Utility Functions Octave Codes (0A)

Young Won Lim
1/24/18

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice and Octave.

Based on
M.J. Roberts, Fundamentals of Signals and Systems

Singularity Functions – Continuous Time

- u(t)
- ramp(t)
- rect(t)
- tri(t)
- drcl(t)

u(t), ramp(t), rect(t), tri(t)

```
# Prevent Octave from thinking that this
# is a function file:
1;

function y = u(t)
    zro = t == 0; pos = t > 0; y = zro/2 + pos;
endfunction

function y = ramp(t)
    y = t.*(t >= 0);
endfunction

function y = rect(t)
    y = u(t+0.5) - u(t-0.5);
endfunction

function y = tri(t)
    y = ramp(t+1) - 2*ramp(t) + ramp(t-1);
endfunction
```

u(t)

```
function y = u(t)
    zro = t == 0; pos = t > 0; y = zro/2 + pos;
endfunction
```

t > 0 :	zro=0, pos=1	y=1
t = 0 :	zro=1, pos=0	y=0.5
t < 0 :	zro=0, pos=0	y=0

ramp(t)

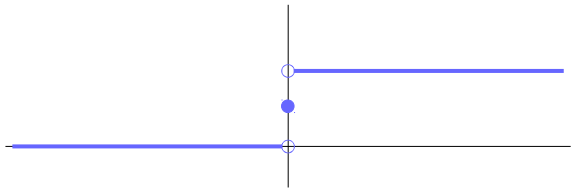
```
function y = ramp(t)
    y = t.*(t >= 0);
endfunction
```

(t>=0)

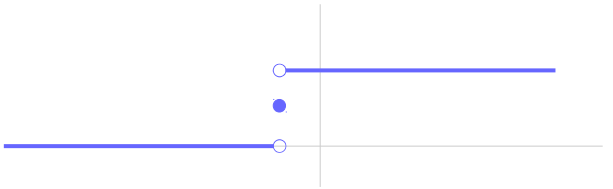
t > 0 :	1	.*	t
t = 0 :	1	.*	t
t < 0 :	0	.*	t

rect(t)

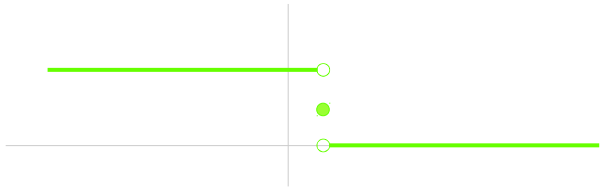
```
function y = rect(t)
    y = u(t+0.5) – u(t-0.5);
endfunction
```



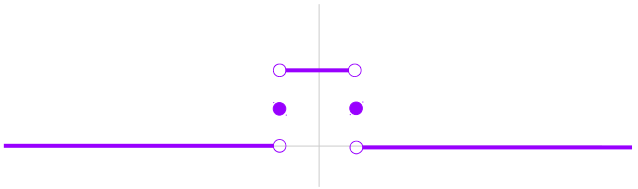
t+0.5 > 0 :	u(t+0.5) = 1
t+0.5 = 0 :	u(t+0.5) = 0.5
t+0.5 < 0 :	u(t+0.5) = 0



