Computing Multivariate Statistics through the Hypergeometric Function of a Matrix Argument

---

Plamen Koev
Massachusetts Institute of Technology

Joint work with Alan Edelman

Foundations of Computational Mathematics Conference
Santander, Spain, July 2005

Goal: Computing Statistics of Random Matrices

- $n \times n$ ***real*** Wishart with l degrees of freedom:

$$A \equiv B^T \cdot \Sigma \cdot B, \quad \text{where}$$

$$B = \begin{bmatrix} N(0,1) & N(0,1) & \dots & N(0,1) \\ N(0,1) & N(0,1) & \dots & N(0,1) \\ \vdots & \vdots & \dots & \vdots \\ N(0,1) & N(0,1) & \dots & N(0,1) \end{bmatrix} \qquad (l \times n)$$

$$\Sigma = \begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_n \end{bmatrix} \qquad \text{(Covariance matrix)}$$

- Interested in computing the CDF (PDF) of
  - $\lambda_{\max}, \lambda_{\min}$
  - $\sigma_{\max}, \sigma_{\min}$
  - trace
  - determinant
  - moments

Explicit Formulas Exist—James 1964

$$P(\lambda_{\max}(A) < x) = \frac{\Gamma_n\left(\frac{n+1}{2}\right)}{\Gamma_n\left(\frac{l+n+1}{2}\right)} \cdot \det\left(\frac{1}{2}x\Sigma^{-1}\right)^{l/2} \cdot {}_1F_1\left(\frac{l}{2}; \frac{n+l+1}{2}; -\frac{1}{2}x\Sigma^{-1}\right)$$

(Blue = easy; Red = hard)

- Impossible until now
  - Previous best algorithm ($n = 5$): 8 weeks

    (Gutiérrez, Rodriguez, Sáez, 2000)

Explicit Formulas Exist—James 1964

$$P(\lambda_{\max}(A) < x) = \frac{\Gamma_n\left(\frac{n+1}{2}\right)}{\Gamma_n\left(\frac{l+n+1}{2}\right)} \cdot \det\left(\frac{1}{2}x\Sigma^{-1}\right)^{l/2} \cdot {}_1F_1\left(\frac{l}{2}; \frac{n+l+1}{2}; -\frac{1}{2}x\Sigma^{-1}\right)$$

(Blue = easy; Red = hard)

- Impossible until now
  - Previous best algorithm ($n = 5$): 8 weeks
    (Gutiérrez, Rodriguez, Sáez, 2000)
  - New algorithm: $\frac{1}{100}$ second

The Hypergeometric Function of a Matrix Argument X

$$ {}_pF_q(a_1,\ldots,a_p;b_1,\ldots,b_q;X) \equiv \sum_{k=0}^{\infty}\sum_{\kappa\vdash k} \frac{(a_1)_\kappa\cdots(a_p)_\kappa}{k!(b_1)_\kappa\cdots(b_q)_\kappa}\cdot C_\kappa(X) $$

- ∞ = HARD! Convergence very slow (even for 1×1 X)
  $\Rightarrow$ Truncating $\infty \to m$ means $m = 20, 40, 60, \ldots$
- $\kappa \vdash k$ means κ is a *partition* of k, e.g.,

$$ 7 = 3+2+1+1, \text{ and } 3 \geq 2 \geq 1 \geq 1 $$

- $(a)_\kappa \equiv \prod_{(i,j)\in\kappa}\left(a - \frac{i-1}{\alpha} + j - 1\right)$ — *Pochhammer symbol* (easy)
  $\Rightarrow$ ${}_pF_q$ as hard as ${}_1F_1$
- $C_\kappa(X)$—*Zonal Polynomial*—Really hard: $O(n^m)$ each!
- Our Contribution: $O(n)$

Computing the Zonal Polynomial

- Depends only on the eigenvalues $x_1, x_2, \ldots, x_n$ of X
- Illustrate $\beta = 2$ (complex); general β analogous

Partition κ	C_κ	Number of terms
(1)	$x_1 + \cdots + x_n$	$O(n)$
(2)	$\sum_{i \leq j} x_i x_j$	$O(n^2)$
$(1,1,1)$	$\sum_{i<j<k} x_i x_j x_k$	$O(n^3)$
κ	$\sum_T x^T$	$O\left(n^{\lvert\kappa\rvert}\right)$

