Spectral Methods to Compute Eigenvalue Histograms for Graph Laplacians

Robert Elsässer
University of Salzburg

Joint work with
Sebastian Arming, Gregor Bankhamer, Christine Gfrerer

Project funded within the Horizon 2020 Framework Programme of the EU

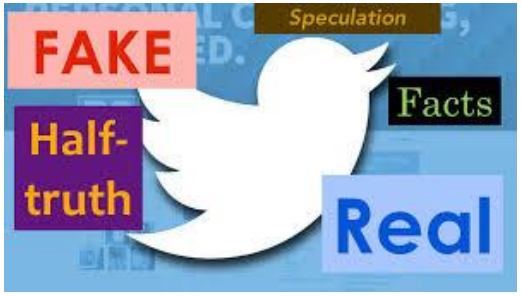
Outline

- Motivation and basic definitions
- Evaluating graph models for real-world networks
- Eigenvalue histograms
  - Exact method
  - Approximative method
- Conclusions

Outline

- **Motivation and basic definitions**
- Evaluating graph models for real-world networks
- Eigenvalue histograms
  - Exact method
  - Approximative method
- Conclusions

Motivation

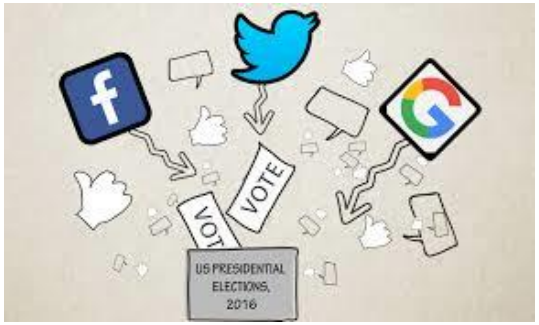


The Telegraph:

Russian trolls sent thousands of pro-Leave messages on day of Brexit referendum, Twitter data reveals

Mirror:

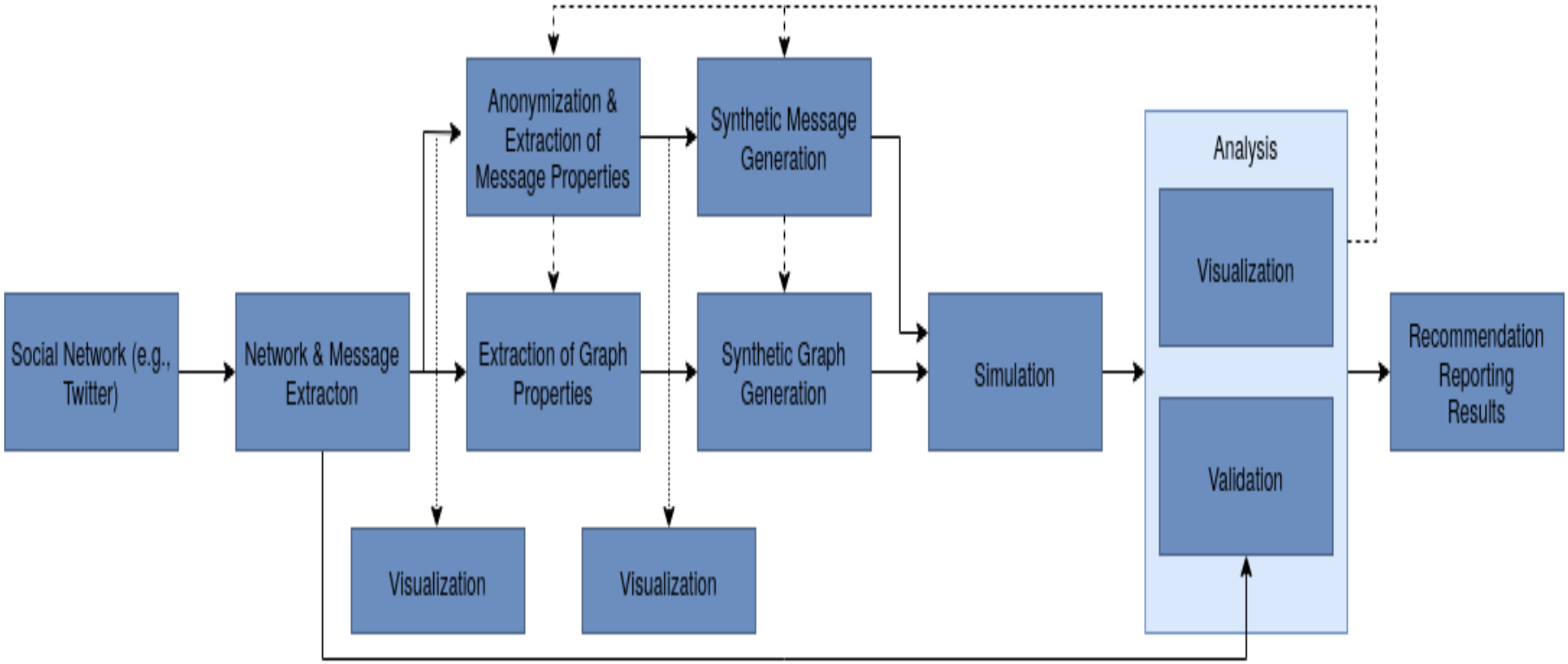
Fake Mirror websites promoted on social media are being used as bait by Bitcoin rackets



