Thesis Presentation

Felix Sarnthein

15. April 2020

Problem Statement: matrix A

$$A = V \cdot \begin{pmatrix} \ddots & \ddots & & & & \\ & \lambda_i & 1 & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & 1 & \\ & & & & \lambda_i & 0 \\ & & & & & \ddots \end{pmatrix} \cdot V^{-1}$$

Problem Statement: polynomial $chev(x)$

$$chev(A) = V \cdot \begin{bmatrix} \ddots & & & & & \\ & chev(\lambda_i) & chev'(\lambda_i) & \cdots & \frac{chev^{(\ell_i - 1)}(\lambda_i)}{(\ell_i - 1)!} & \\ & & \ddots & \ddots & \vdots & \\ & & & \ddots & chev'(\lambda_i) & \\ & & & & chev(\lambda_i) & \\ & & & & & \ddots \end{bmatrix} \cdot V^{-1}$$

For every λ_i, $chev(x)$ has

- a fixpoint at λ_i: $chev(\lambda_i) = \lambda_i$
- a root at all derivatives: $chev^{(k)}(\lambda_i) = 0$

Problem Statement: polynomial $chev(x)$

$$chev(A) = V \cdot \begin{bmatrix} \ddots & & & & & \\ & \lambda_i & 0 & 0 & 0 & \\ & & \ddots & 0 & 0 & \\ & & & \ddots & 0 & \\ & & & & \lambda_i & \\ & & & & & \ddots \end{bmatrix} \cdot V^{-1}$$

For every λ_i, $chev(x)$ has

- a fixpoint at λ_i: $chev(\lambda_i) = \lambda_i$
- a root at all derivatives: $chev^{(k)}(\lambda_i) = 0$

Problem Statement: matrix D

$$D = V \cdot \begin{bmatrix} \ddots & & & & & \\ & \lambda_i & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & \lambda_i & \\ & & & & & \ddots \end{bmatrix} \cdot V^{-1}$$

Problem Statement

Given an integer matrix A

Compute the Jordan-Chevalley decomposition:

- the Chevalley polynomial $chev(x)$
- the diagonalizable matrix $D = chev(A)$

Difficulty: we don’t know the Jordan normal form of A

Algorithm 4.2: Chevalley iteration on matrices

```
  input: matrix $A$ of size $n \times n$
1 $\chi_A(x) \leftarrow$ the characteristic polynomial of $A$        $\prod(\lambda_i - x)^{\alpha_i}$
2 $\mu_D(x) \leftarrow \frac{\chi_A(x)}{gcd(\chi_A(x), \chi'_A(x))}$, the minimal polynomial of $D$        $\prod(\lambda_i - x)$
3 $inv(x) \leftarrow$ the inverse of $\mu'_D(x)$ modulo $\mu_D(x)$

4 $S_0 \leftarrow A$
5 while $S_{k+1} \neq S_k$ do
6 |   $S_{k+1} \leftarrow S_k - \mu_D(S_k) \cdot inv(S_k)$
7 end

8 $D \leftarrow S_\ell$, the converged matrix
9 return $D$
```

Algorithm 4.3: Chevalley iteration on polynomials

```
  input: matrix $A$ of size $n \times n$
1 $\chi_A(x) \leftarrow$ the characteristic polynomial of $A$                    $\prod (\lambda_i - x)^{\alpha_i}$
2 $\mu_D(x) \leftarrow \frac{\chi_A(x)}{gcd(\chi_A(x), \chi'_A(x))}$, the minimal polynomial of $D$      $\prod (\lambda_i - x)$
3 $inv(x) \leftarrow$ the inverse of $\mu'_D(x)$ modulo $\mu_D(x)$
4 $s_0(x) \leftarrow x$, the linear polynomial
5 while $s_{k+1}(x) \neq s_k(x)$ do
6     $s_{k+1}(x) \leftarrow s_k(x) - \mu_D(s_k(x)) \cdot inv(s_k(x)) \pmod{\chi_A(x)}$
7 end
8 chev$(x) \leftarrow s_\ell(x)$, the converged polynomial
9 return $chev(A)$
```

Chevalley Iteration

$$s_k(A) = V \cdot \begin{pmatrix} \ddots & & & & & \\ & s_k(\lambda_i) & {s_k}'(\lambda_i) & \cdots & \dfrac{s_k^{(\ell_i-1)}(\lambda_i)}{(\ell_i-1)!} & \\ & & \ddots & \ddots & \vdots & \\ & & & \ddots & {s_k}'(\lambda_i) & \\ & & & & s_k(\lambda_i) & \\ & & & & & \ddots \end{pmatrix} \cdot V^{-1}$$

