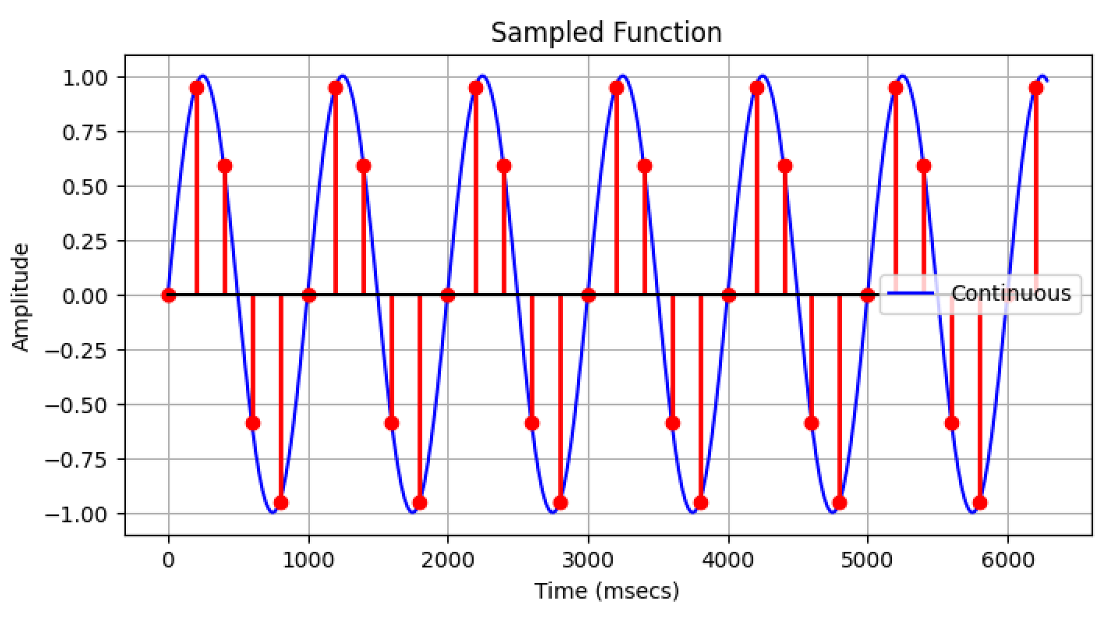
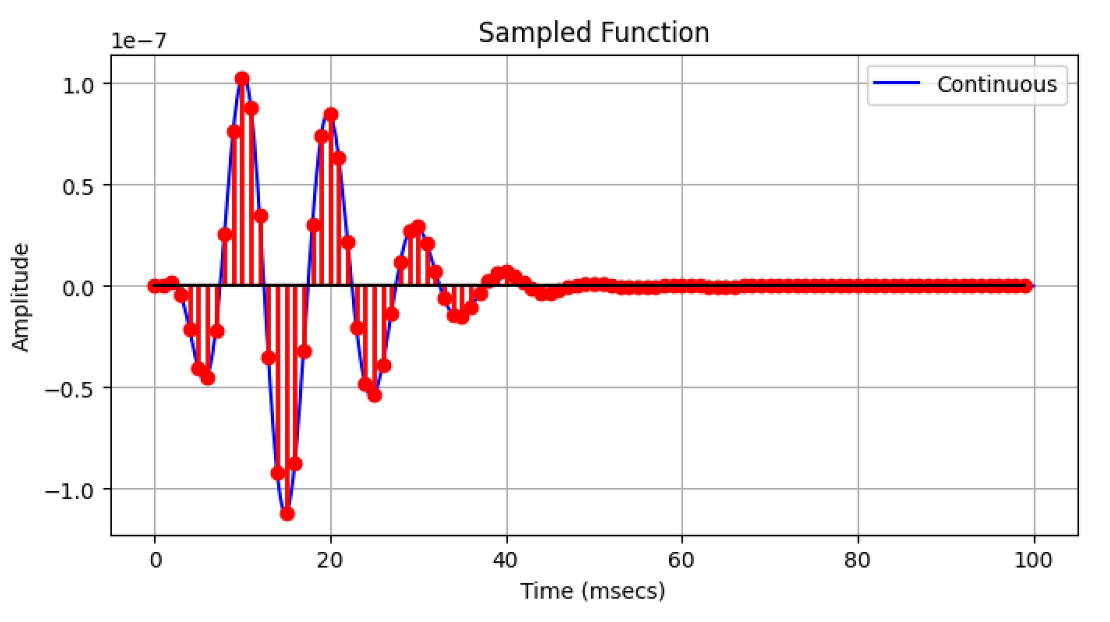
In [1]:
```python
from A3a_rhl72 import *
```

