7.5 Project File LEW4.CPP

This file contains code to calculate the impulse response and the FFT.

Includes:

STDIO.H - library file with the standard input/output routines.
STDLIB.H - standard library file needed for exit function.
MATH.H - library file with math functions.

Defines:

MAXLAYERS - the maximum number of reflecting layers (or reflected rays seen by the receiver) in the ionosphere that the program will handle.
DATA - the number of real data points in the output data streams. Two successive data points represent a complex number. The first is the real part and the second is the imaginary part.
TWOPI - definition of $2\pi = 6.28318530717959$.
SWAP - a data switching macro used by the FFT algorithm, **little_four**.

Structures:

ray_path - structure that contains all input and computed variables characteristic of a path. The elements of **ray_path** are given on p. 28.
compute - structure that contains all the variables specific to the computations or not specific to an individual path. The elements of **compute** are given on p. 29.

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

#define MAXLAYERS 3
#define DATA 4096
#define TWOPI 6.28318530717959

#define SWAP(a,b)    tempr = (a); \
                     (a) = (b); \
                     (b) = tempr

typedef struct ray_path
{
        float path_Distance, center_freq, penetrate_freq, thick_scale, maxD_hgt;
        float peak_amplitude, sigma_tau, sigma_c, sigma_D, fds, fdl;
        double tau_c, sigma_f, slp, tau_L, tau_U, tau_l, alpha, sigma_l, lambda;
};

typedef struct compute
{
        int layers, slices, seed;
        float delta_t, afl;
        double delta_tau, big_el;
};
```

7.5.1. Function **void little_four**

Description:

This function computes the complex FFT of a complex array and is called by **slicedo**. This is an implementation of the FFT found in Press et al. [32, pp. 404-414]. A single data array is passed in where the elements are alternating real and imaginary parts of complex numbers. This data array is replaced (passed back) with the complex coefficients of its Fourier transform. This function begins counting at 1, so the input array must be decremented by one in the function call, e.g., **little_four**(data - 1, 4096, 1). **Little_four** is contained in file LEW4.CPP.

Parameters passed to **little_four**:

data - array of **float**, size 2 × *nn*, passed by reference, contains the complex impulse response array.

nn - **integer**, must be a power of two, indicates size of the complex array.

isign - **integer**, a flag that indicates the desired direction of the FFT, 1 means the normal FFT will be run while -1 means the inverse FFT will be run. Normalization for the inverse case is not done within **little_four**.

Parameters returned by **little_four**:

data - array of **float**, size 2 × *nn*, contains the complex Fourier coefficients, returned by reference.

Macros used:

SWAP - this macro simply swaps the value of two variables. Used by **little_four** in data rearrangement.

Functions called:

sin - library function returns the sine of a real number. Needs MATH.H.

Code listing:

See Press et al. [32, pp. 404-414].

7.5.2. Function **void imp**

Description:

This function computes the impulse response for each time slice by computing and superimposing the impulse responses for each layer. See (2) and (14). **Imp** is located in LEW4.CPP.

Parameters passed to **imp**:

datb - array of **float** of size 2 × DATA, corresponds to *cdat* in slicedo.
pdsi - array of **ray_path** of size *layers*.
cdsi - array of **compute** of size 1.

start - pointer to **float**, current position in the random number array.

timexx - **double**, current slice time.

oo - **integer**, current time slice index.

Local variables:

tau_k - **double**, delay step.
gag - **double**, difference between *tau_k* and *tau_l* for current layer.
gg - **double**, *gag* divided by *sigma_l*.
exparg - **double**, argument of the combined exponential term.
squirt - **double**, result of the square root term.
sine - **double**, sine of *exparg*.
cosine - **double**, cosine of *exparg*.

xkm - **float**, real part of random variable term.
ykm - **float**, imaginary part of random variable term.

now - pointer to **float**, points to current term in the random number array, incremented within **imp**.

r - **integer** counter.

Functions called:

sqrt - library function that takes the square root of a non-negative real number, needs MATH.H.

exp - library function that raises *e* to a real number, requires MATH.H.

log - library function that takes the natural logarithm of a non-negative real number, needs MATH.H.

sin - library function that takes the sine of a real number, needs MATH.H.

cos - library function that takes the cosine of a real number, needs MATH.H.