Computing the Zonal Polynomial – II

IDEA: Only *update* $C_\kappa(X)$ from $C_\lambda(X)$, $\lambda < \kappa$

$$
\begin{aligned}
C_{(1,1)}(X) &= \sum_{i<j} x_i x_j \\
&= x_1x_2 + (x_1+x_2)x_3 + \cdots + (x_1 + \cdots + x_{n-1})x_n
\end{aligned}
$$

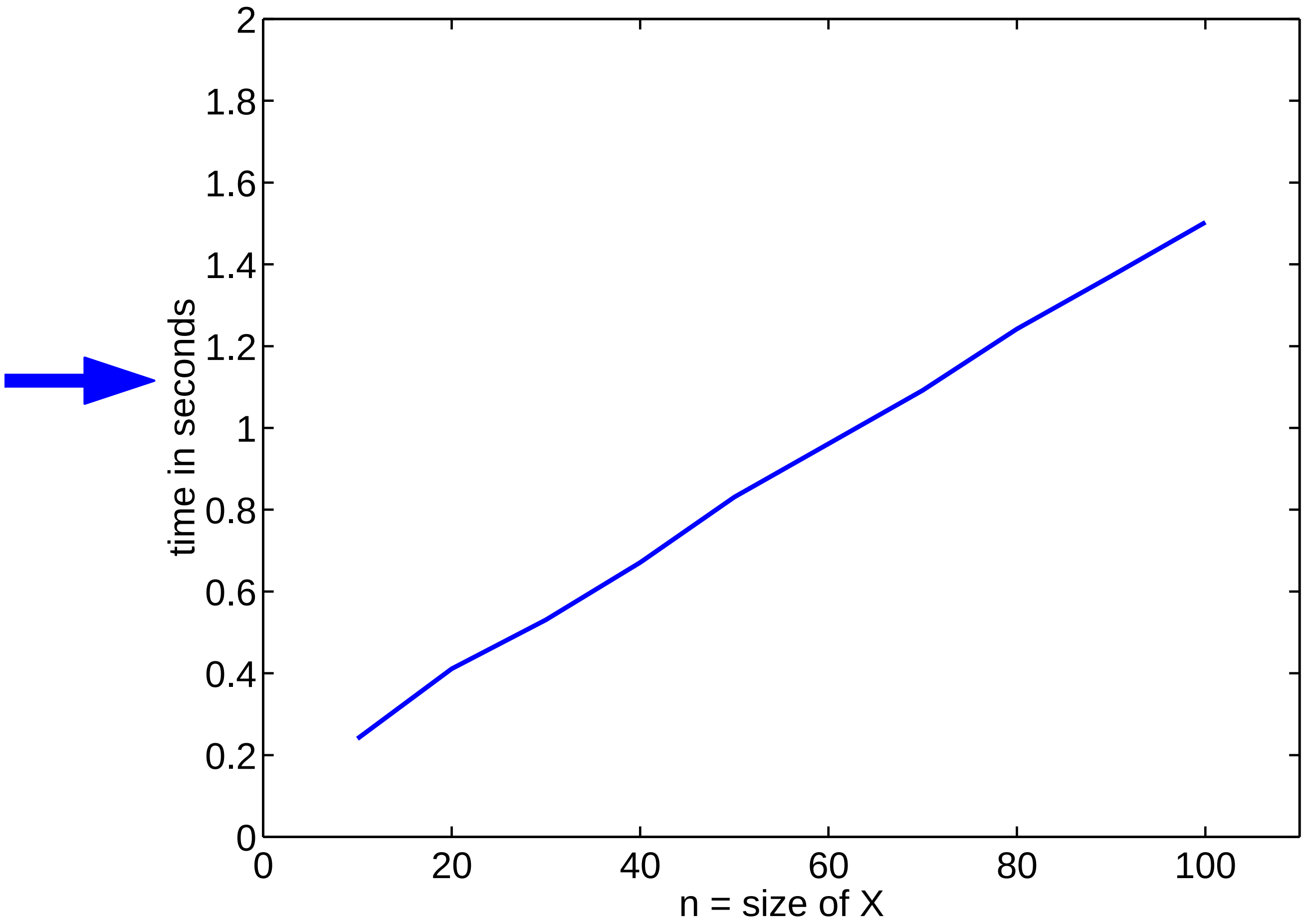
Algorithm:

$$
\begin{aligned}
s_1 &= x_1 & \\
s_2 &= s_1 + x_2 & (= x_1 + x_2) \\
s_3 &= s_2 + x_3 & (= x_1 + x_2 + x_3) \\
&\vdots & \\
s_{n-1} &= s_{n-2} + x_{n-1} & (= x_1 + x_2 + \cdots + x_{n-1})
\end{aligned}
$$

$$
C_{(1,1)}(X) = s_1x_2 + s_2x_3 + \cdots + s_{n-1}x_n
$$

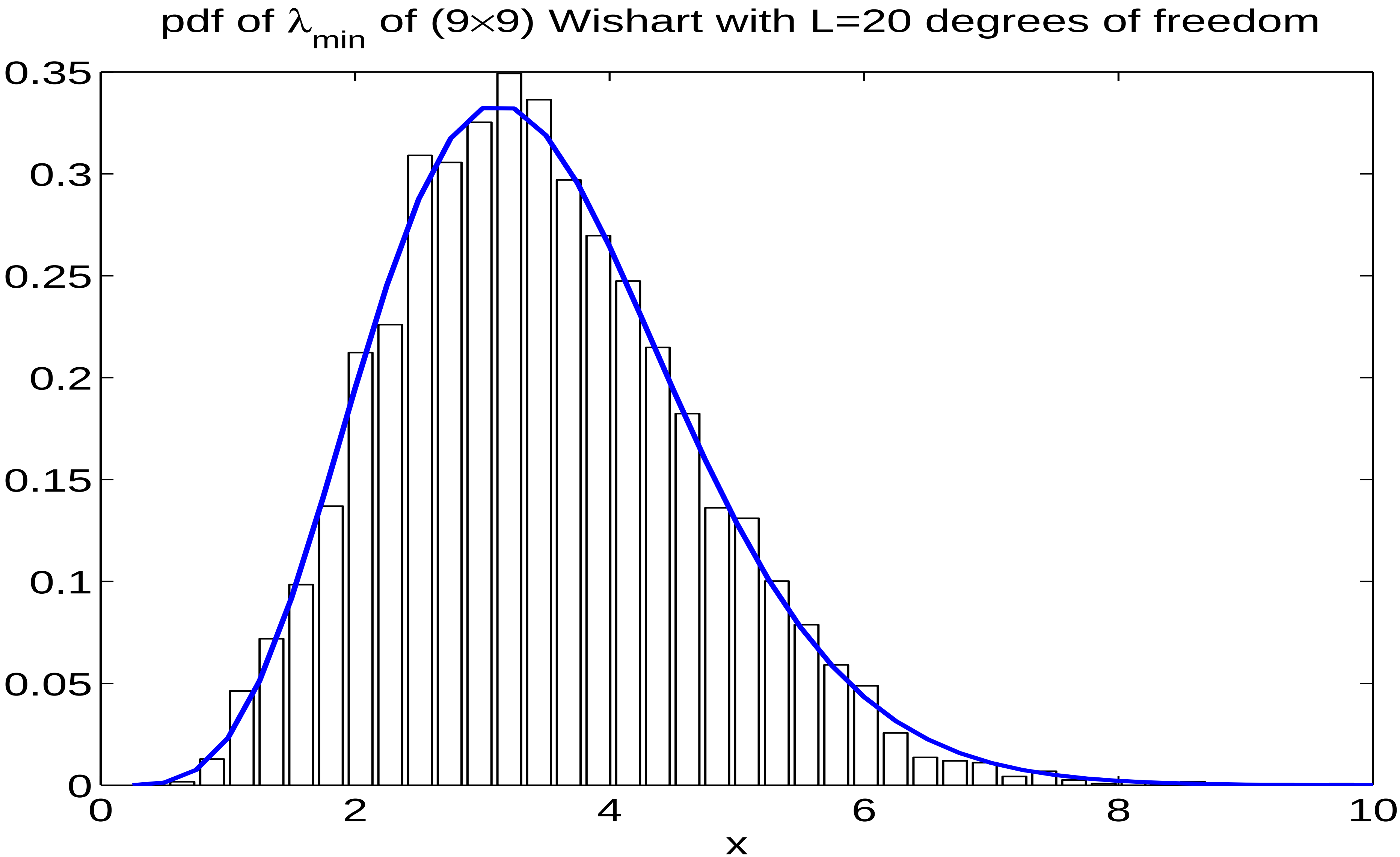
- Cost: $O(n)$ versus $O(n^2)$
- In general: $O(n)$ versus $O(n^{|\kappa|})$