Chevalley Iteration

$s_0(x) = x$

$$s_0(A) = V \cdot \begin{bmatrix} \ddots & & & & & \\ & \lambda_i & {s_0}'(\lambda_i) & \cdots & \dfrac{s_0^{(\ell_i - 1)}(\lambda_i)}{(\ell_i - 1)!} & \\ & & \ddots & \ddots & \vdots & \\ & & & \ddots & {s_0}'(\lambda_i) & \\ & & & & \lambda_i & \\ & & & & & \ddots \end{bmatrix} \cdot V^{-1}$$

For every λ_i, $s_0(x)$ has

- a fixpoint at λ_i: $s_0(\lambda_i) = \lambda_i$

Chevalley Iteration

$$s_0(x) = x$$
$$s_1(x) = x - muD(x) \cdot inv(x)$$

$$s_1(A) = V \cdot \begin{bmatrix} \ddots & & & & & \\ & \lambda_i & 0 & \cdots & \frac{s_1^{(\ell_i-1)}(\lambda_i)}{(\ell_i-1)!} & \\ & & \ddots & \ddots & \vdots & \\ & & & \ddots & 0 & \\ & & & & \lambda_i & \\ & & & & & \ddots \end{bmatrix} \cdot V^{-1}$$

For every λ_i, $s_1(x)$ has

- a fixpoint at λ_i: $s_1(\lambda_i) = \lambda_i$
- a root at the first derivatives: $s_k^{(1)}(\lambda_i) = 0$

Chevalley Iteration

$$s_0(x) = x$$
$$s_1(x) = x \; - muD(x) \cdot inv(x)$$
$$s_k(x) \; - muD\big(s_k(x)\big) \cdot inv\big(s_k(x)\big)$$

$$s_1(A) = V \cdot \begin{pmatrix} \ddots & & & & & \\ & \lambda_i & 0 & \cdots & \frac{s_1^{(\ell_i - 1)}(\lambda_i)}{(\ell_i - 1)!} & \\ & & \ddots & \ddots & \vdots & \\ & & & \ddots & 0 & \\ & & & & \lambda_i & \\ & & & & & \ddots \end{pmatrix} \cdot V^{-1}$$

For every λ_i, $\mathrm{s_k}(x)$ has

- a fixpoint at λ_i: $\mathrm{s_k}(\lambda_i) = \lambda_i$
- a root at the first k derivatives: $s_k^{(j)}(\lambda_i) = 0 \;\; \forall j = 1, \dots, k$

Chevalley Iteration

$s_0(x) = x$
$s_1(x) = x - muD(x) \cdot inv(x)$
$s_k(x) - muD\big(s_k(x)\big) \cdot inv\big(s_k(x)\big)$