The Verge:

How social media platforms influenced the 2016 election

Workflow



Acquisition of data

- A number of social networks from SNAP
- Collected several user defined networks from Twitter
- Analyzed the structural and algorithmic properties of the underlying graphs
  - Friendship networks
  - Follower/followee relationship

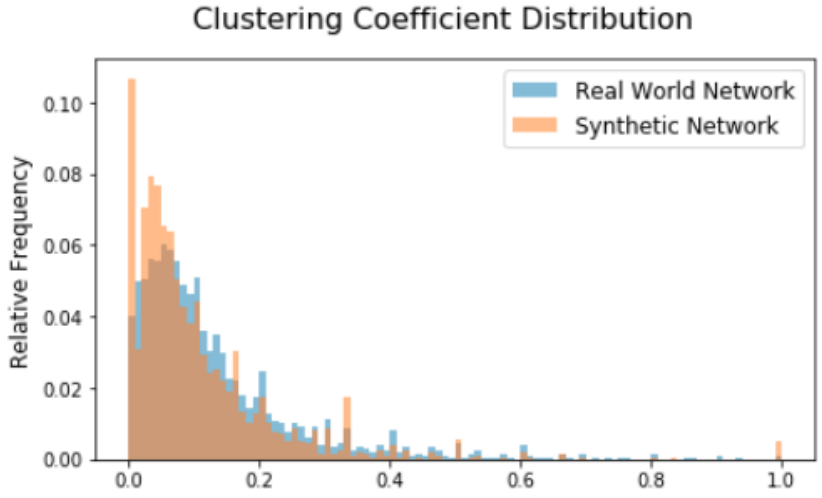
Outline

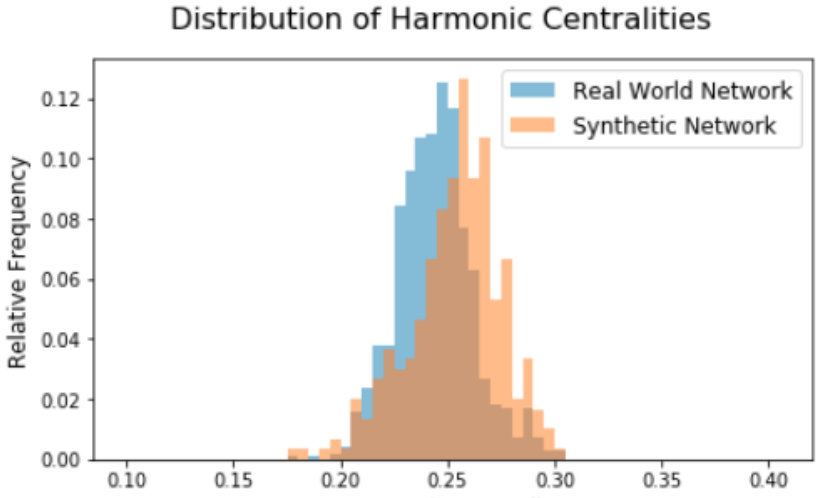
- Motivation and basic definitions
- **Evaluating graph models for real-world networks**
- Eigenvalue histograms
  - Exact method
  - Approximative method
- Conclusions

