ECE 521 Lecture #14 Nov. 7, 2016

Note Title 11/7/2016

HW # 3 – Part 2 due today

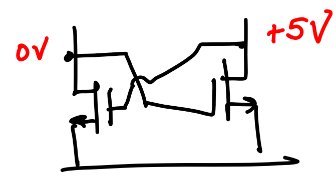
HW # 3 – Part 1 due Nov. 14 (Mon)

HW # 4 – part 2 handed out & due Wed Nov. 16

Project proposal – handed out; due Nov. 16

Initial guess/ conditions in SPICE

- NODESET
- IC

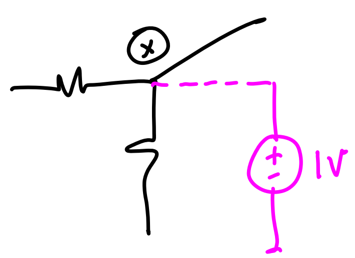


- NODESET is a suggested guess for node voltages in DC analysis

---

- IC holds a node (forces) at a prescribed value for dc analysis. The hold is released for subsequent (transient, ac) analyses

- NODESET is an aid for DC convergence
- IC is useful for different initial conditions for transient analysis, e.g. oscillators



$$\begin{array}{c c} & V_x \\ \text{KCL } x & \begin{bmatrix} & & \\ & 1 & \\ & & \end{bmatrix} \end{array} = \begin{array}{c} \text{RHS} \\ \begin{bmatrix} \\ 1V \\ \\ \end{bmatrix} \end{array}$$

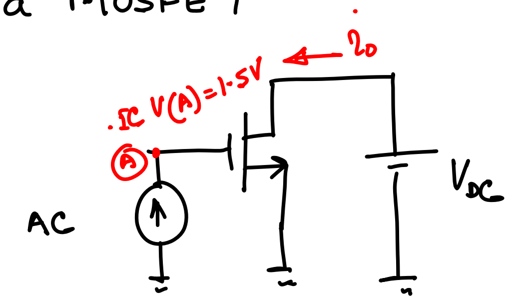
.IC V(x) = 1V

· NODESET 1) implemented as a .IC

2) forced nodes are released another operating point analysis is performed!

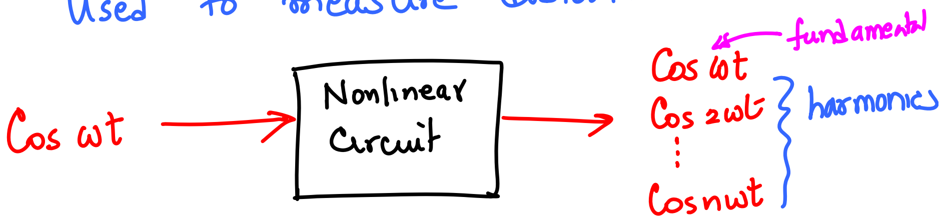
unity gain freq of a MOSFET

$\omega_T \Rightarrow \left| \frac{i_o}{i_i} \right|_{\omega_T} = 1$



Fourier Analysis

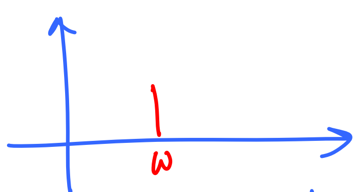
Used to measure distortion in circuits



$i = v^2 = (1 + \cos \omega t)^2$

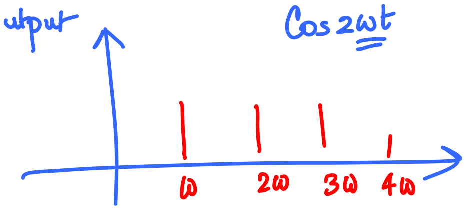
$= 2 + 2\cos \omega t + \underbrace{\cos^2 \omega t}_{\cos 2\omega t}$

