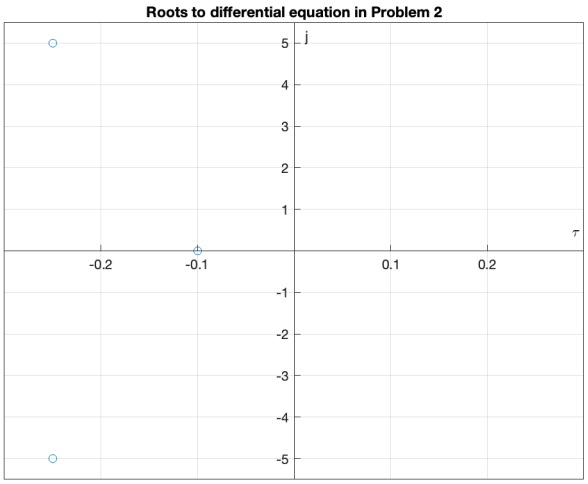


Figure 3: The roots of the differential equation for Problem 2.

Below is the equation that was found as the solution for the differential equation given by $D^3 + 0.6D^3 + 25.1125D + 2.5063y_0(t) = 0$. I used matlab to solve.

$$\frac{21156110196\sqrt{10009}\,\mathrm{e}^{-\frac{3t}{10}}\,\sin\left(\frac{\sqrt{10009}\,t}{20}\right)}{5049641841125} - \frac{5221024\,\mathrm{e}^{-\frac{3t}{10}}\,\cos\left(\frac{\sqrt{10009}\,t}{20}\right)}{504510125} - \frac{25063\,t}{251125} + \frac{1523972423}{1009020250} \tag{1}$$

This is the specific code that was used to calculate the approximation.

```
// Problem 2 Code
int problem_2(void) {
  FILE* fout = fopen("./prog_out/prog_sol_2.txt", "w");
```

```
  if (fout == NULL) {
    perror("output file failed");
    return EXIT_FAILURE;
  }
  double I[][3]  = {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}};
  double A[][3]  = {{0, 1, 0}, {0, 0, 1}, {-2.5063, -25.1125, -0.6}};
  double x_t[3]  = {1.5, 2, -1};
  double delta_t = 0.001f;
  double time;
  // avoid unnecessary function calls in loop
  mat_scale(delta_t, A, A);
  mat_add(I, A, A);
  // Do iterative math
  for (time = 0.0; time < 10.0; time += delta_t) {
    // print the result
    fprintf(fout, "%.3lf\t%.10lf\n", time, x_t[0]);
    // get next x_t value
    mat_vec_mult(A, x_t, x_t);
  }
  return EXIT_SUCCESS;
}
```

3 Problem 3

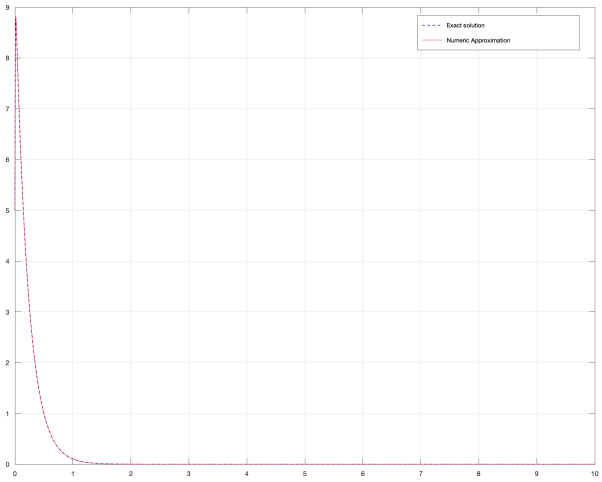


Figure 4: The Exact solution compared to the numeric approximation in Problem 3.

For the figure above we solved the circuit to find the differential equation that work is shown at the end of the document. The differential equation we end up with is $f(t) = D^2LCR_2(\frac{1}{R+1}+\frac{1}{R_2})+D(\frac{L}{R_1}+CR_2)+y(t)$. We see the approximate solution worked well with this equation as it does not oscillate. If the circuit had a dependend source in it the analytical and numeric solutions would both become more complicated as the non-linearity of the circuit would create different frequency responses varying in a non-linear fashion. Additionally The numeric approximation I would expect to see diverge from the exact solution because the exact solution would not be linear.

