Prof. H. Faßbender, Dr. T. Damm
Zentrum Mathematik, Technische Universität Braunschweig
Sommersemester 2003
07.04.2003

Sparse Eigenvalue Systems

Exercise 0: An appetizer

Problem 0. The largest matrix computation of the world

How does a search engine like Google rank the importance of sites in the world wide web? It surfes randomly through the web and computes the probability for each page to be hit. This random surf – or random walk – is defined as follows (cf. [1] for more details):
If you have reached a certain page, then you have two choices. Either you follow randomly one of the links on this page or you pass to some arbitrary, randomly chosen page of the whole web. The first choice you take with probability 0.85 and the second choice with probability 0.15 (approximately you might throw a dice: If it's a 6, you pass to an arbitrary page, otherwise you follow one of the links). After you have made this choice, each possible page is considered with equal probability.
For instance, assume that the web consists of the 4 sites S_1, S_2, S_3, S_4 and S_1 has an outgoing link to S_2 and S_4. Then, from S_1 we go to S_2 with probability $0.85/2 + 0.15/4$. Clearly, we might represent this web by a 4×4 matrix $A = (a_{ij})$ where $a_{ij} = 1$ if S_j has an outgoing link to S_i, and $a_{ij} = 0$ otherwise. The matrix $P = (p_{ij})$ then contains the probabilities p_{ij} to pass from S_j to S_i in the random walk. For example

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \quad \Rightarrow \quad P = \begin{bmatrix} \frac{0.15}{4} & \frac{0.85}{2} + \frac{0.15}{4} & \frac{0.15}{4} & \frac{0.85}{3} + \frac{0.15}{4} \\ \frac{0.85}{2} + \frac{0.15}{4} & \frac{0.15}{4} & \frac{0.85}{1} + \frac{0.15}{4} & \frac{0.85}{3} + \frac{0.15}{4} \\ \frac{0.15}{4} & \frac{0.15}{4} & \frac{0.15}{4} & \frac{0.85}{3} + \frac{0.15}{4} \\ \frac{0.85}{2} + \frac{0.15}{4} & \frac{0.85}{2} + \frac{0.15}{4} & \frac{0.15}{4} & \frac{0.15}{4} \end{bmatrix}.$$

If we start on our favorite page S_1 then the j-th entry of the vector $p_1 = Pe_1$ is the probability that after one step we reach S_j (here $e_1 = (1, 0, 0, 0)^T$). Analogously the vector $p_k = P^k e_1$ contains the probabilities for each page to be reached in the k-th step.
It is a famous result of Perron and Frobenius (1907) that – independently of the starting vector e_0 – the sequence $p_k = P^k e_0$ converges to a fixed vector $p = \lim p_k$. This vector, in fact, contains the page rankings used by Google.

- Show that p must be an eigenvector of P.
- Compute p for our matrix of the example. Which sites are the most important?
- For the world wide web the catch lies in the fact that the dimension n of P is very large. According to [1] there were about $n = 13$ billion $(13 \cdot 10^9)$ web pages in May 2002. How much memory would it take to store the whole matrix P, if each entry took just 1 byte? How long would it take to multiply P with some vector x, if your computer can perform $2 \cdot 10^9$ scalar multiplications in a second while additions simply are neglected?
- In the mean, each web page has about 7 outgoing links. Show under this assumption that P can be written in the form $P = P_0 + ee^T$, where P_0 is sparse, and $e \in \mathbb{R}^{n \times 1}$. How much memory does it (approximately) take to store P_0 and e? How, do you think, the matrix-vector multiplication Px can be performed cheaply?

[1] Cleve Moler, `http://www.mathworks.com/company/newsletter/clevescorner/oct02_cleve.shtml`

Deadline: To be announced