Computation of classical orthogonal polynomials and their associated Gauss quadrature rules

Javier Segura

Departamento de Matemáticas, Estadística y Computación
Universidad de Cantabria, Spain

FASTRELAX, ENS Lyon

1. Gauss-quadratures and OPs: basic ideas
2. Computation of (classical) Gaussian quadratures: overview
   - Golub-Welsch, iterative and asymptotic methods
   - State of the art
   - Iterative, asymptotic or both?
3. Iterative computation of classical Gauss quadratures
   - Some general comments
   - Sturm comparison and the global fourth order method
   - Gauss-Hermite quadrature
   - Gauss-Laguerre quadrature
   - Gauss-Jacobi quadrature (some ideas)
4. Asymptotic methods for OPs and Gauss quadrature
   - An example for $P_n^{(\alpha,\beta)}(x)$ and G-J quadrature
   - Some numerical results

Gauss-quadratures: basic ideas

Given $I(f) = \int_a^b f(x)w(x)dx$, with $w(x)$ a weight function, the n-point quadrature rule

$$Q_n(f) = \sum_{i=1}^{n} w_i f(x_i)$$

is a Gaussian quadrature if $I(f) = Q_n(f)$ for f any polynomial with $\deg(f) \leq 2n-1$.

Gaussian quadrature rules are optimal in a very specific sense and they are one of the more widely used methods of integration.

The difficulty is, of course, computing the nodes x_i and weights w_i.

As it is well known, the nodes x_i, $i = 1, \ldots, n$ of the Gaussian quadrature rule are the roots of the (for instance monic) orthogonal polynomial satisfying

$$\int_a^b x^i p_n(x)w(x)dx = 0,\ i = 0, \ldots, n-1.$$

Gaussian quadrature is the result of approximating the function f by the polynomial f_{n-1} ($\deg(f_{n-1}) \leq n-1$) of lowest degree such that ($f(x_i) = f_{n-1}(x_i)$, $i = 1, \ldots n$).

$$I(f) = \int_a^b f(x)w(x)dx \approx Q(f) = \int_a^b f_{n-1}(x)w(x)dx = \sum_{i=1}^{n} w_i f(x_i),$$

$$f_{n-1}(x) = \sum_{i=1}^{n} f(x_i)L_i(x),\ L_i(x_j) = \delta_{i,j},\ \deg(L_i) = n-1,$$

and then

$$w_j = \int_a^b L_i(x)w(x)dx = \int_a^b \frac{p_n(x)}{(x-x_j)p_n'(x_j)}w(x)dx$$

And with the aid of the Christoffel-Darboux formula a more practical formula is obtained in terms of monic polynomials

$$\boldsymbol{w_j = -\frac{||p_n||^2}{p_n'(x_j)p_{n+1}(x_j)},\ ||p_n||^2 = \int_a^b p_n(x)^2 w(x)dx}$$

Classical Gaussian quadrature

This is the case for which the OPs are solutions of second order ODEs

$$C(x)y_n''(x) + B(x)y_n'(x) + \lambda_n y(x) = 0$$

with C and B polynomials.
Three cases:

1. Hermite: $w(x) = e^{-x^2}$ in $(-\infty, +\infty) \rightsquigarrow \boldsymbol{H_n(x)}$
2. Laguerre: $w(x) = x^{-\alpha}e^{-x}$, $\alpha > -1$, in $(0, +\infty) \rightsquigarrow \boldsymbol{L_n^{(\alpha)}(x)}$
3. Jacobi: $w(x) = (1-x)^{\alpha}(1+x)^{\beta}$, $\alpha, \beta > -1$, in $(-1, 1) \rightsquigarrow \boldsymbol{P_n^{(\alpha,\beta)}(x)}$

Apart from being solution of a second order ODE, the coefficients of the three-term recurrence relation are simple, as well as the coefficients in

$$y_n'(x) = a_n(x)y_n(x) + b_n y_{n-1}(x)$$

Computation of Gauss quadratures: overview

Two mains ways

1. Compute $w_j = -||p_n||^2/(p_n'(x_i)p_{n+1}(x_i))$, for which we need:
   a) A method to compute the polynomials $p_n(x)$ and the first derivative.
   b) A method to compute the roots of $p_n(x)$ (nodes x_i).

2. A second posibility is provided by the recurrence relation (for monic polynomials):

$$p_{k+1}(x) = (x - B_k)p_k(x) - A_k p_{k-1}(x), \quad k = 1, 2, \ldots,$$

where $A_0 p_{-1} \equiv 0$ and

$$A_k = \frac{||p_k||^2}{||p_{k-1}||^2}, \quad k \geq 1, \quad B_k = \frac{\langle xp_k, p_k \rangle}{||p_k||^2}, \quad k \geq 0.$$

$$< f, g >= \int_a^b f(x)g(x)w(x)dx, \; ||f|| =< f, f >$$

The Golub-Welsch algorithm

Let

$$J = \begin{pmatrix} \beta_0 & \alpha_1 & 0 & \dots & 0 \\ \alpha_1 & \beta_1 & \alpha_2 & & \\ 0 & \alpha_2 & \beta_2 & & \vdots \\ & & & & \\ \vdots & & & \ddots & \alpha_{n-1} \\ & & & & \\ 0 & \dots & & \alpha_{n-1} & \beta_{n-1} \end{pmatrix}$$

$\alpha_i = \sqrt{A_i}$, $\beta_i = B_i$. Then the n different eigenvalues of J are the nodes. Furthermore, if $\vec{\Phi}^{(j)}$ is an eigenvector with eigenvalue the node x_j:

$$w_j = \mu_0 \frac{(\Phi_1^{(j)})^2}{||\vec{\Phi}^{(j)}||_E^2}$$

where $\Phi_1^{(j)}$ is the first component of $\vec{\Phi}^{(j)}$ and $\mu_0 = \int_a^b w(x)dx$.

Several approaches to compute classical Gaussian quadratures:

1. Golub-Welsch: straightforward. We just need to diagonalize a tridiagonal matrix with explicitly know entries. Not so good for large degree n.
2. Iterative methods: either we use a globally convergent method or we need previous estimations of the nodes (usually from asymptotic methods for large degree). Better for large degree than GW.
3. Asymptotic methods?: can the initial estimations from asymptotics be accurate enough? What about the weights? Best methods for large n
4. Other methods: numerical integration of the ODE by with a non-oscillatory phase functions (Bremer). Maybe competitive for very large n, but asymptotics will be better in that case.

Some references on the computation of classical Gauss quadrature

G. H. Golub, J. H. Welsch, Calculation of Gauss quadrature rules.
Math. Comp. (1969) ALL

W. Gautschi, Orthogonal polynomials: computation and approximation.
Oxford U. Press (2004) ALL

E. Yakimiw, Accurate computation of weights in classical Gauss-Christoffel quadrature rules.
J. Comput. Phys. (1996) A+I+RR Legendre (Hermite and Laguerre $\alpha = 0$ to a lesser extent)

K. Petras, On the computation of the Gauss-Legendre quadrature formula with a given precision.
J. Comput. Appl. Math. (1999) A^*+I+? Legendre

P. N. Swarztrauber, On computing the points and weights for Gauss-Legendre quadrature.
SIAM J. Sci. Comput. (2002) A+I+FS Legendre

A. Glaser, X. Liu, V. Rokhlin, A fast algorithm for the calculation of the roots of special functions.
SIAM J. Sci. Comput. (2007) A/O+I+**TS** Hermite, Laguerre ($\alpha = 0$), Legendre

J. Segura, Reliable computation of the zeros of solutions of second order linear ODEs using a fourth order method.
SIAM J. Numer. Anal. (2010) □**+I+?** Hermite, Laguerre, Jacobi (*)

I. Bogaert, B. Michiels, J. Fostier, J., O(1) computation of Legendre polynomials and Gauss-Legendre nodes and weights for parallel computing.
SIAM J. Sci. Comput. (2012) A+I+V Legendre

N. Hale, A. Townsend, Fast and accurate computation of Gauss-Legendre and Gauss-Jacobi quadrature nodes and weights.
SIAM J. Sci. Comput. (2013) A+I+A Jacobi

A. Townsend, The race to compute high-order Gauss-Legendre quadrature.
SIAM News (2015)

I. Bogaert, Iteration-free computation of Gauss-Legendre quadrature nodes and weights.
SIAM J. Sci. Comput. (2014) □+A+A Legendre

A. Townsend, T. Trogdon, S. Olver, Fast computation of Gauss quadrature nodes and weights on the whole real line.
IMA J. Numer. Anal (2016) A+I+A Hermite*

J. Bremer, On the numerical calculation of the roots of special functions satisfying second order ordinary differential equations.
SIAM J. Sci. Comput (2017) O Laguerre, Jacobi. Very high degree.

A. Gil, J. Segura, N. M. Temme Asymptotic approximations to the nodes and weights of Gauss-Hermite and Gauss-Laguerre quadratures.
Stud. Appl. Math. (2018) □+A+A Hermite, Laguerre

F. Johansson, M. Mezzarobba Fast and rigorous arbitrary-precision computation of Gauss-Legendre quadrature nodes and weights.
SIAM J. Sci. Comput (2019) A*+I+V Legendre

A. Gil, J. Segura, N. M. Temme Non-iterative computation of Gauss-Jacobi quadrature.
SIAM J. Sci. Comput. (2019) □+A+A Jacobi

A. Gil, J. Segura, N. M. Temme Fast, reliable and unrestricted iterative computation of Gauss–Hermite and Gauss–Laguerre quadratures.
(submitted) □+I+TS/CF Hermite, Laguerre

In the previous list, there are a number of methods which combine asymptotic and convergent iterative methods. We prefer purely asymptotic methods and purely iterative (convergent methods). We argue that these are the best (and complementary) approaches:

1. Iterative-only methods with no asymptotic approximations lead to arbitrary precision algorithms provided that the computations are based on finite or convergent expansions.
2. Asymptotic-only methods can not be used for arbitary precision, but these are the fastest methods, and they can be accurate (15-16 digits) for moderately large degrees (as first proved for Gauss-Legendre in Bogaert's paper).

Methods which combine both iterative and asymptotic methods, may be accurate and efficient, but no so accurate as a purely iterative method and not as fast as a purely asymptotic method.

The present talk is based on:

A. Gil, JS, N. M. Temme Asymptotic approximations to the nodes and weights of Gauss-Hermite and Gauss-Laguerre quadratures. **Stud. Appl. Math. (2018)** □+A+A

A. Gil, JS, N. M. Temme Non-iterative computation of Gauss-Jacobi quadrature. **SIAM J. Sci. Comput. (2019)** □+A+A

A. Gil, JS, N. M. Temme Fast, reliable and unrestricted computation of Gauss-Hermite and Gauss Laguerre quadratures. **(submitted)** □+I+TS/CF

Iterative computation of classical Gauss quadratures

Several remarks regarding iterative methods in the literature:

◯ Most papers use Newton's method for computing the roots (order 2):

Newton's method:

The NM, $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$, has order of convergence 2 because

$$\frac{\epsilon_{n+1}}{\epsilon_n^2} = \frac{f''(\alpha)}{2f'(\alpha)} + \mathcal{O}(\epsilon_n) \text{ as } n \to \infty,\ \epsilon_n = x_n - \alpha.$$

◯ The only proof of convergence for Newton method is for the Legendre case (Petras, 1999).

◯ The defining ODE can be used to speed up the method and also to improve the convergence without increasing the complexity.

For computing zeros of solutions of

$$w''(x) + B(x)w'(x) + C(x)w(x) = 0$$

Newton method gives **order** 2 generally. **But the ODE can be used to speed-up the method.**
Assuming that $B(x)$ is differentiable we can transform (13) by setting

$$y(x) = \exp\left(\int \frac{1}{2}B(x)dx\right) w(x)$$

Then, $y''(x) + A(x)y(x) = 0$, with $A(x) = C(x) - \frac{1}{2}B'(x) - \frac{1}{4}B(x)^2$ and

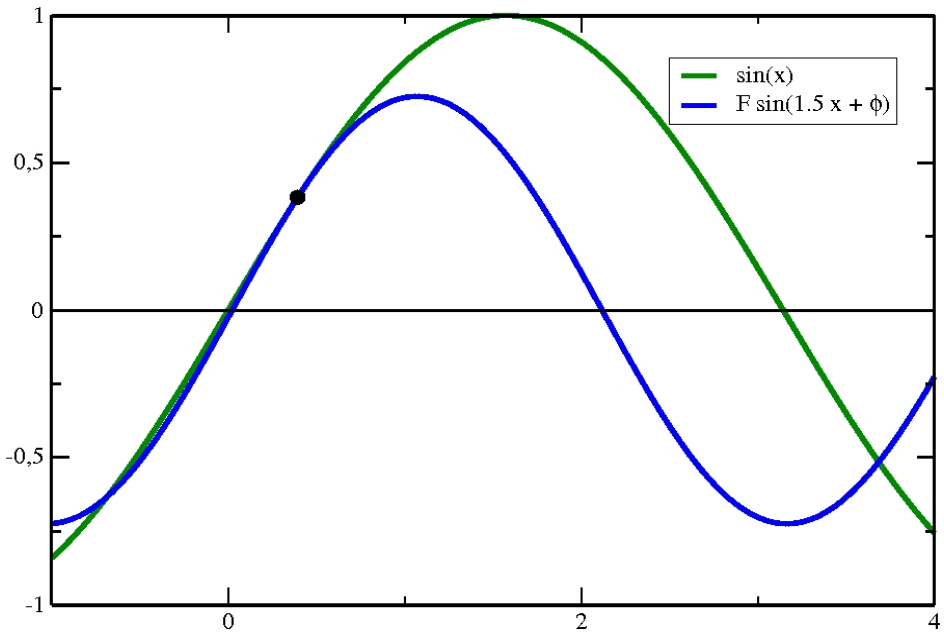
$$\frac{y(x)}{y'(x)} = \frac{w(x)}{\frac{1}{2}B'(x)w(x) + w'(x)}$$

The Newton method $x_{n+1} = x_n - \dfrac{y(x_n)}{y'(x_n)}$ is now of third order.

