```
function [U1, U2] = Chap6_CalculateFemForSystems(x)

% Deduce the number of elements
N = length(x) - 1;

% Initialise A_11, A_12, A_21, A_22, b_1, b_2 to zero
A_11 = zeros(N+1, N+1);
A_12 = zeros(N+1, N+1);
A_21 = zeros(N+1, N+1);
A_22 = zeros(N+1, N+1);
b_1 = zeros(N+1, 1);
b_2 = zeros(N+1, 1);

% Loop over elements, calculating non-zero local
% contributions and inserting them into linear system
for k=1:N
    [A_local_11, A_local_12, A_local_21, ...
        A_local_22, b_local_1, b_local_2] = ...
        CalculateContributionOnElement(x, k);

    A_11(k:k+1, k:k+1) = A_11(k:k+1, k:k+1) + A_local_11;
    A_12(k:k+1, k:k+1) = A_12(k:k+1, k:k+1) + A_local_12;
    A_21(k:k+1, k:k+1) = A_21(k:k+1, k:k+1) + A_local_21;
    A_22(k:k+1, k:k+1) = A_22(k:k+1, k:k+1) + A_local_22;

    b_1(k:k+1) = b_1(k:k+1) + b_local_1;
    b_2(k:k+1) = b_2(k:k+1) + b_local_2;
end

% Handle Neumann boundary condition at x=0
b_1(1) = b_1(1) - 7;

% Handle Neumann boundary condition at x=pi
b_2(N+1) = b_2(N+1) + 2;

% Handle Dirichlet boundary condition at x=pi
A_11(N+1,:) = 0;
A_11(N+1,N+1) = 1;
A_12(N+1,:) = 0;
b_1(N+1) = 0;

% Handle Dirichlet boundary condition at x=0
A_21(1,:) = 0;
A_22(1,:) = 0;
A_22(1,1) = 1;
b_2(1) = 0;

% Form global matrix A and vector b
A = [A_11, A_12; A_21, A_22];
b = [b_1; b_2];

% Solve linear system
U = A\b;

% Decompose the solution U into U_1 and U_2
U1 = U(1:N+1);
U2 = U(N+2:2*N+2);
```

```
% Plot the solution
plot(x,U1,'r-',x,U2,'k-')
xlabel('x')
legend('U_1','U_2')



function [A_local_11, A_local_12, A_local_21, ...
    A_local_22, b_local_1, b_local_2] = ...
    CalculateContributionOnElement(x, k)

% Calculate length of element
h = x(k+1)-x(k);

% Calculate analytical contributions to A
A_local_11 = [1/h, -1/h; -1/h, 1/h] - [h/3, h/6; h/6, h/3];
A_local_12 = 3*[1/h, -1/h; -1/h, 1/h] + ...
    8*[h/3, h/6; h/6, h/3];
A_local_21 = 2*[h/3, h/6; h/6, h/3];
A_local_22 = [1/h, -1/h; -1/h, 1/h] - ...
    4*[h/3, h/6; h/6, h/3];

% Define quadrature rule
M = 3;
QuadWeights = [5/18, 4/9, 5/18];
QuadPoints = [0.5*(1-sqrt(0.6)), 0.5, 0.5*(1+sqrt(0.6))];

% Initialise local contributions to b to zero
b_local_1 = zeros(2,1);
b_local_2 = zeros(2,1);

% Loop over quadrature points to evaluate local
% contributions to b
for m = 1:M
    X = QuadPoints(m);
    phi_local = [1-X, X];
    for i=1:2
        b_local_1(i) = b_local_1(i) + h*QuadWeights(m)* ...
            20*sin(2*(x(k)+h*X))*phi_local(i);
        b_local_2(i) = b_local_2(i) + h*QuadWeights(m)* ...
            2*sin(x(k)+h*X)*phi_local(i);
    end
end
```