Qspice Project - Fourier Series Coefficients and Fourier Synthesis

KSKelvin Kelvin Leung
2-27-2026

Fourier Series Coefficients and Fourier Synthesis

- Fourier Series Coefficients
  - Breaking down a signal into its frequency components (calculating a_0, a_1, b_1, etc.)
- Fourier Synthesis (or Inverse Fourier Series)
  - Rebuilding the signal by summing its sinusoidal components using the coefficients
- Project Description
  - This project builds a DLL-block (.cpp) to compute Fourier series coefficients from input signal, and synthesis an output signal with these coefficients in every signal period

Calculate the Fourier series coefficients and Fourier Synthesis

- To calculate fourier series coefficients, it need to know exactly the signal period (fundamental frequency $T = \frac{1}{f_o}$), assume signal is function of $f(t)$

- Calculate a_0
  - a_0 represents the average value of the function over one period
  - $a_0 = \frac{1}{T}\int_0^T f(t)\,dt$

- Calculate a_n and b_n (convolution)
  - To calculate the Fourier series coefficients a_n (cosine terms) and b_n (sine terms)
  - $a_n = \frac{2}{T}\int_0^T f(t)\cos(n\omega t)\,dt$
  - $b_n = \frac{2}{T}\int_0^T f(t)\sin(n\omega t)\,dt$
  - Here, n is the harmonic order, $\omega_0 = 2\pi f_0 = \frac{2\pi}{T}$ is the period of the signal
  - At each harmonic
    - Magnitude : $MAG_n = \sqrt{a_n^2 + b_n^2}$
    - Phase : $PHASE_n = -\tan^{-1}\frac{b_n}{a_n}$, with respect to cosine wave

- Fourier Synthesis
  - $f(t) = a_0 + \sum_{n=1}^{\infty}(a_n\cos(n\omega t) + b_n\sin(n\omega t)) = a_0 + \sum_{n=1}^{\infty} MAG_n\ \cos(n\omega t + PHASE_n)$

Code Explanation : Only Calculate DC and Fundamental

Qspice : \01 DC and Fundamental (DLL)\fourier.cpp

```
// Calculate time difference since last update
double dT = t - inst->lastT;
clkout = false;

// Perform numerical integration of input signal (in) using trapezoidal rule
inst->cumsumIN += (in + inst->lastIN)*dT/2;  // trap integration
// Perform numerical integration for Fourier coefficients using rectangle method
// (simpler but potentially less accurate than trapezoidal)
inst->cumsumCOS1 += in*cos(1*2*M_PI/Tperiod*t)*dT; // rectangle integration (simplier)
inst->cumsumSIN1 += in*sin(1*2*M_PI/Tperiod*t)*dT; // rectangle integration (simplier)

// Check if a period is reached
if (t > inst->Ttarget){

   // Calculate Fourier coefficients (DC component and first harmonic)
   inst->a0 = 1/Tperiod*inst->cumsumIN;   // DC component (average)
   inst->a1 = 2/Tperiod*inst->cumsumCOS1; // Cosine coefficient of 1st harmonic
   inst->b1 = 2/Tperiod*inst->cumsumSIN1; // Sine coefficient of 1st harmonic

   // Display the calculated coefficients
   Display("t=%f : a0=%f; a1=%f; b1=%f\n",t,inst->a0,inst->a1,inst->b1);

   // Reset integration accumulators for next period
   inst->cumsumIN = 0;
   inst->cumsumCOS1 = 0;
   inst->cumsumSIN1 = 0;

   // Set new target time for next period
   inst->Ttarget += Tperiod;
   clkout = true;
}

// Reconstruct output signal using the calculated Fourier coefficients
out = inst->a0 + inst->a1*cos(1*2*M_PI/Tperiod*t) + inst->b1*sin(1*2*M_PI/Tperiod*t);

// Store current time and input value for next iteration
inst->lastT = t;
inst->lastIN = in;
```

