

```

»x=1 %scalar (% signs are comments and aren't evaluated)
x =
    1

»y=[1 2 3 4 5 6] % sequence vector (row)
y =
    1     2     3     4     5     6

»z=[1;2;3;4;5;6] % list vector (column)
z =
    1
    2
    3
    4
    5
    6

»a=[1 2 3;4 5 6;7 8 9] % matrix
a =
    1     2     3
    4     5     6
    7     8     9

»b=[y;y] %matrix (list of sequences)
b =
    1     2     3     4     5     6
    1     2     3     4     5     6
    1     2     3     4     5     6

»C=[z z z] %matrix (sequences of lists)
c =
    1     1     1
    2     2     2
    3     3     3
    4     4     4
    5     5     5
    6     6     6

»y' %conjugate transpose
ans =
    1
    2
    3
    4
    5
    6

»y=[j*1, j*2, j*3, j*4, j*5, j*6]
y =
Columns 1 through 2
    0 + 1.0000000000000000i    0 + 2.0000000000000000i
Columns 3 through 4
    0 + 3.0000000000000000i    0 + 4.0000000000000000i
Columns 5 through 6
    0 + 5.0000000000000000i    0 + 6.0000000000000000i

»y' %conjugate transpose (watch out, it switches sign of the complex part!) (converts col to row, row to col)
ans =
    0 - 1.0000000000000000i
    0 - 2.0000000000000000i
    0 - 3.0000000000000000i
    0 - 4.0000000000000000i
    0 - 5.0000000000000000i
    0 - 6.0000000000000000i

»y(:) %makes column (this is another way -- this always makes it a column)
ans =
    0 + 1.0000000000000000i
    0 + 2.0000000000000000i
    0 + 3.0000000000000000i
    0 + 4.0000000000000000i
    0 + 5.0000000000000000i
    0 + 6.0000000000000000i

»d=0:1/10:1 %sequence (beg: step: end)
d =
Columns 1 through 4
    0 0.1000000000000000 0.2000000000000000 0.3000000000000000
Columns 5 through 8
    0.4000000000000000 0.5000000000000000 0.6000000000000000 0.7000000000000000
Columns 9 through 11
    0.8000000000000000 0.9000000000000000 1.0000000000000000

»e=sin(2*pi*d) %operation on sequence (operation on each element of sequence)
e =
Columns 1 through 4
    0 0.58778525229247 0.95105651629515 0.95105651629515
Columns 5 through 8
    0.58778525229247 0 -0.58778525229247 -0.95105651629515
Columns 9 through 11
    -0.95105651629515 -0.58778525229247 0

»f=[0:1/8:1;0:1/16:5;0:1/32:25] %list of sequences
f =
Columns 1 through 4
    0 0.1250000000000000 0.2500000000000000 0.3750000000000000
    0 0.0625000000000000 0.1250000000000000 0.1875000000000000
    0 0.0312500000000000 0.0625000000000000 0.0937500000000000
Columns 5 through 8
    0.5000000000000000 0.6250000000000000 0.7500000000000000 0.8750000000000000
    0.2500000000000000 0.3125000000000000 0.3750000000000000 0.4375000000000000

```

```

0.12500000000000 0.15625000000000 0.18750000000000 0.21875000000000
Column 9
1.00000000000000
0.50000000000000
0.25000000000000

»g=sin(2*pi*f) %operation on list of sequences (ex.could have a list of sines sequences with different frequencies)
g =
Columns 1 through 4
    0 0.70710678118655 1.00000000000000 0.70710678118655
    0 0.38268343236509 0.70710678118655 0.92387953251129
    0 0.19509032201613 0.38268343236509 0.55557023301960
Columns 5 through 8
    0 -0.70710678118655 -1.00000000000000 -0.70710678118655
    1.00000000000000 0.92387953251129 0.70710678118655 0.38268343236509
    0.70710678118655 0.83146961230255 0.92387953251129 0.98078528040323
Column 9
    0
    0
1.00000000000000

»whos %variable list and sizes ( helpful when doing vector multiplication)
Name Size Bytes Class

a 3x3 72 double array
ans 1x10 4192 struct array
b 3x6 144 double array
c 6x3 144 double array
d 1x11 88 double array
e 1x11 88 double array
f 3x9 216 double array
g 3x9 216 double array
x 1x1 8 double array
y 1x6 48 double array
z 6x1 48 double array

Grand total is 276 elements using 5264 bytes
leaving 307693008 bytes of memory free.

»
»y-1 %addition
ans =
    0    1    2    3    4    5

»z
z =
    1
    2
    3
    4
    5
    6

»z*(y-1) %outer product 6x1 x 1x6 = 6x6 (makes a list of sequences, each sequence y multiplied by corresponding element of list in z)
ans =
    0    1    2    3    4    5
    0    2    4    6    8   10
    0    3    6    9   12   15
    0    4    8   12   16   20
    0    5   10   15   20   25
    0    6   12   18   24   30

»(y-1)*z %inner product 1x6 x 6x1 = 1x1 (multiplies each corresponding element of y and z, then adds the resulting elements together)
ans =
    70
»0*1+1*2+2*3+3*4+4*5+5*6 inner product
ans =
    70

»[1 2 3]*g %vector x matrix (sequence of inner products)
ans =
Columns 1 through 4
    0 2.05774461196511 3.56226385946836 4.22157654526793
Columns 5 through 8
    4.12132034355964 3.63506112074366 3.18585215990695 3.00061592475332
Column 9
    3.00000000000000

»%g was the list of sine sequences;
»% but you can also think of it as a sequence of lists
»%with each list being the values for the three sinewaves at different times in the sequence.
»%you are doing an inner product between [1 2 3] and each column (list) of the sine sequence.
»%at each time in the sequence, you are scaling each sine wave by a different amount then adding them together.

»%g = [sine1(t0) sine1(t1) sine1(t2)...
      sine2(t0) sine2(t1) sine2(t2)...
      sine3(t0) sine3(t1) sine3(t2)...]

»%[1 2 3]*g=[1*sin1(t0)+2*sin2(t0)+3*sine(t0), 1*sin1(t1)+2*sine2(t1)+3*sine3(t1), ...

=====
Function:
input: f (row),X(row),fs(scalar),dur(scalar)
output: xx(row)

function xx=sumcos(f,X,fs,dur)
.
.
.
xx=?