t-0.5 > 0 :	u(t-0.5) = 1
t-0.5 = 0 :	u(t-0.5) = 0.5
t-0.5 < 0 :	u(t-0.5) = 0

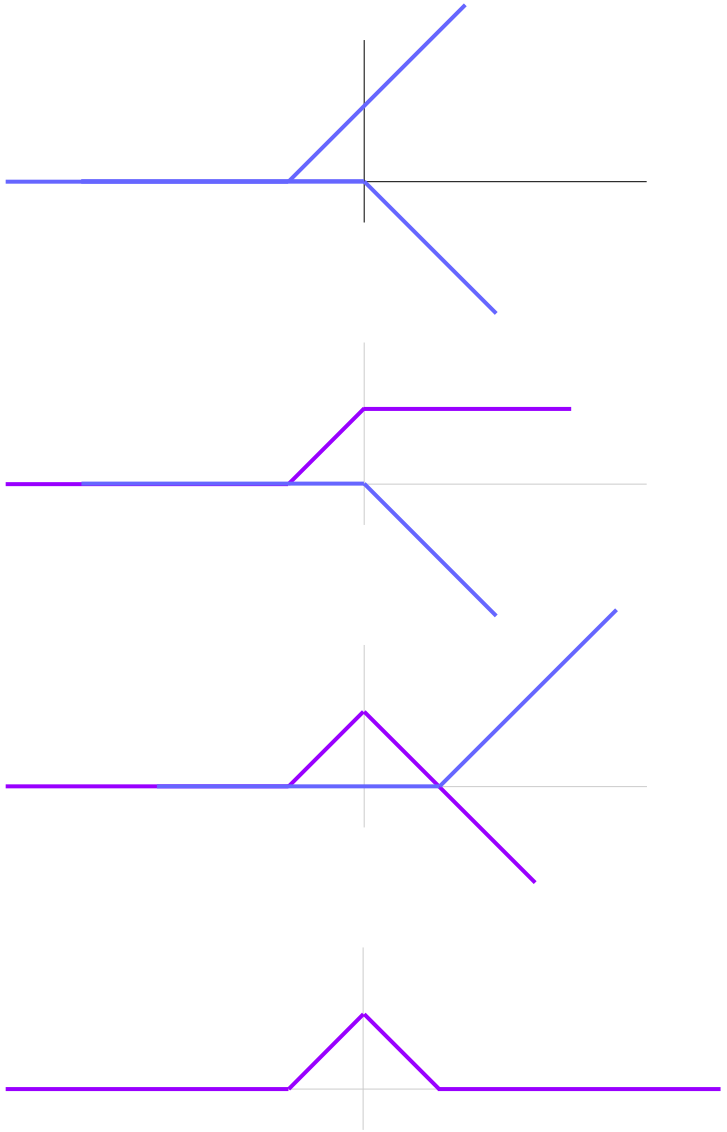


t > +0.5 :	rect(t) = 0
t = +0.5 :	rect(t) = 0.5
-0.5 < t < +0.5 :	rect(t) = 1
t = -0.5 :	rect(t) = 0.5
t < -0.5 :	rect(t) = 0



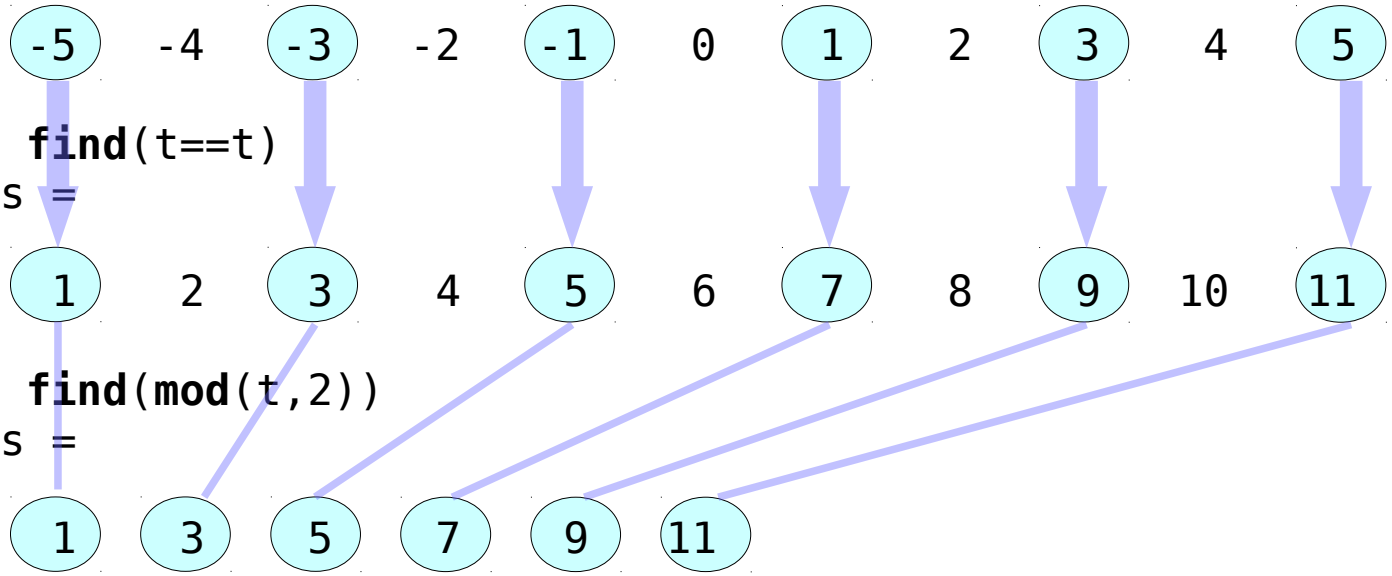
u(t), ramp(t), rect(t), tri(t)

```
function y = tri(t)
    y = ramp(t+1) – 2*ramp(t) + ramp(t-1);
endfunction
```