Below is the code that did the calculation.

```
// Problem 3 Code
int problem_3(void) {
  FILE* fout = fopen("./prog_out/prog_sol_3.txt", "w");
  if (fout == NULL) {
    perror("output file failed");
    return EXIT_FAILURE;
  }
  double A[][2]  = {{0, 1}, {195.651, 869.57}};
  double I[][2]  = {{1, 0}, {0, 1}};
  double x_t[2]  = {5, 847.83};
  double delta_t = 0.001;
  double time;

  mat_scale2(delta_t, A, A);
  mat_add2(I, A, A);
  for (time = 0.0; time <= 10.0; time += delta_t) {
    fprintf(fout, "0.3lf\t0.10lf\n", time, x_t[0]);
    mat_vec_mult2(A, x_t, x_t);
  }
  return EXIT_SUCCESS;
}
```

4 Code

```c
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>

// Functions for the problems
int problem_1(void);
int problem_2(void);
int problem_3(void);
// 2x2 Matrix and Vector Functions
void mat_scale2(double scale, double mat[][2], double prod[][2]);
void mat_sub2(double left[][2], double right[][2], double diff[][2]);
void mat_add2(double left[][2], double right[][2], double sum[][2]);
void mat_vec_mult2(double mat[][2], double* vector, double* prod);
// 3x3 Matrix and Vector Functions
void mat_scale(double scale, double mat[][3], double prod[][3]);
void mat_sub(double left[][3], double right[][3], double diff[][3]);
void mat_add(double left[][3], double right[][3], double sum[][3]);
void mat_vec_mult(double mat[][3], double* vector, double* prod);

int main(int argc, char** argv) {
  //  Begin Program
  int problem;
  // If no arguements prompt and recieve input
  if (argc < 2) {
    printf("Enter problem number: "); // Prompt user to select problem
    scanf("%d", &problem);            // accept user input
  } else {
    // else use command line arguement
    problem = atoi(argv[1]); // Assign the value from the command line
  }
  // Do the selected problem
  switch (problem) {
    case 1: // Print success or fail for problem execution
      !problem_1() ? printf("Problem 1 success\n")
                   : printf("Error in Problem 1\n");
      break;
    case 2: //
      !problem_2() ? printf("Problem 2 success\n")
                   : printf("Error in Problem 2\n");
      break;
    case 3:
      !problem_3() ? printf("Problem 3 success\n")
                   : printf("Error in Problem 3\n");
      break;
    default:
      printf("Error please choose 1-3.\n");
  }
  // End Program
  return EXIT_SUCCESS;
}
// Problem 1 Code
int problem_1(void) {
  FILE* fout = fopen("./prog_out/prog_sol_1.txt", "w");
  if (fout == NULL) {
    perror("output file failed");
    return EXIT_FAILURE;
  }
  // Declare and Initialize variables
```

```
  double y       = 3.0;
  double delta_t = 0.001;
  double time    = 0;
  double a       = -2.5;
  // This line doesn't change in the loop
  a = (1 + a * delta_t);
  // Do the math iteratively in a loop
  for (time = 0.0; time < 10.0; time += delta_t) {
    // Print the t, y(t) coordinate
    fprintf(fout, "%0.3lf\t%0.10lf\n", time, y);
    // Do the calculation
    y = a * y;
  }
  fclose(fout);
  return EXIT_SUCCESS;
}
// Problem 2 Code
int problem_2(void) {
  FILE* fout = fopen("./prog_out/prog_sol_2.txt", "w");
  if (fout == NULL) {
    perror("output file failed");
    return EXIT_FAILURE;
  }
  double I[][3]  = {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}};
  double A[][3]  = {{0, 1, 0}, {0, 0, 1}, {-2.5063, -25.1125, -0.6}};
  double x_t[3]  = {1.5, 2, -1};
  double delta_t = 0.001f;
  double time;
  // avoid unnecessary function calls in loop
  mat_scale(delta_t, A, A);