The reason: if α is such that $y(\alpha) = 0$, then $y''(\alpha) = 0$.
And we haven't used $A(x)$ so far...

Theorem (Sturm comparison)

Let $y(x)$ and $w(x)$ be solutions of $y''(x) + A_y(x)y(x) = 0$ and $w''(x) + A_w(x)w(x) = 0$ respectively, with $A_y(x) > A_w(x) > 0$. If $y(x_0)w'(x_0) - y'(x_0)w(x_0) = 0$ and x_y and x_w are the zeros of $y(x)$ and $w(x)$ closest to x_0 and larger (or smaller) than x_0, then $x_y < x_w$ (or $x_y > x_w$).



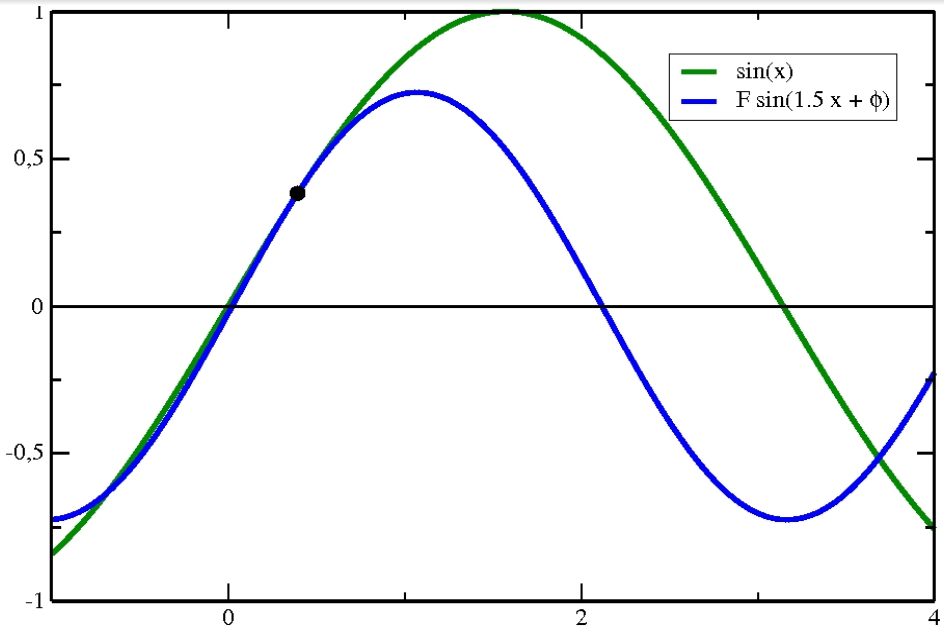
Equations: $y''(x) + y(x) = 0$, $y''(x) + 2.25y(x) = 0$

Algorithm (Zeros of $y''(x) + A(x)y(x) = 0$, $A(x)$ monotonic)

Given x_n, the next iterate x_{n+1} is computed as follows: find a solution of the equation

$$w''(x) + A(x_n)w(x) = 0$$

such that $y(x_n)w'(x_n) - y'(x_n)w(x_n) = 0$. If $A'(x) < 0$ ($A'(x) > 0$) take as x_{n+1} the zero of $w(x)$ closer to x_n and larger (smaller) than x_n.



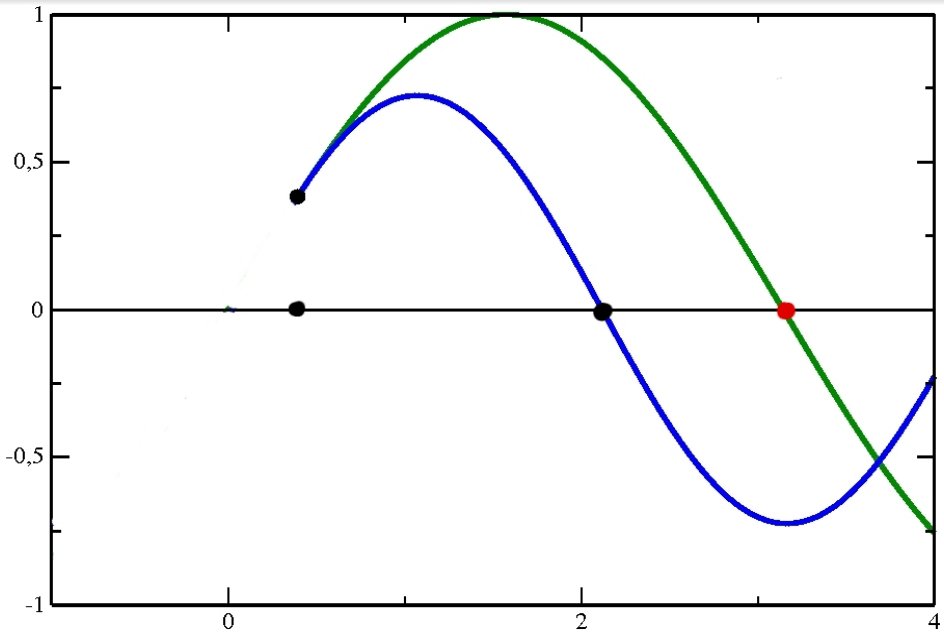
Equations: $y''(x) + y(x) = 0$, $y''(x) + 2.25y(x) = 0$

Algorithm (Zeros of $y''(x) + A(x)y(x) = 0$, $A(x)$ monotonic)

Given x_n, the next iterate x_{n+1} is computed as follows: find a solution of the equation

$$w''(x) + A(x_n)w(x) = 0$$

such that $y(x_n)w'(x_n) - y'(x_n)w(x_n) = 0$. If $A'(x) < 0$ $(A'(x) > 0)$ take as x_{n+1} the zero of $w(x)$ closer to x_n and larger (smaller) than x_n.



Equations: $y''(x) + A(x)y(x) = 0$, $w''(x) + A(x_n)w(x) = 0$, $(A'(x) < 0)$

The method is equivalent to iterating $x_{n+1} = T(x_n)$ with the following fixed point iteration.

Let $j = \text{sign}(A'(x))$, we define

$$T(x) = x - \frac{1}{\sqrt{A(x)}} \arctan_j(\sqrt{A(x)}h(x))$$

with

$$\arctan_j(\zeta) = \begin{cases} \arctan(\zeta) \text{ if } jz > 0, \\ \arctan(\zeta) + j\pi \text{ if } jz \leq 0, \\ j\pi/2 \text{ if } z = \pm\infty \end{cases}$$

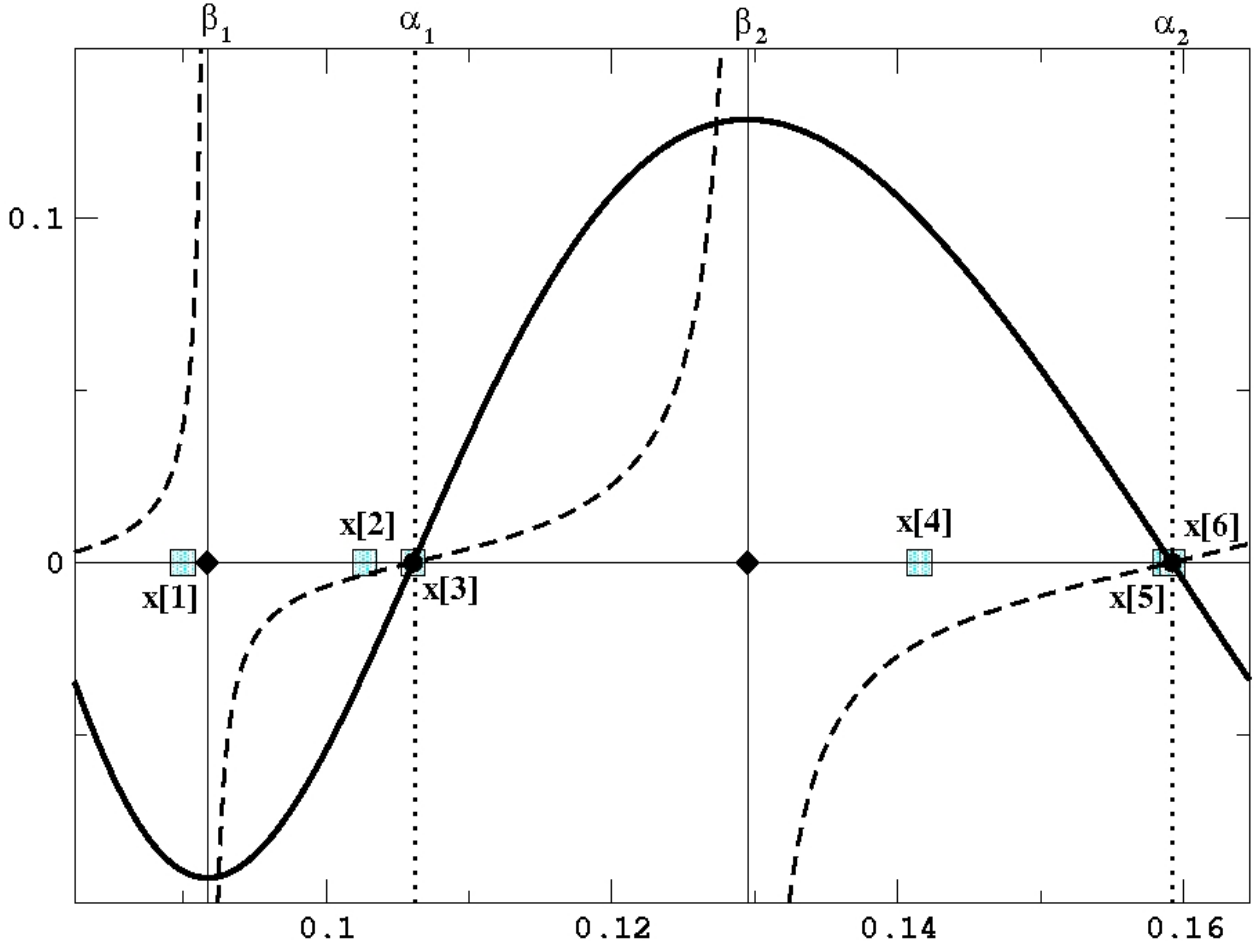
This method converges to α for any x_0 in $[\alpha', \alpha)$ if $A'(x) < 0$, with α' the largest zero smaller than α (analogously for $A'(x) > 0$).

The method has fourth order convergence:

$$\epsilon_{n+1} = \frac{A'(\alpha)}{12}\epsilon_n^4 + \mathcal{O}(\epsilon_n^5),\ \epsilon_k = x_k - \alpha$$

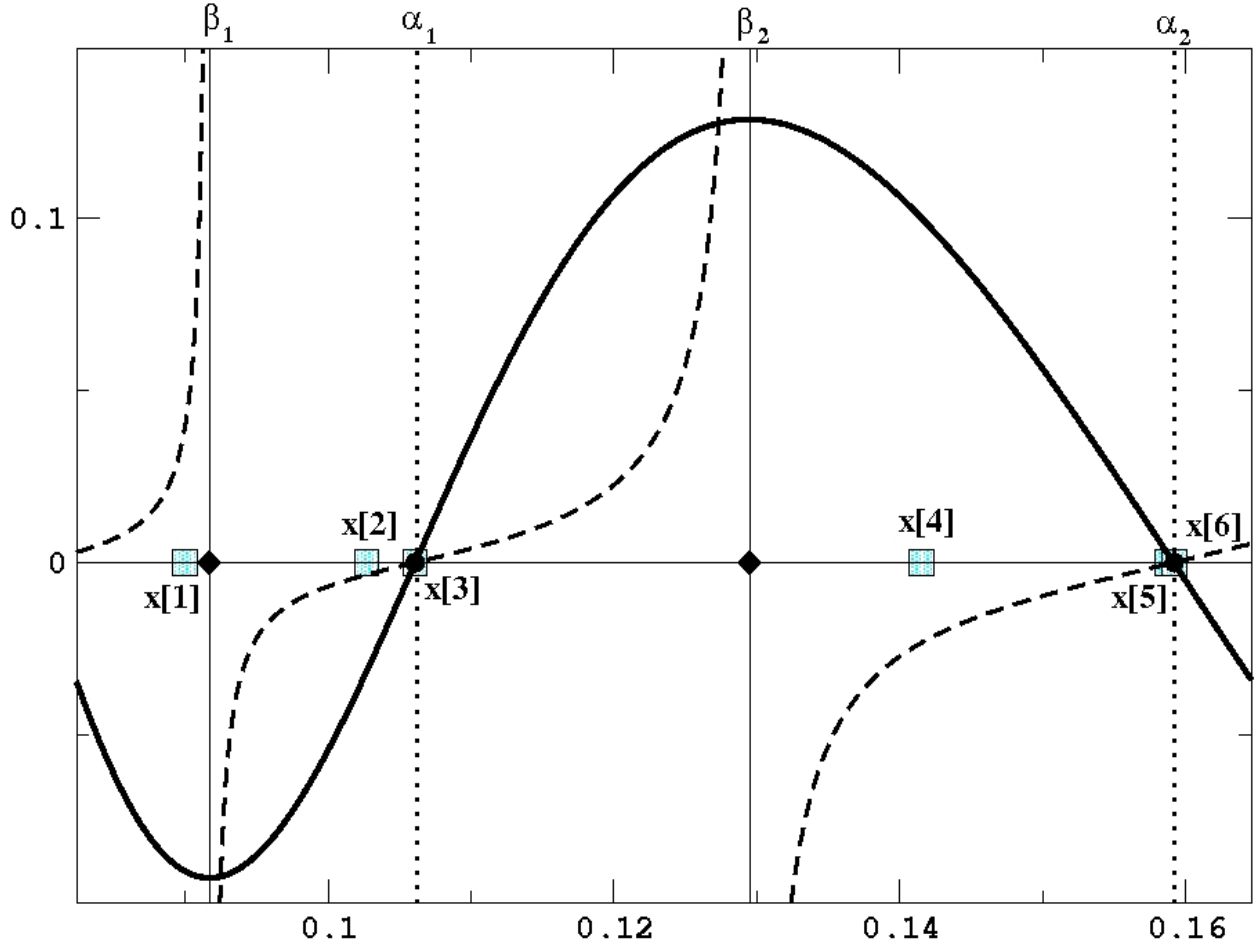
Computing the zeros in an interval where $A(x)$ is monotonic.

