

# Code

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "convolve.h"
4  #include "matrix.h"
5
6  void Ft_conv_Ht(void);
7  int  program_1_code(void);
8
9  int main(void) {
10     int      itr, leny;
11     double*   y;
12     const int  len1 = 6, len2 = 7, len3 = 4, len4 = 5;
13     const double f1[len1] = {0, 1, 2, 3, 2, 1};
14     const double f2[len2] = {-2, -2, -2, -2, -2, -2, -2};
15     const double f3[len3] = {1, -1, 1, -1};
16     const double f4[len4] = {0, 0, 0, -3, -3};
17     const double X[3]      = {0.4, 0.35, 0.25};
18     const double Y[4]      = {0.25, 0.20, 0.20, 0.35};
19
20     printf("f1: ");
21     for (itr = 0; itr < len1; itr++) { printf(" %5.01f", f1[itr]); }
22     printf("\nf2: ");
23     for (itr = 0; itr < len2; itr++) { printf(" %5.01f", f2[itr]); }
24     printf("\nf3: ");
25     for (itr = 0; itr < len3; itr++) { printf(" %5.01f", f3[itr]); }
26     printf("\nf4: ");
27     for (itr = 0; itr < len4; itr++) { printf(" %5.01f", f4[itr]); }
28     printf("\n\n");
29     /* Problem 2.A */
30     leny = conv(f1, len1, f1, len1, &y);
31     printf("f1 * f1: ");
32     for (itr = 0; itr < leny; ++itr) { printf(" %5.01f", y[itr]); }
33     free(y);
34     /* Problem 2.B */
35     printf("\nf1 * f2: ");
36     leny = conv(f1, len1, f2, len2, &y);
37     for (itr = 0; itr < leny; ++itr) { printf(" %5.01f", y[itr]); }
38     free(y);
39     /* Problem 2.C */
40     printf("\nf1 * f3: ");
41     leny = conv(f1, len1, f3, len3, &y);
42     for (itr = 0; itr < leny; ++itr) { printf(" %5.01f", y[itr]); }
43     free(y);
44     /* Problem 2.D */
45     printf("\nf2 * f3: ");
46     leny = conv(f2, len2, f3, len3, &y);
47     for (itr = 0; itr < leny; ++itr) { printf(" %5.01f", y[itr]); }
48     free(y);
49     /* Problem 2.E */
50     printf("\nf1 * f4: ");
51     leny = conv(f1, len1, f4, len4, &y);
52     for (itr = 0; itr < leny; ++itr) { printf(" %5.01f", y[itr]); }
53     free(y);
54     // Get results for the zero input
55     program_1_code();
56     // Perform the convolution and summing of h_t and zero_input
57     Ft_conv_Ht();
58     return 0;
59 }
60
61 int program_1_code(void) {
62     FILE* fout = fopen("zero_input.txt", "w");
63     if (fout == NULL) {
64         perror("output file failed");
65         return EXIT_FAILURE;
66     }
67     double I[][3] = {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}};
68     double A[][3] = {{0, 1, 0}, {0, 0, 1}, {-15, -12, -5}};
69     double x_t[3] = {-2, 3, 4};
70     double delta_t = 0.001f;
71     double time;
72     // avoid unnecessary function calls in loop
```