  mat_add(I, A, A);
  // Do iterative math
  for (time = 0.0; time < 10.0; time += delta_t) {
    // print the result
    fprintf(fout, "%.3lf\t%.10lf\n", time, x_t[0]);
    // get next x_t value
    mat_vec_mult(A, x_t, x_t);
  }
  fclose(fout);
  return EXIT_SUCCESS;
}
// Problem 3 Code
int problem_3(void) {
  FILE* fout = fopen("./prog_out/prog_sol_3.txt", "w");
  if (fout == NULL) {
    perror("output file failed");
    return EXIT_FAILURE;
  }
  double A[][2] = {{0,1},{-869.57,-195.651}};
  double I[][2]  = {{1, 0}, {0, 1}};
  double x_t[2]  = {5, 847.83};
  double delta_t = 0.001;
  double time;

  mat_scale2(delta_t, A, A);
  mat_add2(I, A, A);
  for (time = 0.0; time <= 10.0; time += delta_t) {
    fprintf(fout, "%0.3lf,%0.10lf\n", time, x_t[0]);
    mat_vec_mult2(A, x_t, x_t);
  }
```

```
  return EXIT_SUCCESS;
}
// scale matrix `mat` by a and save in prod
void mat_scale(double scale, double mat[][3], double prod[][3]) {
  int i, j;
  for (i = 0; i < 3; ++i) {
    for (j = 0; j < 3; ++j) {
      prod[i][j] = scale * mat[i][j];
    }
  }
}
// subtract the right from the left and store in diff
void mat_sub(double left[][3], double right[][3], double diff[][3]) {
  int i, j;
  for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
      diff[i][j] = left[i][j] - right[i][j];
    }
  }
}
// add the right and the left and store into sum
void mat_add(double left[][3], double right[][3], double sum[][3]) {
  int i, j;
  for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
      sum[i][j] = left[i][j] + right[i][j];
    }
  }
}
// Unsafe if used incorrectly!
// multiplies 3x3 matrix mat with 3x1 vector storing into 3x1 prod
void mat_vec_mult(double mat[][3], double* vector, double* prod) {
  double sum;
  int    i, j;
  for (i = 0; i < 3; i++) {
    sum = 0;
    for (j = 0; j < 3; j++) {
      sum += mat[i][j] * vector[j];
    }
    prod[i] = sum;
  }
}

// scale matrix `mat` by a and save in prod
void mat_scale2(double scale, double mat[][2], double prod[][2]) {
  int i, j;
  for (i = 0; i < 2; ++i) {
    for (j = 0; j < 2; ++j) {
      prod[i][j] = scale * mat[i][j];
    }
  }
}
// subtract the right from the left and store in diff
void mat_sub2(double left[][2], double right[][2], double diff[][2]) {
  int i, j;
  for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
      diff[i][j] = left[i][j] - right[i][j];
    }
  }
```

```
}
// add the right and the left and store into sum
void mat_add2(double left[][2], double right[][2], double sum[][2]) {
  int i, j;
  for (i = 0; i < 2; i++) {
    for (j = 0; j < 2; j++) {
      sum[i][j] = left[i][j] + right[i][j];
    }
  }
}
// Unsafe if used incorrectly!
// multiplies 3x3 matrix mat with 3x1 vector storing into 3x1 prod
void mat_vec_mult2(double mat[][2], double* vector, double* prod) {
  double sum;
  int    i, j;
  for (i = 0; i < 2; i++) {
    sum = 0;
    for (j = 0; j < 2; j++) {
      sum += mat[i][j] * vector[j];
    }
    prod[i] = sum;
  }
}
```

5 Scratch Work

1.)

a.) $(D+2.5)\,y(t) = 0$

$\lambda = -2.5$

$C_1 e^{-2.5t}$

$C_1 e^{0} = 3$

$C_1 = 3$

$y(t) = 3e^{-2.5t}$

2.) $(D^3 + 0.6D^2 + 25.1125D + 2.5063)\,y(t) = 0$

$y_0(0) = 1.5$, $\dot{y}_0(0) = 2$, $\ddot{y}_0(0) = -1$

Use Matlab to solve for Roots

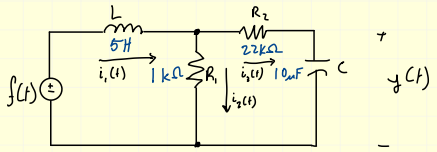
-0.1, $-0.25 + j5$, $-0.25 - j5$

$\lambda = -0.1$
$\lambda = -0.25 + j5$
$\lambda = -0.25 - j5$

$C_1 e^{-0.1t} + C_2 e^{+(-0.25+j5)} + C_3 e^{+(-0.25+j5)}$

using matlab to solve and find equation 1.