Input



Input spectrum

output



output spectrum

Total harmonic distortion

output $= x_0 + x_1 \cos \omega t + x_2 \cos 2\omega t + x_3 \cos 3\omega t + x_4 \cos 4\omega t + \ldots$

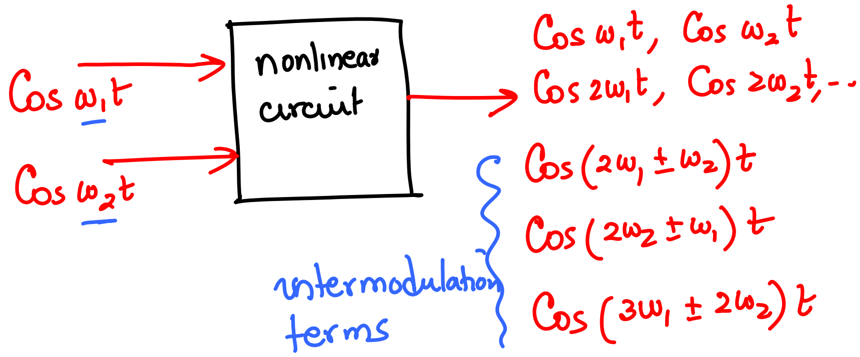
$HD_2 = \frac{|x_2|}{|x_1|}$

⋮

$HD_n = \frac{|x_n|}{|x_1|}$

$THD = \frac{1}{|x_1|} \sqrt{|x_2|^2 + x_3^2 + \cdots}$

Narrowband circuits (intermodulation distortion)



Output is a periodic function with some fundamental frequency

$$x(t) = x(t+T)$$ where T is the period

$$\omega = \frac{2\pi}{T}$$

The Fourier series is given by

$$x(t) = a_0 + \sum_{k=1}^{\infty} \left[a_k \cos k\omega t + b_k \sin k\omega t \right]$$

where

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

$$a_k = \frac{2}{T} \int_0^T x(t) \cos k\omega t \, dt$$

$$b_k = \frac{2}{T} \int_0^T x(t) \sin k\omega t \, dt$$

The above definitions apply to continuous waveforms. In the circuit simulator we have sampled waveforms (discrete time)

$\Rightarrow$ Discrete Fourier Transform (DFT)

FFT faster way

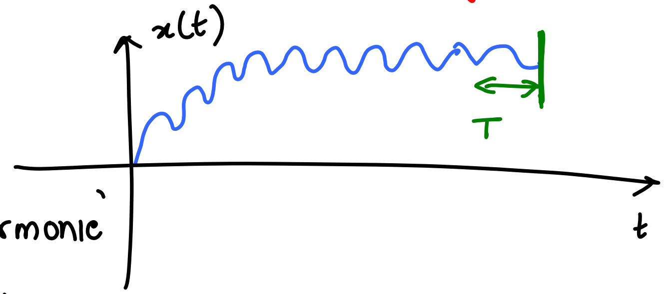
The DFT is calculated on a uniform grid

SPICE Fourier Analysis

- variable timesteps $\Rightarrow$ unequally spaced data

$\Rightarrow$ interpolation to a uniform grid

.FOUR FREQ



SPICE2:

fundamental, 2nd harmonic
⋮
9th "

linear interpolation

SPICE3

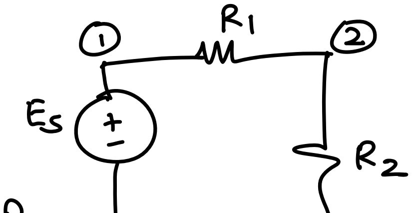
specify # harmonics
Specify type of interpolation
Specify grid size

Network Sensitivity Analysis

- variations of a parameter affecting circuit performance
(tolerance analysis)
- Compare quality of circuits having the same nominal response
- Useful in optimization