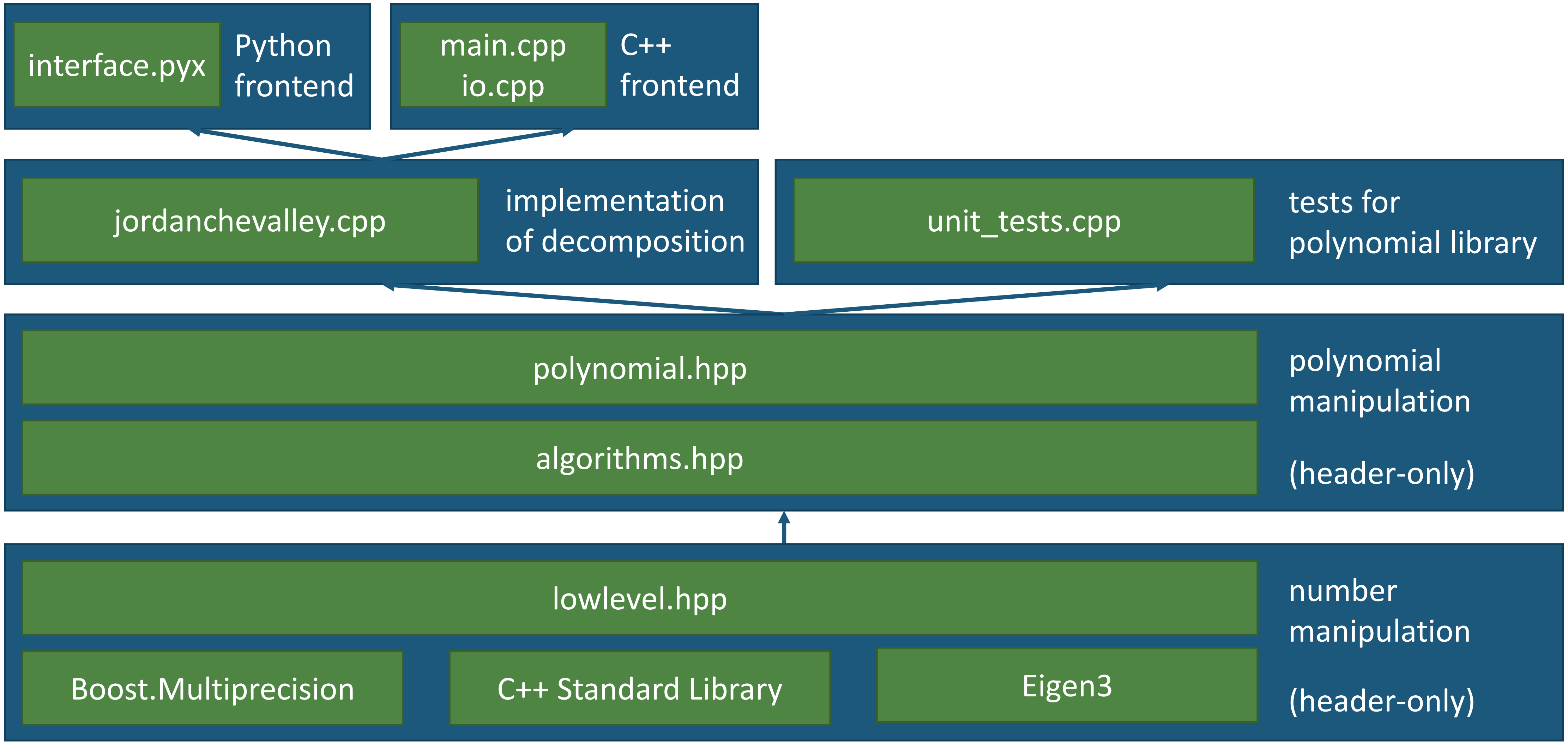
$$s_\ell(A) = V \cdot \begin{pmatrix} \ddots & & & & & \\ & \lambda_i & 0 & 0 & 0 & \\ & & \ddots & \ddots & 0 & \\ & & & \ddots & 0 & \\ & & & & \lambda_i & \\ & & & & & \ddots \end{pmatrix} \cdot V^{-1}$$

For every λ_i, $s_\ell(x)$ has

- a fixpoint at λ_i: $s_\ell(\lambda_i) = \lambda_i$
- a root at all derivatives: $s_\ell^{(j)}(\lambda_i) = 0 \;\; \forall j = 1, \ldots, \ell$