find( )

```
>> t = -5:+5
t =
```




```
>> find(t==t)
ans =
```



```
>> find(mod(t,2))
ans =
```

drcl(t,N)

```
function x = drcl(t,N)
    num = sin(N*pi*t);
    den = N*sin(pi*t);
    I = find(abs(den) < 100*eps);
    num(I) = cos(N*pi*t(I)); den(I) = cos(pi*t(I));
    x = num ./ den ;
endfunction
```

$$f(t) = \frac{\sin(N\pi t)}{N\sin(\pi t)}$$

$$f(0) = \lim_{t\to 0} \frac{\sin(N\pi t)}{N\sin(\pi t)} = \lim_{t\to 0} \frac{N\pi\cos(N\pi t)}{N\pi\cos(\pi t)} = \lim_{t\to 0} \frac{\cos(N\pi t)}{\cos(\pi t)}$$

```
100*eps =  2.2204e-14 : values very close to zero
```

The function **eps**

eps is the relative spacing between any two adjacent numbers
in the machine's floating point system.

system dependent
IEEE floating point arithmetic

eps is approximately
    2.2204e-16 for double precision
    1.1921e-07 for single precision.

eps return a scalar with the value **eps** (1.0).
eps(x) return the distance between x and the next larger value.
eps(n,m, k, ...) matrix dimensions
eps(..., "double")
eps(..., "single")

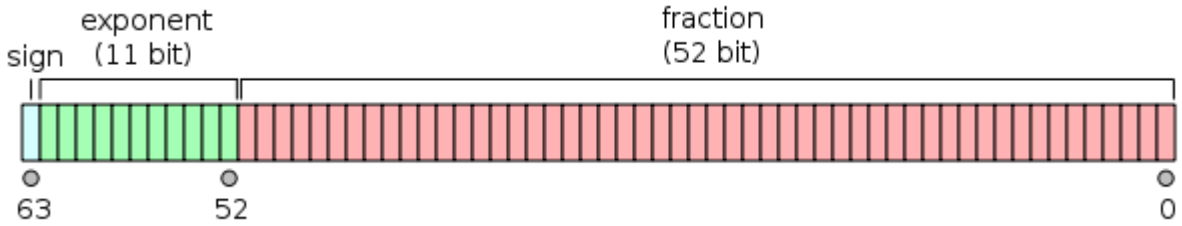
The function **eps**(x)

```
>> num2hex(1)
ans = 3ff0000000000000
>> num2hex(1+eps(1))
ans = 3ff0000000000001
>> num2hex(eps(1))
ans = 3cb0000000000000
>>
>> hex2dec("3cb")-1023
ans = -52
```

1*eps

```
>> num2hex(10)
ans = 4024000000000000
>> num2hex(10+eps(10))
ans = 4024000000000001
>> num2hex(eps(10))
ans = 3ce0000000000000
>>
>> hex2dec("3ce")-1023
ans = -49
```