Performance



- **Series truncated at $m = 25$**
- **Linear complexity in $n =$size(X)**
- **ANSI C MEX function in MATLAB**

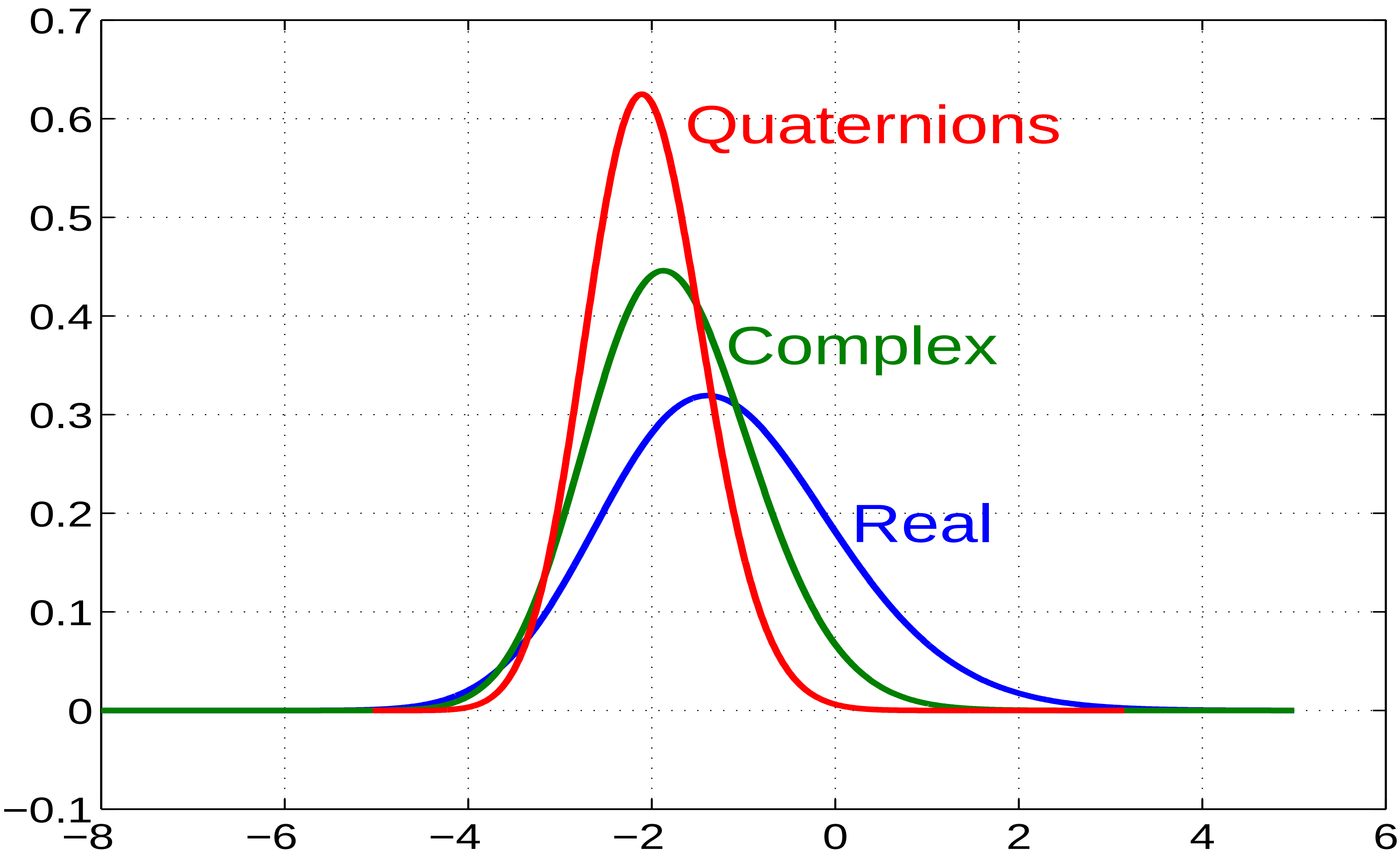
EXAMPLE


pdf of λmin of (9×9) Wishart with L=20 degrees of freedom
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
2
4
6
8
10
x