Example

$$V_2 = \frac{R_2}{R_1+R_2} E_S$$



How does V_2 change with R_2

$$\frac{\partial V_2}{\partial R_2} = \underline{\underline{\frac{R_1}{(R_1+R_2)^2}}} E_S$$

small variations

Normalized sensitivity

$$S^{V_2}_{R_2} = \frac{\Delta V_2 / V_2}{\Delta R_2 / R_2} = \frac{\Delta V_2}{\Delta R_2} \Big/ V_2/R_2$$

$$= \frac{R_2}{V_2} \frac{\partial V_2}{\partial R_2}$$

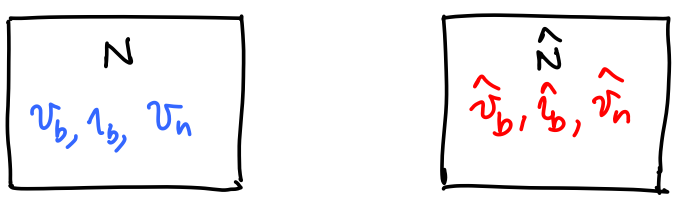
In general, we have some function F that depends on a parameter p

$$S = \frac{\partial F}{\partial p} \; ; \quad S^F_p = \frac{\partial F / F}{\partial p / p}$$

So how do we calculate sensitivity on a computer

→ sensitivity circuits

(the currents and voltages of the sensitivity circuit are the desired sensitivities)



Original Circuit

Sensitivity Circuit

p is the parameter of interest

$$v_R = i_R R$$

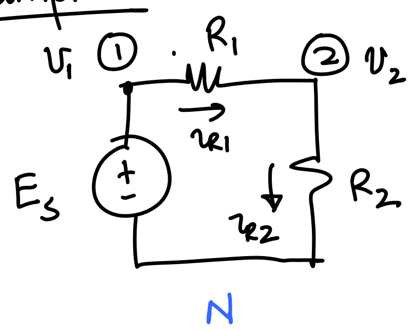
$$\frac{\partial v_R}{\partial p} = R \frac{\partial i_R}{\partial p} + i_R \frac{\partial R}{\partial p}$$

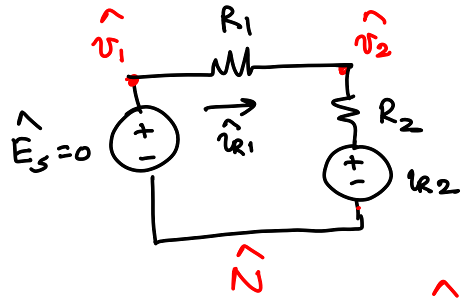
$$\hat{v}_R = R \; \hat{i}_R + \text{voltage source}$$

$$i_I = I_s \Rightarrow \hat{i}_I \; \frac{\partial i_I}{\partial p} = \frac{\partial I_s}{\partial p}$$

$$v_V = V_s \Rightarrow \hat{v}_V \; \frac{\partial v_V}{\partial p} = \frac{\partial V_s}{\partial p}.$$

Example





Here $p = R_2$ and we are interested in $\frac{\partial v_2}{\partial R_2}$ $\hat{v}_2$

$$i_{R2} = \frac{E_s}{R_1 + R_2} \quad ; \quad \hat{v}_2 = i_{R2} - \frac{R_2}{R_1+R_2} i_{R2}$$

$$= \frac{R_1}{(R_1+R_2)} i_{R2}$$

$$\hat{v}_2 = \frac{R_1}{(R_1+R_2)^2} E_s$$

Observations

- The sensitivity circuit has the same topology as the original circuit
- The RHS vector is different for different 'p's
- The LU factors of the original circuit can be used to determine the sensitivity by forward and backward substitution