```

73 mat_scale(delta_t, A, A);
74 mat_add(I, A, A);
75 // Do iterative math
76 for (time = 0.0; time < 10.0; time += delta_t) {
77     // print the result
78     fprintf(fout, "%.10lf\n", x_t[0]);
79     // get next x_t value
80     mat_vec_mult(A, x_t, x_t);
81 }
82 fclose(fout);
83 return EXIT_SUCCESS;
84 }
85
86 void Ft_conv_Ht(void) {
87     double ft[10000] = {0};
88     double ht[10000] = {0};
89     double zi[10000] = {0};
90     char buff[100];
91     int leny = 0;
92     double* y;
93     FILE* FT_FILE = fopen("sine.txt", "r");
94     if (FT_FILE == NULL) {
95         printf("Error: sine.txt doesn't exist\n");
96         exit(1);
97     }
98     FILE* HT_FILE = fopen("h_t.txt", "r");
99     if (HT_FILE == NULL) {
100         printf("Error: h_t.txt doesn't exist\n");
101         exit(1);
102     }
103     FILE* ZI_FILE = fopen("zero_input.txt", "r");
104     if (ZI_FILE == NULL) {
105         printf("error: zero_input.txt didn't open\n");
106         exit(1);
107     }
108     FILE* TOTAL_RESPONSE = fopen("total_result.txt", "w");
109     if (TOTAL_RESPONSE == NULL) {
110         printf("total_result.txt not opened\n");
111         exit(1);
112     }
113     for (int i = 0; i < 10000; i++) {
114         if (!fgets(buff, sizeof(buff), FT_FILE)) { printf("Reached end of file at line %d\n", i); break; }
115         char* endptr;
116         double value = strtod(buff, &endptr);
117         if (endptr == buff) { printf("Invalid number on line %d: %s\n", i, buff); continue; }
118         ft[i] = value;
119     }
120     fclose(FT_FILE);
121     for (int i = 0; i < 10000; i++) {
122         if (!fgets(buff, sizeof(buff), HT_FILE)) { printf("Reached end of file at line %d\n", i); break; }
123         char* endptr;
124         double value = strtod(buff, &endptr);
125         if (endptr == buff) { printf("Invalid number on line %d: %s\n", i, buff); continue; }
126         ht[i] = value;
127     }
128     fclose(HT_FILE);
129     // Perform the convolution
130     leny = conv(ht, sizeof(ht) / sizeof(ht[0]), ft, sizeof(ft) / sizeof(ft[0]), &y);
131     // Scale the convolution by 0.001
132     for(int itr = 0; itr < leny; itr++){
133         y[itr] *= 0.001;
134     }
135     for (int i = 0; i < 10000; i++) {
136         if (!fgets(buff, sizeof(buff), ZI_FILE)) {printf("Reached end of file at line %d\n", i); break; }
137         char* endptr;
138         double value = strtod(buff, &endptr);
139         if (endptr == buff) { printf("Invalid number on line %d: %s\n", i, buff); continue; }
140         zi[i] = value;
141     }
142     fclose(ZI_FILE);
143     double temp = 0;
144     for (int itr = 0; itr < 20000; itr++) {
145         if (itr < 10000) {
146             temp = zi[itr];

```

```

147     } else {
148         temp = 0;
149     }
150     fprintf(TOTAL_RESPONSE, "%lf\\t%.10lf\\n", (double) itr / 1000, y[itr] + temp);
151 }
152 free(y);
153 }

```

```

1  #ifndef __CONVOLVE_H
2  #define __CONVOLVE_H
3
4  int conv(double *f1, int len1, double *f2, int len2, double **y);
5
6  #endif

```

```

1  #include "convolve.h"
2  #include <stdlib.h>
3  #include <stdio.h>
4
5  int conv(double *f1, int len1, double *f2, int len2, double **y){
6      int leny = len1 + len2 - 1;
7      int in_b, m_start;
8      // Allocate The proper amount of memory for y
9      (*y) = (double *)malloc((sizeof(double) * leny));
10     for(int itr = 0; itr < leny; itr++){
11         (*y)[itr] = 0.0f;
12     }
13     // Add the diagonals
14     for(int i = 0; i < len1; i++){
15         for(int j = 0; j < len2; j++){
16             (*y)[i+j] += f1[i] * f2[j];
17         }
18     }
19     return leny;
20 }

```

```

1  #ifndef __MATRIX_H_
2  #define __MATRIX_H_
3  // 2x2 Matrix and Vector Functions
4  void mat_scale2(double scale, double mat[][2], double prod[][2]);
5  void mat_sub2(double left[][2], double right[][2], double diff[][2]);
6  void mat_add2(double left[][2], double right[][2], double sum[][2]);
7  void mat_vec_mult2(double mat[][2], double* vector, double* prod);
8  // 3x3 Matrix and Vector Functions
9  void mat_scale(double scale, double mat[][3], double prod[][3]);
10 void mat_sub(double left[][3], double right[][3], double diff[][3]);
11 void mat_add(double left[][3], double right[][3], double sum[][3]);
12 void mat_vec_mult(double mat[][3], double* vector, double* prod);
13 #endif

```

