```
% Tune-Up #4

% Copy this file into a Matlab script window, add your code and answers to the
% questions as Matlab comments, hit "Publish", and upload the resulting PDF
 file
% to this page for the tune-up assignment.  Please do not submit a link to a
 file
% but instead upload the file itself.   Late penalty: 2 points per minute
 late.

% The tuneup is to solve homework problem 4.3(b) and verify the solution

% Intro.  A step function u[n] is a function that turns “on” at the origin and
% stays on.  Mathematically, u[n] is
%       1 when n >= 0
%       0 otherwise.
% In Matlab, one can implement u[n] as ( (n) >= 0 ).  The logical operator >=
% returns 1 if true and 0 if false.

% Problem. A certain linear time-invariant (LTI) system gives the output
%       y1[n] = d[n] + 2 d[n-1] - d[n-2]
% when the input to the LTI system is
%       x1[n] = u[n]
% y1[n] is called the step response of the LTI system.  Here, d[n]
% is the unit impulse d[n] = 1 when n = 0 and 0 otherwise.  In Matlab,
% rectpuls(n) implements d[n].

% We’ll model the unknown LTI system as a finite impulse response (FIR)
% filter with input signal x[n] and output signal y[n]
%       y[n] = h[0] x[n] + h[1] x[n-1] + h[2] x[n-2] + ...
% From this information, compute the filter coefficients h[n] for
% n = 0, 1, ..., N-1 and manually verify that the step response of the
% FIR filter is y1[n].

% Deconvolution.  We’ll use deconvolution to compute the filter coefficients.
% We derive the time-domain deconvolution algorithm by evaluating the
% output at n = 0 and let the FIR coefficients be b0, b1, b2, ....:
%      y[n] = b0 x[n] + b1 x[n-1] + b2 x[n-2] + ...
% For LTI systems, it is a necessary (but not sufficient) condition for the
 system
% to be “at rest”, which means that all initial conditions x[-1], x[-2], ...,
 x[-(N-1)]
% must be zero.

% Compute b0:  Since we know x[n] and y[n], we have one equation and one
% unknown at n = 0:
%       y[0] = b0 x[0]   --->   b0 = y[0] / x[0]
% For this calculation to be valid, the first value of the test signal, x[0],
 cannot
% be zero.

% Compute b1: The second output value is y[1] = b0 x[1] + b1 x[0], so
```

```
%       b1 = ( y[1] - b0 x[1] ) / x[0]
% For this calculation to be valid, the first value of the test signal, x[0],
 cannot
% be zero.

% -> b1 = 1

% Compute b2: The third output value is y[2] = b0 x[2] + b1 x[1] + b2 x[0], so
%        b2 = ( y[2] - b0 x[2] - b1 x[1] ) / x[0]
% For this calculation to be valid, the first value of the test signal, x[0],
 cannot
% be zero.

% -> b2 = -1

x = [ 1 2 3 4 5 ];
y = [ 1 1 1 1 1 -5 ];

% Determine Nmax based on input signal
%   Finite-length   length(y) - length(x) + 1
%   Infinite-length length(x)
Nmax = length(y) - length(x) + 1;   %% finite-length input signal
if ( Nmax < 2 )
    Nmax = length(x);
end

b = zeros(1, Nmax);
b(1) = y(1) / x(1);
% b(2) = ( y(2) - b(1)*x(2) ) / x(1);
% b(3) = ( y(3) - b(1)*x(3) - b(2)*x(2) ) / x(1);
for k = 2:Nmax
    numer = y(k);
    n = k;
    for m = 1:(k-1)
 if (n >= 1)
            numer = numer - b(m) * x(n);
 end
        n = n - 1;
    end
    b(k) = numer / x(1);
end

% utdeconvolve.m. implements the above algorithm for deconvolution.

% Part (a).  Give the vectors for x and y that you used when running
% utdeconvolve.m. and the filter coefficients in vector b that the code
 computes.

fprintf('Here is the x vector: ');
fprintf('%d ', x);
fprintf('\n')
fprintf('Here is the y vector: ');
fprintf('%d ', y);
fprintf('\n')
```

```
fprintf('Here is the b vector: ');
fprintf('%d ', b);
fprintf('\n')

% Part (b). Verify that the filter coefficients by using them in the
 difference
% equation for the LTI FIR filter.  You can use the Matlab command conv(x, b).
c = conv(x,b);

fprintf('Here are the filter coefficients: ');
fprintf('%d ', c)
```

```
Here is the x vector: 1 2 3 4 5
Here is the y vector: 1 1 1 1 1 -5
Here is the b vector: 1 -1
Here are the filter coefficients: 1 1 1 1 1 -5
```

Published with MATLAB® R2023a