```
__global__ void
culsp_kernel(float *d_t, float *d_X, float *d_P,
             float df, int N_t)
{

  __shared__ float s_t[BLOCK_SIZE];
  __shared__ float s_X[BLOCK_SIZE];

  // Calculate the frequency

  float f = (blockIdx.x*BLOCK_SIZE+threadIdx.x+1)*df;

  // Calculate the various sums

  float XC = 0.f;
  float XS = 0.f;
  float CC = 0.f;
  float CS = 0.f;

  float XC_chunk = 0.f;
  float XS_chunk = 0.f;
  float CC_chunk = 0.f;
  float CS_chunk = 0.f;

  int j;

  for(j = 0; j < N_t; j += BLOCK_SIZE) {

    // Load the chunk into shared memory

    __syncthreads();

    s_t[threadIdx.x] = d_t[j+threadIdx.x];
    s_X[threadIdx.x] = d_X[j+threadIdx.x];

    __syncthreads();

    // Update the sums

    #pragma unroll
    for(int k = 0; k < BLOCK_SIZE; k++) {

      float ft = f*s_t[k];
      ft -= (int) ft;

      float c;
      float s;

      __sincosf(TWOPI*ft, &s, &c);

      XC_chunk += s_X[k]*c;
      XS_chunk += s_X[k]*s;
      CC_chunk += c*c;
      CS_chunk += c*s;

    }

    XC += XC_chunk;
    XS += XS_chunk;
    CC += CC_chunk;
    CS += CS_chunk;

    XC_chunk = 0.f;
    XS_chunk = 0.f;
    CC_chunk = 0.f;
    CS_chunk = 0.f;

  }

  float SS = (float) N_t - CC;

  // Calculate the tau terms

  float ct;
  float st;

  __sincosf(0.5f*atan2(2.f*CS, CC-SS), &st, &ct);

  // Calculate P

  d_P[blockIdx.x*BLOCK_SIZE+threadIdx.x] =
      0.5f*((ct*XC + st*XS)*(ct*XC + st*XS)/
            (ct*ct*CC + 2*ct*st*CS + st*st*SS) +
            (ct*XS - st*XC)*(ct*XS - st*XC)/
            (ct*ct*SS - 2*ct*st*CS + st*st*CC));

  // Finish

}
```

FIG. 1.— Abridged source for the CULSP computation kernel.