Integration (for example)

$$\int_0^T f(t)\,dt$$

$$\int_0^T f(t)\cos(\omega t)\,dt\ ,\ \int_0^T f(t)\sin(\omega t)\,dt$$

Reached a period (calculate coefficient) →

Calculate coefficient

$$a_0 = \frac{1}{T}\int_0^T f(t)\,dt\ ,\ a_1 = \frac{2}{T}\int_0^T f(t)\cos(\omega t)\,dt$$

$$b_1 = \frac{2}{T}\int_0^T f(t)\sin(\omega t)\,dt$$

Fourier Synthesis →

$$f(t) = a_0 + a_1\cos(\omega t) + b_1\sin(\omega t)$$

Code Explanation : General Form

Qspice : \02 General Form (DLL)\fourier.cpp

Calculate coefficient

$$a_0 = \frac{1}{T}\int_0^T f(t)\,dt$$

Calculate coefficient

$$a_n = \frac{2}{T}\int_0^T f(t)\cos(n\omega t)\,dt\ ,\ b_n = \frac{2}{T}\int_0^T f(t)\sin(n\omega t)\,dt$$

Calculate coefficient in magnitude and phase

$$MAG_n = \sqrt{a_n^2 + b_n^2}\ , PHASE_n = -\tan^{-1}\frac{b_n}{a_n}$$

$$f(t) = a_0 + \sum_{n=1}^{\infty} MAG_n\ \cos(n\omega t + PHASE_n)$$

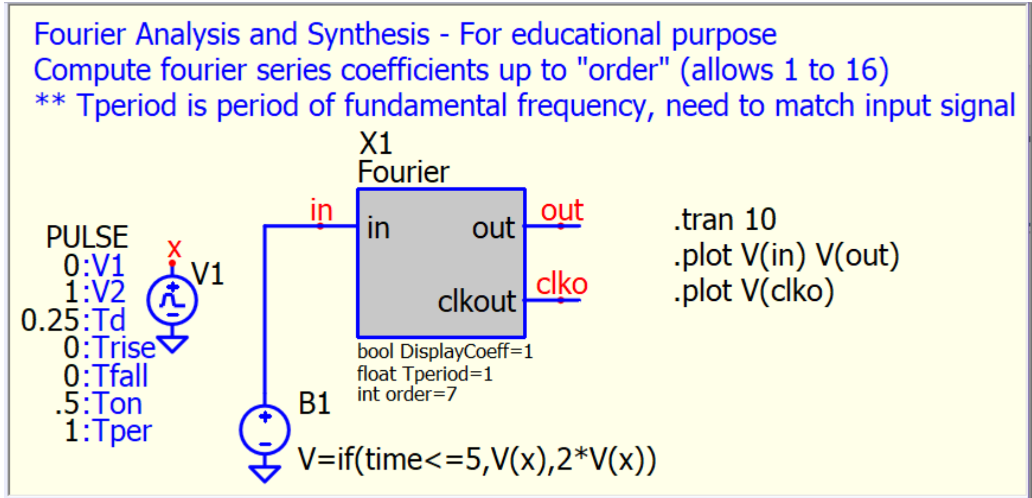
Fourier Synthesis

$$f(t) = a_0 + \sum_{n=1}^{\infty}(a_n\cos(n\omega t) + b_n\sin(n\omega t))$$

```
// Calculate time difference since last update
double dT = t - inst->lastT;
clkout = false;

// Perform numerical integration of input signal (in) using trapezoidal rule
inst->cumsumIN += (in + inst->lastIN)*dT/2;  // trap integration
// Perform numerical integration for Fourier coefficients using rectangle method
// (simpler but potentially less accurate than trapezoidal)
for (int n=1; n <= order; n++){
   inst->cumsumCOS[n] += in*cos(n*2*M_PI/Tperiod*t)*dT; // rectangle integration (simplier)
   inst->cumsumSIN[n] += in*sin(n*2*M_PI/Tperiod*t)*dT; // rectangle integration (simplier)
}

// Check if a period is reached
if (t >= inst->Ttarget){
   // Calculate Fourier coefficients (DC component)