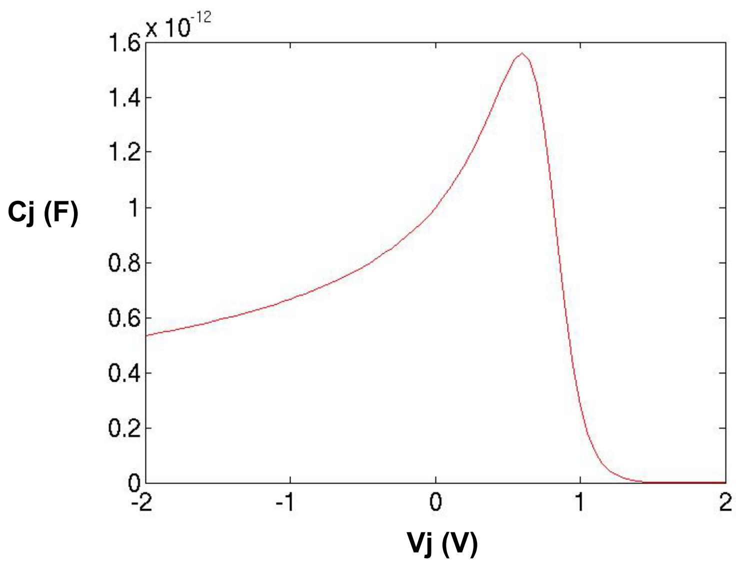
Transient Loop

```
for(time = tstart; time <= tstop; time =+h){
   it_counter = 0;
   converged = FALSE;
   while(!converged){
      // Newton Loop: Clear Rhs, cktMatrix
      //Load circuit matrix
     spFactor(cktMatrix);
     spSolve(cktMatrix,Rhs,Sol);
     it_counter++;
     // Check convergence
  }
  // Current time point -> Previous time point
  tSol = Sol;
}
```

Intgr8 Function

```
void intgr8(x, xdot, h, alpha, beta)
  double x;
  double xdot;
  double h;
  double *alpha, *beta;
{
  /* TR method */
  *alpha = 2.0/h;
  *beta = (-2.0/h * x - xdot);
  /* BE method */
  /*
  *alpha = 1.0/h;
  *beta = -1.0/h * x;
  */
}
```

Nonlinear Capacitor



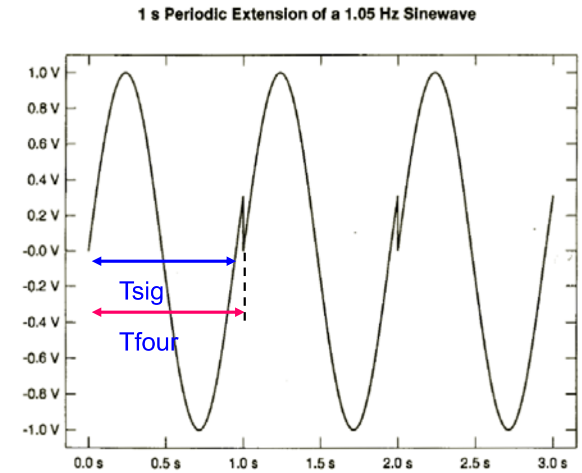
Nonlinear Capacitor Code Template

```
// code template outlining procedure for nonlinear capacitor
  for(i = 1; i <= numNonLinCap; i++) {
     inst = NonLinCap[i];
     Cjo = inst->value;
     na = inst->pNode;
     nb = inst->nNode;
     // Calculate voltage across capacitor. It is assumed that
     // at the first iteration of a new timepoint
     // Sol has the solution from the previous timepoint
     Vc = Sol[na]-Sol[nb];

     // calculate charge and VL
     VL = 0.75-0.1*log(1+exp(-10*(Vc-.75)));
     Q = 1.6*Cjo*(1-sqrt(1-VL/0.8));
     dQdV = (Cjo*exp(-10*Vc+7.5)) / (sqrt(0.0625+0.125*log(1+exp(-10*Vc+7.5)))*(1+exp(-10*Vc+7.5)));
     if(it_count == 0) {
       // first iteration of a given timepoint
       if(time_step_count == 1) {
          // first time point
          inst->qdot = 0;
       }
       else {
          // subsequent time points
          inst->qdot = (inst->alpha)*Q+(inst->beta);
       }
       intgr8(Q,inst->qdot,h,&inst->alpha,&inst->beta);
     }

     Gk = (inst->alpha)*dQdV;
     Ik = (inst->alpha)*Q-(inst->alpha)*dQdV*Vc+(inst->beta);

     // stamp matrix and rhs
  }
```