In [2]:
```python
plot_sampled_function(sinewave, fs=5, tlim=(0, 2*np.pi), tscale=1e3, tunits="msecs", f=1.0, d=0.0)


plot_sampled_function(gammatone, fs=1000, tlim=(0, 0.1), tscale=1e3, tunits="msecs", f=100, n=4, phi=0.0, a=1.0)
```





Niquist Frequency

acts as a maximum before aliasing

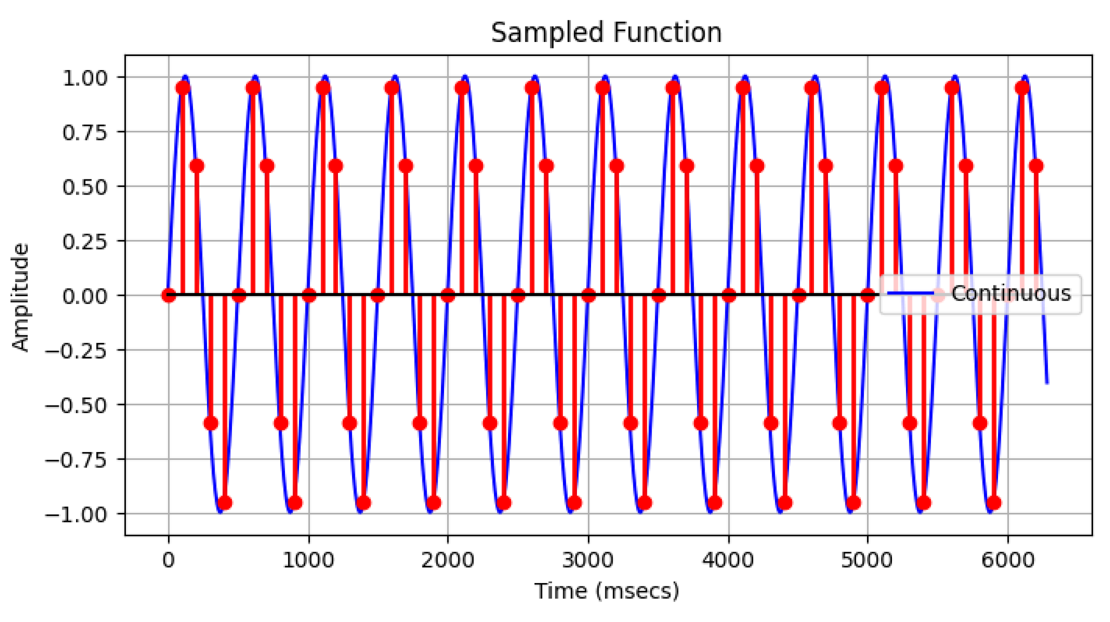
$$f_N = \frac{f_s}{2}$$

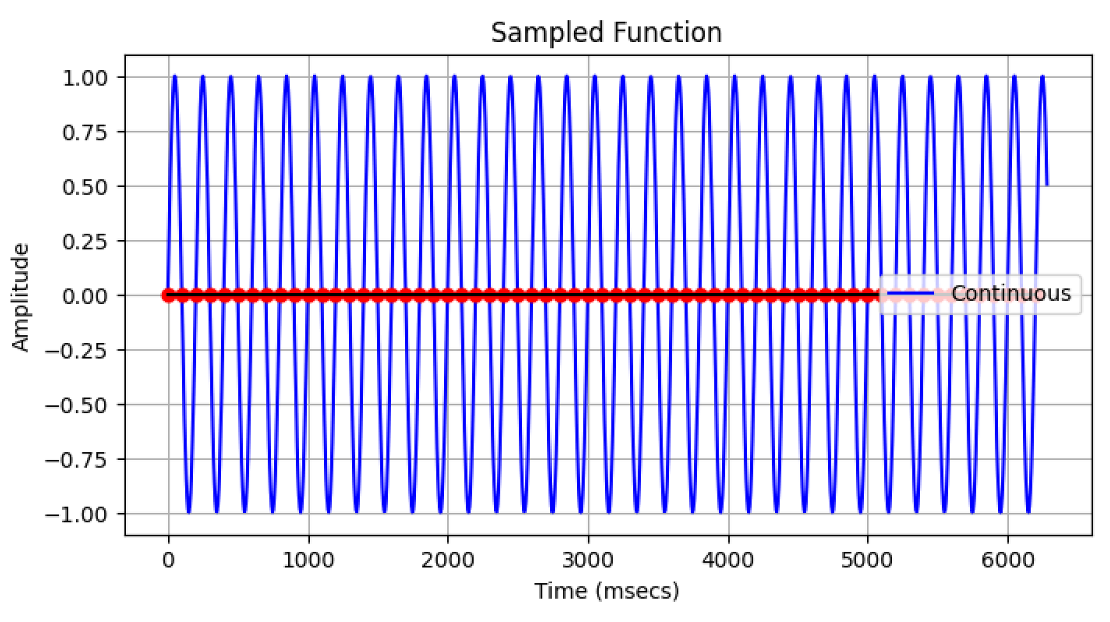
Aliasing

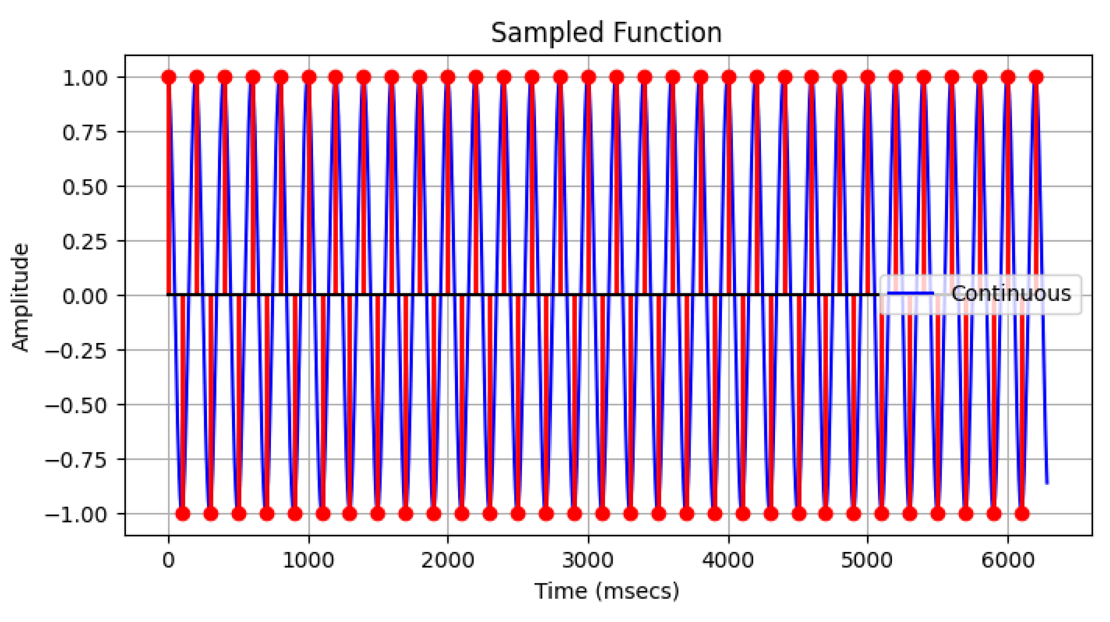
occurs when the frequency of the signal is greater than the Nyquist frequncy

