Computing Eigenvalues of Random Matrices

Plamen Koev

Department of Mathematics, San Jose State University

Joint with: Cy Chan, Jim Demmel, Vesselin Drensky, Alan Edelman, Iain Johnstone, Raymond Kan

March 14, 2012

Why Eigenvalues of Random Matrices

- **Conventional model:**

$$X \in \mathbb{C}^{m \times n}, \quad X \sim \mathcal{N}_m(0, \Sigma), \quad n\text{-variate Gaussian,}$$

  i.e., x_{ij}—normal random variables and $E(X^*X) = \Sigma$.

- **Question: existence and nature of interdependence between columns of X**
- **I.e., $\Sigma =?\ \Sigma = I?\ \Sigma = \Sigma_0?$ ...**

- **$A \equiv X^*X$ is called $n \times n$ Wishart with m DOF and covariance Σ**

- **Cast as tests on $\lambda_{\max}(A)$, a "test statistic"**

- **5%, 1% benchmarks**

- **Thus (distributions of) eigenvalues of Wishart critical**

Wishart Matrix and its eigenvalues (Ref: Muirhead)

- Goal: Distributions of (generalized) eigenvalues and functions thereof
- The density of a Wishart $A = X^T X$, $X \sim \mathcal{N}_n(0, \Sigma)$:

$$|\Sigma|^{-\frac{n}{2}} |A|^{\frac{n-m+1}{2}-1} e^{-\mathrm{tr}(\frac{1}{2}\Sigma^{-1}A)}$$

- Distribution of $\lambda_{\max}$ of Wishart

$$\begin{aligned} P(\lambda_{\max} < x) &= P(A < xI) \\ &= \int_{0<A<xI} C \cdot |\Sigma|^{-\frac{n}{2}} |A|^{\frac{n-m+1}{2}-1} e^{-\mathrm{tr}(\Sigma^{-1}A)} dA \\ &= \dots \qquad (A \to Q^* \Lambda Q, \text{ Selberg integral}) \\ &= C \cdot \det\left(\tfrac{x}{2}\Sigma^{-1}\right)^{\frac{n}{2}} \cdot {}_1F_1\left(\tfrac{n}{2}; \tfrac{n+m-1}{2} + 1; -\tfrac{x}{2}\Sigma^{-1}\right) \end{aligned}$$

- ${}_1F_1$ is the hypergeometric function of a matrix argument

$${}_1F_1(a, b, X) = \sum_{k=0}^{\infty} \sum_{\kappa \vdash k} \frac{1}{k!} \cdot \frac{(a)_\kappa}{(b)_\kappa} \cdot J_\kappa(X)$$

The hypergeometric function of a matrix argument

- ${}_1F_1$ is the hypergeometric function of a matrix argument

$$ {}_1F_1(a, b, X) = \sum_{k=0}^{\infty} \sum_{\kappa \vdash k} \frac{1}{k!} \cdot \frac{(a)_\kappa}{(b)_\kappa} \cdot J_\kappa(X) $$

- Indexed by partitions κ:

  (1), (2), (1,1), (3), (2,1), (1,1,1), (4), (3,1), (2,2), (2,1,1), (1,1,1,1)