```

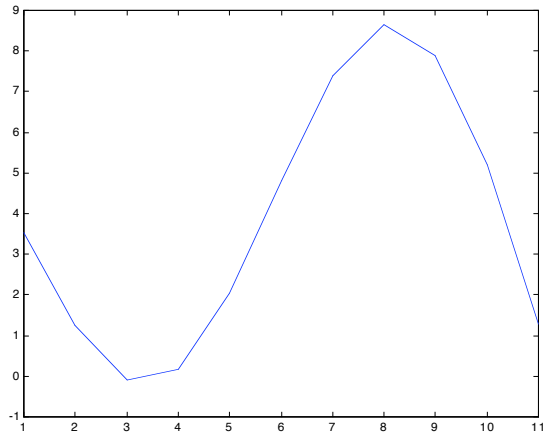
save as sumcos.m

=====

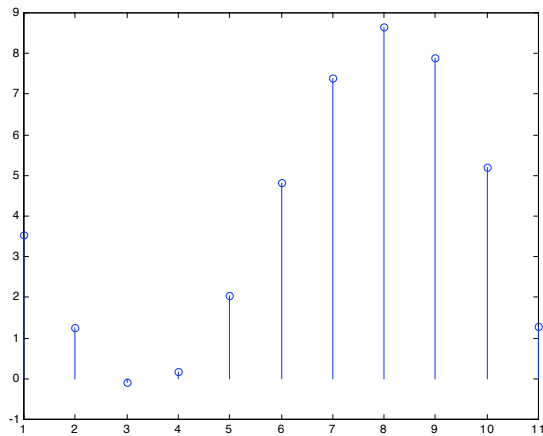
```
»z1=5*exp(j*0.5*pi);  
»z2=5*exp(j*-0.25*pi);  
»zp=sumcos([1, 1/3],[z1,z2],10,1)
```

```
zp =  
Columns 1 through 4  
3.53553390593274 1.25442657826475 -0.08738044898976 0.18315912149992  
Columns 5 through 8  
2.05422141231050 4.82962913144534 7.39395888240421 8.64101238876062  
Columns 9 through 11  
7.90188453672496 5.20887876016010 1.29409522551261
```

```
»plot(zp)
```



```
»stem(zp)
```



=====

output xx and t

```
function [xx,t]=sumcos(f,X,fs,dur)
```

```
%make time sequence from fs & dur
```

```
t=?
```

```
% use outer product to make a list of complex exponential sequences using f and t
```

```
% use the inner product to multiply list of phasors X with list of complex exponential sequences
```

```
xx=?
```

=====

```
»[zp,t]=sumcos([1, 1/3],[z1,z2],10,1)
```

```
zp =  
Columns 1 through 4  
3.53553390593274 1.25442657826475 -0.08738044898976 0.18315912149992  
Columns 5 through 8  
2.05422141231050 4.82962913144534 7.39395888240421 8.64101238876062  
Columns 9 through 11  
7.90188453672496 5.20887876016010 1.29409522551261
```