Example: zeros of $y(x) = x\sin(1/x)$, satisfying $y''(x) + x^{-4}y(x) = 0$ (4 digits of acc.).



Computing the zeros in an interval where $A(x)$ is monotonic.

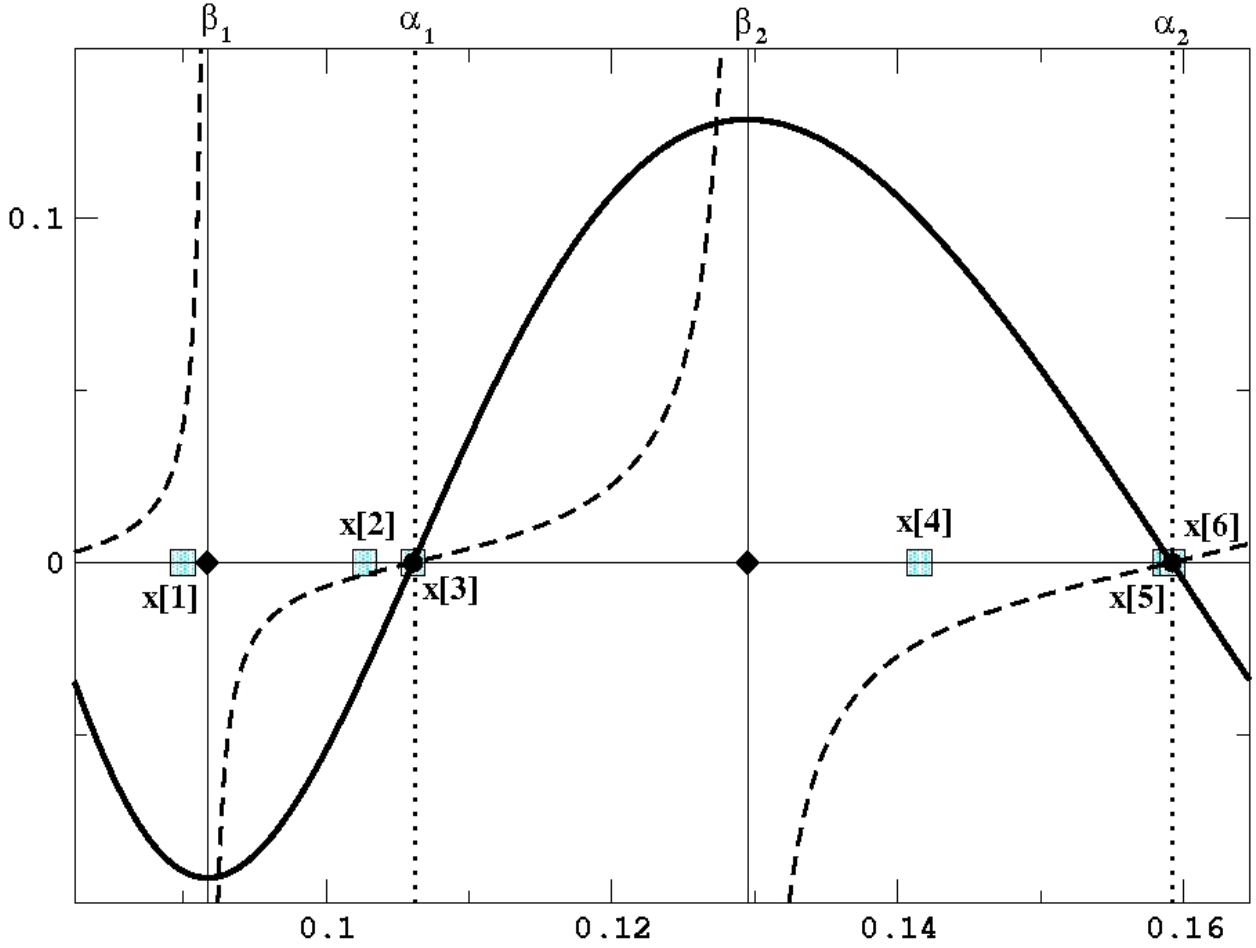
Example: zeros of $y(x) = x\sin(1/x)$, satisfying $y''(x) + x^{-4}y(x) = 0$ (4 digits of acc.).



1. $T(x[1]) = x[2]$, $T(x[2]) = x[3]$ (with four digits acc.)

Computing the zeros in an interval where $A(x)$ is monotonic.

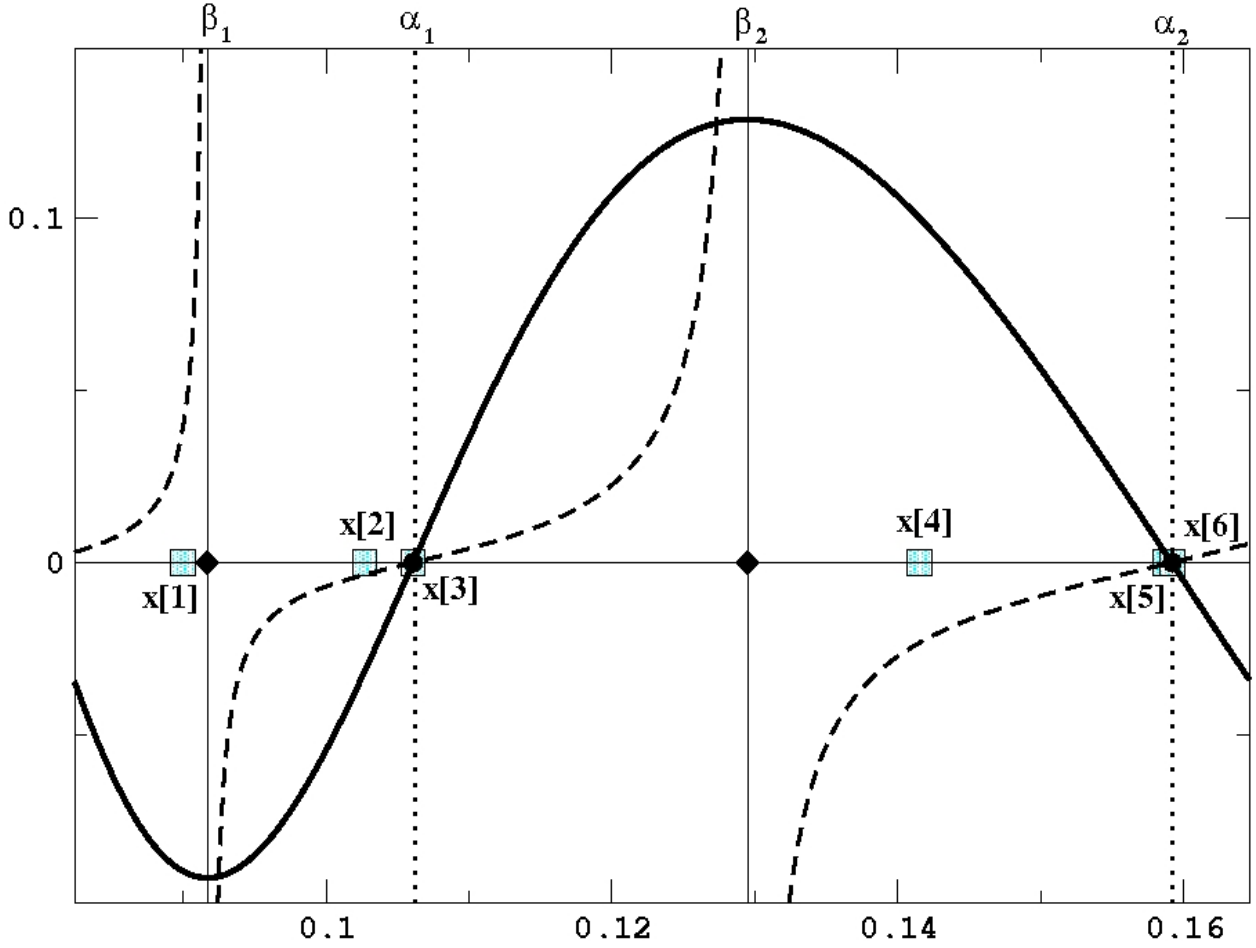
Example: zeros of $y(x) = x\sin(1/x)$, satisfying $y''(x) + x^{-4}y(x) = 0$ (4 digits of acc.).



1. $T(x[1]) = x[2]$, $T(x[2]) = x[3]$ (with four digits acc.)
2. $x[4] = x[3] + \pi/\sqrt{A(x[3])}$ (smaller than the next zero by Sturm comparison)

Computing the zeros in an interval where $A(x)$ is monotonic.

Example: zeros of $y(x) = x\sin(1/x)$, satisfying $y''(x) + x^{-4}y(x) = 0$ (4 digits of acc.).



1. $T(x[1]) = x[2]$, $T(x[2]) = x[3]$ (with four digits acc.)
2. $x[4] = x[3] + \pi/\sqrt{A(x[3])}$ (smaller than the next zero by Sturm comparison)
3. $T(x[4]) = x[5]$, $T(x[5]) = x[6]$ (with four digits acc.)

1. Guaranteed convergence. Does not require initial estimates.

1. Guaranteed convergence. Does not require initial estimates.
2. Fourth order convergence (Newton has order 2)

1. Guaranteed convergence. Does not require initial estimates.
2. Fourth order convergence (Newton has order 2)
3. Same computational cost per iteration as Newton!

1. Guaranteed convergence. Does not require initial estimates.
2. Fourth order convergence (Newton has order 2)
3. Same computational cost per iteration as Newton!
4. And more: it will be asymptotically exact if a convenient variable is chosen.

Gauss-Hermite quadrature

We have that $y_n(x) = C_n e^{-x^2/2} H_n(x)$ satisfies

$$y_n''(x) + A(x) y_n(x) = 0,\ A(x) = 2n + 1 - x^2.$$

$A(x)$ has its maximum at $x = 0$. The nodes are symmetric around the origin.

We compute the positive roots in the direction of decreasing $A(x)$, starting at $x = 0$ until we have computed $\lfloor n/2 \rfloor$ zeros.
The first step is

$$x = T_{-1}(0) = \begin{cases} \dfrac{\pi}{\sqrt{2n+1}}, & n \text{ odd} \\ \dfrac{\pi}{2\sqrt{2n+1}}, & n \text{ even} \end{cases}$$

As $n \to +\infty$ the coefficient $A(x)$ is essentially constant for not too large x.
In this sense, the method will be asymptotically exact.

Methods are available for computing efficiently and reliably $y_n(x)$ (see A. Gil , JS, N.M. Temme. ACM Trans. Math. Softw. (2006)), but here we prefer to avoid asymptotics so that arbitary accuracy for any degree is available:

The three-term recurrence relation IS NOT a good idea, particularly for large degree: the complexity is bad as for Golub-Welsch. A good alternative: use local Taylor series (as done in Glaser, Liu, Rokhlin (2007)).

Given $y(x) = C_n e^{-x^2/2} H_n(x)$, and assuming that the derivatives at x_0 are known:

$$y(x) = \sum_{k=0}^{\infty} \frac{y^{(k)}(x_0)}{k!}(x - x_0)^k.$$

and similarly for $y'(x)$, truncating the series for a given precision.

From $y(x_0)$ and $y'(x_0)$, we compute the sucessive derivatives by differentiating $y''(x) + (2n + 1 - x^2)y(x) = 0$:

$$y^{(k+2)} + (2n + 1 - x^2)y^{(k)} - 2kxy^{(k-1)} - k(k-1)y^{(k-2)} = 0.$$

Perron-Kreuser theorem: all the solutions of the difference equation are such that $\limsup_{k \to +\infty} \left(|y^{(k)}| / (k!)^{2/3} \right)^{1/k} = 1$ (the series converges everywhere, as expected).

$h = x - x_0$ will be always less than the maximal distance between zeros of $H_n(x)$.

Algorithm for Gauss-Hermite based on local Taylor series.
Take $y(x) = (2^n n!)^{-1/2} e^{-x^2/2} H_n(x)$.

1. With $x = \pi/\sqrt{2n+1}$ for n odd and $x = \pi/(2\sqrt{2n+1})$ for n even, compute $y(x)$ and $y'(x)$ by Taylor series starting from the known values $y(0)$ and $y'(0)$ **(alternatively, we can take $y(0) = 1$ and $y'(0) = 0$ for n even and $y(0) = 0$ and $y'(0) = 1$ for n odd, and rescale later)**. Let $i = 1$.
2. Iterate the fixed point method

$$T(x) = x - \frac{1}{\sqrt{A(x)}} \arctan_{-1}(\sqrt{A(x)} h(x))$$

   until convergence is reached within a given accuracy. The values of $y(x)$ and $y'(x)$ are computed from Taylor series, starting with the values at the previous point. Let x_i be the resulting zero (node)
3. The corresponding weight is given by $w_i = 2e^{-x_i^2}/(y_n'(x_i))^2$.
4. Set $x = x_i + \pi/\sqrt{A(x_i)}$, $i = i + 1$, and if additional nodes have to be computed then go to 2.

If rescaling is used, we just have to take into account that the sum of all the weights is $\sqrt{\pi}$.

Some (nice) features of the algorithm

1. The cost of local Taylor series does not increase with the degree.
2. The cost of computing each zero decreases as $n \to \infty$ (asymptotic exactness)
3. Because of the fourth order convergence and the error relation, when two succesive iterates are such that $|x^{(n+1)} - x^{(n)}| < 6^{1/4}\, 10^{-p/4}$ then we can estimate that $|x_{n+1}/\alpha - 1| < 10^{-p}$.
4. The so-called scaled weights $\omega_i = 1/y'(x_i)^2$ are well-conditioned as a function of x_i because $\omega_i = W(x_i)$ with $W'(x_i) = 0$.
5. The code is short (< 100 lines), reliable and efficient.