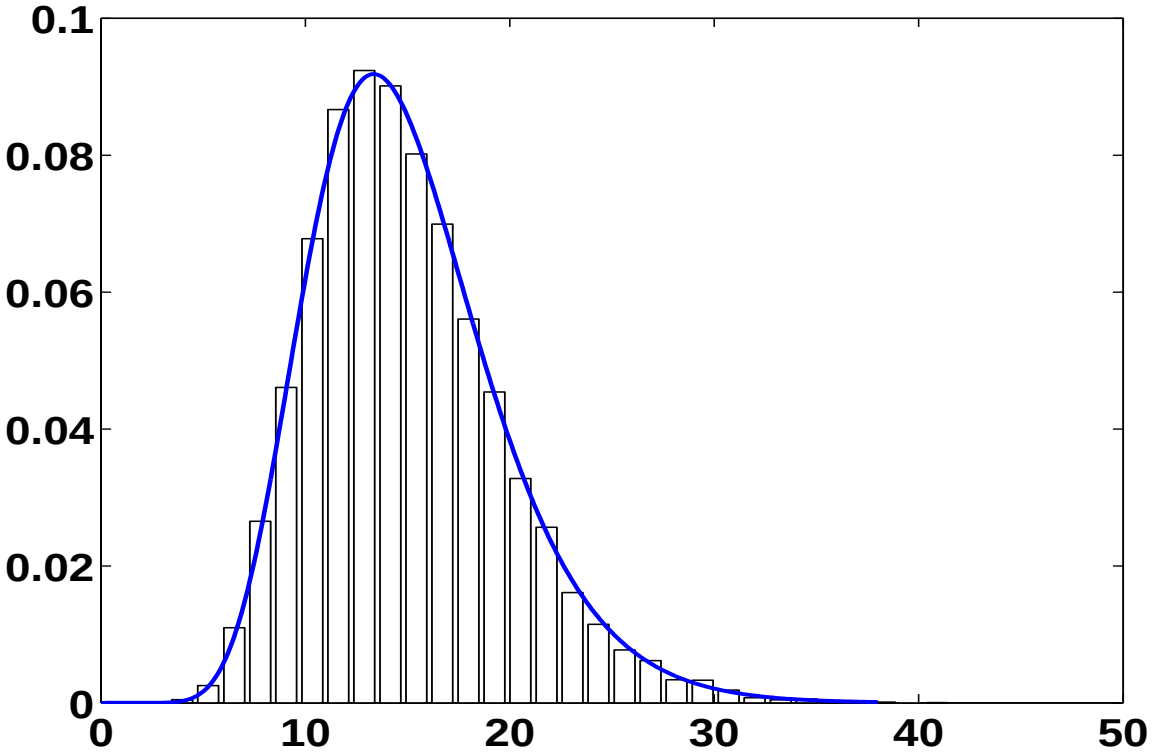
- Generalized Pochhammer symbol (= multivariate raising factorial)

$$ (a)_\kappa \equiv \prod_{i=1}^{n} \prod_{j=1}^{\kappa_i} \left(a - \frac{i-1}{2} + j - 1 \right) = \prod_{i=1}^{n} \left(a - \frac{i-1}{2} \right)_{\kappa_i} $$

- $J_\kappa(X)$ is the Jack function:
  - Real case: $\beta = 1$, zonal polynomials
  - Complex case $\beta = 2$, Schur functions
  - Works for any $\beta > 0$

- Computing ${}_1F_1$ extremely hard (too many terms, each term too complicated)

Example: $\lambda_{\max}$ of 4×4 Wishart with 7 DOF, $\Sigma = I$



Exact vs Empirical with 20,000 replications

Hypergeometric Fn of Matrix Argument, Previous Best Algorithm

Electronic Transactions on Numerical Analysis.
Volume 11, pp. 121-130, 2000.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

APPROXIMATION OF HYPERGEOMETRIC FUNCTIONS WITH MATRICIAL ARGUMENT THROUGH THEIR DEVELOPMENT IN SERIES OF ZONAL POLYNOMIALS*

R. GUTIÉRREZ†, J. RODRIGUEZ‡, AND A. J. SÁEZ§

Hypergeometric Fn of Matrix Argument, Previous Best Algorithm

Electronic Transactions on Numerical Analysis.
Volume 11, pp. 121-130, 2000.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

APPROXIMATION OF HYPERGEOMETRIC FUNCTIONS WITH MATRICIAL ARGUMENT THROUGH THEIR DEVELOPMENT IN SERIES OF ZONAL POLYNOMIALS*

R. GUTIÉRREZ†, J. RODRIGUEZ‡, AND A. J. SÁEZ§

more time would be needed. (We spent about 8 days to obtain the 627 zonal polynomials of degree 20 with a 350 MHz Pentium II processor.)