8*eps

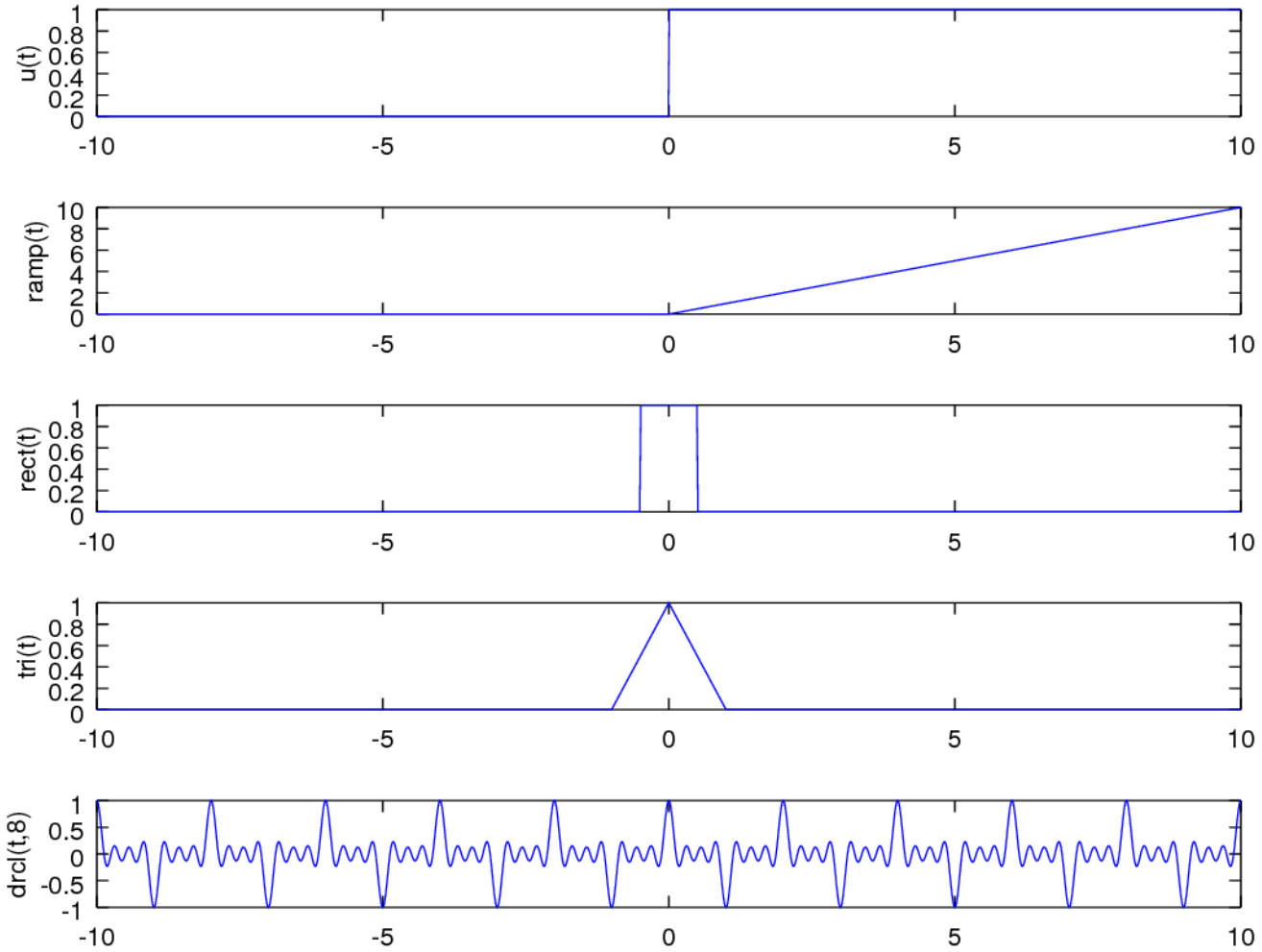


https://en.wikipedia.org/wiki/Double-precision_floating-point_format

Testing CT Singularity Functions

```
t = -10:0.1:+10;

subplot(5, 1, 1);
plot(t, u(t)); ylabel('u(t)');

subplot(5, 1, 2);
plot(t, ramp(t)); ylabel('ramp(t)');

subplot(5, 1, 3);
plot(t, rect(t)); ylabel('rect(t)');

subplot(5, 1, 4);
plot(t, tri(t)); ylabel('tri(t)');

subplot(5, 1, 5);
plot(t, drcl(t, 2)); ylabel('drcl(t,2)');
```