The method does not need initial estimations for the roots. It seems they don't improve significantly the performance (typical running time: 0.5 seconds for 10^6 nodes in my laptop).

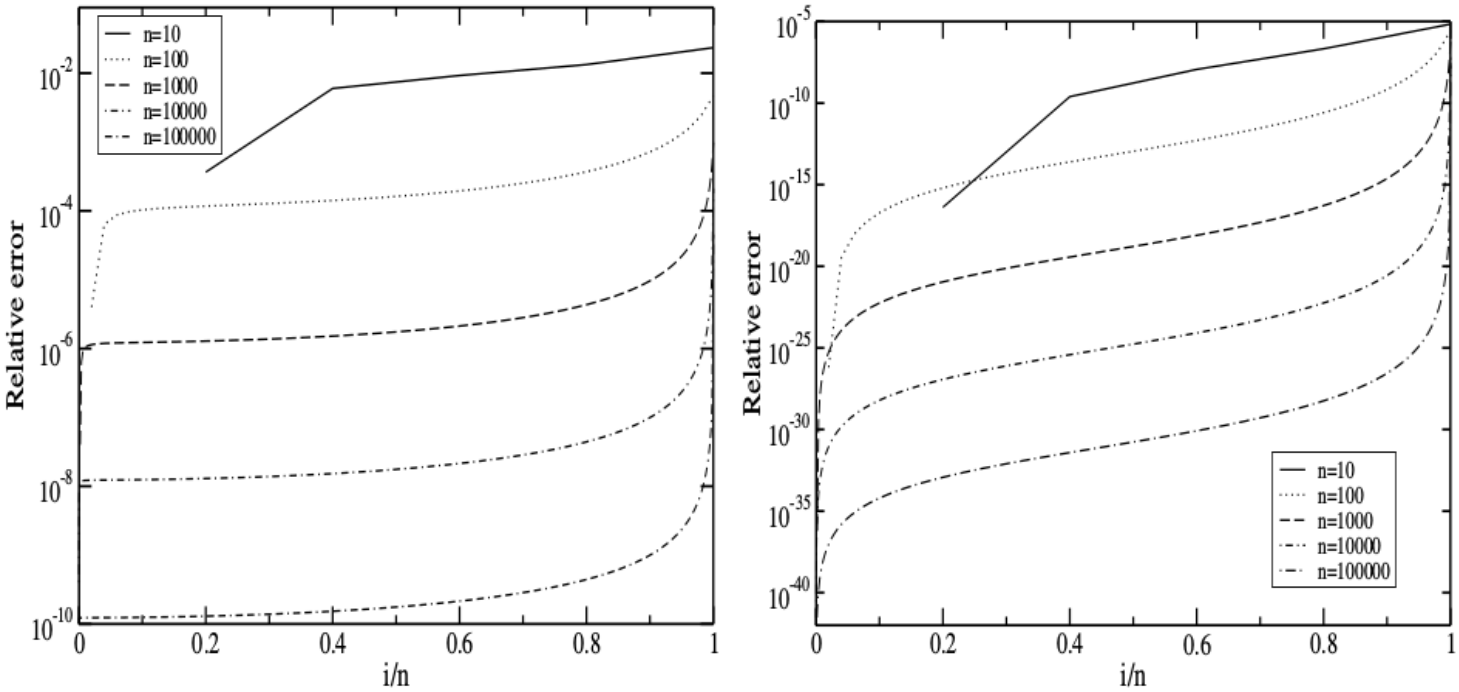
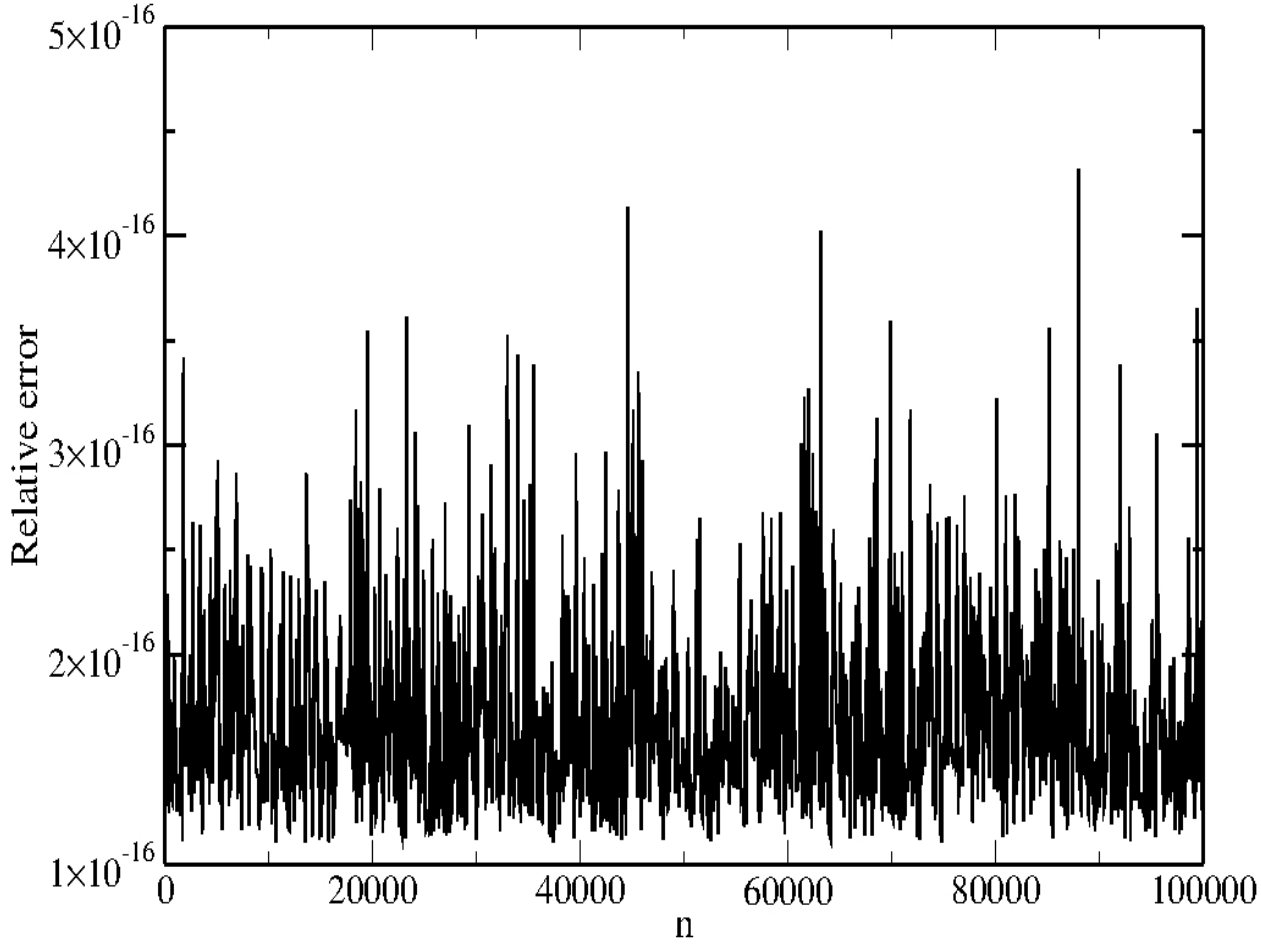
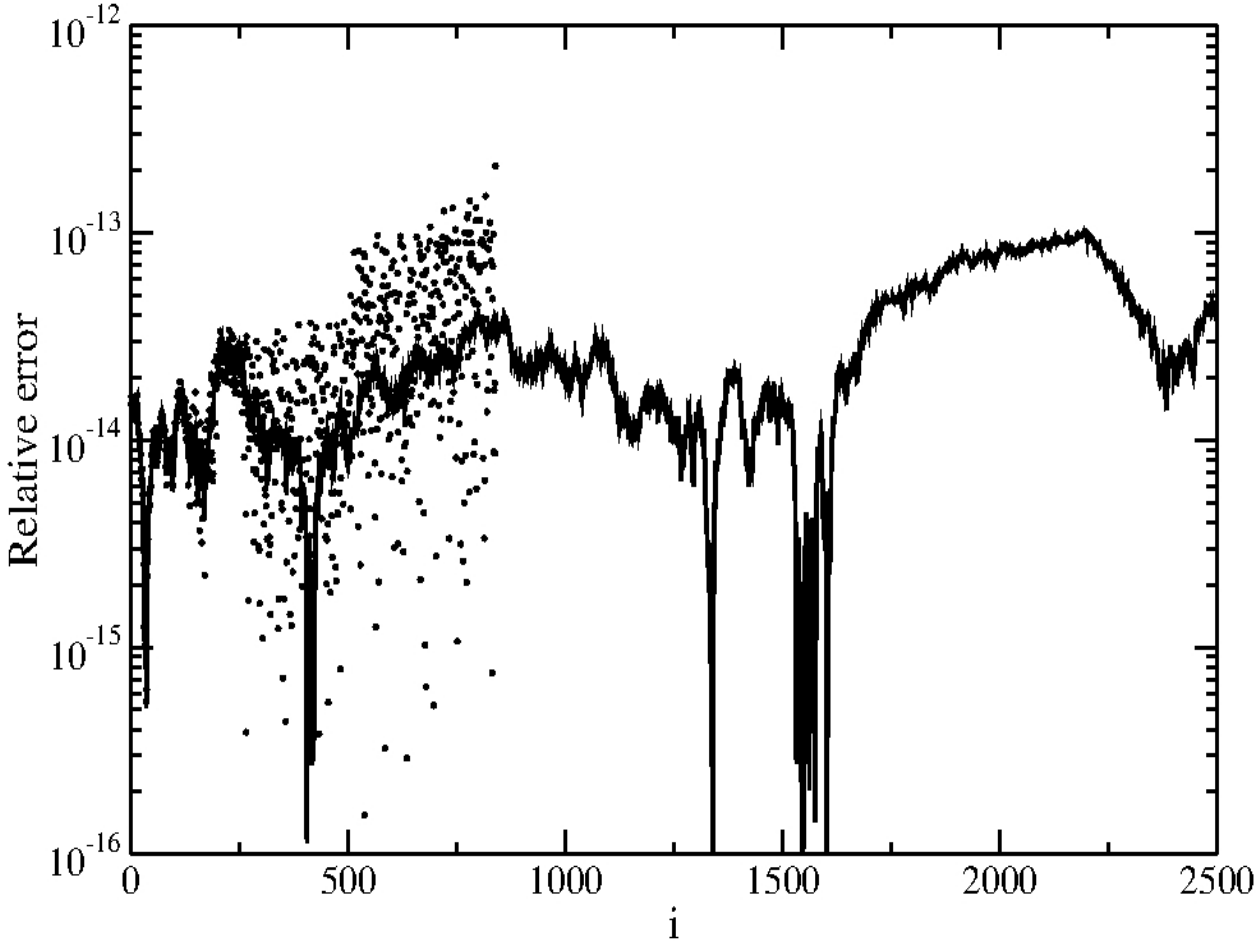


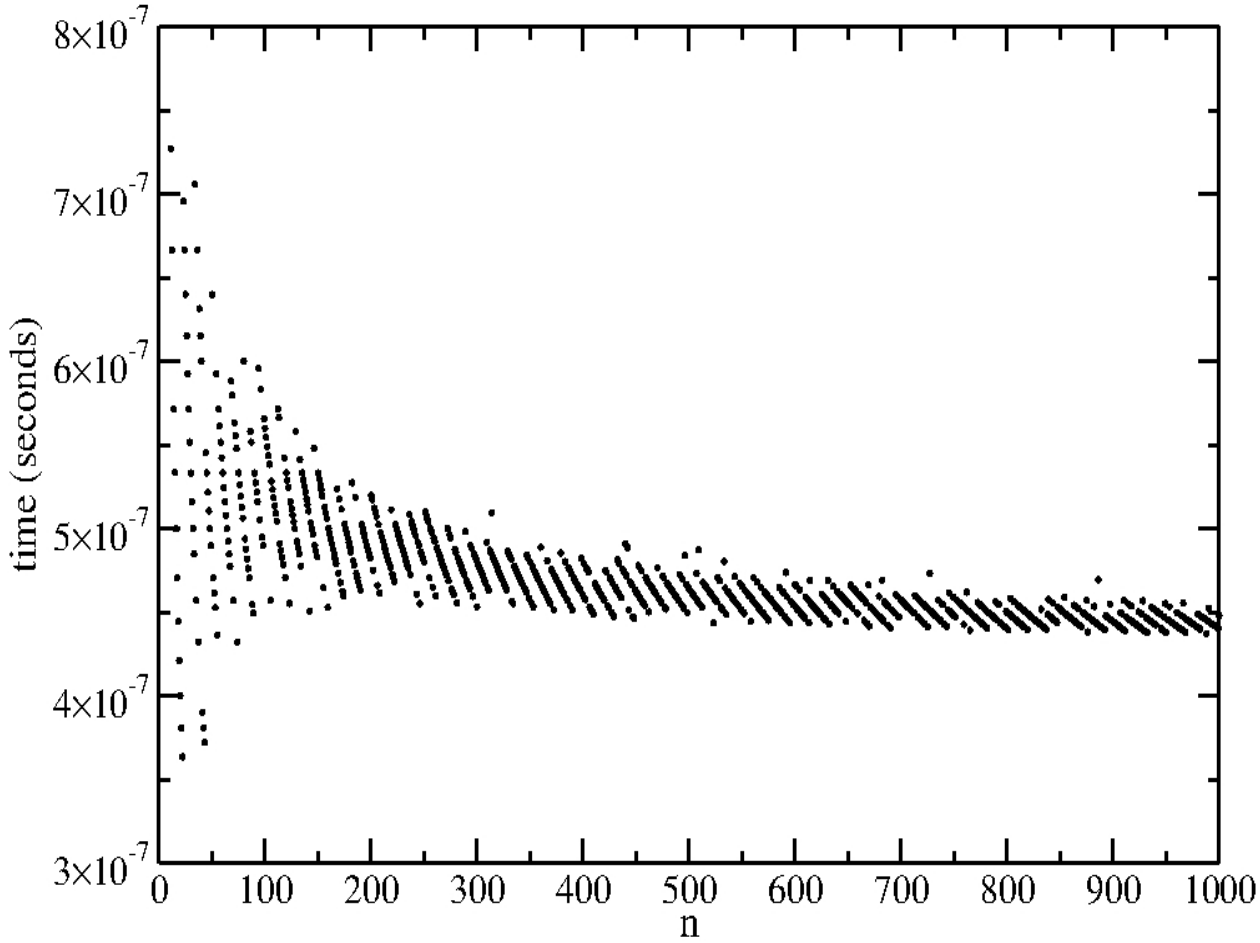
Fig. 5 Left: relative error in the first estimation for the nodes provided by the algorithm as a functions of i/n, where n is the degree and i is the index enumerating the positive nodes in increasing order, $i = 1, \ldots n/2$. Right: same but for the second estimation.