```

1  #include "matrix.h"
2
3  // scale matrix 'mat' by a and save in prod
4  void mat_scale(double scale, double mat[][3], double prod[][3]) {
5      int i, j;
6      for (i = 0; i < 3; ++i) {
7          for (j = 0; j < 3; ++j) {
8              prod[i][j] = scale * mat[i][j];
9          }
10     }
11 }
12 // subtract the right from the left and store in diff
13 void mat_sub(double left[][3], double right[][3], double diff[][3]) {
14     int i, j;
15     for (i = 0; i < 3; i++) {
16         for (j = 0; j < 3; j++) {
17             diff[i][j] = left[i][j] - right[i][j];
18         }
19     }
20 }
21 // add the right and the left and store into sum

```

```

22 void mat_add(double left[][3], double right[][3], double sum[][3]) {
23     int i, j;
24     for (i = 0; i < 3; i++) {
25         for (j = 0; j < 3; j++) {
26             sum[i][j] = left[i][j] + right[i][j];
27         }
28     }
29 }
30 // Unsafe if used incorrectly!
31 // multiplies 3x3 matrix mat with 3x1 vector storing into 3x1 prod
32 void mat_vec_mult(double mat[][3], double* vector, double* prod) {
33     double sum;
34     int i, j;
35     for (i = 0; i < 3; i++) {
36         sum = 0;
37         for (j = 0; j < 3; j++) {
38             sum += mat[i][j] * vector[j];
39         }
40         prod[i] = sum;
41     }
42 }
43
44 // scale matrix 'mat' by a and save in prod
45 void mat_scale2(double scale, double mat[][2], double prod[][2]) {
46     int i, j;
47     for (i = 0; i < 2; ++i) {
48         for (j = 0; j < 2; ++j) {
49             prod[i][j] = scale * mat[i][j];
50         }
51     }
52 }
53 // subtract the right from the left and store in diff
54 void mat_sub2(double left[][2], double right[][2], double diff[][2]) {
55     int i, j;
56     for (i = 0; i < 3; i++) {
57         for (j = 0; j < 3; j++) {
58             diff[i][j] = left[i][j] - right[i][j];
59         }
60     }
61 }
62 // add the right and the left and store into sum
63 void mat_add2(double left[][2], double right[][2], double sum[][2]) {
64     int i, j;
65     for (i = 0; i < 2; i++) {
66         for (j = 0; j < 2; j++) {
67             sum[i][j] = left[i][j] + right[i][j];
68         }
69     }
70 }
71 // Unsafe if used incorrectly!
72 // multiplies 3x3 matrix mat with 3x1 vector storing into 3x1 prod
73 void mat_vec_mult2(double mat[][2], double* vector, double* prod) {
74     double sum;
75     int i, j;
76     for (i = 0; i < 2; i++) {
77         sum = 0;
78         for (j = 0; j < 2; j++) {
79             sum += mat[i][j] * vector[j];
80         }
81         prod[i] = sum;
82     }
83 }

```

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <complex.h>
5
6  int main(void){
7      double stop = 10.0, d_T = 0.001, c_dt;
8      FILE *fout = fopen("sine.txt", "w");
9      for(c_dt = 0; c_dt < stop; c_dt += d_T){
10         fprintf(fout, "%lf\n", sin(2.5 * M_PI * c_dt));

```

```
11 }
12 fclose(fout);
13 return 0;
14 }
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include <complex.h>
5
6 int main(void){
7     double stop = 10.0,d_T = 0.001, c_dt;
8     double result = 0;
9     FILE *fout = fopen("h_t.txt","w");
10    for(c_dt =0; c_dt < stop; c_dt += d_T){
11        result = -0.175 * exp(-2.604*c_dt) + 2 * (0.201*exp(-1.198*c_dt)*cos(-2.08*c_dt + 1.12));
12        fprintf(fout,"%lf\n",result);
13    }
14    fclose(fout);
15    return 0;
16 }
```