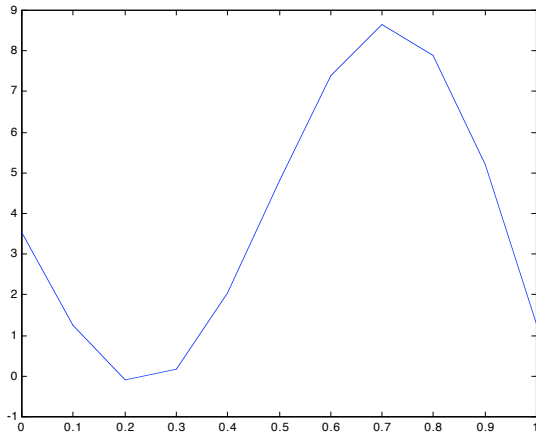
```
t =
```

```

Columns 1 through 4
    0 0.100000000000000 0.200000000000000 0.300000000000000
Columns 5 through 8
    0.400000000000000 0.500000000000000 0.600000000000000 0.700000000000000
Columns 9 through 11
    0.800000000000000 0.900000000000000 1.000000000000000

```

```
»plot(t,zp)
```



```

»cd dspfirst % change directory (you either must be in directory to use the functions, or add path of directory)
»ls % list directory

```

```
Writing_Fast_MATLAB_Code.pdf pkinterp.m
```

```

alphacon.m      pkpick.m
amdemod.m      pretty_w.m
b5syn.mat      pumpkin.m
baboon.mat     replace.m
beatcon.m      recks.mat
beatconb.m     show_img.m
clip.m         showspec.m
cosgen.m       sinedri
dspf.mat       spectgr.m
dspf1.mat      striplot.m
dspf2.mat      sumprod.m
dspfirst.m    tools.gif
dtmfchck.m    tools.mat
dtmfmain.m    trusize.m
echar512.mat  ucplot.m
echart.mat     vowel_d.m
expand.m      wavesnds.m
fnotes.m      wavwisc.m
firfilt.m     wngui.m
furelise.mat  wngui.mat
imsample.m    woodwenv.m
inout.m       wreck.mat
lab6dat.mat   wrinotes.m
lab7dat.mat   zcat.m
lenna.mat     zcoords.m
lenna512.mat  zdrill
lenna_bl.mat  zone.mat
manipsin.m    zone512.mat
mattostr.m    zone_mak.m
musiclab      zprint.m
mysound.m     zvect.m
nveloper.m
pez_31

```

```
»help zprint % if you don't know what the command does, use help
```

```
ZPRINT printout complex # in rect and polar form
```

```
-----
```

```
usage: zprint(z)
```

```
z = vector of complex numbers; each one will be printed
in a format showing real, imag, mag and phase
```

```
»zprint([z1 z2])
```

```

Z = X + jY Magnitude Phase Ph/pi Ph(deg)
    -0 + 5j 5 1.571 0.500 90.00
    3.536 -3.536 5 -0.785 -0.250 -45.00

```

```
»help zvect
```

```
ZVECT Plot vectors in complex z-plane from zFrom to zTo
```

```
usage: HV = zvect(zFrom, <zTo>, <LTYPE>, <SCALE>)
```

```

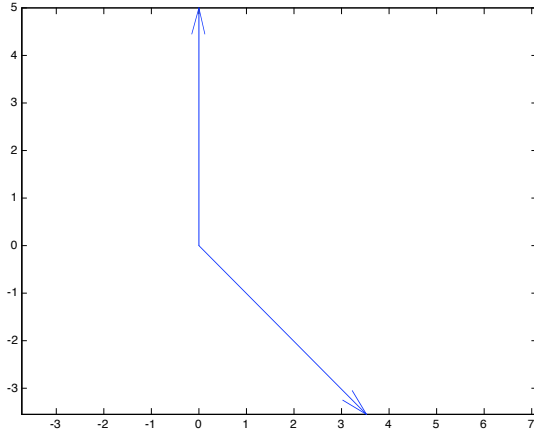
Z: is a vector of complex numbers; each one will be
displayed as an arrow emanating from the origin.
LTYPE: string containing any valid line type (see PLOT)
SCALE: controls size of arrowhead (default = 1.0)
(order of LTYPE and SCALE args doesn't matter)
HV: output handle from graphics of vector plot

```

** With only one input vector: `zvect(Z)`
displays `Z` as arrows emanating from the origin.
** If either `zFrom` or `zTo` is a scalar all vectors will
start or end at that point.

See also `ZCAT`, `PLOT`, `COMPASS`, `ROSE`, `FEATHER`, `QUIVER`.

»`zvect([z1, z2])`



»`help zcat`

`ZCAT` Plot vectors in z-plane end-to-end

usage: `hv = zcat(Z, <LTYPE>, <SCALE>)`

`Z` = vector of complex numbers; each complex # is displayed

as a vector, with the arrows placed end-to-end

`LTYPE`: string containing any valid line type (see `PLOT`)

`SCALE`: varies size of arrowhead (default = 1.0)

(order of `LTYPE` and `SCALE` args doesn't matter)

`hv`: output handle from graphics of vector plot

See also `ZVECT`, `COMPASS`, `ROSE`, `FEATHER`, `QUIVER`.

»`zcat([z1,z2])`

