Project 1
Due on Monday, Feb 20

Consider the following simplified model of opinion formation: each person has an expressed opinion x_i which is a weighted average of an "intrinsic" opinion s_i and the expressed opinions of some set of neighbors. That is, if w_{ij} represents how much weight person i gives to the opinion of person j, then

$$x_i = \frac{s_i + \sum_{j \neq i} w_{ij} x_j}{1 + \sum_{j \neq i} w_{ij}}.$$

Rearranging this expression slightly, we have a matrix equation

$$(L + I)x = s,$$

where L is the so-called (directed) graph Laplacian

$$L_{ij} = \begin{cases} \sum_{k \neq i} w_{ik}, & i = j \\ -w_{ij}, & \text{otherwise.} \end{cases}$$

Under this model, define $\phi(W, s)$ to be the *mean opinion*:

$$\phi(W, s) = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

In this assignment, we will investigate the sensitivity of the mean opinion to small changes in the weights w_{ij} or the intrinsic opinions s_i. That is, we would like a function

```
function [phi,Ws,ss] = phi_sensitivity (W,s)
```

where

- `W` is a sparse matrix of edge weights
- `s` is a column vector of node "intrinsic" opinions
- `phi` is the mean opinion for the given graph
- `Ws` is a sparse matrix of partial derivatives of ϕ with respect to the (nonzero) edge weights

- `ss` is a column vector of partial derivatives with respect to the intrinsic opinions

We want *efficient* solutions to each of these problems. Assume that n is fairly large (say $n = 10^4$), but that the topology of the graph is nice enough that it is practical to factor network matrices using MATLAB's sparse direct Gaussian elimination code. Ideally, you should look in the documentation of `sparse` to see how to compute factorizations of the form $LU = PAQ$, where L and U are sparse lower and upper triangular matrices and P and Q are permutation matrices. You should be able to do one factorization, then only ever solve sparse triangular linear systems (using backslash).

Notes

1. This project should not involve much code if you do it right. My solution takes nine lines of MATLAB (not including the function prototype). However, this code is fairly compact because it uses MATLAB's vector operations — I could have done the same with loops, but it would have been longer, and it would have taken longer to run.

2. You will lose points if your code does way too much work, or if you store something in dense form that I requested be sparse. As a point of reference, my code takes a bit under a quarter second to process the `glinkW` network (available on the class web page) on my PowerBook laptop using MATLAB R2011a.

3. You may want to look at the MATLAB help pages for `lu` (particularly the form `[L,U,P,Q] = lu(A)`), for `spdiags`, for `find`, and for `sparse`. The only other functions my code uses (apart from built-in matrix intrinsics like backslash) are `length` and `ones`.

4. It's easy to make minor goofs, so you should carefully check your results. You can check your sensitivity calculations using a finite difference approximation for the derivative; for example:

```
% i,j are indices of a weight to check;
% h is a step size (e.g. h = 1e-4)

Wp = W; Wp(i,j) = Wp(i,j) + h;
```

```
Wm = W; Wm(i,j) = Wm(i,j) - h;
[phi,Ws,ss] = phi_sensitivity(W,s);
phip        = phi_sensitivity(Wp,s);
phim        = phi_sensitivity(Wm,s);
Ws_ij_est   = (phip-phim)/2/h;
fprintf('Ws(i,j) = %g; check vs fd = %g\n', ...
        Ws(i,j), abs(Ws_ij_est-Ws(i,j))/abs(Ws(i,j)));
```

5. If you find this application interesting, enough that you think you'd like to see more, you might enjoy the paper by Bindel, Oren, and Kleinberg, *How Bad is Forming Your Own Opinion?* (FOCS 2011).