   inst->a[0] = 1/Tperiod*inst->cumsumIN;    // DC component (average)
   inst->cumsumIN = 0;  // Reset integration accumulators for next period
   if (DisplayCoeff){
      Display("\nSimulation time t = %f \n",t); // Display simulation time when period is reached
      Display("   a0 = %9.6f\n",inst->a[0]);   // Display the calculated coefficients
   }

   // Calculate Fourier coefficients (Harmonics)
   for (int n=1; n <= order; n++){
      // Calculate Fourier coefficients
      inst->a[n] = 2/Tperiod*inst->cumsumCOS[n]; // Cosine coefficient of n-th harmonic
      inst->b[n] = 2/Tperiod*inst->cumsumSIN[n]; // Sine coefficient of n-th harmonic
      // Reset integration accumulators for next period
      inst->cumsumCOS[n] = 0;
      inst->cumsumSIN[n] = 0;
      if (DisplayCoeff){
         // Calculate magnitude/phase for each harmonic
         double mag = sqrt(inst->a[n]*inst->a[n] + inst->b[n]*inst->b[n]);
         double phase = -atan2(inst->b[n], inst->a[n]);
         // Display the calculated coefficients
         Display("   a%d = %9.6f; b%d = %9.6f",n,inst->a[n],n,inst->b[n]);
         Display("; mag%d = %4.2f; phase%d = %4.2f\n",n,mag,n,phase*180/M_PI);
      }
   }

   // Set new target time for next period
   inst->Ttarget += Tperiod;
   clkout = true;
}

// Reconstruct output signal using the calculated Fourier coefficients
out = inst->a[0];
for (int n=1; n <= order; n++){
   out += inst->a[n]*cos(n*2*M_PI/Tperiod*t) + inst->b[n]*sin(n*2*M_PI/Tperiod*t);
}

// Store current time and input value for next iteration
inst->lastT = t;
inst->lastIN = in;
```

Example of Fourier Analysis and Synthesis

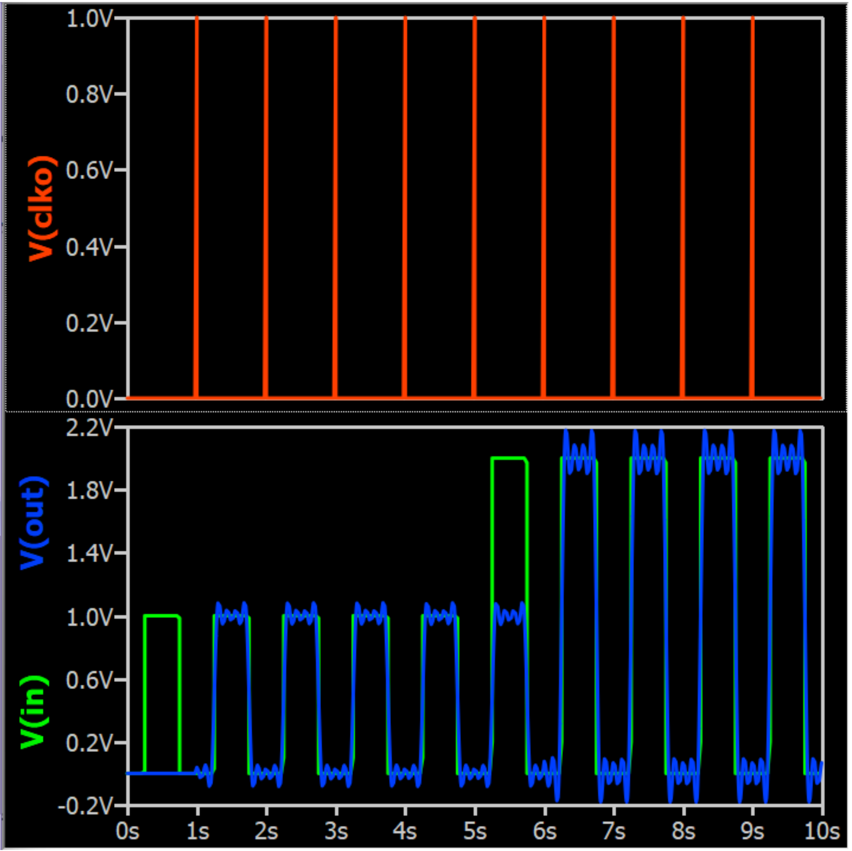
Qspice : \02 General Form (DLL)\example.basic.Fourier.qsch