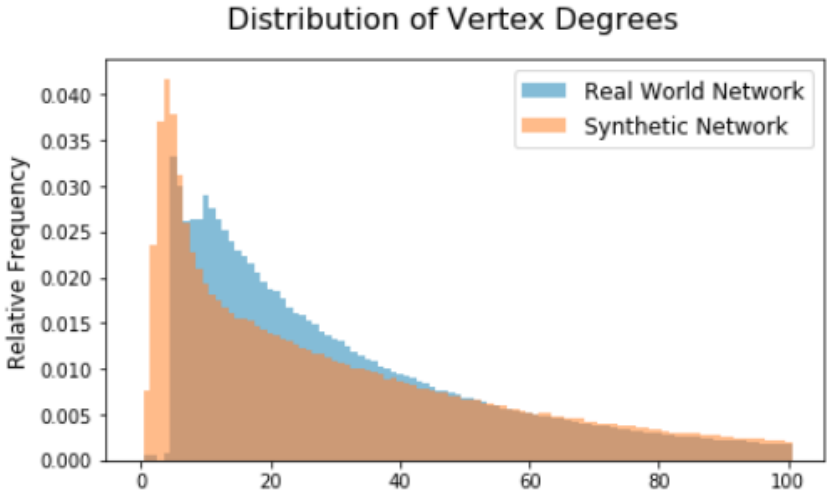
Real vs. synthetic graphs

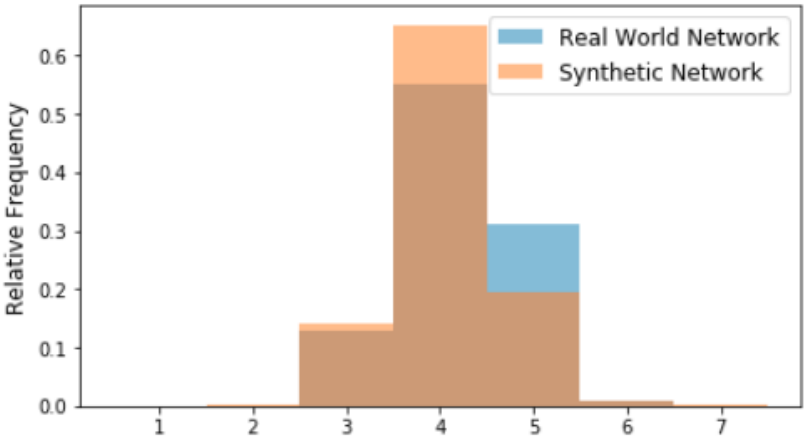
- The size of the graphs ranges between
$$5 \cdot 10^4 \ - 3 \cdot 10^6$$
- Mostly sparse graphs (highest degree $O(\sqrt{n})$)
- Properties of produced graphs
  - Clustering coefficients
  - Distribution of node degrees
  - Distribution of distances for randomly sampled nodes
  - Eigenvalue distribution

Validation with structural properties









Spectral properties of graphs

- Laplacian of the (undirected, unweighted) graph

$$L = D - A$$

A – adjacency matrix, D – diagonal matrix with degree d_i of the i^{th} node as the i^{th} entry

- The normalized Laplacian is

$$\mathcal{L} = D^{-1/2} L D^{-1/2}$$

$$\text{i.e., } l_{ij} = \begin{cases} -1/\sqrt{d_i d_j} & \text{if } (i,j) \in E \\ 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Spectral properties of graphs

- The eigenvalues of $\mathcal{L}$ lie between 0 and 2.
- We are interested in the eigenvalue histogram,

  e.g., how many eigenvalues lie in each interval of length 0.02.
- Compare the eigenvalue histogram of the synthetic graph with the one of the real world network.
- Parts of eigenvalue distributions have been used to compare random graph models to real world networks in the past.