Test Results for CT Singularity Functions



Singularity Functions – Discrete Time

- impD(n)
- uD(n)
- rampD(n)
- rectD(W,n)
- impND(N,n)

impD(n), uD(n), rampD(n)

```
# Prevent Octave from thinking that this
# is a function file:
1;

function y = impD(n)
  y = double(n == 0);
  ss = find(round(n) ~= n);
  y(ss) = NaN;
endfunction

function y = uD(n)
  y = double(n >= 0);
  ss = find(round(n) ~= n);
  y(ss) = NaN;
endfunction

function y = rampD(n)
  pos = double(n>0);
  y = n.*pos;
  ss  = find(round(n) ~= n);
  y(ss) = NaN;
endfunction
```

rectD(W,n), impND(N,n)

```
function y = rectD(W,n);
  if (W == round(W))
     y = double(abs(n) <= abs(W));
     ss = find(round(n) ~= n);
     y(ss) = NaN;
  else
    disp('In rectD, width paraemter, W, is not an
integer');
  endif
endfunction

function y = impND(N, n)
  if (N == round(N))
     y = double(n/N == round(n/N));
     ss = find(round(n) ~= n);
     y(ss) = NaN;
  else
    disp('In impND, period paraemter, N, is not an
integer');
  endif
endfunction
```

impD(n)

```
function y = impD(n)
  y = double(n == 0);
  ss = find(round(n) ~= n);
  y(ss) = NaN;
endfunction
```

n = -1	(n==0) 0	double 0.	round(n)=-1
n = 0	(n==0) 1	double 1.	round(n)= 0
n = +1	(n==0) 0	double 0.	round(n)=+1

n = +1.5	(n==0) 0	NaN	round(n)=+1

uD(n)

```
function y = uD(n)
  y = double(n >= 0);
  ss = find(round(n) ~= n);
  y(ss) = NaN;
endfunction
```

n = -1	(n>=0) 0	double 0.	round(n)=-1
n = 0	(n>=0) 1	double 1.	round(n)= 0
n = +1	(n>=0) 1	double 1.	round(n)=+1

n = +1.5	(n>=0) 1	NaN	round(n)=+1

rampD(n)

```
function y = rampD(n)
  pos = double(n>0);
  y = n.*pos;
  ss  = find(round(n) ~= n);
  y(ss) = NaN;
endfunction
```

n = -1	(n>0) 0	n * double 0.	round(n)=-1
n = 0	(n>0) 0	n * double 0.	round(n)= 0
n = +1	(n>0) 1	n * double 1.	round(n)=+1

n = +1.5	(n>0) 1	NaN	round(n)=+1

rectD(W,n)

```
function y = rectD(W,n);
  if (W == round(W))
     y = double(abs(n) <= abs(W));
     ss = find(round(n) ~= n);
     y(ss) = NaN;
  else
    disp('width paraemter W is not an integer');
  endif
endfunction
```

n = -2	abs(n) 2	abs(W) 1	double 0.	round(n)=-2
n = -1	abs(n) 1	abs(W) 1	double 1.	round(n)=-1
n = 0	abs(n) 0	abs(W) 1	double 1.	round(n)= 0
n = +1	abs(n) 1	abs(W) 1	double 1.	round(n)=+1
n = +2	abs(n) 2	abs(W) 1	double 0.	round(n)=+2