Maximum relative error in the computation of the nodes as a function of the degree n



Relative error in the computation of scaled weights (solid line) and unscaled weights (dots) for the 5000-point Gauss Hermite formula as a function of i, with i numbering the positive nodes in increasing order.



Unitary time spent (time per node and corresponding weight) in seconds as a function of the degree

Gauss-Laguerre quadrature

Take $y(z) = z^{\alpha+1/2} e^{-z^2/2} L_n^{(\alpha)}(z^2)$ which satisfies $\boldsymbol{\ddot{y}(z) + A(z)y(z) = 0}$ with

$$\boldsymbol{A(z(x)) = -x + 2(2n + \alpha + 1) + \frac{\frac{1}{4} - \alpha^2}{x}, \; x = z^2}$$

$A(x)$ for $x > 0$: decreasing if $|\alpha| \leq 1/2$.
with a maximum at $x_e = \sqrt{\alpha^2 - 1/4}$ if $|\alpha| > 1/2$.

Depence of $A(x)$ on $n \to$ asymptotic exactness.

The Gauss-Laguerre weights are

$$w_i = \frac{4\Gamma(n+\alpha+1)}{n!(\dot{y}(z_i))^2} x_i^{\alpha+\frac{1}{2}} e^{-x_i} \equiv \omega_i x_i^{\alpha+\frac{1}{2}} e^{-x_i}.$$

Scaled weights ω_i : **well conditioned as a function of the nodes**
$(\omega_i = W(z_i)$ with $\dot{W}(z_i) = 0)$

Our algorithm computes the **doubly scaled weights** $\hat{\omega}_i = \omega_i/\Gamma(\alpha+1)$.

The computation of $y(z)$ for the Laguerre case is not so easy as for Hermite.

Taylor series must be supplemented with an additional starting method.
A good choice is the continued fraction which follows from iterating

$$r^{(\alpha)} = \frac{a_\alpha}{b_\alpha + r^{(\alpha+1)}},$$

where $r^{(\alpha)} = L_n^{(\alpha)}(x)/L_n^{(\alpha-1)}(x)$, $b_\alpha = -(1+\alpha/x)$, $a_\alpha = -(n+\alpha)/x$.
Using the derivative rule

$$xL_n^{(\alpha)'}(x) = -\alpha L_n^{(\alpha)}(x) + (\alpha+n)L_n^{(\alpha-1)}(x)$$

we get

$$\frac{\dot{y}(z)}{y(z)} = \frac{1/2-\alpha}{z} - z + \frac{2(n+\alpha)}{zr^{(\alpha)}(z^2)}.$$

with $y(z)$ as defined before.

We only need this for initiating Taylor series (the ratio is enough, because we can rescale with the moment of order zero).

Taylor series

The function $y(z)$ satisfies

$$P(z)y^{(2)}(z) + Q(z)y(z) = 0,$$

with

$$P(z) = z^2, \quad Q(z) = -z^4 + 2Lz^2 + \frac{1}{4} - \alpha^2$$

taking successive derivatives and using that $P^{(n)}(z) = 0$, $n > 2$ and $Q^{(n)}(z) = 0$, $n > 4$, we obtain the following recursion formula for the derivatives with respect to z:

$$\sum_{m=0}^{2} \binom{j}{m} P^{(m)}(z)y^{(j+2-m)}(z) + \sum_{m=0}^{4} \binom{j}{m} Q^{(m)}(z)y^{(j-m)}(z) = 0,$$

where $\binom{j}{m}$ are binomial coefficients.

This is a seven-term recurrence relation and therefore the space of solutions has dimension 6.

Perron-Kreuser theorem: the solutions of this difference equation lie in two subspaces: a subspace of dimension two of solutions satisfying

$$\limsup_{n\to+\infty} |y^{(n)}/n!|^{1/n} = |1/x|,$$

and a subspace of dimension four of solutions satisfying

$$\limsup_{n\to+\infty} |y^{(n)}/\sqrt{n!}|^{1/n} = 1.$$

The solutions of the first subspace are dominant over the second subspace.

The derivatives of solutions of the ODE are in this dominant subspace because the Taylor series centered at x has radius of convergence $R = |x|$ (as corresponds to a differential equation with a singularity at $x = 0$).

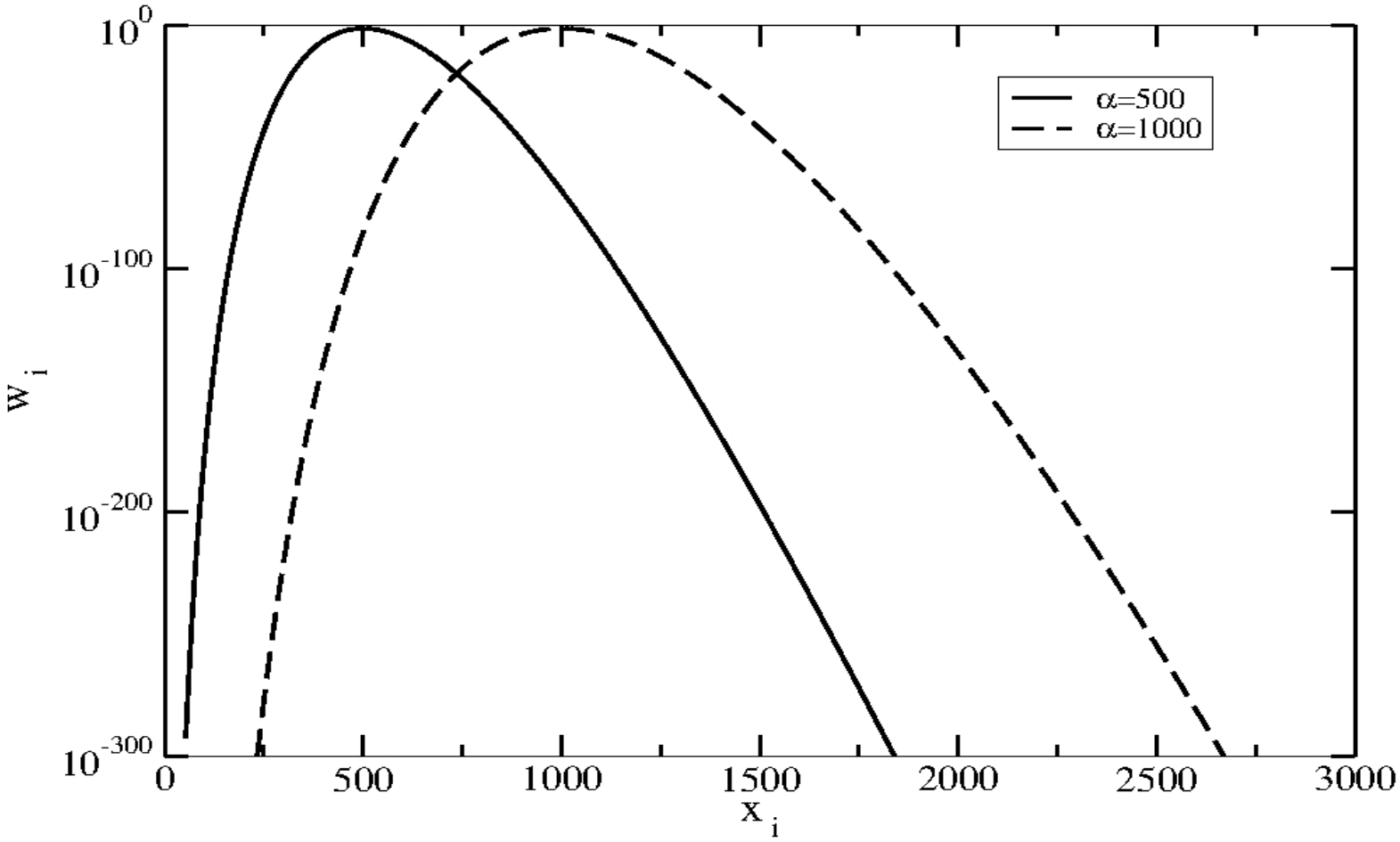
Because of the dominance of these solutions, the computation of the derivatives in the forward direction is well conditioned.

The ingredients for the method for Gauss-Laguerre are:

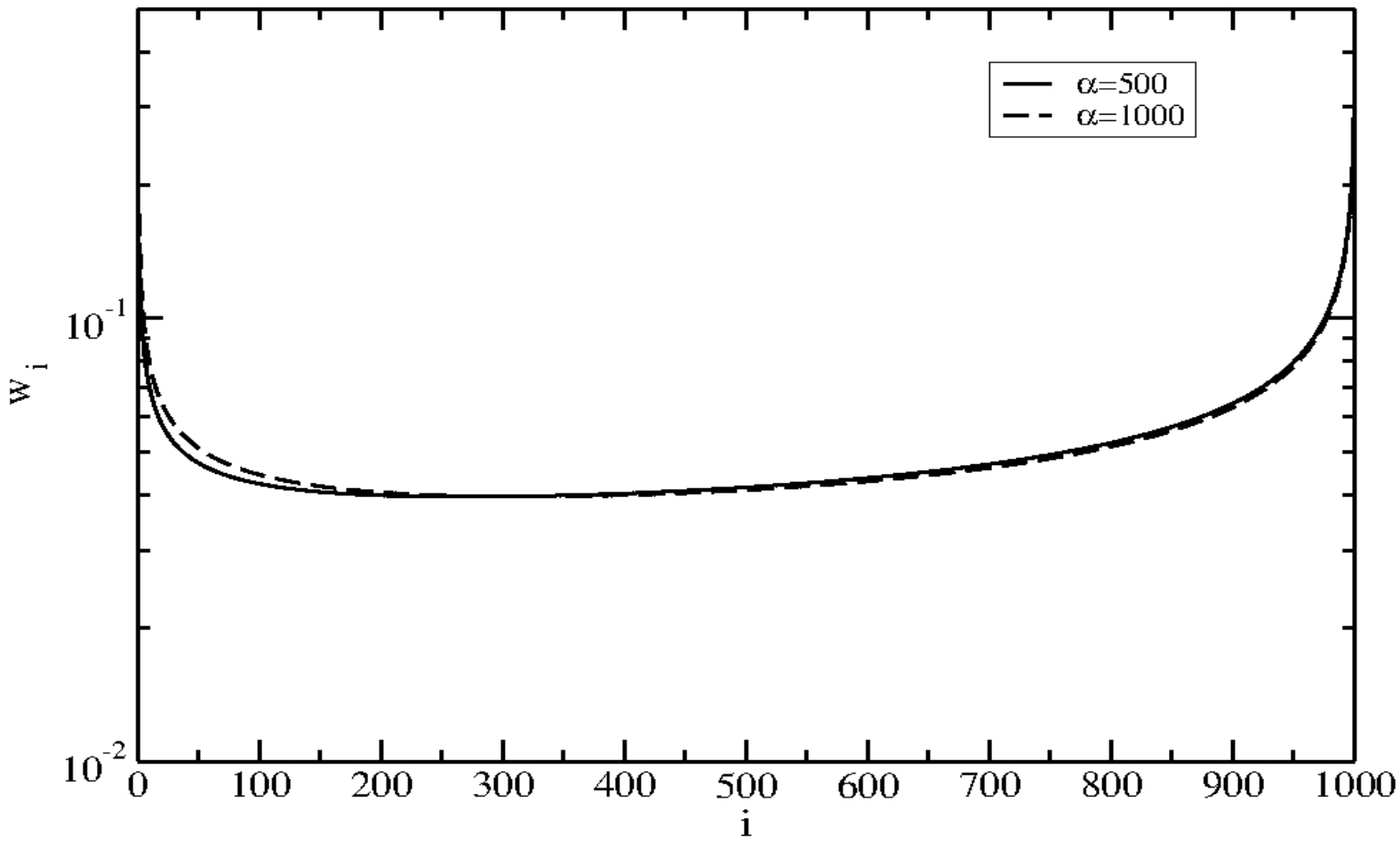
1. The fourth order fixed point method in the z variable
2. The continued fraction as a starting value (in some cases two values are required)
3. Taylor series in the z variable

