# Efficient computation of Hankel transforms based on Levin's method

Oskar Grocholski
in collaboration with Markus Diehl

DESY Hamburg

February 17, 2023

# Motivation

TMD factorization: unpolarized structure functions expressed as

$$\tilde{W}(q_\perp) = \int \frac{d^2\mathbf{z}_\perp}{(2\pi)^2}\, e^{-i\mathbf{z}_\perp\cdot\mathbf{q}_\perp} W(z_\perp) = \int_0^\infty \frac{dz_\perp}{2\pi} J_0(q_\perp z_\perp) z_\perp W(z_\perp)\,. \quad (1)$$

$W(z_\perp)$ – product of TMDs and FFs.
Polarized structure functions $\rightarrow$ also integrals involving $J_1, J_2$.

# Motivation

TMD factorization: unpolarized structure functions expressed as

$$\tilde{W}(q_\perp) = \int \frac{d^2\mathbf{z}_\perp}{(2\pi)^2} e^{-i\mathbf{z}_\perp\cdot\mathbf{q}_\perp} W(z_\perp) = \int_0^\infty \frac{dz_\perp}{2\pi} J_0(q_\perp z_\perp) z_\perp W(z_\perp)\,. \quad (1)$$

$W(z_\perp)$ – product of TMDs and FFs.
Polarized structure functions $\rightarrow$ also integrals involving $J_1, J_2$.

Adaptative quadrature methods use different grid points in $z_\perp$, depending on $q_\perp \implies$ if computation of $W(z)$ becomes costly, then computation of every point $\tilde{W}(q_\perp)$ becomes proportionally longer.

$\implies$ Find a method that can use a fixed grid in a wide range of $q_\perp$ !

# Levin's method: the general idea

D. Levin *Fast integration of rapidly oscillatory functions*,
J. of Computational and Applied Mathematics 67 (1996) 95-101

Rewrite the integral as

$$\int_{z_0}^{z_1} \vec{\omega} \cdot \vec{g}\, dz, \tag{2}$$

with the quickly oscillating part $\vec{\omega}$ such that

$$\frac{d}{dz}\vec{\omega} = A^T \vec{\omega}, \tag{3}$$

e.g.

$$\vec{\omega}(z) = \left[J_\nu(qz),\ J_{\nu+1}(qz)\right]^T, \quad \vec{g}(z) = \left[f(z),\ 0\right]^T.$$

# Levin's method: the general idea

Find a function $\vec{h}(z)$ such that

$$\left(\frac{d}{dz} + A\right)\vec{h} = \vec{g}, \tag{4}$$

then

$$\frac{d}{dz}\left(\vec{h}\cdot\vec{\omega}\right) = \left(\frac{d}{dz}\vec{h}\right)\cdot\vec{\omega} + \vec{h}\cdot\left(A^T\vec{\omega}\right) = \vec{g}\cdot\vec{\omega}. \tag{5}$$

# Levin's method: the general idea

Find a function $\vec{h}(z)$ such that

$$\left(\frac{d}{dz} + A\right)\vec{h} = \vec{g}, \tag{4}$$

then

$$\frac{d}{dz}(\vec{h} \cdot \vec{\omega}) = \left(\frac{d}{dz}\vec{h}\right) \cdot \vec{\omega} + \vec{h} \cdot \left(A^T \vec{\omega}\right) = \vec{g} \cdot \vec{\omega}. \tag{5}$$

The integral can be easily computed:

$$\int_{z_0}^{z_1} \vec{\omega} \cdot \vec{g} \, dz = \vec{h}(z_1) \cdot \vec{\omega}(z_1) - \vec{h}(z_0) \cdot \vec{\omega}(z_0). \tag{6}$$

# Levin's method: the general idea

One can choose

$$\vec{\omega}(z) = \begin{bmatrix} J_\nu(qz) \\ J_{\nu+1}(qz) \end{bmatrix}, \quad \vec{g}(z) = \begin{bmatrix} f(z) \\ 0 \end{bmatrix}. \tag{7}$$

In that case:

$$A = \begin{bmatrix} \nu/z & -q \\ q & -(\nu+1)/z \end{bmatrix}. \tag{8}$$

Side remark: integral with $J_{\nu+1}$ $\rightarrow$ just use $\vec{g} = \left[0,\ f(z)\right]^T$.

