

# ***Parameterized Matrices and PageRank***

**Paul G. Constantine   David F. Gleich**

Stanford University

*Sandia National Labs, Livermore*

*August 6th, 2009*

# PageRank by Google

$$(\mathbf{I} - \alpha\mathbf{P})\mathbf{x}(\alpha) = (1 - \alpha)\mathbf{v}$$

## Setup

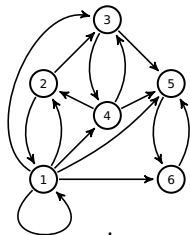
$\mathbf{P}$  : column stochastic

$\mathbf{v}$  : probability distribution

## The Random Surfer Model

1. follow edges with probability  $\alpha$
2. "do something else" according to  $\mathbf{v}$  with probability  $1 - \alpha$

$$\mathbf{x} = (\alpha\mathbf{P} + (1 - \alpha)\mathbf{v}\mathbf{e}^T)\mathbf{x}$$



$$\begin{matrix} \downarrow \\ \left[ \begin{array}{cccccc} 1/6 & 1/2 & 0 & 0 & 0 & 0 \\ 1/6 & 0 & 0 & 1/3 & 0 & 0 \\ 1/6 & 1/2 & 0 & 1/3 & 0 & 0 \\ 1/6 & 0 & 1/2 & 0 & 0 & 0 \\ 1/6 & 0 & 1/2 & 1/3 & 0 & 1 \\ 1/6 & 0 & 0 & 0 & 1 & 0 \end{array} \right] \end{matrix}$$

$\mathbf{P}$

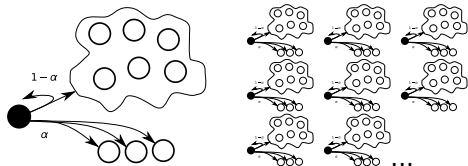
Note Dangling nodes patched to  $\mathbf{v}$  and  $v_i = 1/n$  for most of this talk.

# Random Alpha PageRank

RAPr

$\alpha$  : probability of following a link

$A$  : distribution of  $\alpha$  from people's behavior



$\downarrow$   
 $\mathbf{x}(E[A])$

$\downarrow$   
 $E[\mathbf{x}(A)]$

$\swarrow \quad \searrow$   
 $\mathbf{x}(E[A]) \neq E[\mathbf{x}(A)]$

$$E[\mathbf{x}(A)] =$$

$$\int_0^1 \mathbf{x}(\alpha) \rho(\alpha) d\alpha$$

**“expected PageRank”**

$$\text{Std}[\mathbf{x}(A)] =$$

**“PageRank stability”**

$$\text{Cov}[x_i(A), x_j(A)] =$$

**“covariance”**

*Gleich and Constantine, Workshop on Algorithms on the Web Graph, 2007*

# Just one second ...

$$\int_0^1 \mathbf{x}(\alpha) \rho(\alpha) d\alpha = \int_0^1 (1 - \alpha)(\mathbf{I} - \alpha\mathbf{P})^{-1} \mathbf{v} \rho(\alpha) d\alpha$$

$$\begin{array}{ccc} \alpha = 1 & \longrightarrow & (\mathbf{I} - \mathbf{P})^{-1} \\ \mathbf{P} \text{ stochastic} & & \text{singular?} \end{array}$$

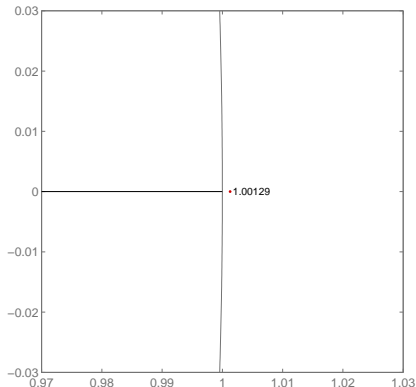
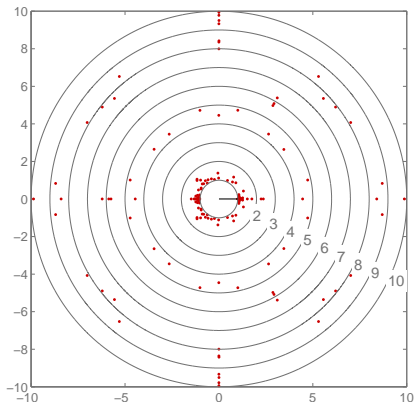
Yes, but ...

$$\lim_{\alpha \rightarrow 1} (1 - \alpha)(\mathbf{I} - \alpha\mathbf{P})^{-1} \mathbf{v} = \mathbf{x}^* \text{ is unique}$$

(Think about  $\mathbf{P} = \mathbf{1}$ , use Jordan Form of  $\mathbf{P}$  to generalize)