Incorrect Period

- **Assumption for Fourier analysis is that the waveform is T-periodic**
  - T = Fourier analysis interval
  - X(t) = X(t+T)
- **If the period of signal does not match the Fourier analysis interval**
  - A discontinuity is produced in the waveform
  - A broad frequency spectrum that corrupts results of Fourier analysis
  - Small signals cannot be resolved when an incorrect period is used

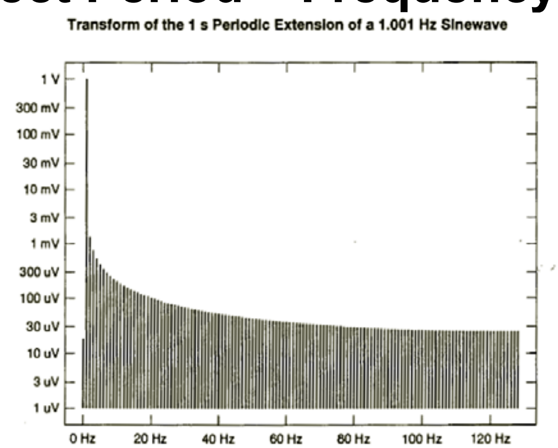
Incorrect Period – Time-domain



Kundert: SPICE & SPECTRE Book

Figure 5.7: The 1 second periodic extension of a 1.05 Hz sinewave. A large difference between the period of the sinewave and the period of the extension was used to make the discontinuity visible in a time-domain waveform. The spectrum of a sinewave with a much smaller difference is shown in Figure 5.8 on the following page.

Incorrect Period – Frequency-domain



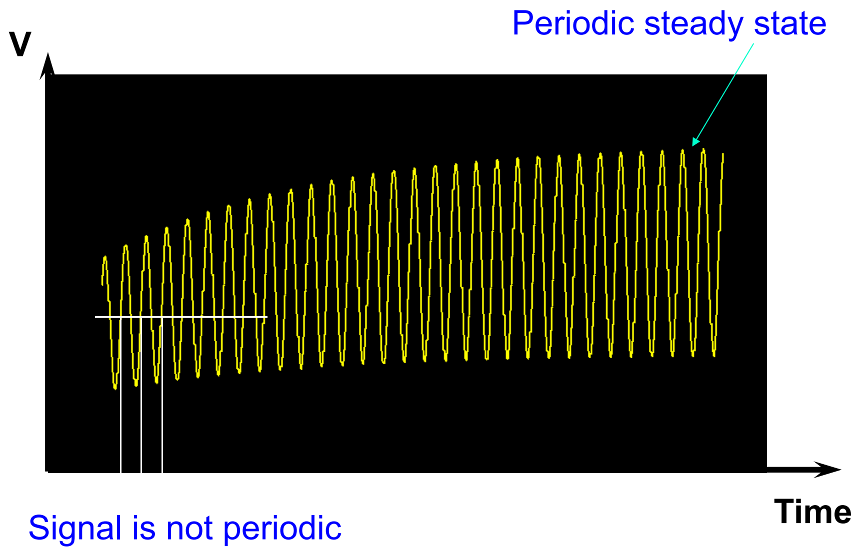
Kundert: SPICE & SPECTRE Book

Figure 5.8: The spectrum of the 1 second periodic extension of 1.001 Hz sinewave. This illustrates how a discontinuity that is so small as to be invisible in the time-domain can severely contaminate a spectrum. Notice how slowly the spectrum drops off as frequency increases. Such discontinuities routinely result when Fourier analysis is applied to waveforms with periods different from the Fourier analysis interval or waveforms generated by circuits that have not completely settled.

