```
include("hw6_helpers.jl");

function conjugate_gradient(h::Real; tol::Real=1e-7)

  # initialize data
  grid = Grid(h);
  enforce_bcs!(grid);
  us = grid.us;
  rs = zeros(grid.nx, grid.ny);
  qs = zeros(grid.nx, grid.ny);
  const start_time = time();

  for j = 2:grid.ny-1, i = 2:grid.nx-1
    x, y = grid.xs[i, j], grid.ys[i, j];
    rs[i, j] = (-h*h * force(x, y) + us[i+1, j] + us[i-1, j] + us[i, j+1] +
                us[i, j-1] - 4.0 * us[i, j]);
  end
  r2_prev = dot(rs, rs);
  ps = copy(rs);
  for j = 2:grid.ny-1, i = 2:grid.nx-1
    qs[i, j] = 4*rs[i, j] - rs[i+1, j] - rs[i-1, j] - rs[i, j+1] - rs[i, j-1];
  end

  pdotq = dot(ps, qs);
  for iteration = 1:(grid.nx*grid.ny)
    alpha = r2_prev / pdotq;
    for j = 2:grid.ny-1, i = 2:grid.nx-1
      us[i, j] += alpha * ps[i, j];
    end

    if norm(alpha * ps, 2) < tol
      return us, iteration, time() - start_time, true;
    end

    for j = 2:grid.ny-1, i = 2:grid.nx-1
      rs[i, j] -= alpha * qs[i, j];
    end
    r2_new = dot(rs, rs);
    beta = r2_new / r2_prev;
    for j = 2:grid.ny-1, i = 2:grid.nx-1
      ps[i, j] = rs[i, j] + beta * ps[i, j];
      qs[i, j] = (4*rs[i, j] - rs[i+1, j] - rs[i-1, j] - rs[i, j+1] -
                  rs[i, j-1] + beta * qs[i, j]);
    end
    r2_prev = r2_new;
    pdotq = dot(ps, qs);
  end

  return us, grid.nx*grid.ny, time() - start_time, false;
end

for h in [0.1; 0.05; 0.025]
  println("Conjugate gradient, h = ", h);
  approx_soln, iterations, time_elapsed, did_converge = conjugate_gradient(h);
  println("   time elapsed:     ", time_elapsed);
  println("   iterations:       ", iterations);
  println("   sec/iteration:    ", time_elapsed / iterations);
  println("   converged:        ", (did_converge) ? "true" : "false");

  const n = size(approx_soln, 1);
  num = 0.0;
  den = 0.0;
  for (j, y) in zip(1:n, 0.0:h:1.0), (i, x) in zip(1:n, 0.0:h:1.0)
    num += (approx_soln[i, j] - analytical_soln(x, y))^2;
```

```
65        den += (analytical_soln(x, y))^2;
66      end
67
68      println("   rel. L2 error:    ", sqrt(num / den));
69      println();
70    end
```