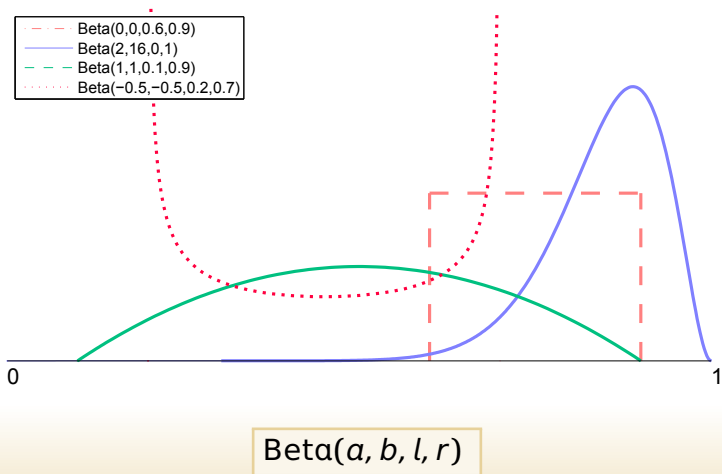
*Serra-Capizzano, Jordan canonical form of the Google matrix, 2005.*

# Singularities

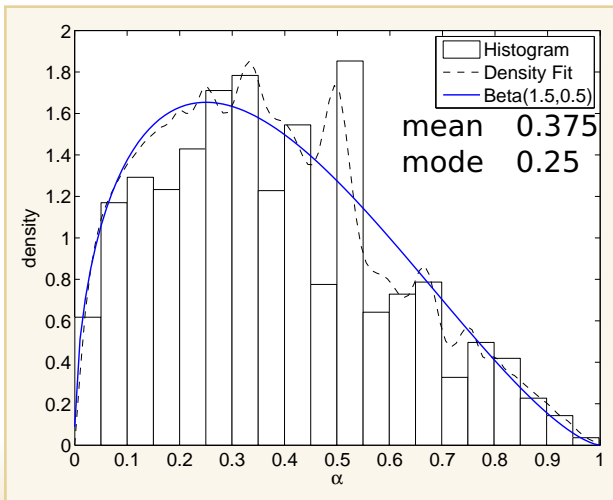


Note 500-node harvard500 graph from Cleve Moler, left plot is  $\log_{10}(9 + |1/\lambda|)e^{\arg(1/\lambda)}$ ; right plot is  $1/\lambda$  for eigenvalues  $\lambda \neq 0, 1$  of  $\mathbf{P}$

# What is A?



# Alpha is



Data provided by Abraham Flaxman and Asela Gunawardana at Microsoft.

# Quadrature implementation

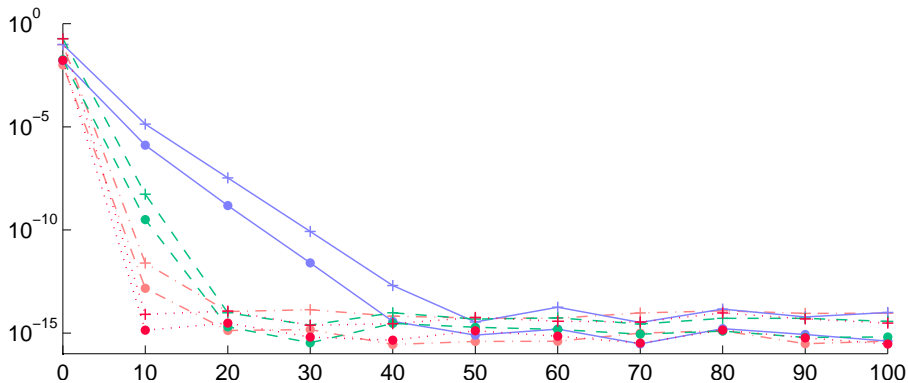
$$\int_l^r \mathbf{x}(\alpha)\rho(\alpha) d\alpha = \sum_{i=1}^N \mathbf{x}(\lambda_i)w_i$$

```
% first run these commands to get the OPQ codes
% urlwrite('http://www.cs.purdue.edu/archives/2002/wxg/codes/gauss.m','gauss.m')
% urlwrite('http://www.cs.purdue.edu/archives/2002/wxg/codes/r_jacobi.m','r_jacobi.m')
% urlwrite('http://www.cs.purdue.edu/archives/2002/wxg/codes/r_jacobi01.m','r_jacobi01.m')
% input P,N,a,b,l,r
tol=1e-9; maxit=1000; n=size(P,1); v=1/n;
ab=r_jacobi01(N,a,b); xw=gauss(N,ab); xw(:,2) = (1./beta(b+1,a+1))*xw(:,2);
xw(:,1) = (r-l).*xw(:,1)+l;          % generate the quadrature rule by scale and shift
ex = zeros(n,1); stdx = zeros(n,1); % initialize running sums
for i=1:N
    % solve the PageRank system
    x = inoutpr(P,xw(i,1),v,min(tol./xw(i,2),1e-2), ... % adjust tol and maxit
        2*ceil(log(min(tol./xw(i,2),1e-2))/log(xw(i,1))));% for mult by xw(i,2)
    ex = ex+xw(i,2).*x; stdx = stdx+xw(i,2).*(x.^2);
end
stdx = sqrt(stdx - ex.^2);          % convert to stdx
```