Error Due to Transients

- **If signal does not settle to its periodic steady state**
  - Discontinuity
  - Spectral contamination
- **Transient simulation run should be sufficiently long to ensure all transients have settled**
- **Check to make sure signal is periodic or use periodic steady-state analysis (PSS) in Spectre**

Incomplete Settling



Errors Due to Aliasing

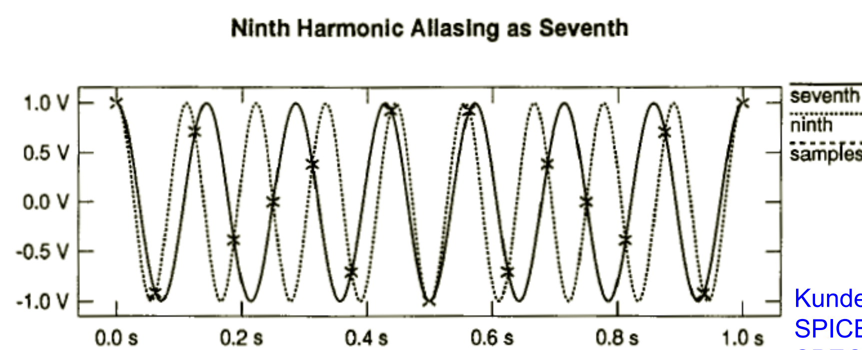
- DFT computes Fourier coefficients of a finite number of frequencies
- Aliasing occurs when energy from higher-order harmonics that are not computed shows up as energy of the computed harmonics
- Sampled data points can represent many signals
- To avoid aliasing signal must be sampled at a rate faster that the Nyquist frequency of the highest frequency signal

Errors Due to Aliasing - Example

- T = 1 sec: sinusoidal waveform $\cos 2\pi t$
  - 7^{th} harmonic $\cos 2\pi 7t$
  - 9^{th} harmonic $\cos 2\pi 9t$

t	$\cos 2\pi 7t$	$\cos 2\pi 9t$
1/16	-0.924	-0.924
2/16	0.707	0.707
…		
8/16	-1	-1
…		
16/16	1	1

Sampled Waveform and Aliasing



Kundert: SPICE & SPECTRE Book

Figure 5.9: An example of aliasing. The seventh and ninth harmonics are sampled at a rate of 16 per period of the fundamental. Notice that the values of both waveforms are identical at the sample points.

Interpolation Errors

- **DFT requires equally spaced data points**
- **Circuit simulators naturally produce unequally spaced time points**

  ⇒Interpolate data onto a uniform grid
- **Interpolation errors can cause problems when resolving small signals**

Interpolation Error Example-1

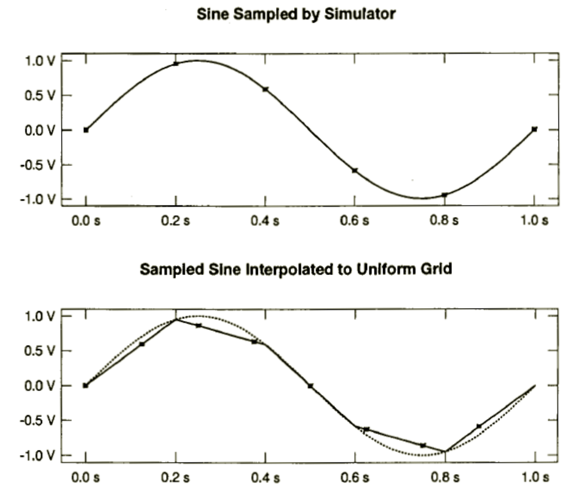


Figure 5.11: The top waveform is sampled directly at 6 equally spaced points. To illustrate the errors caused by interpolating, the 6 equally spaced sample points are linearly interpolated and sampled a 8 equally spaced points, as shown in the top waveform. The 8 sample points do not actually fall on the sine wave and so are in error.