In our previous notation, this is $\square$**+I+TS/CF**

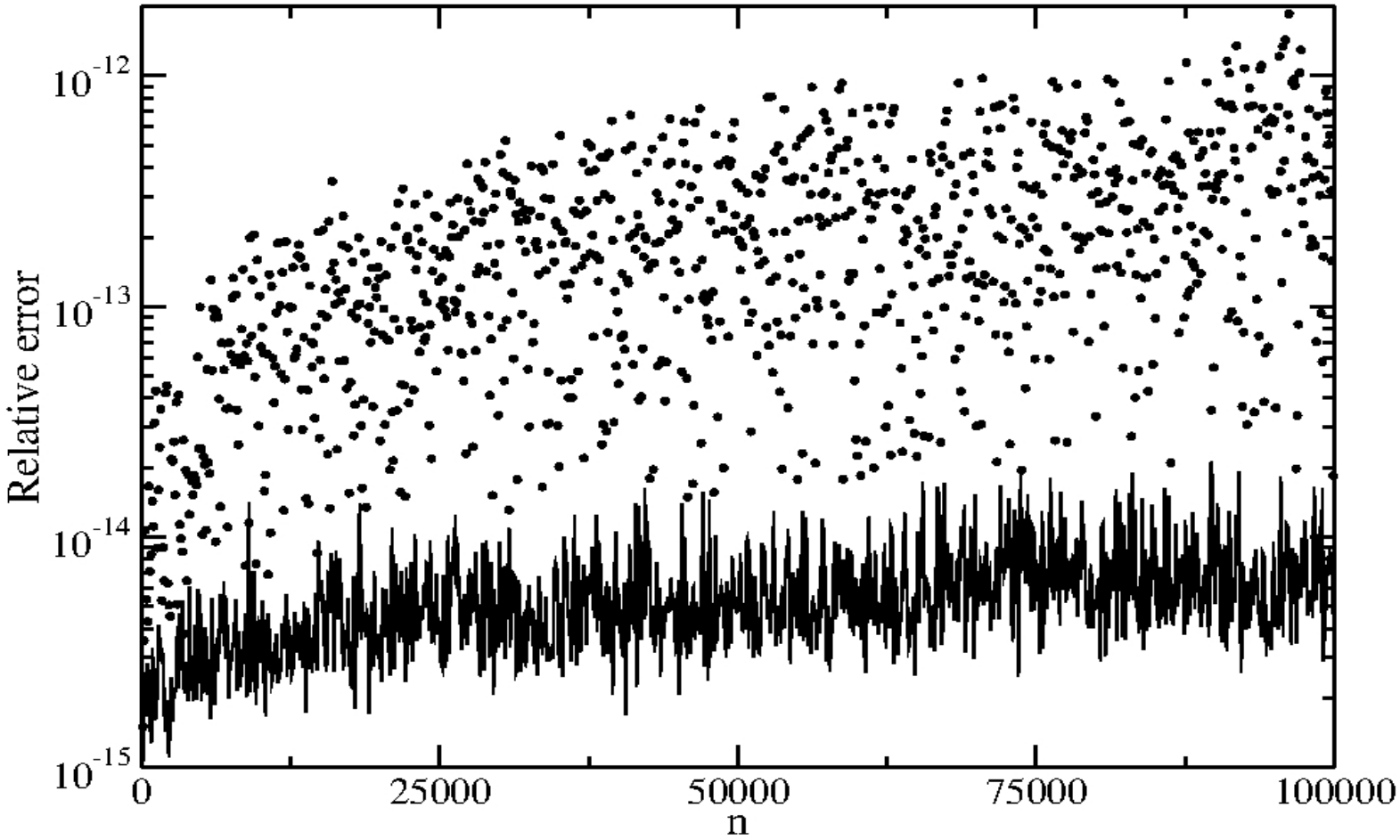
Some numerical results follow



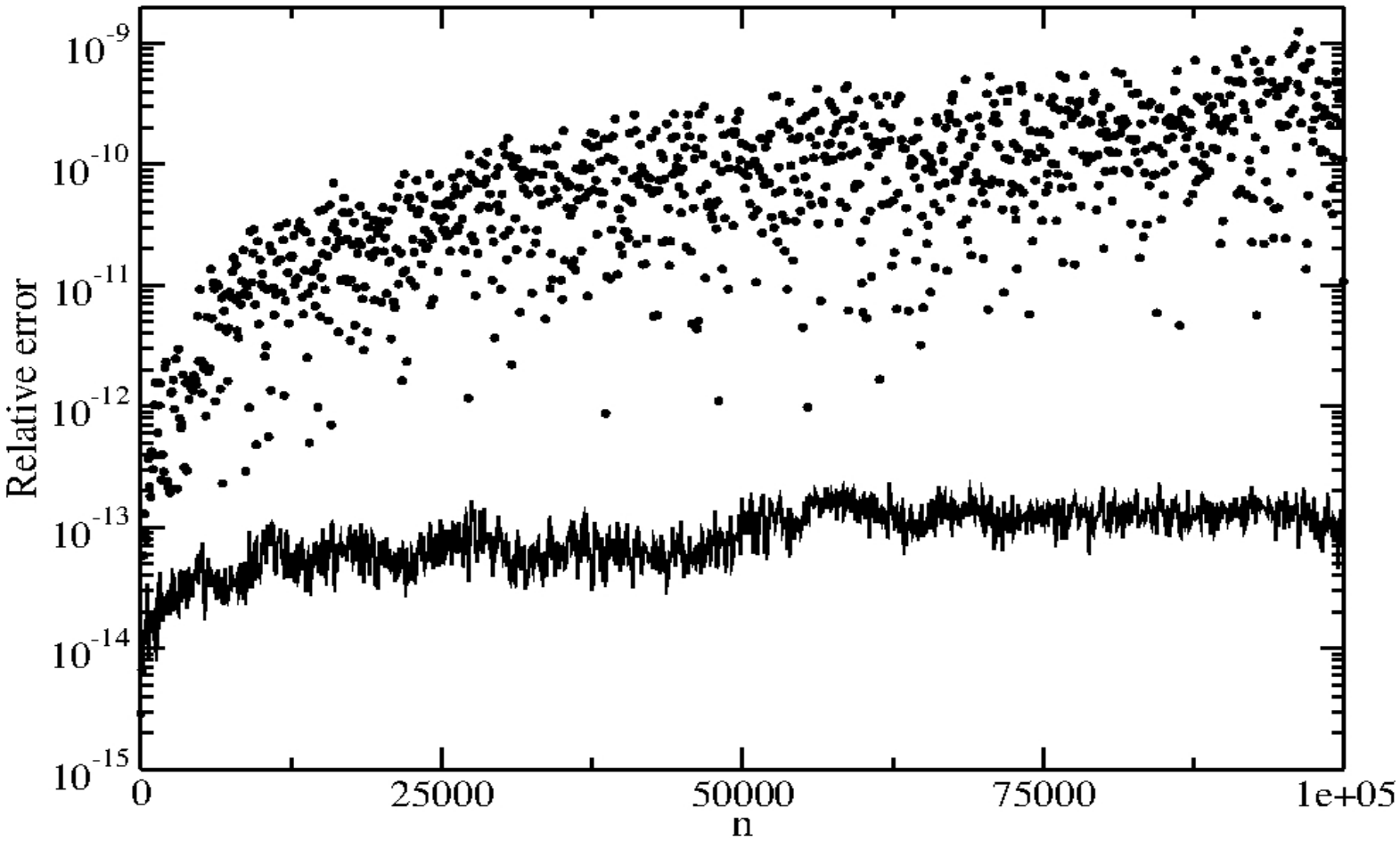
Gauss-Laguerre weights as a function of the nodes x_i, where the degree is $n = 1000$. Two values of α are considered: $\alpha = 500$ and $\alpha = 1000$



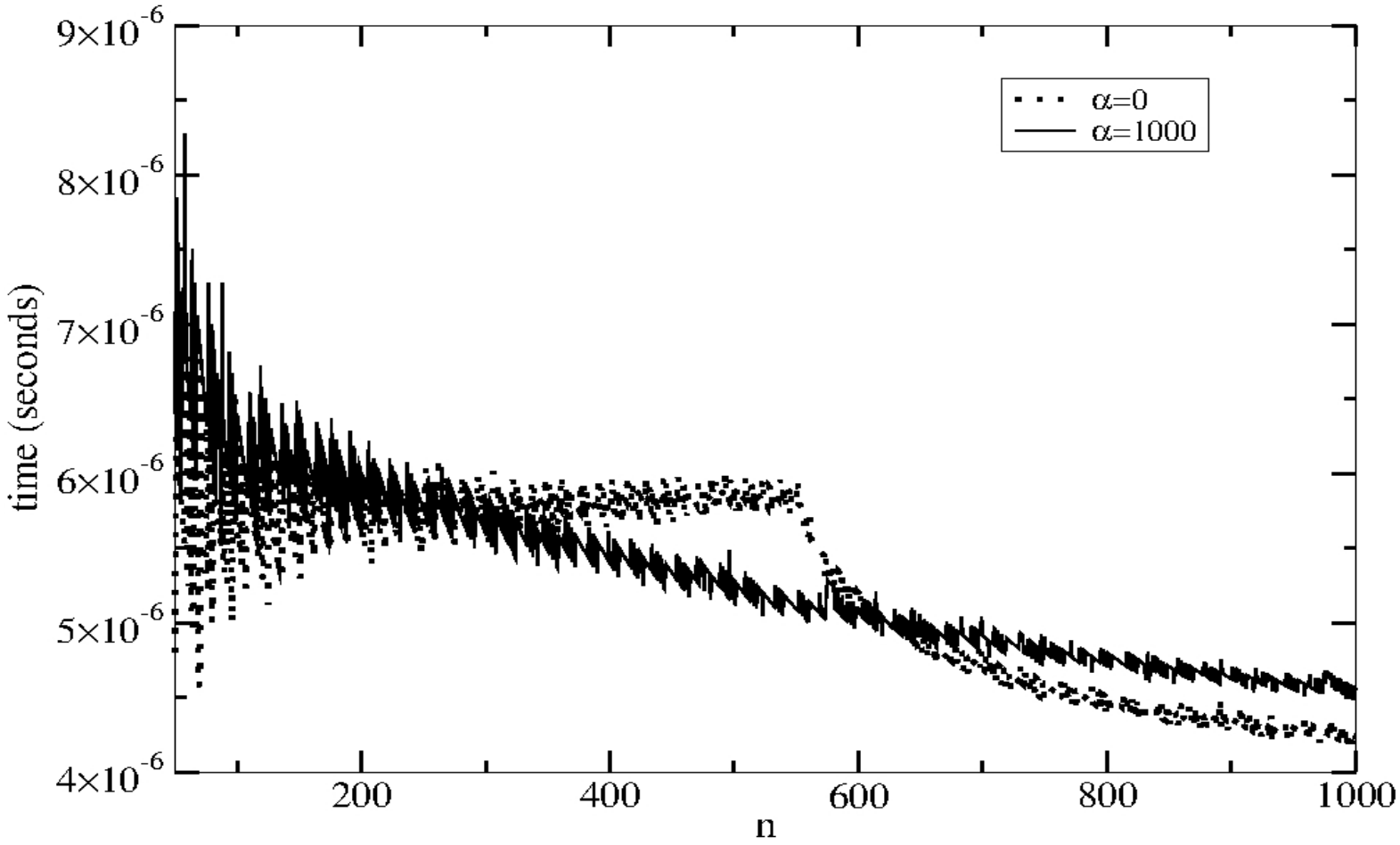
Gauss-Laguerre doubly-scaled weights as a function of the number i of the nodes x_i $(x_0 < x_1 < \cdots)$, where the degree is $n = 1000$. Two values of α are considered: $\alpha = 500$ and $\alpha = 1000$



Maximum relative errors in the computation of the nodes for n-point Gauss–Laguerre quadrature with $\alpha = 0$ (dots). We also show the maximum error when it is evaluated only for the nodes for which the weights are larger than 10^{-30} (solid line); the error in this case is smaller.



Same as the previous figure but for the scaled weights.



Unitary CPU-time spent as a function of the degree n for Gauss-Laguerre with $\alpha = 0$ and $\alpha = 1000$

Gauss-Jacobi quadrature

We only mention few details.

The most appropriate starting point is the ODE satisfied by $y(\theta) = \left(\sin\frac{\theta}{2}\right)^{\alpha+1/2}\left(\cos\frac{\theta}{2}\right)^{\beta+1/2} P_n(\cos\theta)$:

$$\frac{d^2y}{d\theta^2} + \frac{1}{4}\left[L^2 + \frac{\frac{1}{4}-\alpha^2}{\sin^2(\theta/2)} + \frac{\frac{1}{4}-\beta^2}{\cos^2(\theta/2)}\right] y = 0$$

$L = 2n + \alpha + \beta$

The change $x = \cos\theta$ is not the only possibility (see Deaño, Gil, Segura (2004)) but in this variable the method is asymptotically exact as $n \to +\infty$, and well-conditioned scaled weights are also available.

With

$$u(\theta) = M_{n,\alpha,\beta}^{-1/2} \left(\sin\frac{\theta}{2}\right)^{\alpha+1/2} \left(\cos\frac{\theta}{2}\right)^{\beta+1/2} P_n^{(\alpha,\beta)}(\cos\theta),$$

which satisfies the previous ODE, we can write the Jacobi weights as

$$w_i = |\dot{u}(\theta_i)|^{-2} \left(\sin\frac{\theta_i}{2}\right)^{2\alpha+1} \left(\cos\frac{\theta_i}{2}\right)^{2\beta+1},$$

where

$$M_{n,\alpha,\beta} = 2^{\alpha+\beta+1} \frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{n!\Gamma(n+\alpha+\beta+1)},$$

and the scaled weights can be defined as

$$\omega_i = |\dot{u}(\theta_i)|^{-2},$$

which are well conditioned as a function of θ_i.
Taylor series in the θ variable are harder, because the derivatives do not satisfy a recurrence relation with a fixed number of terms.

Asymptotic methods

Until recently, only the Gauss-Legendre quadrature was available with $15-16$ accuracy and moderately large n with asymptotic methods(Bogaert's paper). We have used different expansions in terms of other functions and zeros for computing accurately all the zeros and and weight for $n \geq 100$:

1. Hermite, in terms of: elementary functions and Airy functions.
2. Laguerre: two approximations in terms of Bessel functions and one with Airy functions.
3. Jacobi, in terms of: elementary functions and Bessel functions.

We only describe one of the simplest expansions (which is new): the elementary expansion for the Jacobi case, which can be used for computing most of the nodes and weights.

Notice: the Bogaert expansion for Gauss-Legendre is in terms of Bessel functions, and our elementary expansion for the more general Jacobi case is simpler but quite powerful.