- ▶ tolerance adjusted to  $w_i$
- ▶ inner-outer solves for PageRank
- ▶ parallel C++ implementation too

*Gleich et al. An inner-outer iteration for PageRank. To appear.*

# Convergence



Convergence to semi-exact solutions on a 335-node graph (*harvard500* strong component). Blue is  $\text{Beta}(2, 16)$ , green is  $\text{Beta}(1, 1, 0.1, 0.9)$ , salmon is uniform  $(0.6, 0.9)$ , red is  $\text{Beta}(-1/2, -1/2, 0.2, 0.7)$ .

# RAPr on Wikipedia

E [x(A)]
United States
C:Living people
France
United Kingdom
Germany
England
Canada
Japan
Poland
Australia

Std [x(A)]
United States
C:Living people
C:Main topic classif.
C:Contents
C:Ctgs. by country
United Kingdom
France
C:Fundamental
England
C:Ctgs. by topic

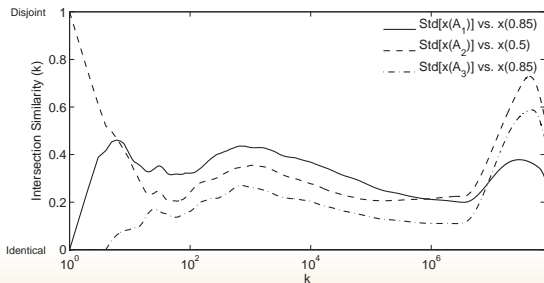
*Note Using a Beta(0.5, 1.5, [0, 1]) distribution.*  
Constantine and Gleich (Stanford)

# Std vs. PageRank

Does it tell us more than just PageRank?

uk2006 — 77M nodes and 2B edges

$$\text{isim}(k) = \frac{1}{k} \sum_{i=1}^k \frac{1}{2i} |\text{Diff}[Y(1:i), Z(1:i)]|$$



Kendall's  $\tau$

$$\tau(\mathbf{x}(E_1), S_1) = +0.3$$

$$\tau(\mathbf{x}(E_2), S_2) = -0.5$$

$$\tau(\mathbf{x}(0.85), S_3) = -0.2$$

$$A_1 \sim \text{Beta}(2, 16, [0, 1]) \quad A_2 \sim \text{Beta}(1, 1, [0, 1])$$

$$A_3 \sim \text{Beta}(1.5, 0.5, [0, 1])$$

# Webspam application

Hosts of uk-2006 are labeled as **spam**, **not-spam**, **other**

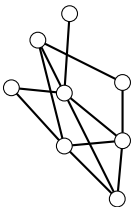
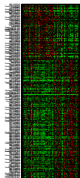
	<b>P</b>	<b>R</b>	<b>f</b>	<b>FP</b>	<b>FN</b>
Baseline	0.694	0.558	0.618	0.034	0.442
Beta(1.5,0.5)	0.692	0.557	0.617	0.034	0.443
Beta(0.5,1.5)	0.695	0.561	0.621	0.034	0.439
Beta(1,1)	0.698	0.562	0.622	0.033	0.438
Beta(2,16)	0.699	0.562	0.623	0.033	0.438

*Note Bagged (10) J48 decision tree classifier in Weka, mean of 50 repetitions from 10-fold cross-validation of 4948 non-spam and 674 spam hosts (5622 total).*

*Becchetti et al. Link analysis for Web spam detection, 2008.*

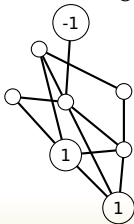
# Similar problems

## GeneRank



Use  $(\mathbf{I} - \alpha \mathbf{G} \mathbf{D}^{-1}) \mathbf{x} = \mathbf{w}$  to find “nearby” important genes.

## Labeled Graph Interpolation



Use  $(\mathbf{I} - \alpha \mathbf{L}) \mathbf{f} = \mathbf{y}$  for graph Laplacian  $\mathbf{L}$  to “interpolation”  $\mathbf{y}$

# Summary and Future Work

- ▶ broad overview of parameterized matrix equations
- ▶ connections between polynomial approximation techniques
- ▶ error estimates and intuition
- ▶ PageRank example
- ▶ multivariate approximation
- ▶ exploring parameterized Lanczos and parameterized eigenvalues
- ▶ techniques for large scale problems
- ▶ singularity detection
- ▶ singularity matching/rational interpolation
- ▶ non-linear models
- ▶ SOFTWARE
- ▶ MORE APPLICATIONS