Kolda et al., SIAM J. Scientific Computing, 2014

Outline

- Motivation and basic definitions
- Evaluating graph models for real-world networks
- Eigenvalue histograms
  - **Exact method**
  - Approximative method
- Conclusions

Eigenvalue count in $[a, b]$

- Let A be an $n \times n$ Hermetian matrix
- Let $n_{[a,b]} = n - n_+(A - bI) - n_-(A - bI)$
  be the number of eigenvalues in $[a, b]$
- Define $M_x = A - xI$
- We are interested in the number of positive eigenvalues of M_b and the number of negative eigenvalues of M_a.

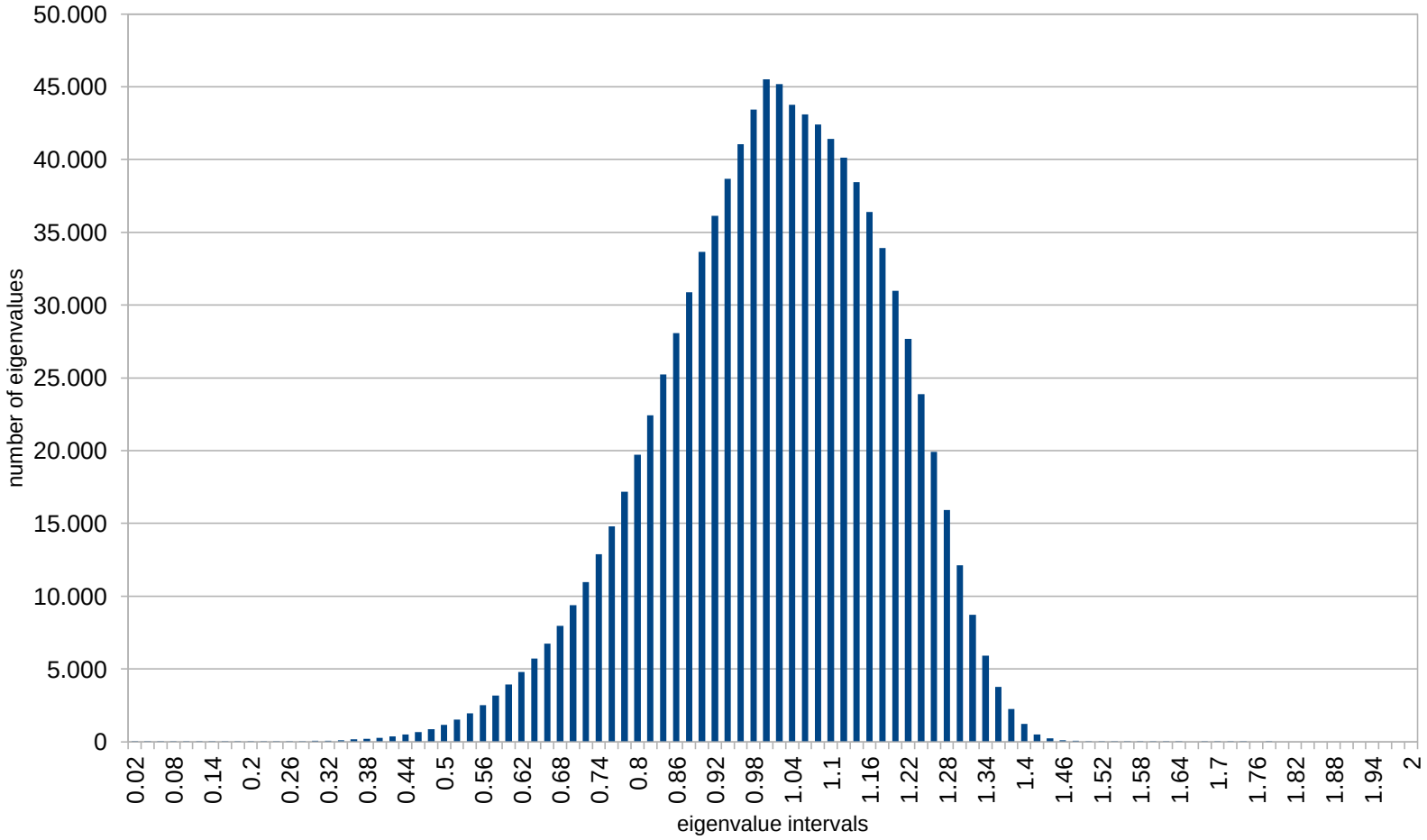
Eigenvalue count in $[a, b]$

- Factorize $M_x = LDL^T$
  - L – lower triangular matrix, D – diagonal matrix
- Compute the number of positive entries in D
- According to the inertia theorem, this number corresponds to $n_+(M_x)$
- For dense matrices $\theta(n^3)$ operations per factorization are needed
- Only suitable for sparse graphs