The starting point

$$P_n^{(\alpha,\beta)}(x) = \frac{(-1)^n}{2^n n!\, w(x)} \frac{1}{2\pi i} \int_{\mathcal{C}} e^{-\kappa\phi(z)} \frac{w(z)(z-x)^{\gamma-1}}{(1-z^2)^{\gamma}}\, dz,$$

where $\gamma = \frac{1}{2}(\alpha+\beta+1)$, $\kappa = n+\gamma$, and $\phi(z) = \ln(z-x) - \ln(1-z^2)$.

$$y(\theta) = \frac{G_\kappa(\alpha,\beta)}{\sqrt{\pi\kappa}}(\cos\chi\, U(x) - \sin\chi\, V(x)),\ \chi = \kappa\theta + \left(\alpha+\frac{1}{2}\right)\frac{\pi}{2},\ x = \cos\theta$$

with expansions

$$U(x) \sim \sum_{m=0}^{\infty} \frac{u_{2m}(x)}{\kappa^{2m}}, \quad V(x) \sim \sum_{m=0}^{\infty} \frac{v_{2m+1}(x)}{\kappa^{2m+1}}.$$

The first coefficients are

$$u_0(x) = 1, \quad v_1(x) = \frac{2\alpha^2 - 2\beta^2 + (2\alpha^2+2\beta^2-1)x}{8\sin\theta},$$

$$u_2(x) = \frac{1}{384\sin^2\theta}(12(5-2\alpha^2-2\beta^2)(\alpha^2-\beta^2)x +$$
$$4(-3(\alpha^2-\beta^2)^2 + 3(\alpha^2+\beta^2) - 6 + 4\alpha(\alpha^2-1+3\beta^2) +$$
$$(-12(\alpha^2+\beta^2)(\alpha^2+\beta^2-1) - 16\alpha(\alpha^2-1+3\beta^2) - 3)x^2).$$

For computing the weights we also need $\dot{v}(\theta)$.

Let see how to compute the nodes. Let $W(\theta) = \cos\chi\, U(x) - \sin\chi\, V(x)$,

$$U(x) \sim \sum_{m=0}^{\infty} \frac{u_{2m}(x)}{\kappa^{2m}}, \quad V(x) \sim \sum_{m=0}^{\infty} \frac{v_{2m+1}(x)}{\kappa^{2m+1}}.$$

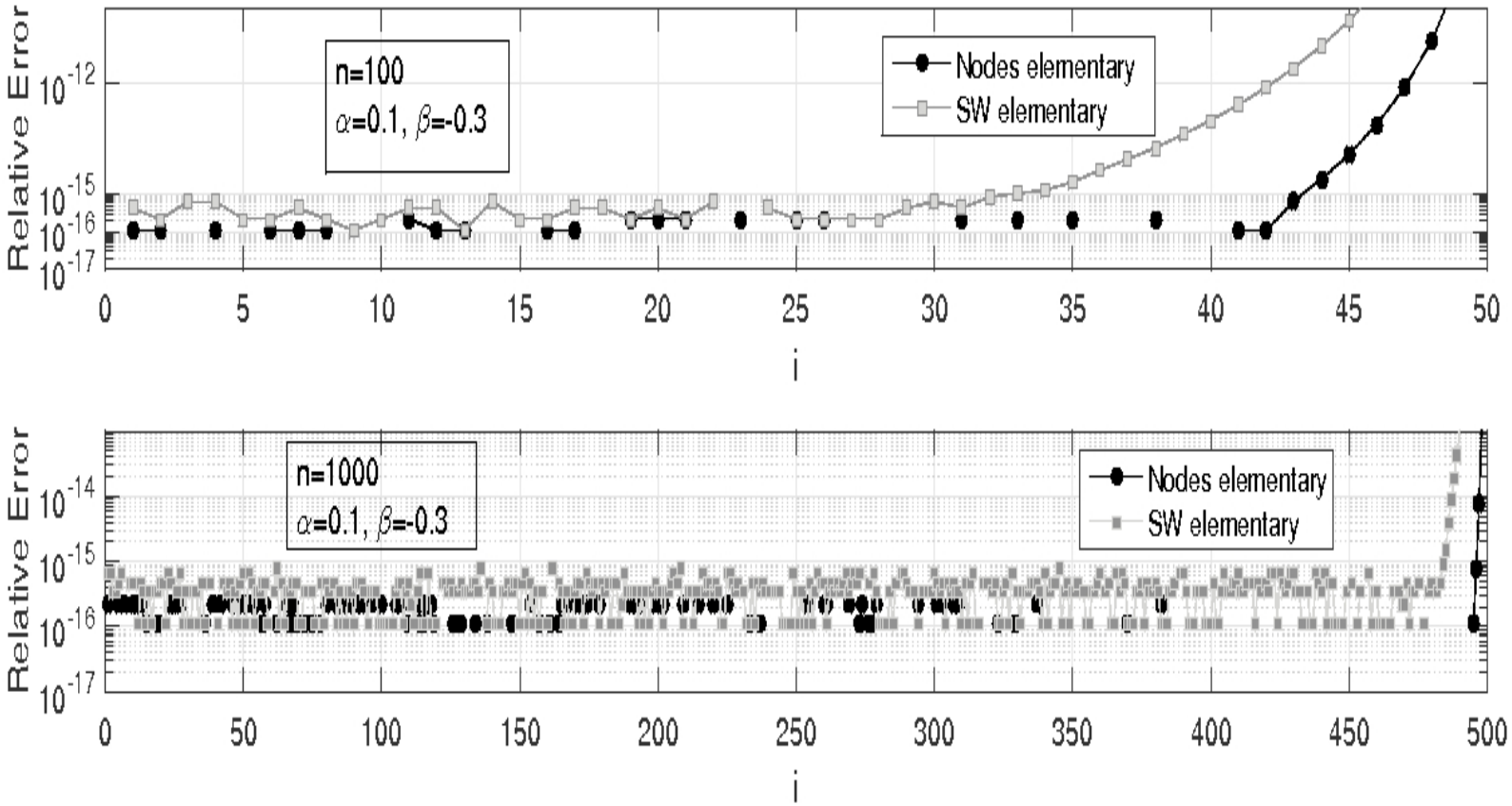
A first approximation is $\cos\chi = 0 \to \theta = \theta_0 = \left(n - k + \frac{3}{4} + \frac{\alpha}{2}\right)\frac{\pi}{\kappa}$.

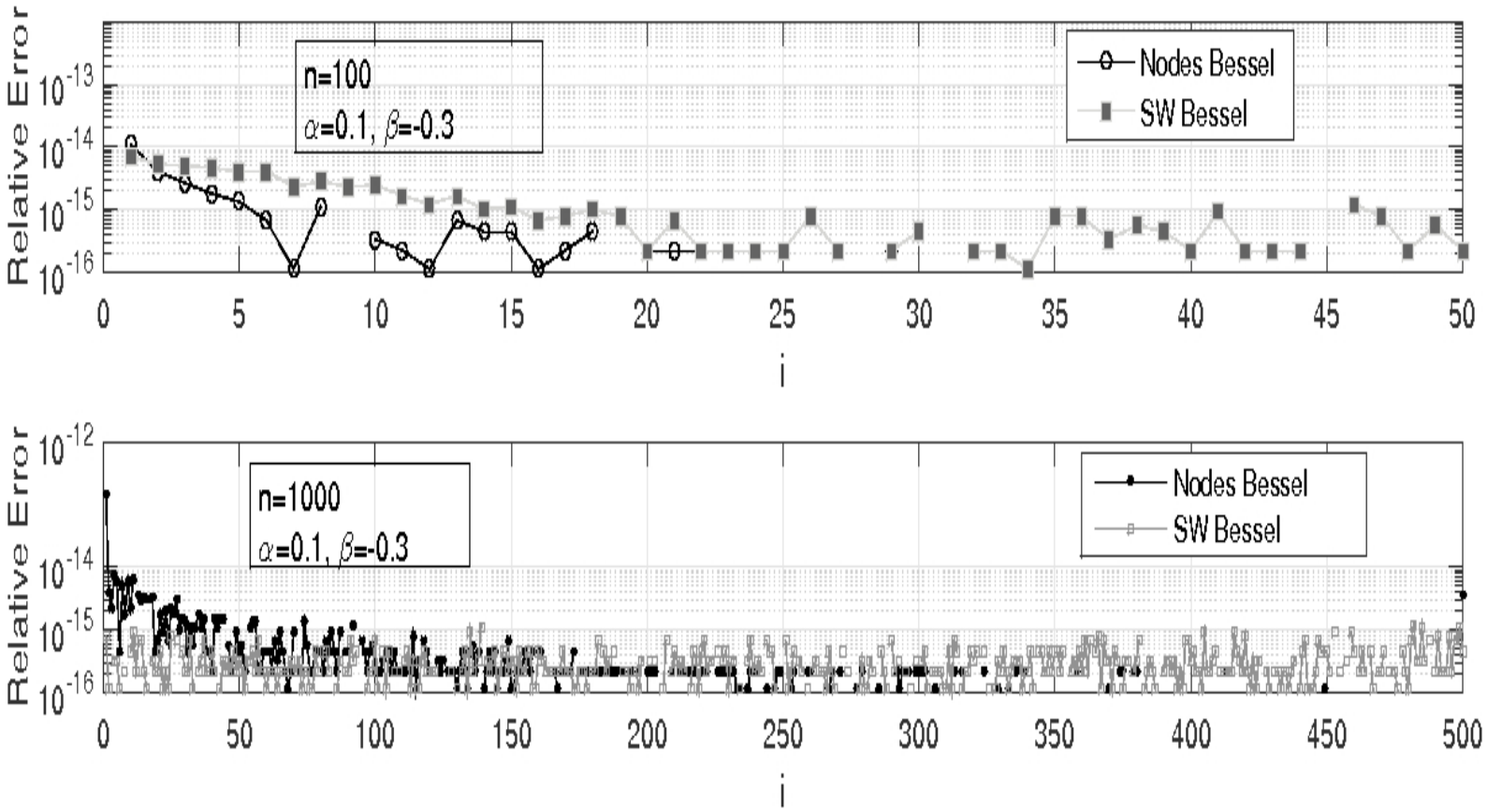
Now we write $W(\theta) = W(\theta_0 + \epsilon) = W(\theta_0) + \epsilon \dot{W}(\theta_0) + \frac{\epsilon^2}{2}\ddot{W}(\theta) + \ldots = 0$, we assume the expansion $\epsilon = \frac{\theta_1}{\kappa^2} + \frac{\theta_2}{\kappa^3} + \frac{\theta_3}{\kappa^4} + \ldots$, and using the expansions of U and V and comparing equal powers of κ we determine the coefficients θ_i, $i \geq 1$, (depending on θ_0). The first two cofficients are:

$$\theta_1 = -\frac{1}{8\sin\theta_0}\left(2\beta^2 x + 2\alpha^2 x - x - 2\beta^2 + 2\alpha^2\right),$$

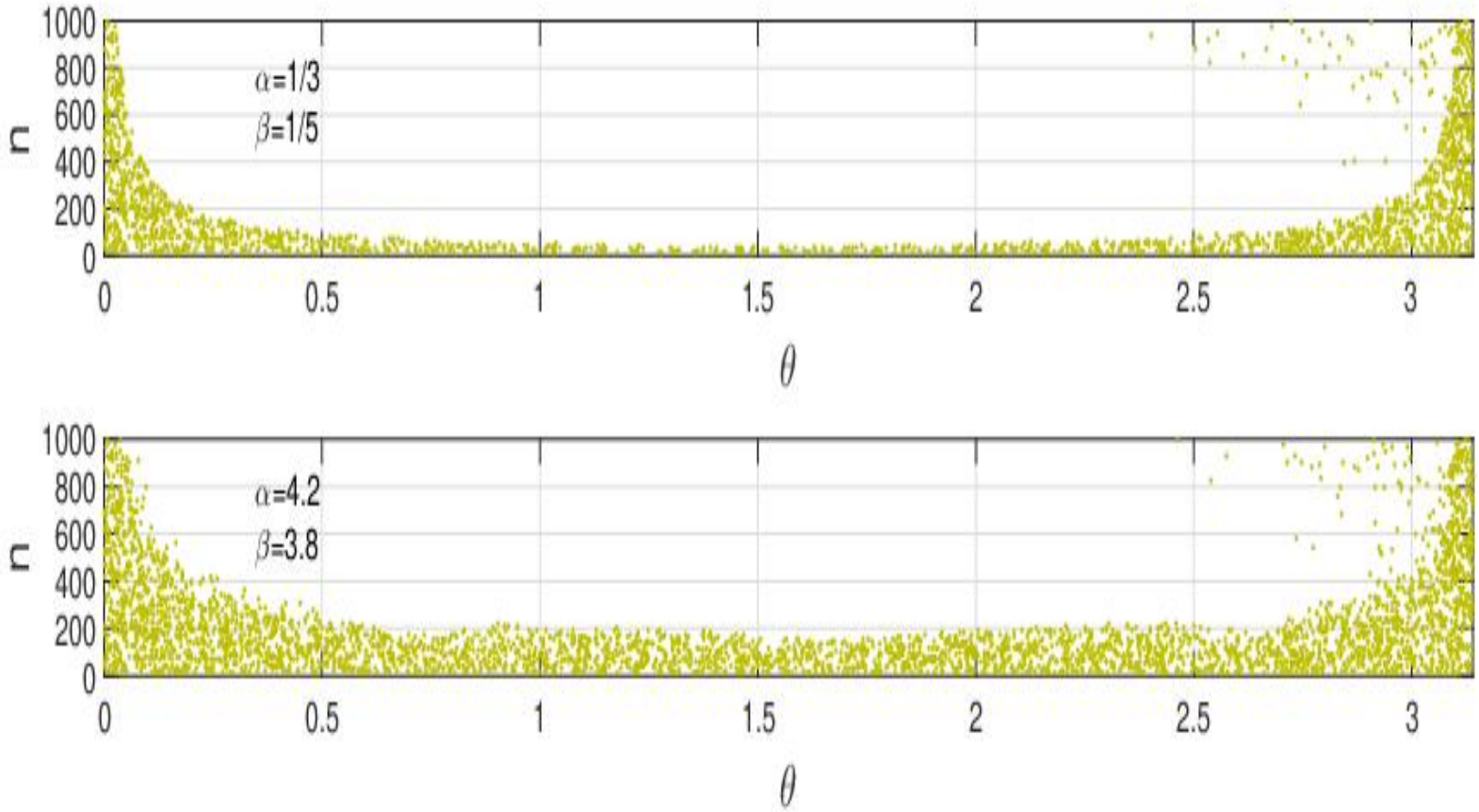
$$\begin{aligned}\theta_2 = \frac{1}{384\sin^3\theta_0}(&-33x - 36\beta^2x^2 + 36\alpha^2x^2 + 24\beta^4x^2 - 24\alpha^4x^2 + 2x^3 + \\ &84\beta^2x - 60\alpha^4x - 60\beta^4x + 84\alpha^2x + 4\beta^4x^3 + 4\alpha^4x^3 - 8\beta^2x^3 + \\ &40\alpha^2 - 8\alpha^2x^3 - 40\beta^2 + 32\beta^4 - 32\alpha^4 + 24\alpha^2\beta^2x^3 - 24\alpha^2\beta^2x),\end{aligned}$$

where $x = \cos\theta_0$.
With this we compute $\theta = \theta_0 + \epsilon$ and then $x_k \sim \cos\theta$

