

***=====Create_Arbitrary_waveforms=====**

Instead of creating PWL files, and then including them back into a simulation, one can now create any type of waveform needed without any external files.

This example needs just four voltage sources.

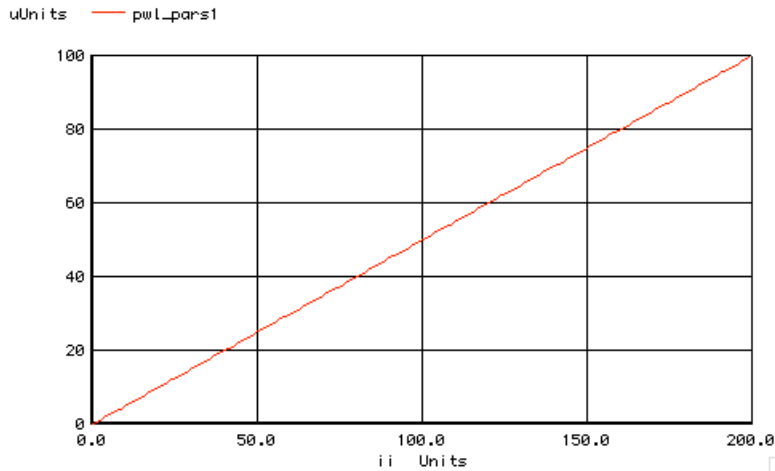
```
Arbitrary_Waveform_Generator
V1 1 0 0 dc
V2 2 0 0 dc
V3 3 0 0 dc
V4 4 0 0 dc
```

The **PWL** array is created in the control section using the "**vector(N)**" function. The intention is to create several arrays which alternates between a time point and a voltage point. If 100 times points are desired, then 200 points for the **pwl_par** array needs to be created. The "**vector(N)**" creates a ramp up array.

```
.control
*=====Want_100_lus_steps=====
let n = 100
let tstep = 1us
*=====The PWL needs 200 point .5us each=====
let pwl_pars1 = vector(2*n)*tstep*0.5
let pwl_pars2 = vector(2*n)*tstep*0.5
let pwl_pars3 = vector(2*n)*tstep*0.5
let pwl_pars4 = vector(2*n)*tstep*0.5
```

The following index **ii** was done so that the **pwl_par** array can be viewed.

```
*=====View The PWL_array=====
let ii=vector(2*$&n)
plot pwl_pars1 vs ii
```



Four different PWL arrays are created by adding voltage values that alternate with time points.

***=====Define_a_array_called_values=====**

```

let values = rnd(unitvec(n)*32768)
*=====PWL_array_alt_time_and_value=====
let index = 0
repeat $n
  let pwl_pars1[1+2*index] = values[index]
  let pwl_pars2[1+2*index] = rnd(8)
  let pwl_pars3[1+2*index] = rnd(8)
  let tt = 3.60*index
  let pwl_pars4[1+2*index] = sin(tt)
  let index = index + 1
end

```

Apparently the following code allows the four arrays to be applied to the four voltage sources as piece wise linear arrays.

```

*=====Install the PWL_arrays=====
alter @v1[pwl] = pwl_pars1
alter @v2[pwl] = pwl_pars2
alter @v3[pwl] = pwl_pars3
alter @v4[pwl] = pwl_pars4

```

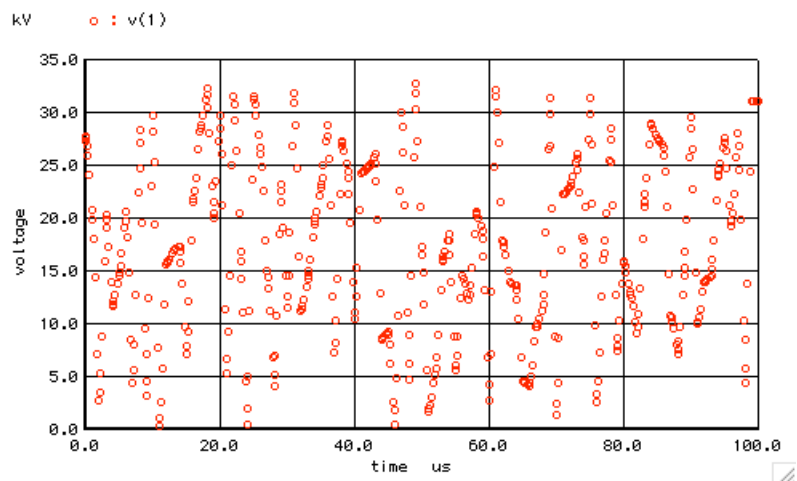
Run a transient analysis, and view the waveforms.