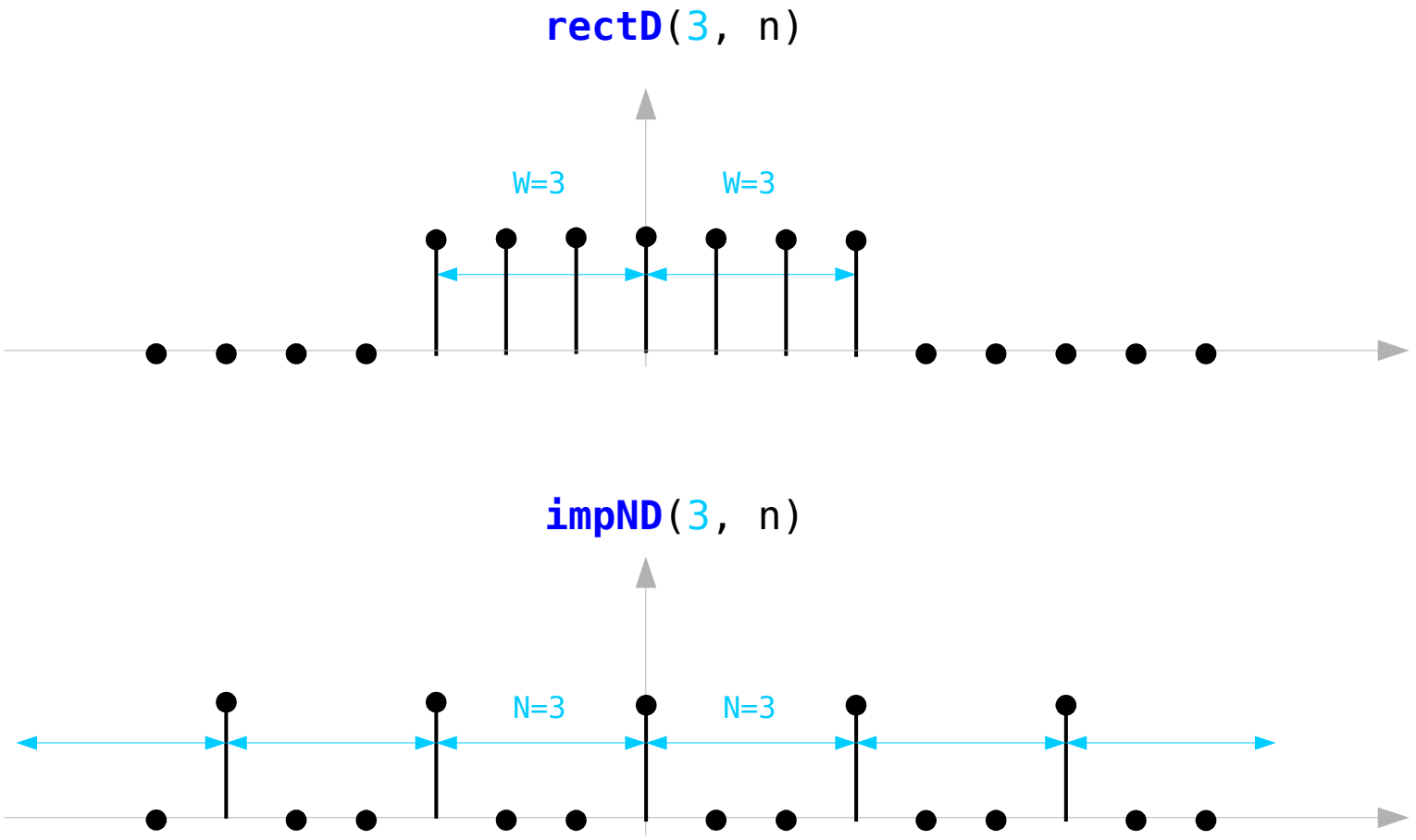
n = +1.5	abs(n)1.5	abs(W) 1	NaN	round(n)=+1

impND(N,n)

```
function y = impND(N, n)
  if (N == round(N))  ---------------------- integer N
     y = double(n/N == round(n/N)); ------- integer (n/N)
     ss = find(round(n) ~= n);
     y(ss) = NaN;
  else
    disp('period paraemter N is not an integer');
  endif
endfunction
```