Output Window

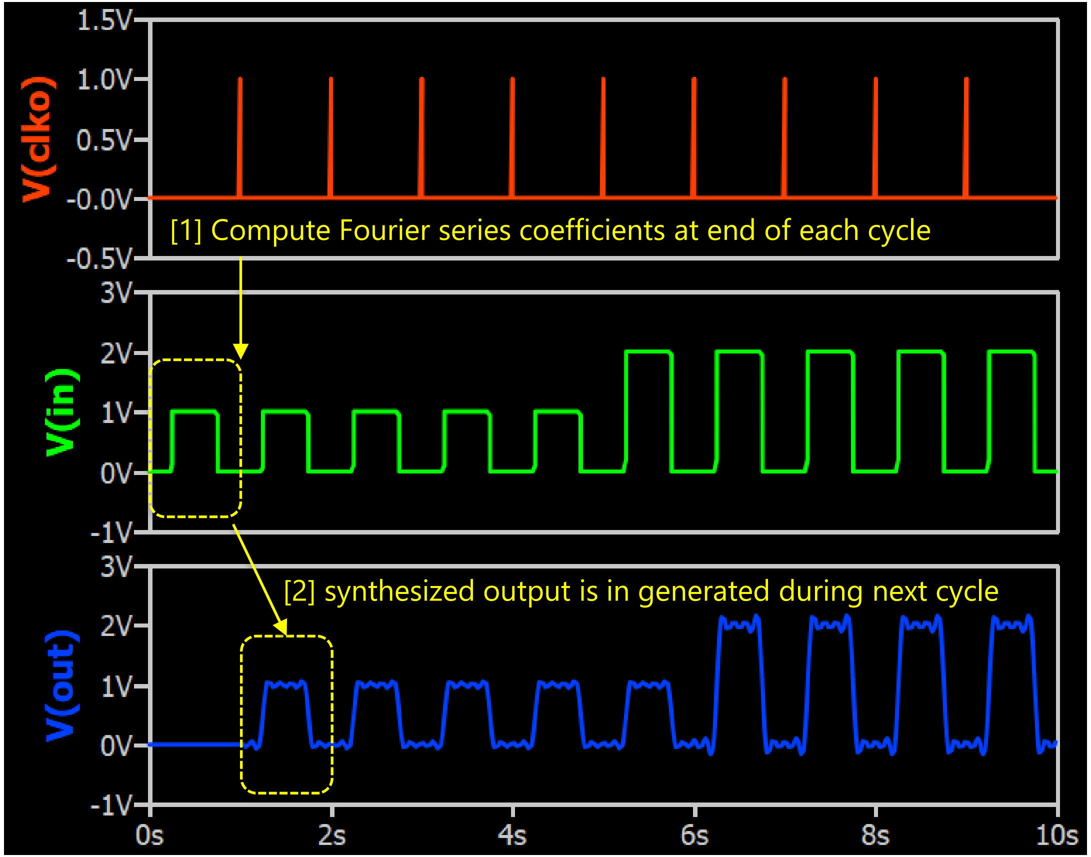
```
X1:
X1: Simulation time t = 1.000024
X1:     a0 =  0.501000
X1:     a1 = -0.636603; b1 = -0.004018; mag1 = 0.64; phase1 = -179.64
X1:     a2 = -0.002000; b2 = -0.000025; mag2 = 0.00; phase2 = -179.28
X1:     a3 =  0.212156; b3 =  0.004018; mag3 = 0.21; phase3 = 1.08
X1:     a4 =  0.001999; b4 =  0.000050; mag4 = 0.00; phase4 = 1.44
X1:     a5 = -0.127240; b5 = -0.004017; mag5 = 0.13; phase5 = -178.19
X1:     a6 = -0.001998; b6 = -0.000075; mag6 = 0.00; phase6 = -177.84
X1:     a7 =  0.090828; b7 =  0.004015; mag7 = 0.09; phase7 = 2.53
```