Software requirements

- Implemented in C++ with SLEPc and PETSc libraries
- MUMPS is used for the parallel LDL decomposition
- As preprocessing – before factorization – PARMETIS is applied
- MPI is used for message-passing communication
- The graphs are given in Metis format – size ranging from 250 MB to 4 GB

Histogram of the normalized Laplacian of the Pokec social network
input size n = 990 908, computed on Hazelhen@HLRS

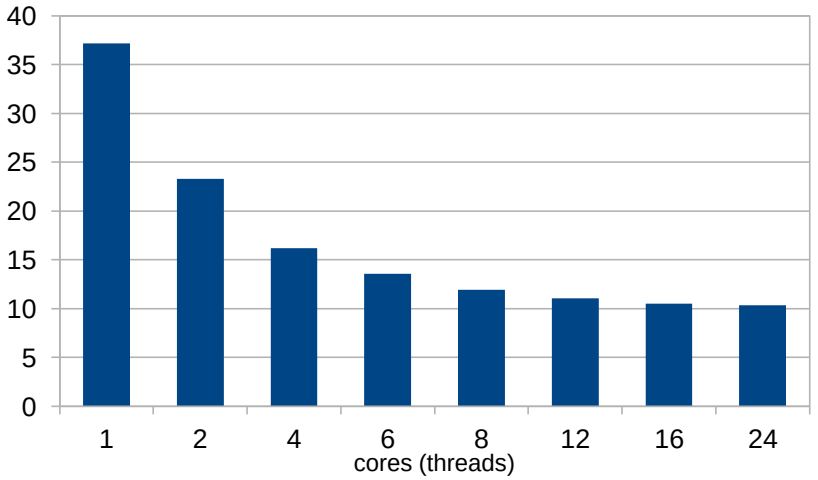


Resource requirements

Problem size	Compute nodes	Average memory per processor (GB)	Total memory (GB)	Worst case memory per processor (GB)
200,000	15	11.8	177	19.4
400,000	40	12.5	500	24.4
700,000	80	17.7	1,416	45.3
850,000	100	18.4	1,840	51.8
925,000	105	20.0	2,100	69.2
970,000	110	20.0	2,200	71.2
990,000	115	20.6	2,369	66.3

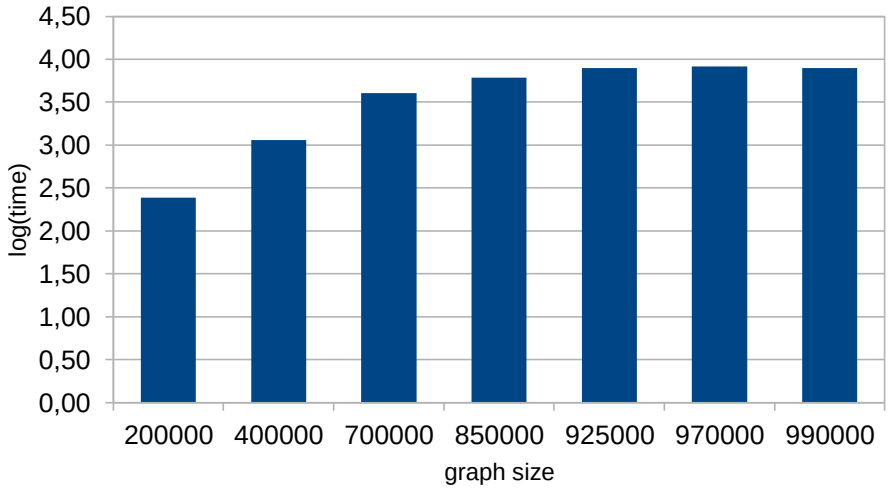
Benchmark results

Graph size	cores	threads	time
50.000	1	1	37.18
50.000	2	2	23.31
50.000	4	4	16.18
50.000	8	8	11.91
50.000	12	12	11.08
50.000	16	16	10.53
50.000	24	24	10.34



