Implementation of LP Transformations

Recall the Levinson-Durbin recursion:

$$\begin{aligned}
&for\ i = 1, 2, \ldots, p \\
&\quad E_0 = r(0) \\
&\quad k_i = \left(r(i) - \sum_{j=1}^{i-1} a_{i-1}(j) r(i-j) \right) / E_{i-1} \\
&\quad for\ \mathrm{j} = 1, 2, \ldots, i-1 \\
&\qquad a_i(i) = k_i \\
&\qquad a_i(j) = a_{i-1}(j) - k_i a_{i-1}(i-j) \\
&\quad E_i = (1 - k_i^2) E_{i-1}
\end{aligned}$$

Predictor to reflection coefficient transformation:

$$\begin{aligned}
&for\ i = p, p-1, \ldots, 1 \\
&\qquad k_i = a_i(i) \\
&\qquad a_{i-1}(j) = \frac{a_i(j) + k_i a_i(i-j)}{1 - k_i^2} \qquad 1 \le j \le i-1
\end{aligned}$$

Reflection to predictor coefficient transformation:

$$\begin{aligned}
&for\ i = 1, 2, \ldots, p \\
&\qquad a_i(i) = k_i \\
&\qquad a_i(j) = a_{i-1}(j) - k_i a_{i-1}(i-j) \qquad 1 \le j \le i-1
\end{aligned}$$

Equivalent Code:

```
#define P_MAX   8       /* order p of LPC analysis, typically 8..14 */

double levinson_durbin(double const * ac, double * ref, double * lpc);
void autocorrelation(int n, double const * x, int lag, double * ac);

/* LPC- and Reflection Coefficients
  * The next two functions calculate linear prediction coefficients
 * and/or the related reflection coefficients from the first P_MAX+1
 * values of the autocorrelation function.
 * Invented by N. Levinson in 1947, modified by J. Durbin in 1959. */

double levinson_durbin( /* returns minimum mean square error    */

        double const * ac,  /*  in: [0...p] autocorrelation values  */
        double       * ref, /* out: [0...p-1] reflection coef's     */
        double       * lpc) /*      [0...p-1] LPC coefficients      */ {
        int i, j;  double r, error = ac[0];

        if (ac[0] == 0)
             {for (i = 0; i < P_MAX; i++) ref[i] = 0; return 0; }
       for (i = 0; i < P_MAX; i++) {
              /* Sum up this iteration's reflection coefficient.*/
              r = -ac[i + 1];
              for (j = 0; j < i; j++) r -= lpc[j] * ac[i - j];
              ref[i] = r /= error;
              /*  Update LPC coefficients and total error.*/
              lpc[i] = r;
              for (j = 0; j < i/2; j++) {
                     double tmp  = lpc[j];
                     lpc[j]     += r * lpc[i-1-j];
                     lpc[i-1-j] += r * tmp;
              }
              if (i % 2) lpc[j] += lpc[j] * r;
          error *= 1.0 - r * r;
    }
    return error;
}
```

```
/* Compute the autocorrelation
 *                    ,--,
 *             ac(i) = >  x(n) * x(n-i)  for all n
 *                    `--'
 * for lags between 0 and lag-1, and x == 0 outside 0...n-1
 */
void autocorrelation(
        int   n, double const * x,   /*  in: [0...n-1] samples x   */
        int lag, double       * ac)  /* out: [0...lag-1] ac values */
{
        double d; int i;
        while (lag--) {
                for (i = lag, d = 0; i < n; i++) d += x[i] * x[i-lag];
                ac[lag] = d;
        }
}
```

Gain Matching

How do we match the LP spectrum to the DFT?