Kundert: SPICE & SPECTRE Book

Interpolation Error Example-1

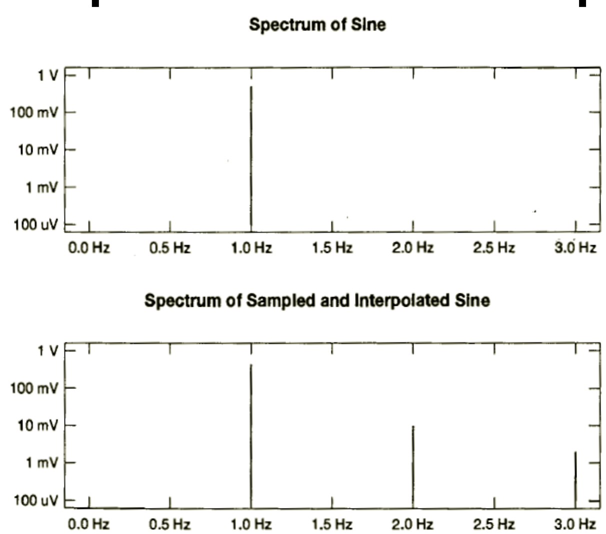


Figure 5.12: The spectrum of the 6 point sampled waveform from the previous figure is shown at the top. It exhibits no distortion. The spectrum of the 8 point interpolated and sampled waveform is shown on the bottom. It shows considerable distortion.

Kundert: SPICE & SPECTRE Book

Interpolation Error Example-2

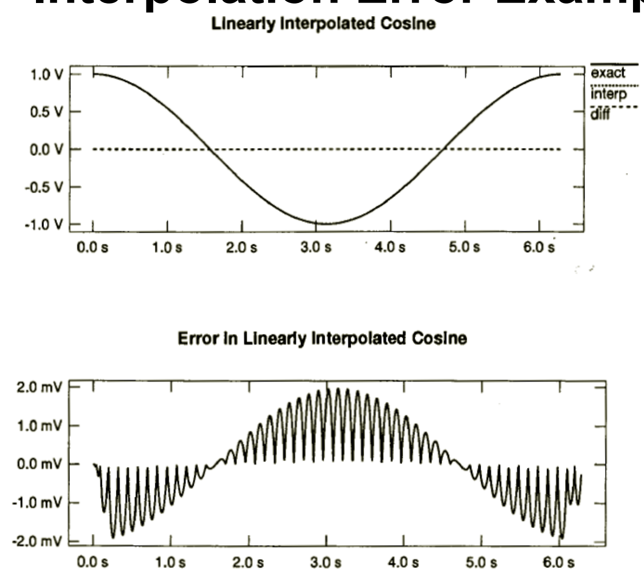
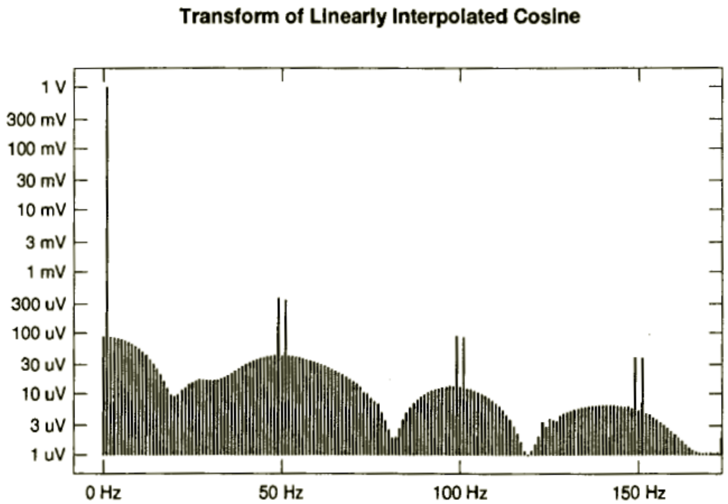


Figure 5.13: A cosine interpolated to 55 roughly equally-spaced points. The difference between the exact cosine and the interpolated cosine is shown in the lower graph.