Implementation

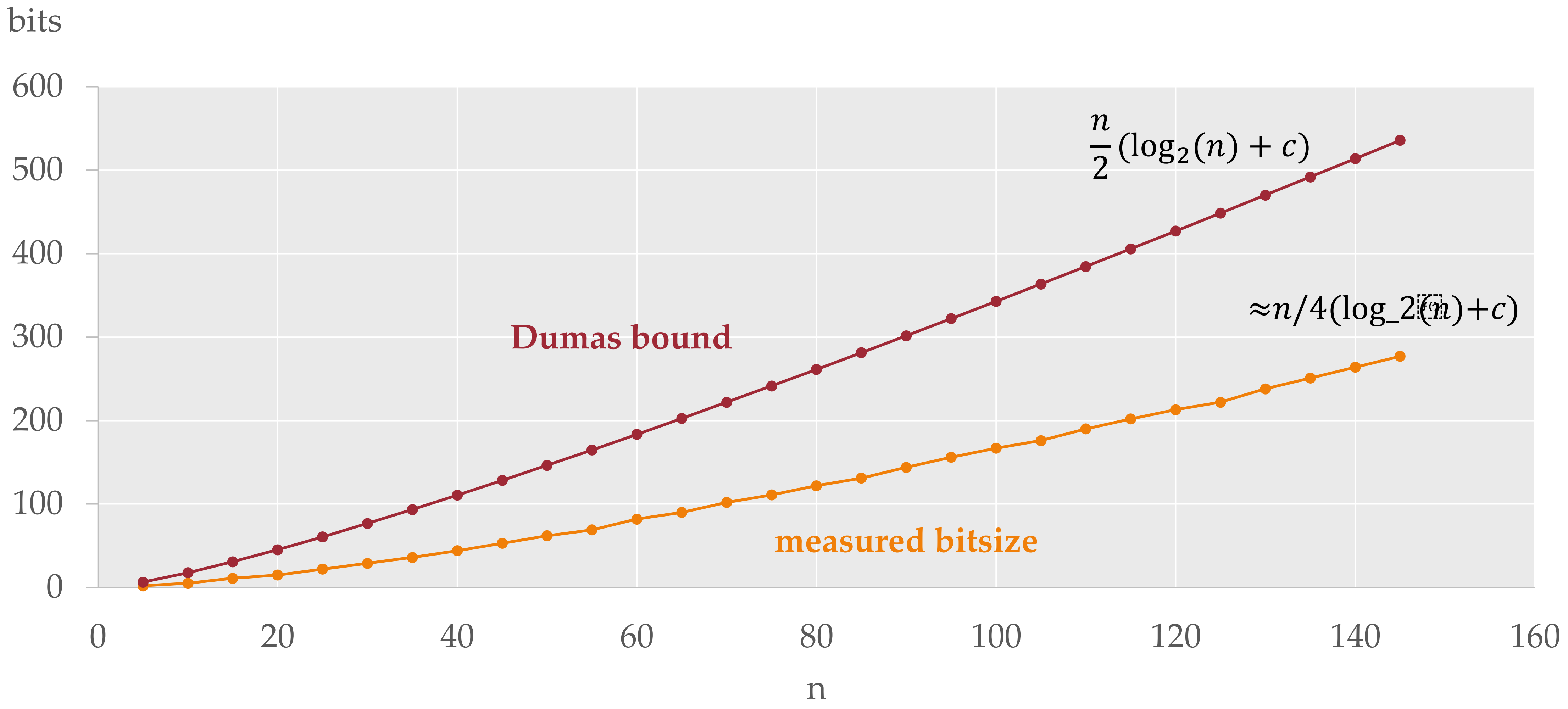
Structure of the Project



Python Extension

	Step	Method	Accessor
0.	Initialize decomposition	dec = JCDec(A)	dec.__A__
1.	characteristic polynomial $chiA(x)$	dec.compute_chiA(string precision, bool round)	dec.__chiA__
2.	minimal polynomial $muD(x)$	dec.compute_muD(bool resultant)	dec.__muD__
3.	inverse polynomial $inv(x)$	dec.compute_inv( )	dec.__inv__
4.	iteration $s_k(x) \to chev(x)$	dec.compute_chev( )	dec.__chev__
5.	diagonalizable matrix D	dec.compute_D(bool mat)	dec.__D__
6.	nilpotent matrix N	dec.compute_N( )	dec.__N__
	compute all steps	dec.compute( )	

Coefficients Bitsizes of Characteristic Polynomial



⇒ Need variable precision integers

dec.compute_chiA( )

- Variable precision floating-point implementation in $O(n^3)$
  - ➢ round coefficients to integers

floating point precision	bits in mantissa	exact for matrix size $\leq$
single / mp::quad	24	20
double / mp::double	53	40
long double	64	45
mp::quad	113	75
mp::oct	237	130
mp::50	168	100
mp::100	334	175
mp::1000	3324	1260

dec.compute_muD( )

$$muD(x) = \frac{chiA(x)}{\gcd(\ chiA(x), chiA'(x))}$$

1. Derivative of monomial in $O(n) \Rightarrow chiA'(x)$
2. Euclidean algorithm in $O(n^2) \Rightarrow$ $\gcd(\ chiA(x), chiA'(x))$
3. Polynomial division in $O(n^2)$ $\Rightarrow muD(x)$

$\Rightarrow$ Need rational number type

Resultant algorithm: gcd for integer polynomials

$\Rightarrow$ Coefficient bitsize grows in $O(n \log n)$

dec.compute_inv( )