y becomes 1.0 when n is integer multiple of N (n = kN)
Otherwise y is 0.0

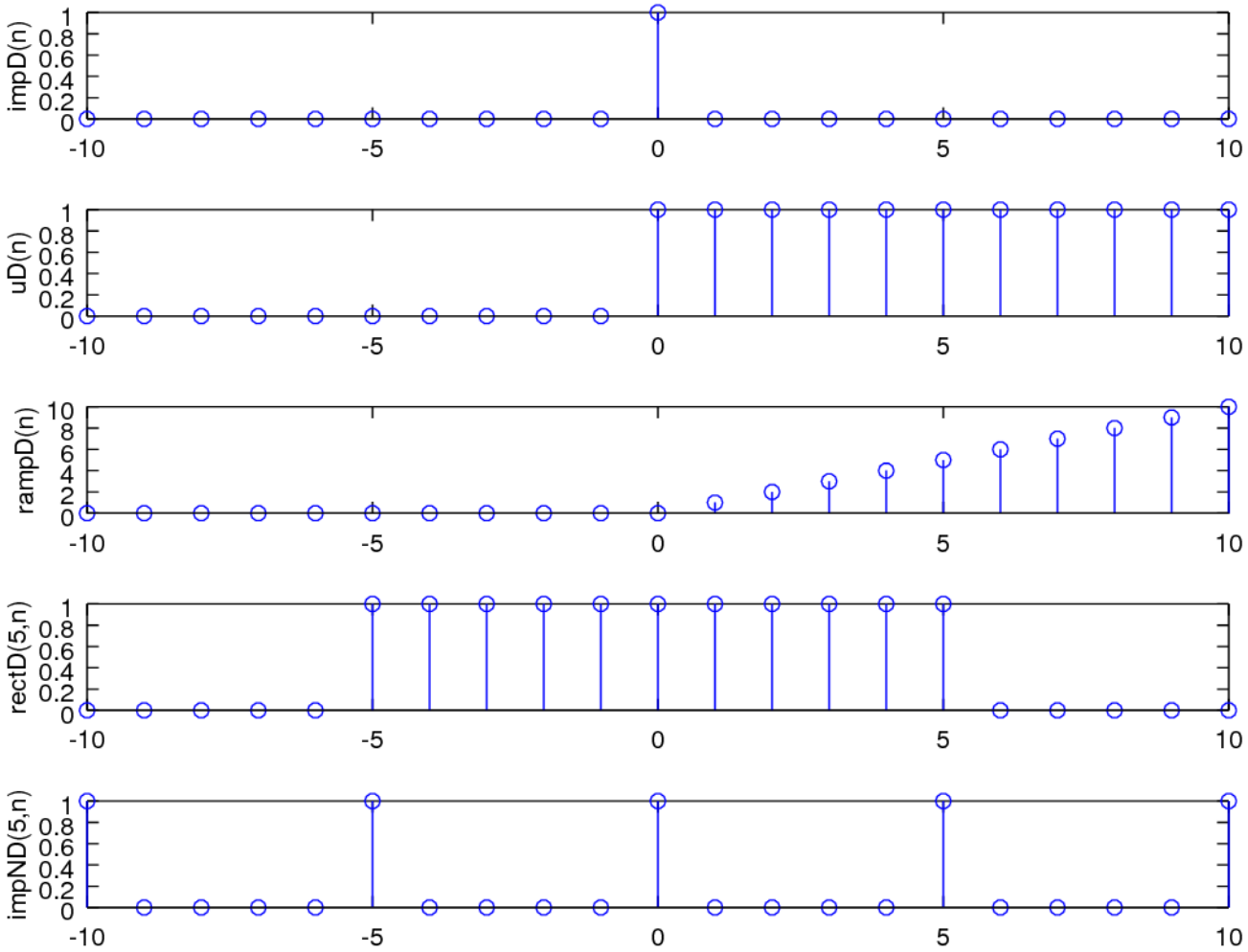
rectD(3,n) and impND(3,n)



Testing DT Singularity Functions

```
n = -10: 10;

subplot(5, 1, 1);
stem(n, impD(n)); ylabel('impD(n)');

subplot(5, 1, 2);
stem(n, uD(n)); ylabel('uD(n)');

subplot(5, 1, 3);
stem(n, rampD(n)); ylabel('rampD(n)');

subplot(5, 1, 4);
stem(n, rectD(5, n)); ylabel('rectD(5,n)');

subplot(5, 1, 5);
stem(n, impND(5, n)); ylabel('impND(5,n)');
```

Test Results for DT Singular Functions



References

[1] http://en.wikipedia.org/
[2] J.H. McClellan, et al., Signal Processing First, Pearson Prentice Hall, 2003
[3] M.J. Roberts, Fundamentals of Signals and Systems
[4] S.J. Orfanidis, Introduction to Signal Processing
[5] K. Shin, et al., Fundamentals of Signal Processing for Sound and Vibration Engineerings

[6] A “graphical interpretation” of the DFT and FFT, by Steve Mann