Hypergeometric Fn of Matrix Argument, Previous Best Algorithm

Electronic Transactions on Numerical Analysis.
Volume 11, pp. 121-130, 2000.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

APPROXIMATION OF HYPERGEOMETRIC FUNCTIONS WITH MATRICIAL ARGUMENT THROUGH THEIR DEVELOPMENT IN SERIES OF ZONAL POLYNOMIALS*

R. GUTIÉRREZ†, J. RODRIGUEZ‡, AND A. J. SÁEZ§

more time would be needed. (We spent about 8 days to obtain the 627 zonal polynomials of degree 20 with a 350 MHz Pentium II processor.)

- **New result (2.33 GHz Pentium):**

```
>> tic; mhg(20,2,[],[],[1:20]/210), toc
ans =
   2.71828182845905
elapsed_time =
   0.03100000000000
```

How to compute ${}_1F_1$?

$$ {}_1F_1(a, b, X) = \sum_{k=0}^{\infty} \sum_{\kappa \vdash k} \frac{1}{k!} \cdot \frac{(a)_\kappa}{(b)_\kappa} \cdot J_\kappa(X) $$

- We truncate to $k \leq M$ for some M and compute instead

$$ {}_1F_1(a, b, X) = \sum_{k=0}^{M} \sum_{\kappa \vdash k} \frac{1}{k!} \cdot \frac{(a)_\kappa}{(b)_\kappa} \cdot J_\kappa(X) $$

- Will demonstrate the complex case, $\beta = 2$, Schur functions s_κ
- Even a single s_κ is exponential if computed naively

Cost of Computing Schur Polynomials Naively

Degree	Partition κ	s_κ	Number of terms
1	(1)	$x_1 + \cdots + x_n$	$\mathcal{O}(n)$
2	(2)	$\sum_{i \le j} x_i x_j$	$\mathcal{O}(n^2)$
2	$(1,1)$	$\sum_{i < j} x_i x_j$	$\mathcal{O}(n^2)$
3	$(1,1,1)$	$\sum_{i < j < k} x_i x_j x_k$	$\mathcal{O}(n^3)$
$\lvert\kappa\rvert$	κ	$\sum x_1^{\kappa_1} \cdots x_n^{\kappa_n}$	$\mathcal{O}\left(n^{\lvert\kappa\rvert}\right)$

- **New result: $\mathcal{O}(n)$ each, as long as one wants them all (and we do!)**

Computing the Schur Polynomial "Cleverly"

- s_κ of higher degree "contain" s_λ of lower degree. Redundancy.
- E.g.:

$$\begin{aligned} s_{(1,1)}(x_1, \ldots, x_n) &= \sum_{i<j} x_i x_j \\ &= x_1 x_2 + (x_1 + x_2)x_3 + \cdots + (x_1 + \cdots + x_{n-1})x_n \end{aligned}$$

- In general:

$$s_\kappa(x_1, x_2, \ldots, x_n) = \sum_{\lambda<\kappa} s_\lambda(x_1, x_2, \ldots, x_{n-1}) \cdot x_n^{|\kappa|-|\lambda|}$$

- Connection with representation theory:
  - s_κ are the irreducible characters of $\mathbf{GL}_n(\mathbb{C})$
  - the characters of $\mathbf{GL}_{n-1}(\mathbb{C})$ induce those of $\mathbf{GL}_n(\mathbb{C})$
- Result long known (Macdonald), but missed for 40 years

Computing the Schur Polynomial "Cleverly"

- **Example:** $s_{(1,1)}(x_1, \ldots, x_n)$

$$= \sum_{i<j} x_i x_j \qquad (\ \sim n^2 \ \text{operations}\ )$$

$$= \underbrace{x_1}_{s_1} x_2 + (\underbrace{x_1+x_2}_{s_2})x_3 + (\underbrace{x_1+x_2+x_3}_{s_3})x_4 + \cdots + (\underbrace{x_1+\cdots+x_{n-1}}_{s_{n-1}})x_n$$