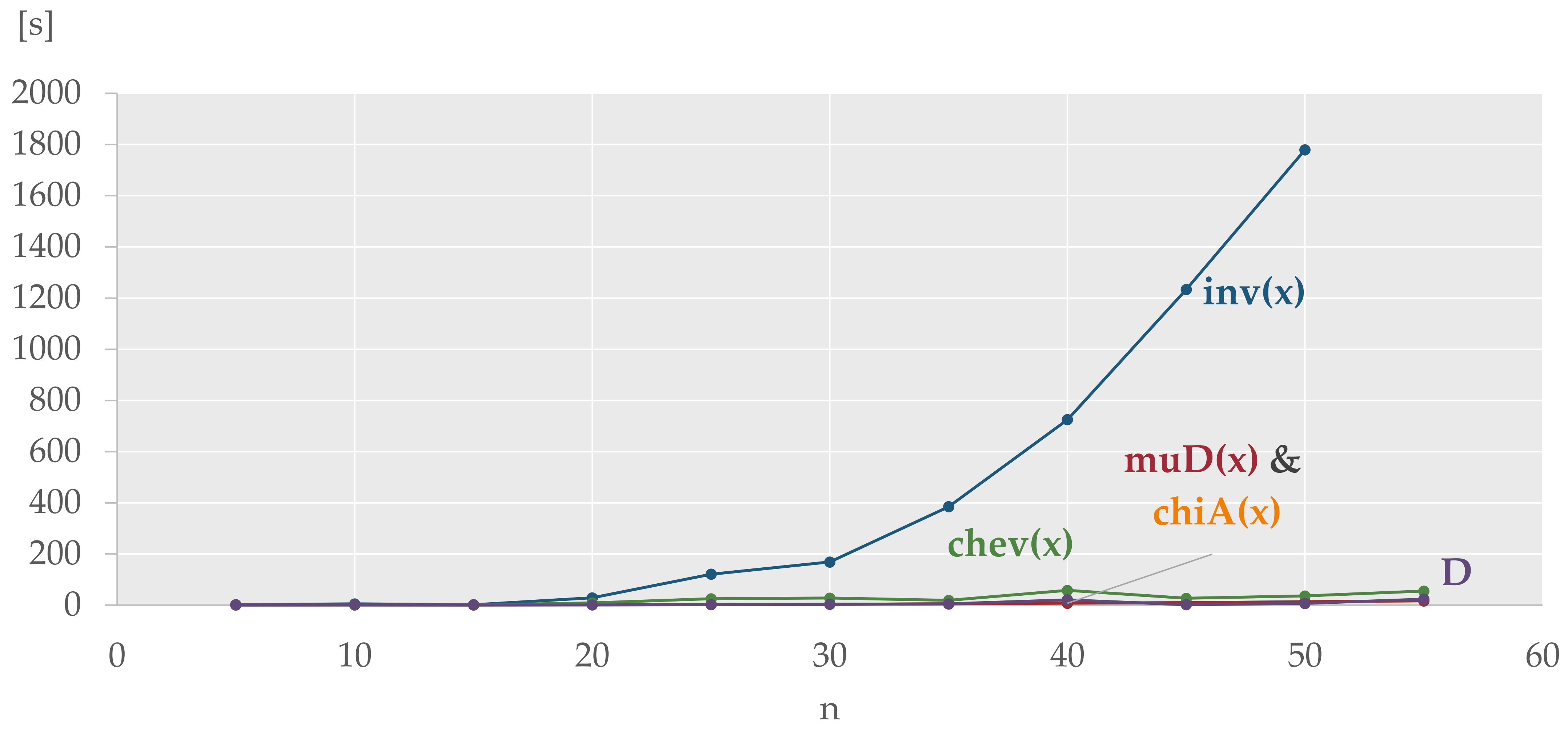
Extended Euclidean Algorithm in $O(n^2)$ yields Bézout Identity