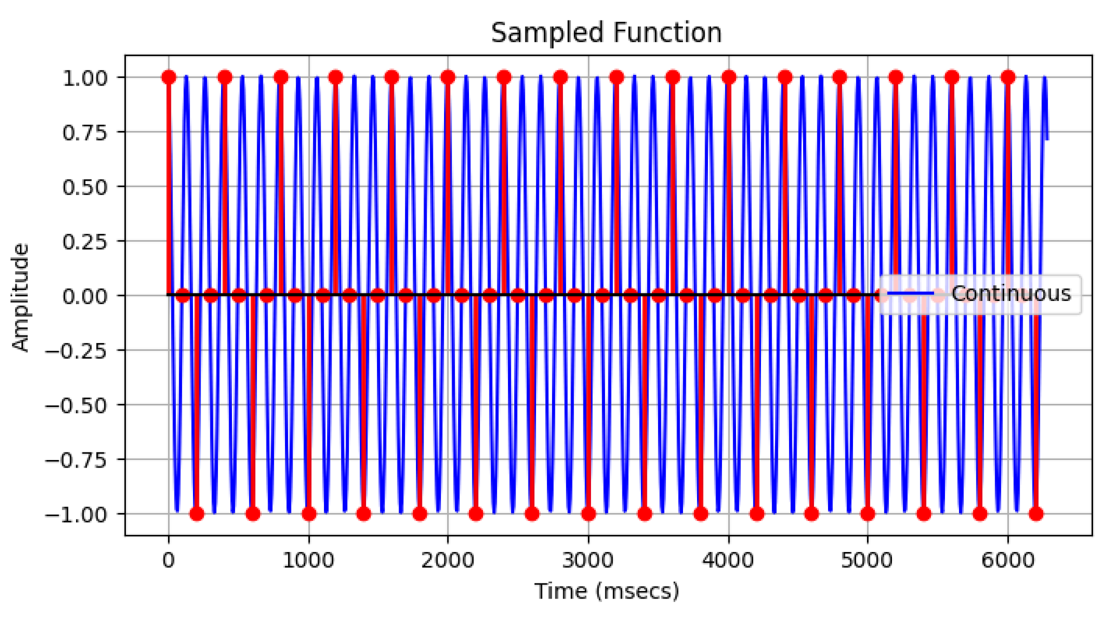
$$f > f_N$$

In [3]:
```python
# Below Nyquist: fs = 10 Hz, f = 2 Hz (clearly sampled with only a few points per cycle)
plot_sampled_function(sinewave, fs=10, tlim=(0, 2*np.pi), tscale=1e3, tunits="msecs", f=2)

# At Nyquist: fs = 10 Hz, f = 5 Hz
plot_sampled_function(sinewave, fs=10, tlim=(0, 2*np.pi), tscale=1e3, tunits="msecs", f=5)

# Cosine at Nyquist: using cosine via phase shift
plot_sampled_function(lambda t, **kw: np.cos(2*np.pi*5*t), fs=10, tlim=(0, 2*np.pi), tscale=1e3, tunits="msecs")

# Above Nyquist: fs = 10 Hz, f = 7.5 Hz, which aliases
plot_sampled_function(lambda t, **kw: np.cos(2*np.pi*7.5*t), fs=10, tlim=(0, 2*np.pi), tscale=1e3, tunits="msecs")
```









Dirac Delta Function

Used to model an impulse or discrete event as a brief impulse of energy.

$$\delta(t) = \begin{cases} \text{undefined} & t = 0, \\ 0 & t \neq 0. \end{cases}$$