$$x\Delta t_{i+1} = \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \Delta t \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2.5063 & -25.1125 & -0.6 \end{bmatrix} \right) \begin{bmatrix} 1.5 \\ 2 \\ -1 \end{bmatrix}$$



(a)

$$i_1(t) = i_2(t) + i_3(t)$$

$$i_2(t) = \frac{V - y(t)}{R_2} = C\frac{dy(t)}{dt} + y(t)$$

$$i_3(t) = \frac{V}{R_1}$$

$$i_1 = \frac{V}{R_1} + \frac{V - y(t)}{R_2}$$

$$\frac{di_1(t)}{dt} = \frac{1}{R_1}\frac{dv}{dt} + \frac{1}{R_2}\left(\frac{dv}{dt} - \frac{dy(t)}{dt}\right)$$

$$f(t) = L\frac{di_1(t)}{dt} + V$$

$$f(t) = L\left[\frac{1}{R_1}\frac{dv}{dt} + \frac{1}{R_2}\left(\frac{dv}{dt} - \frac{dy}{dt}\right)\right] + V$$

$$f(t) = L\left(\frac{1}{R_1} + \frac{1}{R_2}\right)\frac{dv}{dt} - \frac{L}{R_2}\frac{dy}{dt} + V$$

$$V = y + CR_2\frac{dy}{dt} \Rightarrow \frac{dv}{dt} = \frac{dy}{dt} + CR_2\frac{d^2y}{dt^2}$$

$$f(t) = L\left(\frac{1}{R_1} + \frac{1}{R_2}\right)\left(\frac{dy}{dt} + CR_2\frac{d^2y(t)}{dt^2}\right) - \frac{L}{R_2}\cdot\frac{dy}{dt} + y + CR_2\frac{dy}{dt}$$

use D operators

$$f(t) = DL\left(\frac{1}{R_1} + \frac{1}{R_2}\right) + D^2LCR_2\left(\frac{1}{R_1} + \frac{1}{R_2}\right) - D\frac{L}{R_2} + y + DCR_2$$

$$f(t) = D^2LCR_2\left(\frac{1}{R_1} + \frac{1}{R_2}\right) + D\left(\frac{L}{R_1} + CR_2\right) + y$$

(b)

$L = 5$ H
$C = 10\,\mu$F
$R_2 = 22$ kΩ
$R_1 = 1$ kΩ

$y(0) = 5$ V
$i_1(0) = 0.2$ A

$$1.1\left(\frac{1}{1k} + \frac{1}{22k}\right)D^2 + (0.005 + 0.22)D + y$$

$$\frac{23}{20{,}000}D^2 + \frac{9}{40}D + y$$

$$\lambda_1 = -4.5503$$
$$\lambda_2 = -191.1019$$

$$C_1e^{-4.5503t} + C_2e^{-191.1019t} = 0$$

$$= 847.83$$

$$V = CR_2\left(\frac{dy(t)}{dt}\right) + y(t)$$

$$i_1(0) = \frac{y(0)}{R_1} + \left(\frac{R_2C}{R_1}C\right)y'(0)$$

$$i_1(0) = 0.2 - \frac{\frac{5}{1000}}{\frac{22{,}000}{1000}}$$

(C)

$$C_1 + C_2 = 5$$
$$-4.5503C_1 + -191.1019C_2 = 847.83$$

$$\begin{bmatrix} 1 & 1 & 5 \\ -4.5503 & -191.1019 & 847.83 \end{bmatrix}$$

$$\begin{matrix} 1 & 0 & -181.105 \\ 0 & 1 & -4.5503 \end{matrix}$$

$$9.6667e^{-4.5503t} - 4.667e^{-191.1019t}$$

(d)

$$\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \Delta t\cdot\begin{bmatrix} 0 & 1 \\ -869.57 & -195.651 \end{bmatrix}\right)\cdot\begin{bmatrix} 5 \\ 847.83 \end{bmatrix}$$