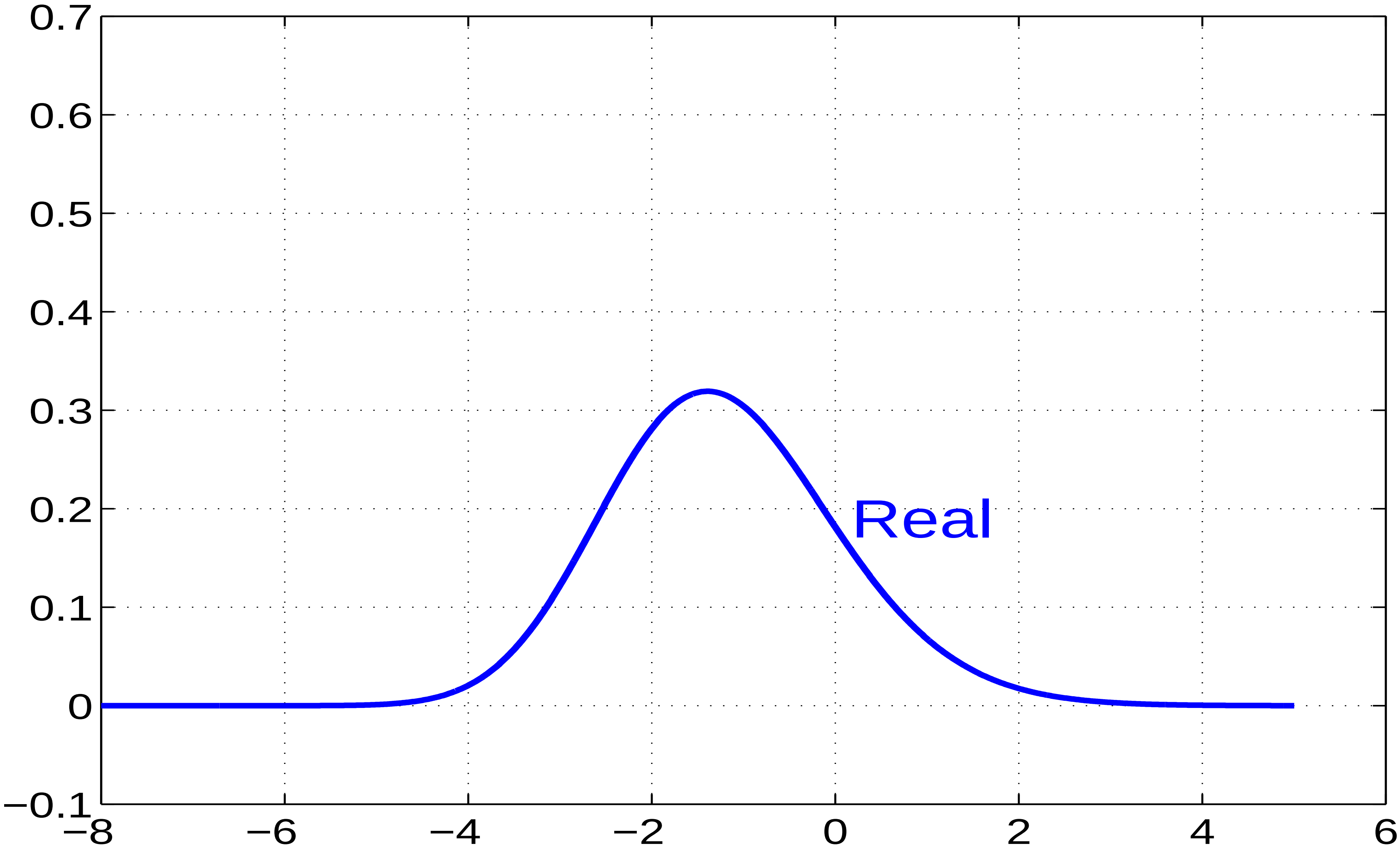
Statistics We Can Now Compute

Statistic	Wishart, *any* Σ	β-Laguerre, $\Sigma = I$
$\lambda_{\min}$	✓	✓
$\lambda_{\max}$	✓	✓
trace	✓	✓
$E(\det^k)$	✓	✓