but also,

$$\int_{-\infty}^{\infty} \delta(t) dt = 1$$

another way to interpret it is,

$$\int_{-\infty}^{\infty} f(t)\delta(t-\tau) dt = f(\tau)$$

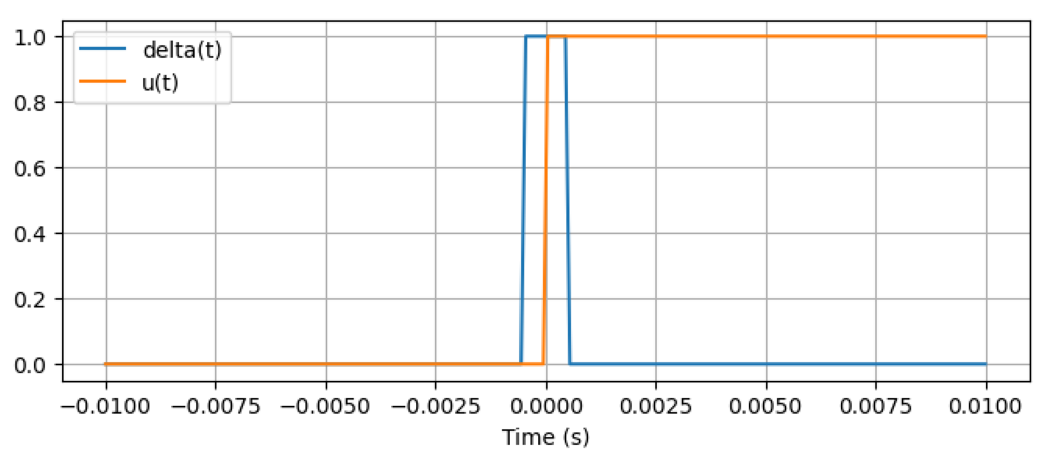
- where $\delta(t-\tau)$ is zero everywhere except at $t = \tau$
- and at the infinitesimal point $f(\tau)$ is a constant and so so multiplies the integral, which is one.

Unit Step Function

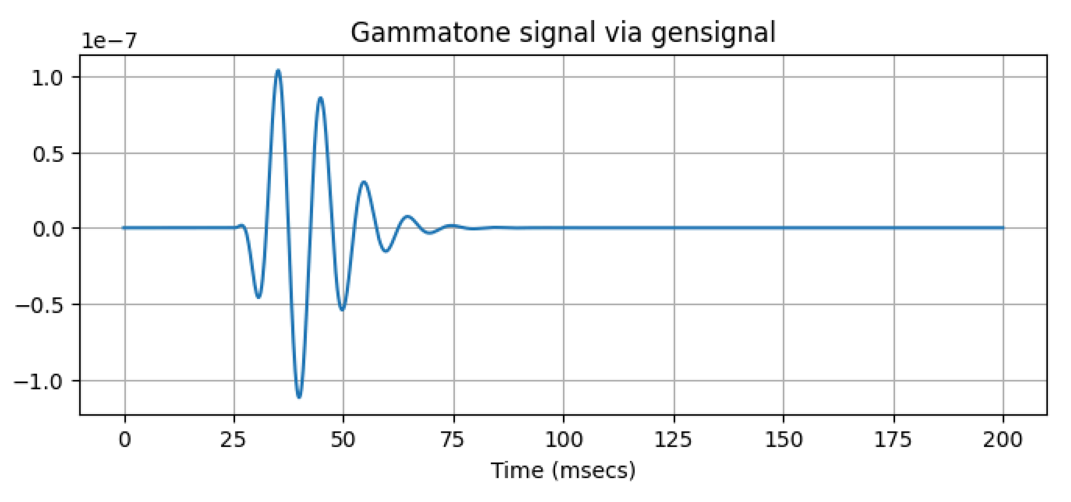
The step function is used to indicate a constant signal that starts at t=0.

$$u(t) = \begin{cases} 1 & t \geq 0, \\ 0 & t < 0. \end{cases}$$

In [4]:
```python
# Example plots:
t = np.linspace(-0.01, 0.01, 200)
plt.figure(figsize=(8,3))
plt.plot(t, delta(t, fs=1000), label="delta(t)")
plt.plot(t, u(t), label="u(t)")
plt.xlabel("Time (s)")
plt.legend()
plt.grid(True)
plt.show()
```



In [5]:
```python
# Example usage: a gammatone signal delayed by 0.025 s and lasting 0.1 s
t = np.linspace(0, 0.2, 1000)
x = gensignal(t, gammatone, tau=0.025, T=0.1, f=100, n=4, phi=0.0, a=1.0)
plt.figure(figsize=(8,3))
plt.plot(t*1e3, x)
plt.xlabel("Time (msecs)")
plt.title("Gammatone signal via gensignal")
plt.grid(True)
plt.show()
```



Power

the average energy over a period.