$\Rightarrow muD'(x) \cdot inv(x) + muD(x) \cdot v(x) = 1$

$\Rightarrow muD'(x) \cdot inv(x) \equiv 1 \pmod{\mathrm{muD(x)}}$

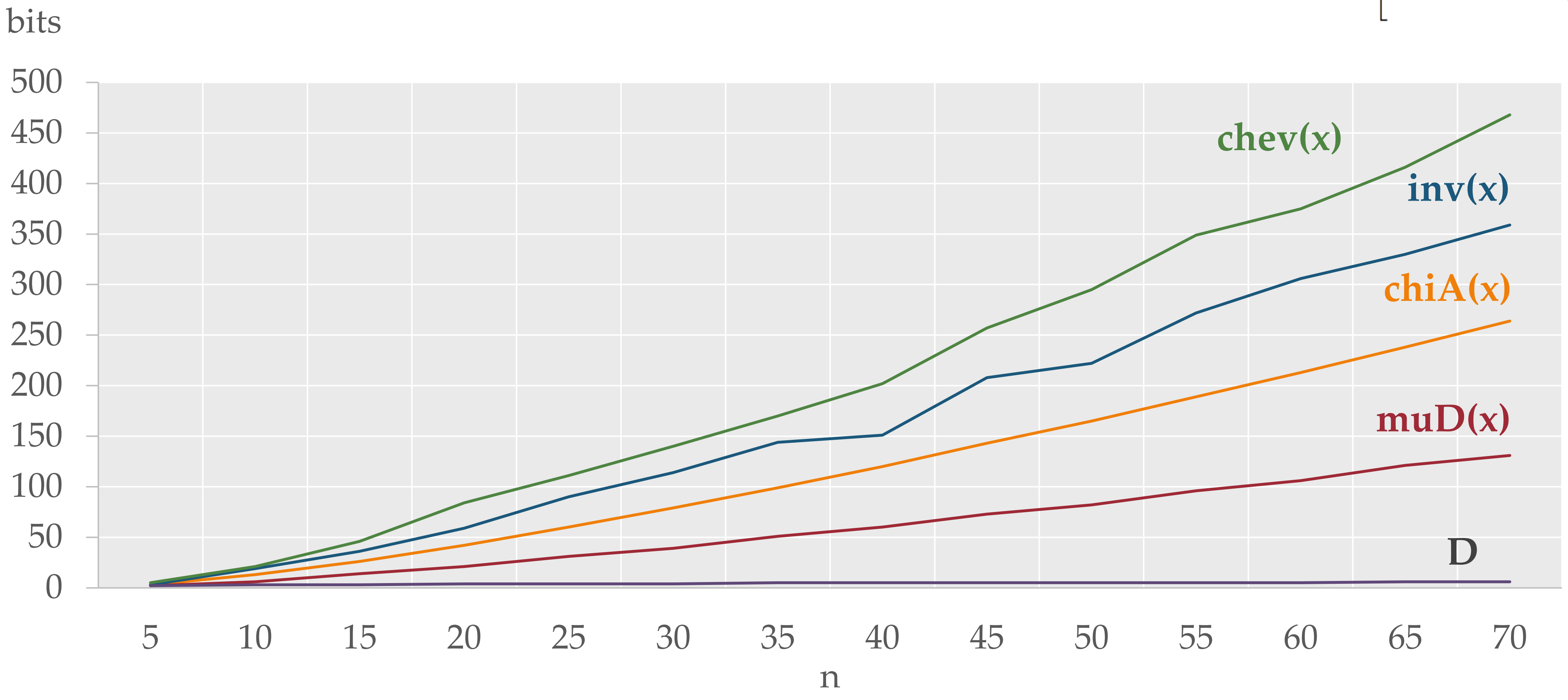
BUT: coefficient bitsizes might grow very fast

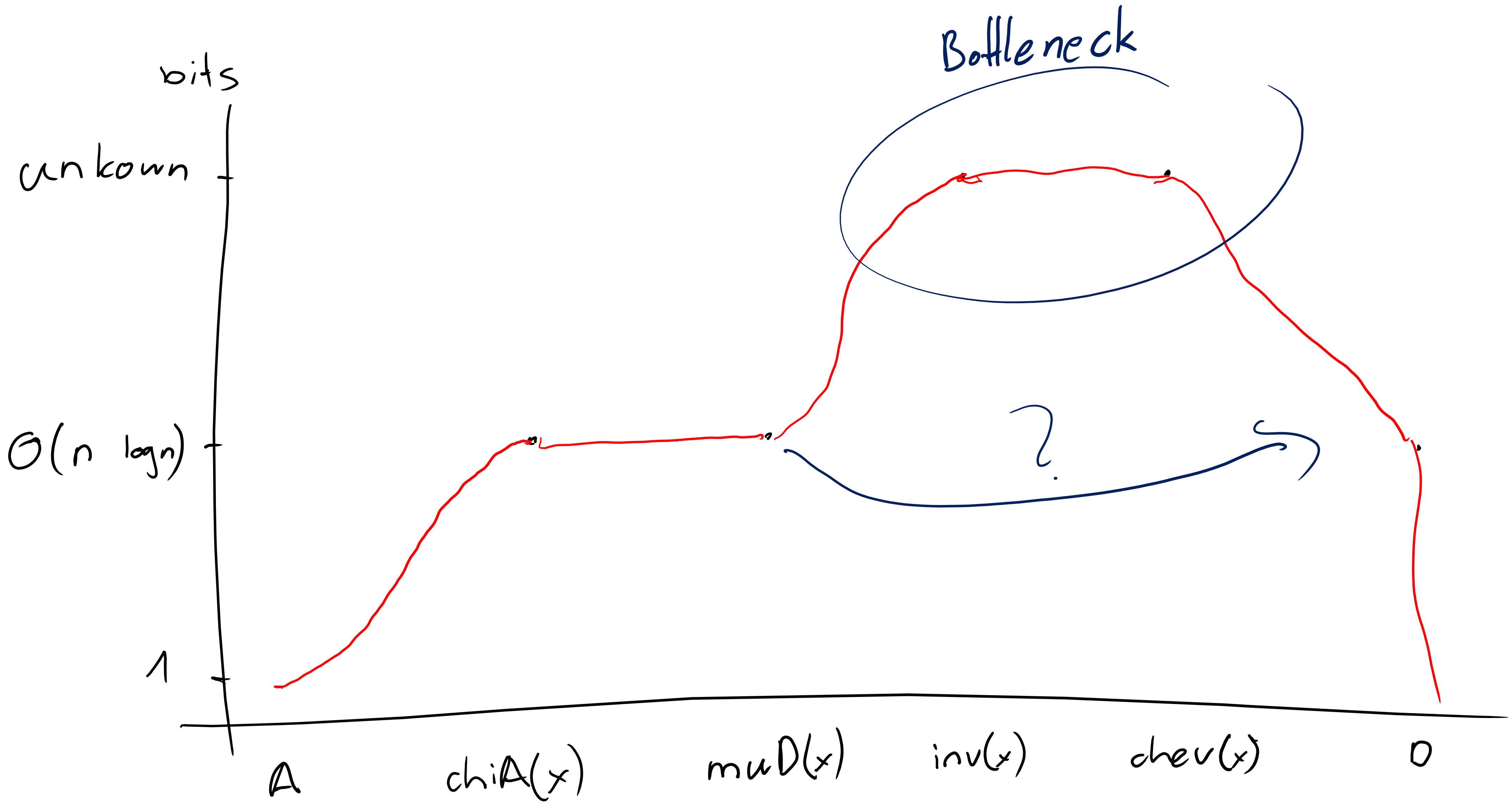
Bitsizes of Intermediate Results (Bernoulli p=1/16)



