Code for part_a.m:

```
% Kian Kaas, 301372500
% -----Part A-----
% make arrays that will store the mean times for each type of matrix and
% matrix size
meantimes_full = zeros (5,1);
meantimes_upper = zeros (5,1);
meantimes_tri = zeros (5,1);
meantimes_sparse = zeros (5,1);
p=1; % counter for storing in meantimes arrays
for N_a=[100 200 400 800 1600]    % N values for matrix size N x N

   % create arrays for each trial's time for each type of matrix
   times_full = zeros(50, 1);
   times_upper = zeros(50, 1);
   times_tri = zeros(50, 1);
   times_sparse = zeros(50, 1);
   for j=1:50 % run 50 trials

       % full matrix
       A_full = randn(N_a);   % randomly generated full A matrix (size N x N)
       b_full = randn(N_a,1); % randomly generated full b matrix (size N x 1)
       tic;              % begin timer
       A_full\b_full;    % solve Ax = b with gaussian elimination
       time_taken_full = toc;    % end timer
       times_full(j) = time_taken_full;  % store the trial's time to times_full array
       % upper triangular matrix
       A_upper = randn(N_a);
       A_upper = triu(A_upper);
       b_upper = randn(N_a,1);
       tic;
       A_upper\b_upper;
       time_taken_upper = toc;
       times_upper(j) = time_taken_upper;
       % tridiagonal matrix
       A_tri = diag(randn(N_a-1,1),-1) + diag(randn(N_a,1)) + diag(randn(N_a-1,1),1);   % randomly generated A matrix
(size N x N) (i x i)
       b_tri = randn(N_a,1); % randomly generated b matrix (size N x 1) (i x 1)
       tic;
       A_tri\b_tri;
       time_taken_tri = toc;
       times_tri(j) = time_taken_tri;
       % sparse tridiagonal matrix
       b_sparse = randn(N_a,3);  % Nx3 random matrix (columns are 3 random N-vectors)
       A_sparse = spdiags(b_sparse, -1:1, N_a, N_a);
       tic;
       A_sparse\b_sparse;
       time_taken_sparse = toc;
       times_sparse(j) = time_taken_sparse;
   end
   mean_time_full = mean(times_full);    % calculate mean time of full N x N matrix
   meantimes_full(p) = mean_time_full;   % store mean time for matrix of size N into meantimes_full array
   mean_time_upper = mean(times_upper);
   meantimes_upper(p) = mean_time_upper;

   mean_time_tri = mean(times_tri);
   meantimes_tri(p) = mean_time_tri;
   mean_time_sparse = mean(times_sparse);
   meantimes_sparse(p) = mean_time_sparse;
   p= p+1;
   % print mean times for each type of matrix and its size
   disp("mean time for " + N_a + " x " + N_a + " full matrix: " + mean_time_full + " seconds")
   disp("mean time for " + N_a + " x " + N_a + " upper triangular matrix: " + mean_time_upper + " seconds")
   disp("mean time for " + N_a + " x " + N_a + " tridiagonal matrix: " + mean_time_tri + " seconds")
   disp("mean time for " + N_a + " x " + N_a + " sparse tridiagonal matrix: " + mean_time_sparse + " seconds" + newline)
end
% graph matrices
x = [100 200 400 800 1600]; % set x-axis to N Value (matrix size)
```

```
% graph full matrix
y_full = meantimes_full;
hold on
plot (log10(x), log10(y_full), 'r.', 'MarkerSize', 16)
c_full = polyfit (log10(x), log10(y_full), 1)
v_full = polyval(c_full, log10(x));
plot (log10(x), v_full)  % best fit line
title("Graph of the Log of Mean Time against Log of N for Different Types of Matrices")
xlabel("Log of N Value (log of matrix size)")
ylabel("Log of Mean Time (in seconds)")
% graph upper triangular matrix
y_upper = meantimes_upper;  % y-axis: Mean time (seconds)
plot (log10(x), log10(y_upper), 'm.', 'MarkerSize', 16)
c_upper = polyfit (log10(x), log10(y_upper), 1)
v_upper = polyval(c_upper, log10(x));
plot (log10(x), v_upper)  % best fit line
% graph tridiagonal matrix
y_tri = meantimes_tri;  % y-axis: Mean time (seconds)
plot (log10(x), log10(y_tri), 'b.', 'MarkerSize', 16)
c_tri = polyfit (log10(x), log10(y_tri), 1)
v_tri = polyval(c_tri, log10(x));
plot (log10(x), v_tri)  % best fit line
% graph sparse tridiagonal matrix
y_sparse = meantimes_sparse;  % y-axis: Mean time (seconds)
plot (log10(x), log10(y_sparse), 'k.', 'MarkerSize', 16)
c_sparse = polyfit (log10(x), log10(y_sparse), 1)
v_sparse = polyval(c_sparse, log10(x));
plot (log10(x), v_sparse)  % best fit line
legend('full matrix data points', 'full matrix best fit line', 'upper triangle data points', 'upper triangle best fit
line', 'tridiagonal data points', 'tridiagonal best fit line', 'sparse tridiagonal data points', 'sparse tridiagonal best
fit line', 'Location', 'best')
hold off
```

Code for part_b.m:

```
% -----Part B-----
mean_errors = zeros(5,1);
m= 1;
for N_b=[62500 125000 250000 500000 1000000]    % N values for matrix size N x N
%for N_b = (10000)  % use this to find sig digs reliable in solution N = 10^4
   errors = zeros(100,1);
   x_b = ones(N_b, 1);
   for j=1:100 % run 50 trials
       % sparse tridiagonal matrix
       B = randn(N_b, 3);
       A_sparse = spdiags(B, -1:1, N_b, N_b);
       b_sparse = A_sparse*x_b;
       z = A_sparse\b_sparse;
       trial_error = norm(x_b - z, inf);
       errors(j) = trial_error;
   end
   meanerror = mean(errors);    % calculate mean time of full N x N matrix
   mean_errors(m) = meanerror;
   m = m+1;
   disp("mean error for " + N_b + " x " + N_b + " sparse tridiagonal matrix: " + meanerror) % print matrix size and error
end
%mean_errors  % use this to go through each error and find largest error
x = [62500 125000 250000 500000 1000000]; % x-axis: N Value (matrix size)
% graph mean error against N for sparse tridiagonal matrix
y = mean_errors;  % y-axis: Mean error
hold on
plot (log10(x), log10(y), 'r.', 'MarkerSize', 16)
c = polyfit (log10(x), log10(y), 1)
v = polyval(c, log10(x));
plot (log10(x), v)  % best fit line
title("Graph of Log of Mean Error against Log of N for a Sparse Tridiagonal Matrix")
xlabel("Log of N Value (log of matrix size)")
ylabel("Log of Mean Error")
hold off
```