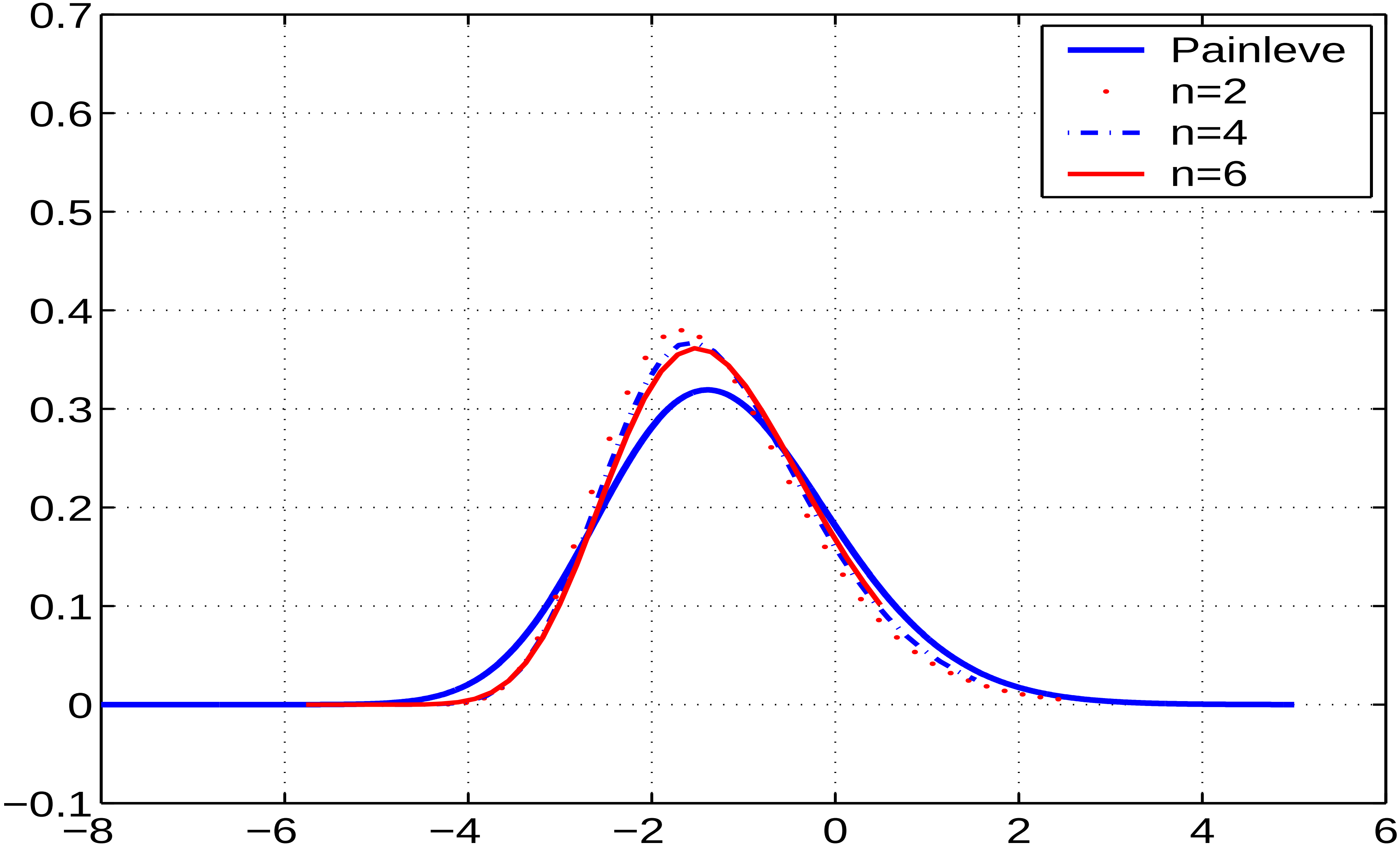
Tracy–Widom Laws for λ_{max} of a Wishart Matrix



Tracy–Widom Laws for $\lambda_{\max}$ of a Wishart Matrix



Tracy–Widom meet Muirhead ($\lambda_{\max}(A_n) \to \lambda_{\max}(A_\infty)$)



Conclusions

- **New efficient algorithm for ${}_pF_q$: Takes seconds**
- **Solves a 40-year-old problem**

- **Future Work: Cooley–Tukey–like algorithm**

$$\text{Cost} = O(\sqrt{\text{Current cost}})$$

- **MATLAB software, paper, slides:**

`http://math.mit.edu/~plamen` = **GOOGLE(Plamen Koev)**