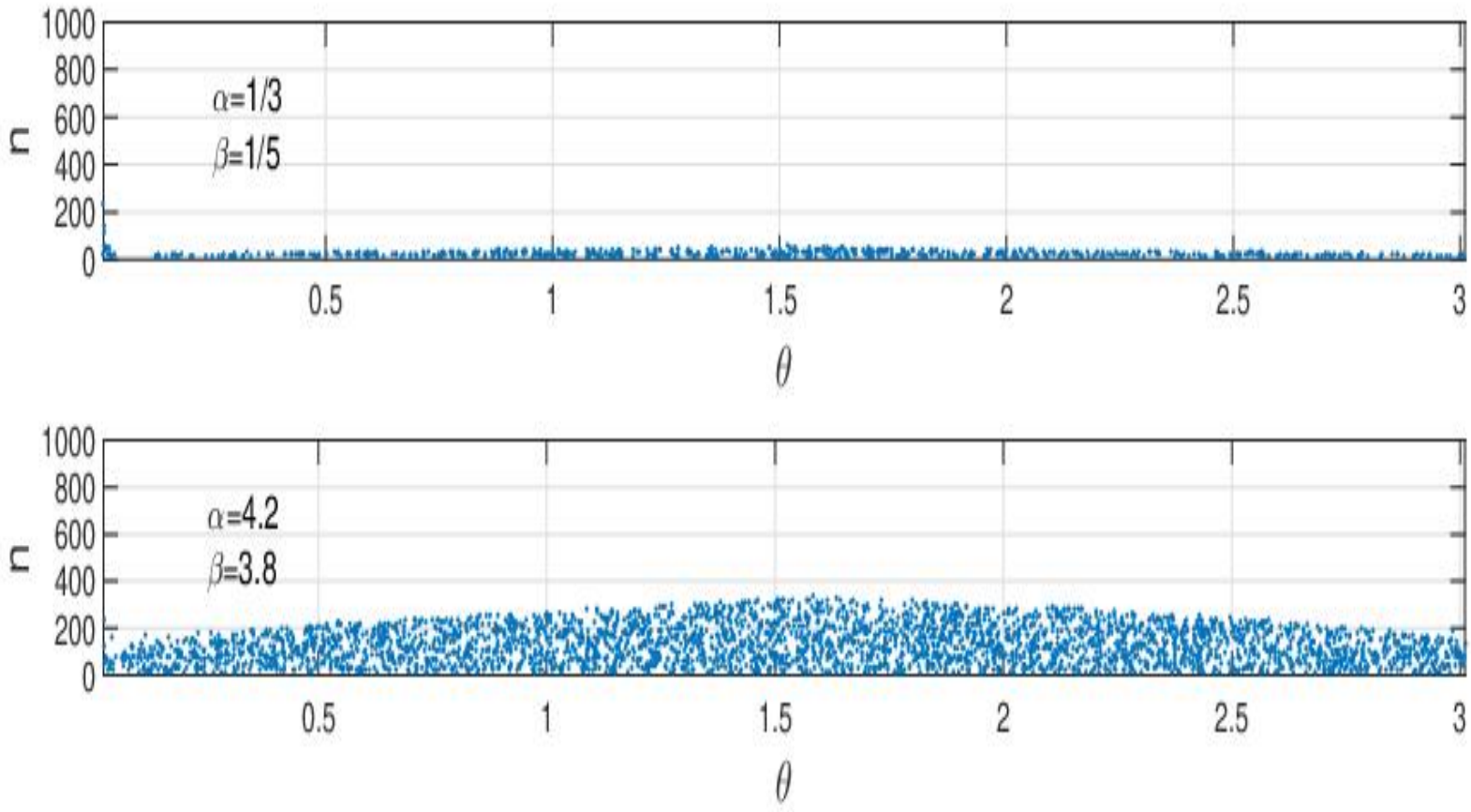

Relative Error
n=100
α=0.1, β=-0.3
Nodes elementary
SW elementary
i
Relative Error
n=1000
α=0.1, β=-0.3
Nodes elementary
SW elementary
i


Relative Error
n=100
α=0.1, β=-0.3
Nodes Bessel
SW Bessel
i
Relative Error
n=1000
α=0.1, β=-0.3
Nodes Bessel
SW Bessel
i

Asymptotic computation of Jacobi polynomials

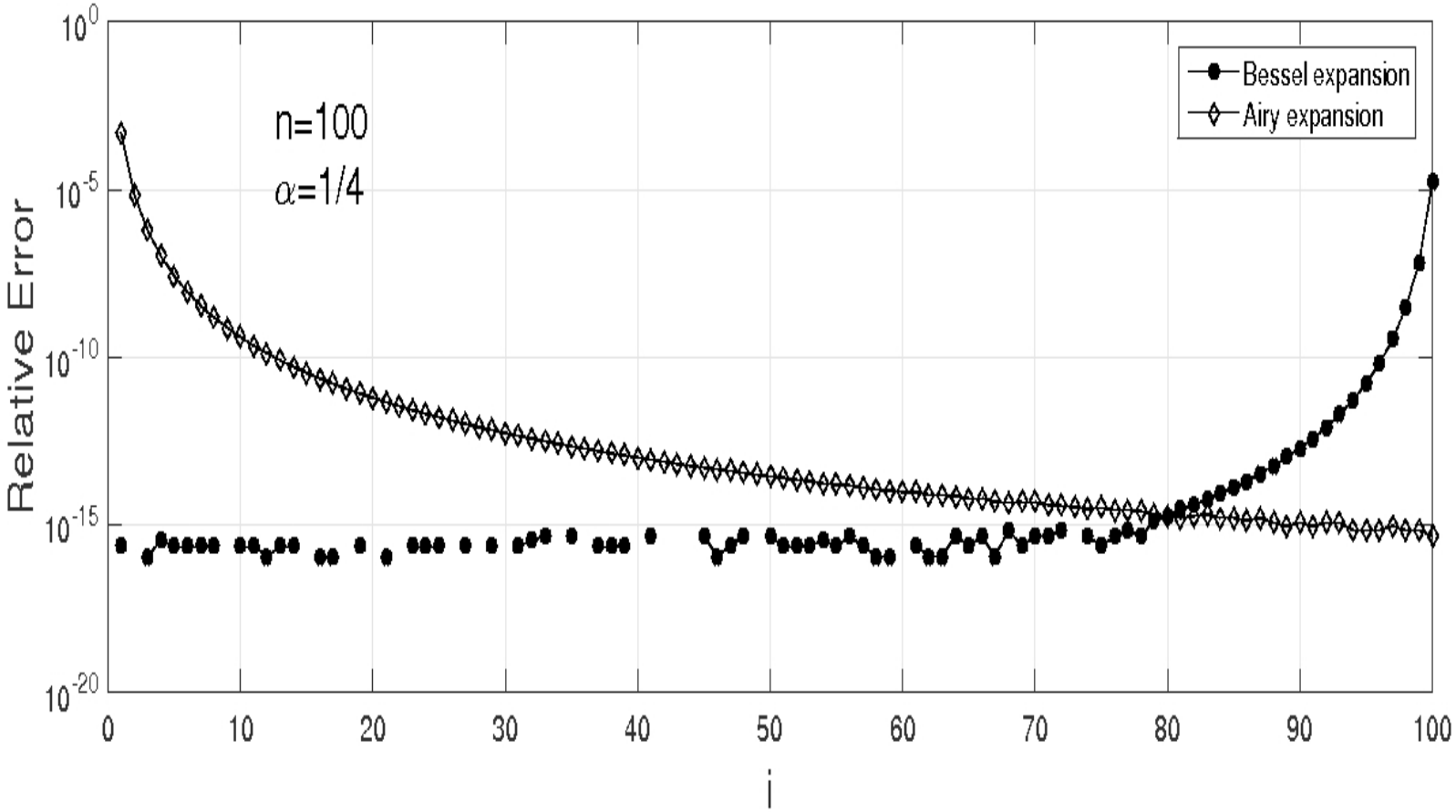


Points in the (θ, n) plane for which 10^{-12} relative accuracy is not reached in the computation of Jacobi polynomials for two pairs of values of α and β. The **elementary expansion** is used.

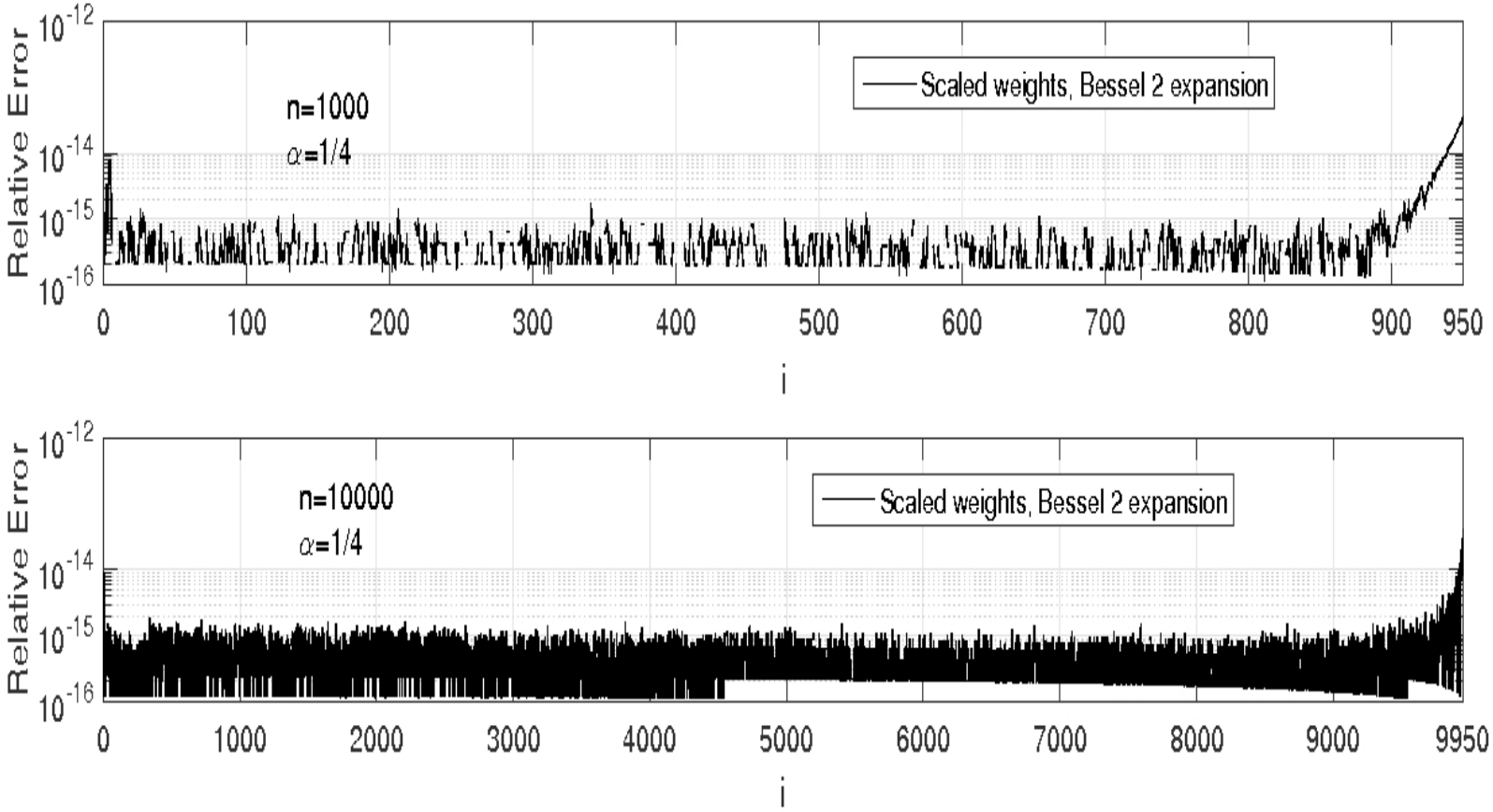


Points in the (θ, n) plane for which 10^{-12} relative accuracy is not reached in the computation of Jacobi polynomials for two pairs of values of α and β. The **Bessel-type expansion** is used.

We finish with two more plots (now for Gauss-Laguerre).



Relative accuracy for computing the zeros of $L_n^{(1/4)}(x)$ for $n = 100$ with two different asymptotic expansions. The points not shown in the plot correspond to values with all digits correct in double precision accuracy.



Relative accuracy obtained for the computation of the Laguerre doubly-scaled weights for $n = 1000, 10000$ (with $\alpha = 1/4$) using an asymptotic expansion in terms of Bessel functions.

THANK YOU!