Benchmark results

Graph size	cores	time	log(time)
200.000	360	243	2.39
400.000	970	1,149	3.06
700.000	1,920	4,035	3.61
925.000	2,520	7,883	3.90
970.000	2,640	8,296	3.92
990.000	2,760	7,909	3.90



Outline

- Motivation and basic definitions
- Evaluating graph models for real-world networks
- Eigenvalue histograms
  - Exact method
  - **Approximative method**
- Conclusions

Di Napoli et al., Numerical Linear Algebra Applications, 2016

Approximation method

- Approximate the trace of $P_{[a,b]} = \sum_{\lambda_i \epsilon [a,b]} u_i \, u_i^T$
  - λ_i: i^{th} smallest eigenvalue of $\mathcal{L}$ with eigenvector u_i
  - $P_{[a,b]}$ - spectral projector associated with $[a, b]$
- Let $\mu_{[a,b]} = \text{Trace}(P_{[a,b]})$ the number of eigenvalues in $[a, b]$
- $P_{[a,b]} = h(\mathcal{L})$, where

$$h(t) = \begin{cases} 1 & \text{if } t \epsilon [a, b] \\ 0 & \text{otherwise} \end{cases}$$

Approximation method

- Expand the function h in a sum of Chebyshev polynomials.
- Use Hutchinson's unbiased estimator to approximate the trace of $P_{[a,b]}$

$$\mathrm{Trace}\left(P_{[a,b]}\right) \approx \frac{n}{n_v} \sum_{k=1}^{n_v} v_k^T P_{[a,b]} v_k$$

  - n_v is the number of samples
  - v_k are vectors with entries sampled u.a.r. from $[0,1]$ and normalized to unit l_2-norm.

Approximation method

- Polynomial expression filtering

$$\mu_{[a,b]} \approx \frac{n}{n_v} \sum_{k=1}^{n_v} v_k^T \psi(P_{[a,b]}) v_k$$

  - ψ approximates h with the sum of Chebyshev polynomials
- That is,

$$h(t) \approx \psi_p(t) = \sum_{j=0}^{p} \gamma_j T_j(t)$$

  - T_j are the j–degree Chebyshev polynomials
  - γ_j are expansion coefficients (defined below)

Approximation method

- The coefficients

$$\gamma_j = \begin{cases} \frac{1}{\pi}(\cos^{-1} a - \cos^{-1} b) & \text{if } j = 0 \\ \frac{2}{\pi} \cdot \frac{\sin(j \cos^{-1} a) - \sin(j \cos^{-1} b)}{j} & \text{if } j > 0 \end{cases}$$

- This results in

$$\mu_{[a,b]} \approx \frac{n}{n_v} \sum_{k=1}^{n_v} \left(\sum_{j=1}^{p} \gamma_j \, v_k^T \, T_j(P_{[a,b]}) v_k \right)$$

Approximation method

- Take care of oscillations near the boundaries and transform the solution to the right interval.
- No LDL factorization needed, “just” matrix-vector multiplications.
- The accuracy is governed by the degree of the polynomial p and the number of samples n_v
- The computations can be parallelized (each interval and each sample can run concurrently)