```
void imp(float datb[2 * DATA], struct ray_path pdsi[MAXLAYERS],
                struct compute cdsi[1], float *start, double timexx, int oo)
{
        /* Variables */

    int r;
    float xkm, ykm;
    float *now;
    double tau_k, gg, exparg, squirt, sine, cosine, gag;

        /* Code */

    now = start;
    tau_k = cdsi[0].big_el;

    for (r = 1; r < DATA / 2; r++)
    {
        tau_k += cdsi[0].delta_tau;

        if ((gag = tau_k - pdsi[oo].tau_l) <= 0)
            continue;

            /* Bypass since log(gg) in squirt computation below will be
             * undefined when tau_k is less than or equal to .tau_l */
        else
            gg = gag / pdsi[oo].sigma_l;

        exparg = TWOPI * (timexx * (pdsi[oo].fds + pdsi[oo].slp *
                                    (tau_k - pdsi[oo].tau_c)));
        squirt = sqrt((pdsi[oo].peak_amplitude * exp(pdsi[oo].alpha *
                                (log(gg) - gg + 1))));
        sine = sin(exparg);
        cosine = cos(exparg);
        xkm = *now;
        now++;
        ykm = *now;
        now++;
        datb[r + r] += (float)(squirt * (cosine * xkm - sine * ykm));
        datb[r + r + 1] += (float)(squirt * (cosine * ykm + sine * xkm));
    }
    return;

} /* End of imp */
```

7.6. Additional Information

This section contains additional information that may be of use to understanding, executing, manipulating, and changing the code.

7.6.1 Library Functions Used

exit - terminates the program, used to terminate for improper input arguments, and for unsuccessful input or output file openings and closings, must include STDLIB.H.
log - returns the natural logarithm of a positive real number, must include MATH.H.
sqrt - returns the square root of a non-negative real number, must include MATH.H.
fmod - returns the fractional remainder of one positive **double** divided by another, must include MATH.H.
pow - library function returns *x* to the power of *y* where *x* and *y* are type **double**, needs MATH.H.
cos - returns trigonometric cosine of a real number, must include MATH.H.
sin - returns trigonometric sine of a real number, must include MATH.H.
exp - returns *e* raised to the real argument, must include MATH.H.
fopen - opens files, requires STDIO.H.
fclose - closes files, requires STDIO.H.
fscanf - library function reads from files, requires STDIO.H.
printf - library function reads to files, requires STDIO.H.
fprintf - prints to file, requires STDIO.H.
malloc - allocates memory for the large data arrays, needs STDLIB.H.
free - unallocates memory block, needs STDLIB.H.
sinh - library function takes the hyperbolic sine of a real number, requires MATH.H.

7.6.2. Input Data File Format

This program reads input data from an ASCII file in the following order with white space between values.

slices (**integer**)
delta_t (**float**) [Δt]{microseconds}
afl (**float**)
layers (**integer** between 1 and 3 inclusive)
seed (**integer** between 1 and 30268 inclusive)

[For each layer]

path_Distance (**float**) [D] {kilometers}
center_freq (**float**) [f_c] {megaHertz}
penetrate_freq (**float**) [f_p] {megaHertz}
thick_scale (**float**) [σ] {kilometers}
maxD_hgt (**float**) [h_o] {kilometers}
peak_amplitude (**float**) [A]
sigma_tau (**float**) [σ_τ] {microseconds}
sigma_c (**float**) [σ_c] {microseconds}
sigma_D (**float**) [σ_D] {Hertz}
fsl (**float**) [f_s] {Hertz}
fdl (**float**) [f_{sL}] {Hertz}

[] Indicates variable symbol.
{} Indicates units.

7.6.3. Function Calling Hierarchy

This section contains a modified function calling tree that indicates the functions, including library functions that each function calls. A function calls the functions indented once immediately below it. The hierarchy also indicates the function that calls particular functions. For example, function **main** calls the functions **init** and **doit**. **Little_el** calls the function **funvalue** and the library function **pow**. **Ran1** is called by the function **get_2i_normals** and the library function **log** is called by the functions **comp_arrays**, **big_c**, **funvalue**, **get_2i_normals**, **slicedo**, and **imp**.

main
- **init**
  - **input_data**
    - **{exit, fscanf, printf}**
  - **{exit, fopen, fclose}**
- **doit**
  - **comp_arrays**
    - **{sqrt, log}**
    - **big_c**
      - **{sqrt, sinh, log}**
    - **little_el**
      - **funvalue**
        - **{log}**
      - **{pow}**
    - **{exit}**
    - **out1**
  - **slicedo**
    - **rvgexp**
      - **get_2i_normals**
        - **ran1**
          - **{fmod}**
        - **ran2**
          - **{sqrt, log}**
    - **{cos, sin, log, sqrt}**
    - **outit**
    - **imp**
      - **{sqrt, exp, log, sin, cos}**
    - **little_four**
      - **{sin}**

{} - indicates library functions.