

```

clc; clear; clf;

% Fundamental parameters
To = pi;
wo = 2*pi/To; % = 2 rad/s

% ----- ORIGINAL FS COMPUTATION (with tiny fixes) -----
h = 0.001;
t = 0:h:(To-h);

% ORIGINAL SIGNAL x(t) = exp(-t/2) % <<< CHANGE 1
y = exp(-t/2); % was exp(t/2) before
N = length(y);

Co = sum(y)/(N-1);
for n = 1:10 % Number of Fourier terms
    a(n) = 2*sum(y.*cos(n*wo*t))/(N-1);
    b(n) = 2*sum(y.*sin(n*wo*t))/(N-1);
end

Cn = sqrt(a.^2 + b.^2);
thetan = atan(-b./a);
n = 0:10;

figure;
subplot(2,2,1); stem(n,[Co a],'k'); ylabel('a_n'); xlabel('n');
subplot(2,2,2); stem(n,[0 b],'k'); ylabel('b_n'); xlabel('n'); % <<< CHANGE 2 (a -> b)
subplot(2,2,3); stem(n,[Co Cn],'k'); ylabel('c_n'); xlabel('n');
subplot(2,2,4); stem(n,[0 thetan],'k'); ylabel('theta [rad]'); xlabel('n');

% ----- NEW PART FOR Q2: HARMONICS + PARTIAL SUM -----
N_harm = 5; % first 5 harmonics
t_plot = linspace(0, 3*To, 2000); % for plotting several periods

figure;
hold on; grid on;
for k = 1:N_harm
    xk = a(k)*cos(k*wo*t_plot) + b(k)*sin(k*wo*t_plot); % k-th harmonic
    plot(t_plot, xk, 'DisplayName', sprintf('Harmonic n = %d', k));
end
xlabel('t (s)');
ylabel('Amplitude');
title(sprintf('First %d Harmonics of x(t)', N_harm));
legend('show');

% Partial sum using same coefficients
x_partial = Co * ones(size(t_plot)); % DC term
for k = 1:N_harm
    x_partial = x_partial + a(k)*cos(k*wo*t_plot) + b(k)*sin(k*wo*t_plot);
end

figure;
plot(t_plot, x_partial);
grid on;
xlabel('t (s)');
ylabel('x_N(t)');
title(sprintf('Partial Sum with First %d Harmonics', N_harm));

```

Nice, this is a good time to make a “cheat sheet” for your code.
I’ll treat this as starting from the original code for $x(t) = e^{-t/2}$:

```
To = pi; wo = 2*pi/To;
h = 0.001; t = 0:h:(To-h);

y = exp(-t/2); % <-- ORIGINAL SIGNAL x(t) on [0, To]

N = length(y);
Co = sum(y)/(N-1);
for n = 1:10
    a(n) = 2*sum(y.*cos(n*wo*t))/(N-1);
    b(n) = 2*sum(y.*sin(n*wo*t))/(N-1);
end

Cn = sqrt(a.^2 + b.^2);
thetan = atan2(-b, a);
```

To handle **time operations** correctly for the **periodic** signal, it’s safest to always:

```
tau = mod(something_with_t, To); % bring argument inside [0, To)
y = exp(-tau/2); % because original definition is e^{-t/2} on [0, To)
```

1

Time Reversal: ($x(-t)$)

Goal: coefficients of the periodic signal ($x_p(-t)$).
Change only these lines:

```
t = 0:h:(To-h);
tau = mod(-t, To); % argument = -t wrapped into [0, To)
y = exp(-tau/2); % x(-t)
```

Everything else (N, sums, plots) stays the same.

2

Time Shift: ($x(t - t_0)$) and ($x(t + t_0)$)

Let (t_0) be some shift (e.g. ($t_0 = To/4$)).

(a) Delay: ($x(t - t_0)$)

```
t0 = To/4; % choose any shift
t = 0:h:(To-h);
```

```

tau = mod(t - t0, To); % argument = t - t0 wrapped into [0,To)
y   = exp(-tau/2);      % x(t - t0)

```

(b) Advance: ($x(t + t_0)$)

```

t0  = To/4;
t   = 0:h:(To-h);
tau = mod(t + t0, To); % argument = t + t0 wrapped into [0,To)
y   = exp(-tau/2);      % x(t + t0)

```

Again, everything after y is unchanged.

3 Time Scaling (compression / expansion): ($x(a t)$)

This one is trickier because **time scaling changes the period**.

- Original: period (T_0), signal ($x_p(t)$).
 - New signal ($x_a(t) = x_p(a t)$) has **period** ($T_0' = T_0 / |a|$).
- So you must change **both** the grid and the period.

Let's say ($a = 2$) (**time compression**, "faster" signal):

```

a    = 2;                      % >1: compressed, <1: expanded
To  = pi;                     % original period
To_new = To / abs(a);        % new period
wo  = 2*pi/To_new;

h   = 0.001;
t   = 0:h:(To_new - h);     % integrate over ONE period of the new signal

tau = mod(a * t, To);       % argument = a*t wrapped in ORIGINAL [0,To)
y   = exp(-tau/2);          % x(a t)

```

- For **expansion** (e.g. $a = 0.5$), use the same formula; To_{new} will be larger. Then the rest (N , $a(n)$, $b(n)$, etc.) is the same.

4 Simple amplitude scaling: ($k, x(t)$)

If you just multiply the signal by a constant (k):

```

k   = 3;                      % for example
t   = 0:h:(To-h);
y   = k * exp(-t/2);          % k * x(t)

```

All coefficients just get multiplied by k (including Co, a, b, Cn).

5

Combining operations

You can combine them by putting all operations inside the tau = mod(...):

Example: **time-reversed and delayed**: $(x(-(t - t_0))) = x(t_0 - t)$

```
t0 = To/4;  
t = 0:h:(To-h);  
tau = mod(t0 - t, To); % combine shift + reversal  
y = exp(-tau/2);
```

Example: **compressed and shifted**: $(x(a(t - t_0)))$

```
a = 2;  
t0 = To/4;  
To = pi;  
To_new = To / abs(a);  
wo = 2*pi/To_new;  
  
t = 0:h:(To_new - h);  
tau = mod(a*(t - t0), To);  
y = exp(-tau/2);
```

🔑 Mental template (most important takeaway)

Whenever you want a new signal built from the periodic ($x(t) = e^{-t/2}$):

1. Write its continuous-time form as $(x(\phi(t)))$
(e.g. $(\phi(t) = -t)$, $(t - t_0)$, $(a t)$, etc.)
2. In MATLAB do:

```
tau = mod(phi(t), To); % wrap argument into one period  
y = exp(-tau/2); % because original definition is on [0, To)
```

Then reuse the **same coefficient- and plotting code**.

If you want, next step I can give you a final neat version of the script with a **function handle** like $x_fun(t)$ and a switch for each case (original / reversed / shifted / scaled) so you just change one parameter.