$$\begin{bmatrix} 0 & 1 & & & & & \\ & 0 & 0 & & & & \\ & & 1 & 1 & & & \\ & & & 1 & 0 & & \\ & & & & 2 & 1 & \\ & & & & & 2 & \ddots \\ & & & & & & \ddots \end{bmatrix}$$

Coefficient Bitsizes of Intermediate Results




Bottleneck
bits
unkown
O(n logn)
1
?
A
chiA(x)
muD(x)
inv(x)
chev(x)
O

dec.compute_D(mat = True)

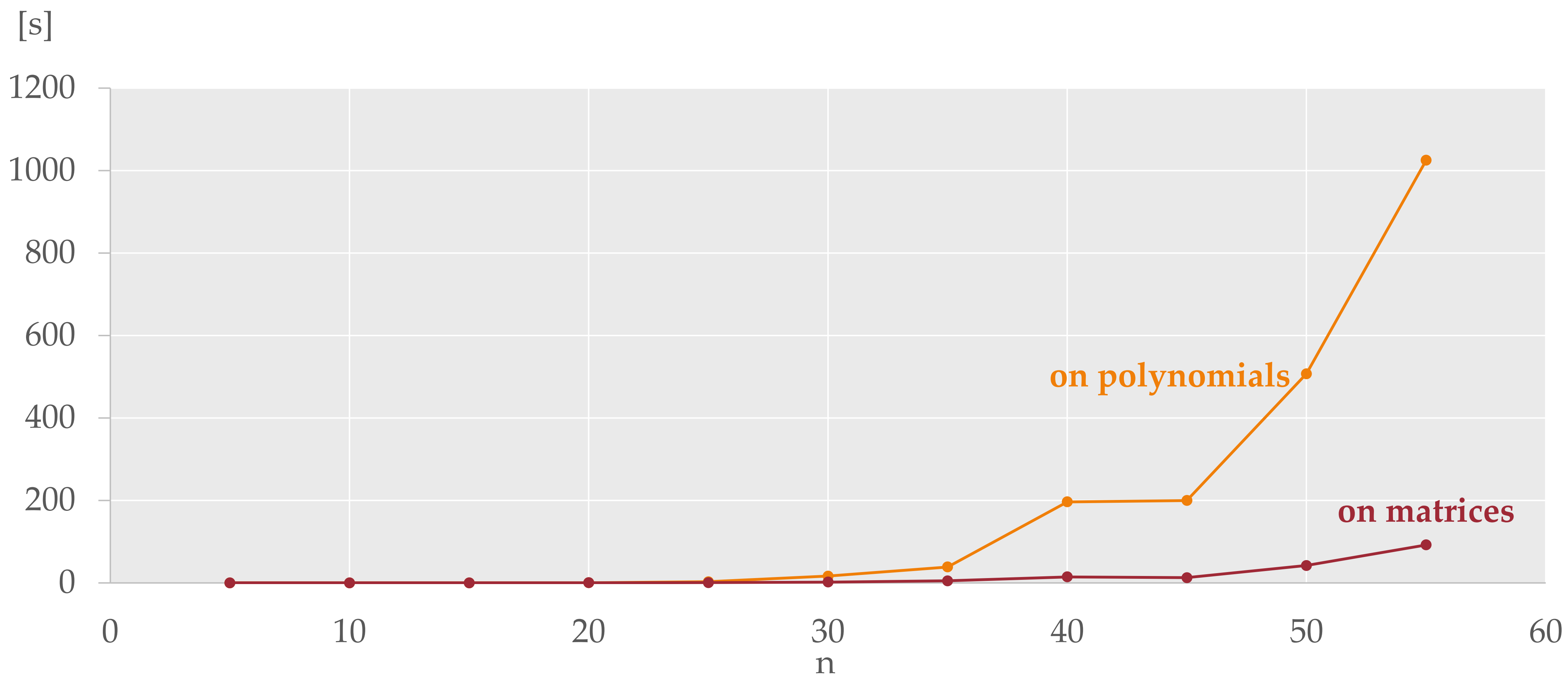
Algorithm 4.3: Chevalley iteration on matrices

```
input: matrix $A$ of size $n \times n$
$\chi_A(x) \leftarrow$ the characteristic polynomial of $A$
$\mu_D(x) \leftarrow \frac{\chi_A(x)}{gcd(\chi_A(x), \chi'_A(x))}$, the minimal polynomial of $D$
$S_0 \leftarrow A$
while $S_{k+1} \neq S_k$ do
    $S_{k+1} \leftarrow S_k - \mu_D(S_k) \cdot \mu'_D(S_k)^{-1}$
end
$D \leftarrow S_\ell$, the converged matrix
return $D$
```