- **New cost:** $3n-2$ **instead of** n^2
- **Generalizes to all** κ**:**
  Cost of $s_\kappa(x_1, \ldots, x_n)$ **goes down from** $\mathcal{O}(n^{|\kappa|})$ **to** $\mathcal{O}(N_\kappa n)$
- **Enumeration of the** κ **in memory – lexicographic ordering**

- **For** $\beta = 2$ **it gets better: We can get rid of** $N_\kappa \equiv \{\#\lambda | \lambda < \kappa\}$

Analogy with the FFT

- **Idea: (DFT)$_{ij}$—characters of $\mathbb{Z}/n\mathbb{Z}$ $\longleftrightarrow$ s_λ—characters of GL$_n(\mathbb{C})$**
- **Write our main identity**

$$s_\kappa(x_1, x_2, \ldots, x_n) = \sum_{\lambda<\kappa} s_\lambda(x_1, x_2, \ldots, x_{n-1}) \cdot x_n^{|\kappa|-|\lambda|}$$

in matrix form: $\mathcal{C}_n = \mathcal{C}_{n-1} \cdot Y_n(x_n)$, where

$$Y_2(x) = \left[\begin{array}{cccc|cccc|cccc|cccc} 1 & x & x^2 & x^3 & x & x^2 & x^3 & x^4 & x^2 & x^3 & x^4 & x^5 & x^3 & x^4 & x^5 & x^6 \\ & 1 & x & x^2 & & x & x^2 & x^3 & & x & x^2 & x^3 & & & & \\ & & 1 & x & & & x & x^2 & & & & & & & & \\ & & & 1 & & & & & & & & & & & & \end{array}\right]$$

$$\begin{bmatrix} 1 & x & x^2 & x^3 \\ & 1 & x & x^2 \\ & & 1 & x \\ & & & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -x & & \\ & 1 & -x & \\ & & 1 & -x \\ & & & 1 \end{bmatrix}$$

- **Matrix-vector multiplication by Y_n costs $\mathcal{O}(n)$ per s_λ instead of $\mathcal{O}(nN_\kappa)$**

Our New Fast Algorithm

- Let A be the lower shift matrix $a_{i+1,i} = 1$; $B = A^T$
- Structure of Y_n :

$$\begin{aligned}
U_n(x_n) &= I_{(N+1)^{n-1}} + x_n(A \otimes B_{n-1}) + \cdots + x_n^N(A^N \otimes B_{n-1}^N) \\
&= \left(I_{(N+1)^{n-1}} - x_n(A \otimes B_{n-1})\right)^{-1}, \\
C_n(x_n) &= U_n(x_n)K_{n-1}(x_n), \\
K_n(x_n) &= I_{N+1} \otimes C_n(x_n), \\
B_n &= B_{n-1} \otimes I_{N+1} = B \otimes I_{(N+1)^{n-2}}, \\
Q_n(x_n) &= \left(I_{(N+1)^{n-1}} \,|\, x_nB_n \,|\, \ldots \,|\, x_n^NB_n^N\right) \\
Y_n &= Q_n(x_n)K_n(x_n)
\end{aligned}$$

- New algorithm:

```
for i=n:-1:1
   for all λ such that |λ| ≤ M in reverse lexicographic order
      s_λ = s_λ + s_{λ^(i)} x_n
```

(where $\lambda^{(i)} \equiv (\lambda_1, \ldots, \lambda_i - 1, \ldots, \lambda_n)$)

- Final cost: $\mathcal{O}(n)$ per each s_λ, optimal

$\beta \neq 2$? Open problem

- **Generalize the FFT idea to real, $\beta = 1$, $\alpha = 2$ case**
- **How does one multiply quickly by the matrix:**

$$A = \begin{bmatrix} 1 & 1 & \frac{\alpha+1}{2} & \frac{(\alpha+1)(2\alpha+1)}{6} \\ & 1 & 1 & \frac{\alpha+1}{2} \\ & & 1 & 1 \\ & & & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -2 & & \\ & 1 & -2 & \\ & & 1 & -2 \\ & & & 1 \end{bmatrix}^{-1/2}$$

- **Resources:** `www-math.mit.edu/~plamen`