Approximation method

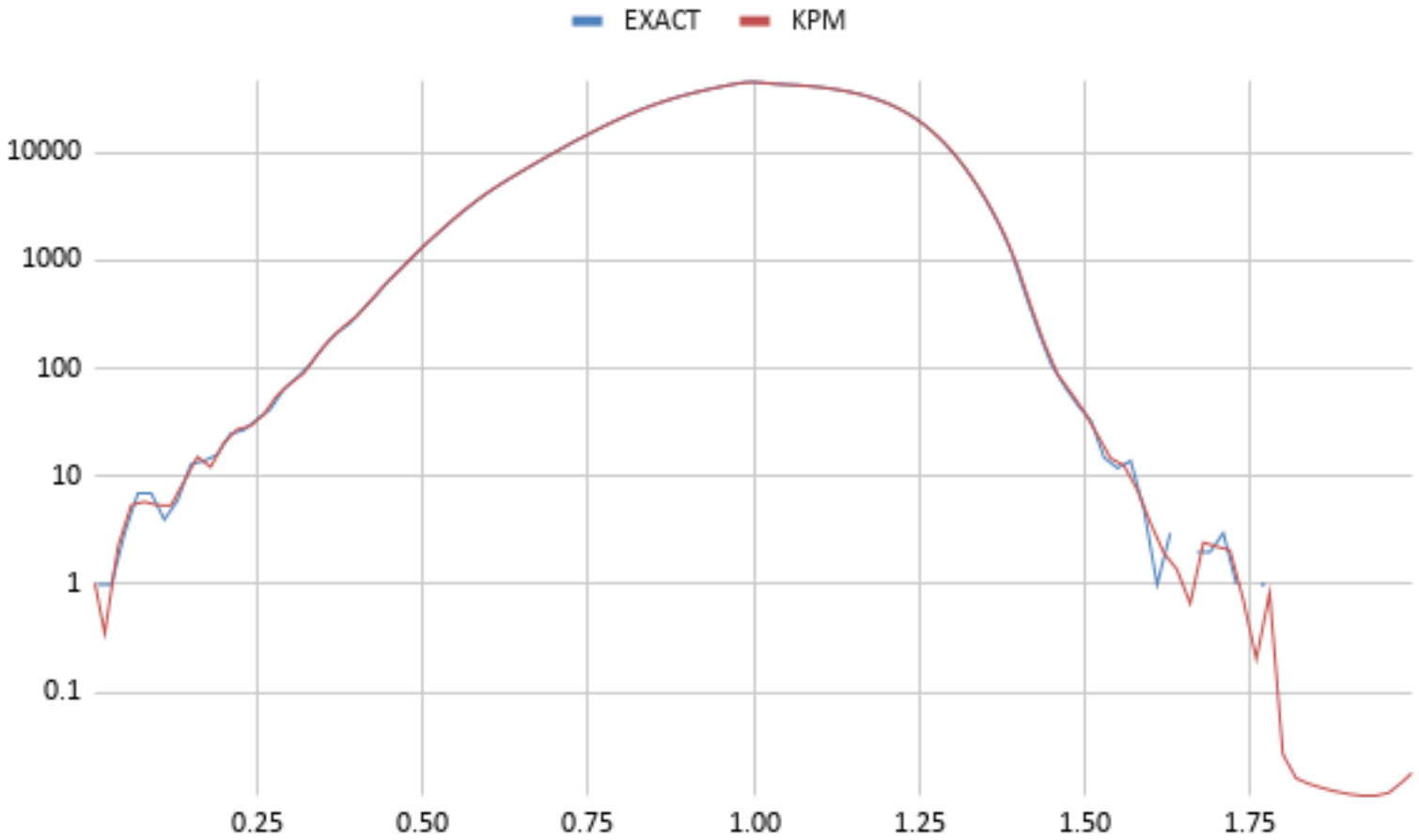
- For our computations we use

$$n_v \in [200,300] \text{ and } p \in [150,250]$$

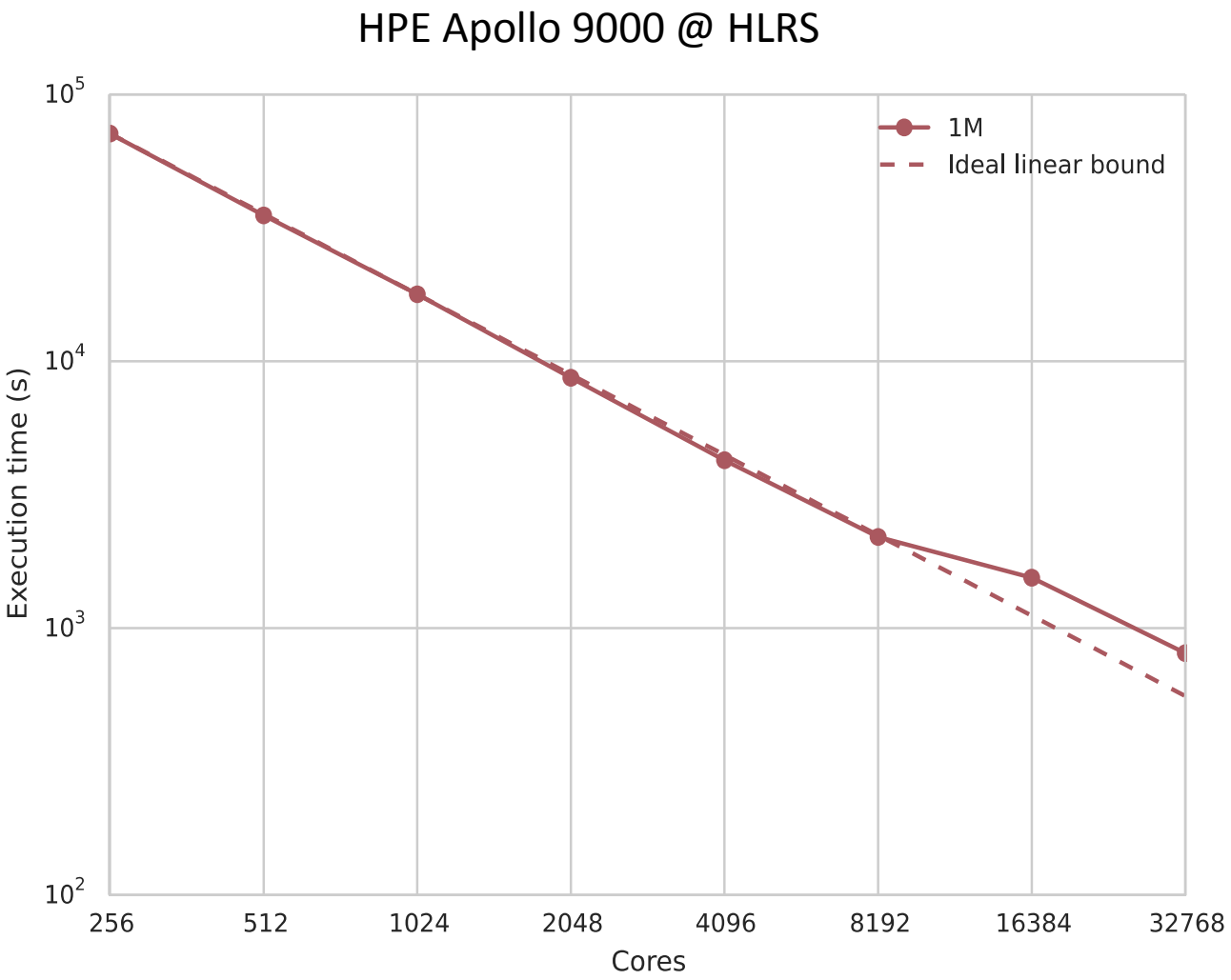
- Computations are far less resource intensive than the exact method
- One thread is used for each interval and every computation of $\sum_{j=1}^{p} \gamma_j v_k^T T_j(P_{[a,b]}) v_k$
- Running time $O(p\ n_v \cdot n^2)$

Synthetic graphs – Validation

Eigenvalue histogram – appox. 1 million nodes



Synthetic graphs – Validation



Synthetic graphs

- Properties of produced graphs
  - Clustering coefficients
  - Distribution of node degrees
  - Distribution of distances for randomly sampled nodes
  - Eigenvalue distribution
- Successive relaxation of input parameters
  - Construct graphs for a desired input size with properties obtained from real-world social networks
  - Edge assignments provide very good results, node assignments to clusters/communities are in progress

Take aways

- Eigenvalue histograms well suited to evaluate synthetic graphs for real-world networks
- Exact method extremely resource consuming (memory and time)
- Approximation method saves resources by orders of magnitude and with the right choice of parameters it provides very good results

THANK YOU !

QUESTIONS ?