```
    xk_list = np.array(xk_list)
    plot_line(xk_list, subfig_num)

    xstar_x1 = xk_list[-1][0][0]
    xstar_x2 = xk_list[-1][1][0]
    xstar = np.array([round(xstar_x1), xstar_x2]).reshape(2, 1)
    for i in range(xk_list.shape[0]):
        xk_xstar_list.append(norm(xk_list[i] - xstar))
    xk_xstar_list = np.array(xk_xstar_list)
    plot_convergence(xk_xstar_list, subfig_num)
# main begin

x1 = np.arange(-10, 11, 4)
x2 = np.arange(-2, 3, 4)

plt.figure(1, figsize=(10, 5))
plot_contour()
subfig_num = 1

time_list = []
for i in range(6):
    for j in range(2):
        initial = np.zeros(2).reshape(2,1)
        initial[0][0] = x1[i]
        initial[1][0] = x2[j]
        plt.figure(1)
        plt.scatter(initial[0], initial[1], s=40, marker='s',
                    facecolors ='none', edgecolor= color_list[subfig_num-1])

        start = time.clock()
        Global_Newton(initial, subfig_num)
        end = time.clock()

        time_list.append(end - start)
        subfig_num = subfig_num + 1

plt.figure(1)
plt.xlabel('x1')
plt.ylabel('x2')
plt.xlim(-17, 12)
plt.ylim(-2.5, 2.5)
plt.savefig('A4_1_a', dpi=700)

plt.figure(2)
plt.savefig('A4_1_a_convergence', dpi=700)

print()
print('Number of iterations from different initial points:', Number_iterations)
print('Average number of iterations from different initial points:',
      sum(Number_iterations)/len(Number_iterations))

print('Calculating time from different initial points:', time_list)
print('Average calculating time from different initial points:',
      sum(time_list)/len(time_list))
```

Subproblem (b)

By comparing the performance of the Newton method and gradient method with backtracking, we can derive Table 2. From the table, we can see that the Newton method is much better than the gradient method with backtracking. The Newton method always utilize the Newton direction. Newton method and gradient method both do not always use full step sizes $\alpha_k = 1$. The figures which contains the paths is showing as

Assignment A4.1 continued on next page...