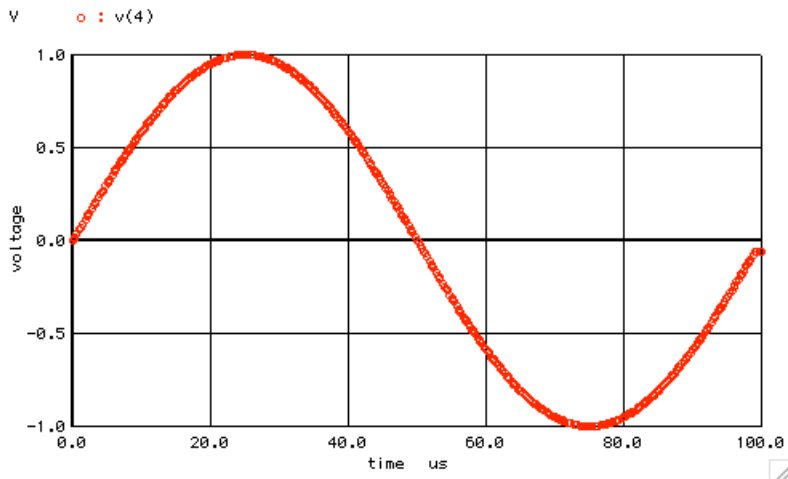
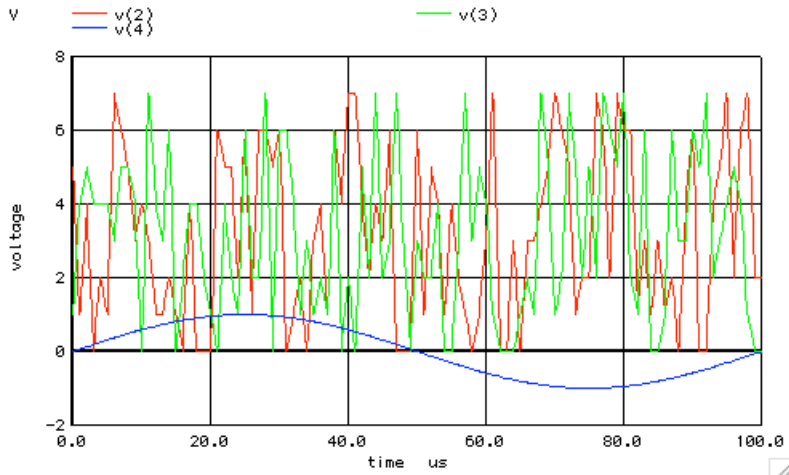
```

*=====Run_and_Plot=====
tran 0.5us 100us
plot v(1) pointplot
plot v(2) v(3) v(4)
plot v(4) pointplot

```

The result are four independent waveforms.





Looks like Spice can now do a arbitrary waveform generator function.

```

=====Full_Netlist_For_Copy_Paste=====
Arbitrary_Waveform_Generator
V1 1 0 0 dc
V2 2 0 0 dc
V3 3 0 0 dc
V4 4 0 0 dc
.control
*=====Want_100_lus_steps=====
let n = 100
let timestep = 1us
*=====The_PWL_needs_200_point_.5us_each=====
let pwl_pars1 = vector(2*n)*timestep*0.5
let pwl_pars2 = vector(2*n)*timestep*0.5
let pwl_pars3 = vector(2*n)*timestep*0.5
let pwl_pars4 = vector(2*n)*timestep*0.5
*=====View_The_PWL_array=====
let ii=vector(2*$&n)
plot pwl_pars1 vs ii

*=====Define_a_array_called_values=====
let values = rnd(unitvec(n)*32768)

*=====PWL_array_alt_time_and_value=====
let index = 0
repeat $&n
  let pwl_pars1[1+2*index] = values[index]
  let pwl_pars2[1+2*index] = rnd(8)
  let pwl_pars3[1+2*index] = rnd(8)
  let tt = 3.60*index
  let pwl_pars4[1+2*index] = sin(tt)
  let index = index + 1
end

*=====Look_at_PWL_array=====
*plot abs(pwl_pars+1e-6) vs ii ylog
*plot abs(pwl_pars2+1e-6) vs ii ylog

```

```
*display
*=====Install_the_PWL_arrays=====
alter @v1[pwl] = pwl_pars1
alter @v2[pwl] = pwl_pars2
alter @v3[pwl] = pwl_pars3
alter @v4[pwl] = pwl_pars4
*=====Run_and_Plot=====
tran 0.5us 100us
plot v(1) pointplot
plot v(2) v(3) v(4)
plot v(4) pointplot
*=====To_View_Arrays=====
*display
*destroy
*display

.endc
.end
```