Kundert: SPICE & SPECTRE Book

Interpolation Error Example-2



Kundert: SPICE & SPECTRE Book

Figure 5.14: The spectrum of the interpolated cosine of Figure 5.13 on the preceding page.

Errors Due to Simulator

- **Numerical algorithms generate errors**
  - Newton convergence
  - Integration method
    - $\Rightarrow$ **Error in Fourier analysis**
- **These errors can be reduced by making RELTOL smaller**

The Fourier Integral Method

Recall:

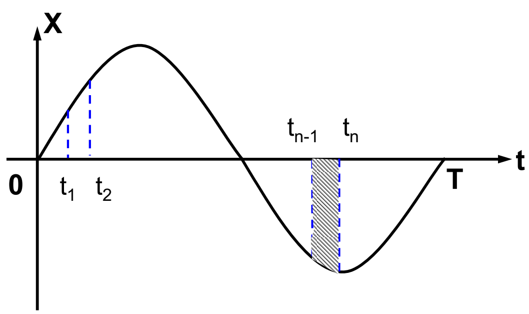
$$x(t) \approx a_0 + \sum_{i=1}^{s} \left(a_i \cos(\omega_i t) + b_i \sin(\omega_i t)\right)$$

$$a_k = \frac{2}{T} \int_0^T x(t) \mathrm{Cos} \frac{2\pi k t}{T} dt$$

For transient analysis: simulation interval discretized at $t_0, t_1, t_2, \ldots, t_N$ and problem solved at each timepoint

$$\therefore a_k = \frac{2}{T} \sum_{n=1}^{N} \int_{t_{n-1}}^{t_n} x(t) \mathrm{Cos} \frac{2\pi k t}{T} dt$$

where $T = t_N - t_0$



The Fourier Integral Method - Observations

The simulator approximates x(t) by a low degree polynomial over the interval $[t_{n-1}, t_n]$ due to the integration method

$$x(t) = \sum_{i=0}^{p} c_{i,n} t^i \qquad \text{for} \quad t_{n-1} \le t \le t_n$$

Use this approximation to write

$$a_k = \frac{2}{T} \sum_{n=1}^{N} \int_{t_{n-1}}^{t_n} \sum_{i=0}^{p} c_{i,n} t^i \mathrm{Cos} \frac{2\pi k t}{T} dt$$

$$= \frac{2}{T} \sum_{n=1}^{N} \sum_{i=0}^{p} c_{i,n} \underbrace{\int_{t_{n-1}}^{t_n} t^i \mathrm{Cos} \frac{2\pi k t}{T} dt}_{\text{Can be integrated in closed form}}$$

$$\int t \mathrm{Cos} k\omega t dt = \frac{1}{(k\omega)^2} \mathrm{Cos} k\omega t + \frac{t}{k\omega} \mathrm{Sin} k\omega t$$

Advantages of Fourier Integral

- **Not subject to aliasing**
- **Can compute only the desired harmonics**
  - Don't require calculation of higher order harmonics for accuracy
- **Not subject to external interpolation errors**
- **Compatible with the non-uniform timesteps used by a simulator**
- **Accuracy similar to that of the integration method used**

Summary

- **Accuracy can be improved by reducing RELTOL**
  - Smaller timesteps
- **Fourier integral method available in Spectre**
- **Stability in computing with higher order ($k \geq 3$) integration methods questionable**