$$E_x = \sum_{n=1}^{N} |x[n]|^2$$

power of x is then,

$$P_x = \frac{1}{N} \sum_{n=1}^{N} |x[n]|^2 = \sigma_x^2$$

for a signal with additive noise

$$y[t] = x[t] + \epsilon[t]$$

the SNR is simply

$$\frac{P_x}{P_\epsilon}$$

However, it is usually described in decibals

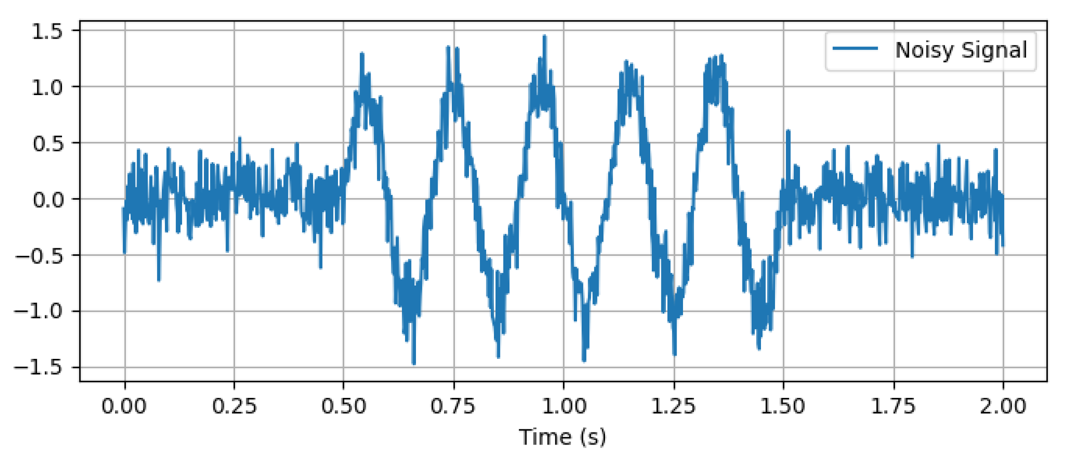
$$dB\ SNR = 10log_{10}(\frac{P_x}{P_\epsilon}) = 20log_{10}(\frac{\sigma_x}{\sigma_\epsilon})$$

Peak Signal Noise Ratio

Is used more often in image processing.

$$PSNR = 10log_{10}(\frac{max_t(y[t])^2}{\sigma_y^2}) = 20log_{10}(\frac{max_t(y[t])}{\sigma_y})$$

In [6]:
```python
# Example: Noisy sinewave signal.
t = np.linspace(0, 2, 1000)
y = noisysignal(t, sinewave, tau=0.5, T=1, sigma=0.2, f=5, d=0.0)
plt.figure(figsize=(8,3))
plt.plot(t, y, label="Noisy Signal")
plt.xlabel("Time (s)")
plt.legend()
plt.grid(True)
plt.show()
```



In [7]:
```python
# Example:
# Suppose x is our signal generated over a time vector.
t = np.linspace(0, 2, 1000)
x = gensignal(t, sinewave, tau=0.5, T=1, f=5, d=0.0)
desired_dBsnr = 10
sigma_required = snr2sigma(x, dBsnr=desired_dBsnr)
print("Noise sigma for", desired_dBsnr, "dB SNR:", sigma_required)
```

```
Noise sigma for 10 dB SNR: 0.15803476716592346
```

In [8]:
```python
# Demonstration:
y_test = noisysignal(t, sinewave, tau=0.5, T=1, sigma=0.2, f=5, d=0.0)
ind = extent(y_test, theta=0.05)
print("Estimated signal extent:", ind)

# One could then compute the SNR using the signal's energy over that extent
if ind is not None:
    sig_range = slice(ind[0], ind[1]+1)
    Ps = power(x[sig_range])
    Pn = power(y_test[sig_range] - x[sig_range])
    print("Estimated SNR (dB):", snr(Ps, Pn))
```

```
Estimated signal extent: (0, 999)
Estimated SNR (dB): 8.411478733825959
```

In [9]:
```python
import numpy as np
from scipy.io.wavfile import write

# Synthesize a 5-second waveform with 20 random gammatones.
t, waveform = grand_synthesis(duration=5, fs=44100, num_tones=20, T=0.1, fmin=200, fmax=2000)

# Save to a .wav file.
```

```python
write("synthesized.wav", 44100, (waveform * 32767).astype(np.int16))
print("Synthesized audio saved as synthesized.wav")
```

```
Synthesized audio saved as synthesized.wav
```