Simulation | Post Process



Example of Fourier Analysis and Synthesis

Qspice : \Fourier Series Coefficients in Real Time\02 General Form (DLL)\fourier.cpp



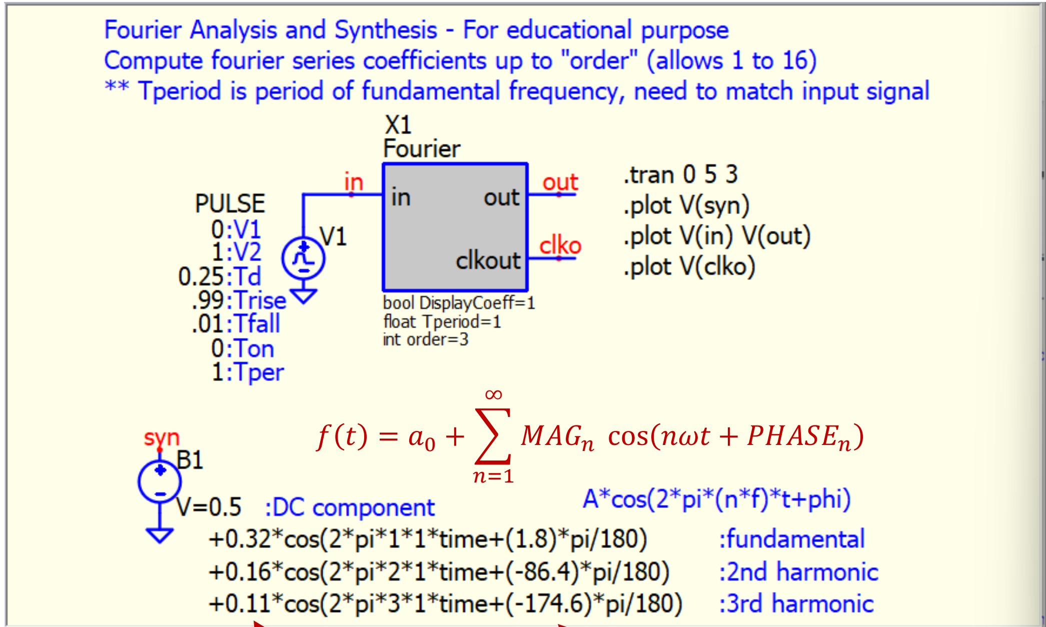
$$a_n = \frac{2}{T}\int_0^T f(t)\cos(n\omega t)\,dt$$

$$b_n = \frac{2}{T}\int_0^T f(t)\sin(n\omega t)\,dt$$

a_n and b_n can only be computed after obtaining the full cycle integration results
Therefore, the synthesized output is delayed by at least one full cycle.

Example of Fourier Analysis and Synthesis

Qspice : \02 General Form (DLL)\example.verify.Fourier.qsch



Output Window

```
X1: Simulation time t = 5.000000
X1:     a0 =  0.500000
X1:     a1 =  0.321314; b1 = -0.010098; mag1 = 0.32; phase1 = 1.80
X1:     a2 =  0.010088; b2 =  0.160340; mag2 = 0.16; phase2 = -86.40
X1:     a3 = -0.106542; b3 =  0.010071; mag3 = 0.11; phase3 = -174.60
```