$\prod (\lambda_i - x)^{\alpha_i}$

$\prod (\lambda_i - x)$

Runtime of Chevalley Iterations (Bernoulli p=1/16)



Summary

- No need for eigenvalues and multiplicities
  - ➢ exact computation possible

- Bitsize of coefficient grows superlinearly
  - ➢ requires variable precision computation

- Bitsize of $chev(x)$ is significantly larger than D
  - ➢Iteration on matrices is more efficient

Ideas for Optimization (Meeting Notes)

- Is a floating-point implementation feasible?
  - Other Algorithms for Newton Iteration on matrices

- Sparsity
  - Leverage sparsity for evaluation of matrix polynomials
  - Leverage sparsity for computation of characteristic polynomial

- Alternatives to Horner scheme?

- Use the minimal polynomial of A as a starting point (instead of $chiA$)?
  - Bitsize complexity would be $O(N_\lambda \cdot \log_2 \rho(A))$, where
    - N_λ is the number of distinct eigenvalues
    - $\rho(A)$ is the spectral radius, which is bound for example by the nnz of a matrix A

Appendix

Algorithm 4.1: Hermite interpolation

```
input: matrix $A$ of size $n \times n$
1 for $i \leftarrow 0$ to $m$ do
2     $\lambda_i \leftarrow$ the $i^{th}$ distinct eigenvalue of $A$
3     $\ell_i \leftarrow$ the Jordan index of eigenvalue $\lambda_i$
4 end
5 chev$(x) \leftarrow$ the solution to the interpolation problem in lemma 3.3
6 $D \leftarrow chev(A)$, the evaluation of $chev(x)$ on the matrix $A$
7 return $D$
```

Hermite Interpolation

$$chev(A) = V \cdot \begin{bmatrix} \ddots & & & & & \\ & chev(\lambda_i) & chev'(\lambda_i) & \cdots & \dfrac{chev^{(\ell_i - 1)}(\lambda_i)}{(\ell_i - 1)!} & \\ & & \ddots & \ddots & \vdots & \\ & & & \ddots & chev'(\lambda_i) & \\ & & & & chev(\lambda_i) & \\ & & & & & \ddots \end{bmatrix} \cdot V^{-1}$$

For every λ_i, $chev(x)$ has

- a fixpoint at λ_i: $chev(\lambda_i) = \lambda_i$
- a root at its derivatives: $chev^{(k)}(\lambda_i) = 0$

Hermite Interpolation

$$chev(A) = V \cdot \begin{pmatrix} \ddots & & & & & \\ & \lambda_i & 0 & 0 & 0 & \\ & & \ddots & 0 & 0 & \\ & & & \ddots & 0 & \\ & & & & \lambda_i & \\ & & & & & \ddots \end{pmatrix} \cdot V^{-1}$$

For every λ_i, $chev(x)$ has

- a fixpoint at λ_i: $chev(\lambda_i) = \lambda_i$
- a root at its derivatives: $chev^{(k)}(\lambda_i) = 0$