# Levin's method: application to Hankel transform

Obtained system of differential equations:

$$\left(\frac{d}{dz} + \begin{bmatrix} \nu/z & -q \\ q & -(\nu+1)/z \end{bmatrix}\right) \begin{bmatrix} h_1(z) \\ h_2(z) \end{bmatrix} = \begin{bmatrix} f(z) \\ 0 \end{bmatrix}. \tag{9}$$

✗ Singularity at $z = 0$ !

# Levin's method: singularity at $z = 0$

Make a different splitting of the oscillatory and non-oscillatory part of the integral:

$$\int_0^\infty J_\nu(qz)f(z)\,dz = \int_0^\infty \Big(z^{-\nu} J_\nu(qz)\Big) z^\nu f(z)\,dz. \tag{10}$$

# Levin's method: singularity at $z = 0$

Make a different splitting of the oscillatory and non-oscillatory part of the integral:

$$\int_0^\infty J_\nu(qz)f(z)\,dz = \int_0^\infty \Big(z^{-\nu}J_\nu(qz)\Big)z^\nu f(z)\,dz. \tag{10}$$

Use the rescaled vector of oscillatory functions:

$$\vec{\omega}_2 = \begin{bmatrix} z^{-\nu}J_\nu\big(qz\big) \\ \\ z^{-\nu}J_{\nu+1}\big(qz\big) \end{bmatrix}, \qquad \lim_{z\to 0}\vec{\omega}_2(z) \text{ is finite.} \tag{11}$$

# Levin's method: singularity at $z = 0$

Make a different splitting of the oscillatory and non-oscillatory part of the integral:

$$\int_0^\infty J_\nu(qz) f(z)\, dz = \int_0^\infty \Big(z^{-\nu} J_\nu(qz)\Big) z^\nu f(z)\, dz. \tag{10}$$

Use the rescaled vector of oscillatory functions:

$$\vec{\omega}_2 = \begin{bmatrix} z^{-\nu} J_\nu\big(qz\big) \\ \\ z^{-\nu} J_{\nu+1}\big(qz\big) \end{bmatrix}, \qquad \lim_{z\to 0} \vec{\omega}_2(z) \text{ is finite.} \tag{11}$$

The resulting matrix $A_2$ is

$$A_2 = \begin{bmatrix} 0 & -q \\ \\ q & -(2\nu+1)/z \end{bmatrix}. \tag{12}$$

# Levin's method: singularity at $z = 0$

Introduce $z\tilde{h}_3 = \tilde{h}_2$, to remove the $1/z$ factor.
$\nu \geq 1 \implies \tilde{h}_3$ is well-behaved at $z = 0$.

# Levin's method: singularity at $z = 0$

Introduce $z\tilde{h}_3 = \tilde{h}_2$, to remove the $1/z$ factor.
$\nu \geq 1 \implies \tilde{h}_3$ is well-behaved at $z = 0$.

Differential equations for the rescaled functions:

$$\begin{cases} z^\nu f(z) & = & \frac{d}{dz}\tilde{h}_1(z) - qz\tilde{h}_3, \\ 0 & = & z\frac{d}{dz}\tilde{h}_3 - 2\nu\tilde{h}_3 + q\tilde{h}_1. \end{cases} \tag{13}$$

Remark: for larger $z$, it is better to use $\frac{z}{z+1}\tilde{h}_3 = \tilde{h}_2$
$\implies$ longer formulas, but the method is the same.

In fact, one takes also $z/(1+z)$ instead of $z$ in $\omega_2$.

# The case $0 \leq \nu < 1$

Can integrate by parts:

$$\int_{z_0}^{z_1} J_\nu(qz)f(z)\,dz = \frac{1}{q}J_{\nu+1}(qz)f(z)\Big|_{z_0}^{z_1} - \frac{1}{q}\int_{z_0}^{z_1} z^\nu\Big(\frac{d}{dz}(zf(z)) - (\nu+2)f(z)\Big)z^{-(\nu+1)}J_{\nu+1}(qz). \tag{14}$$

# Infinite interval

In general, one can make a variable transformation:

$$\int_0^\infty f(z)J_\nu(qz)\,dz = \int_{-1}^1 \left(\frac{dz}{du}\right) f\big(z(u)\big) J_\nu\big(z(u)\big)\,du. \tag{15}$$

The resulting equation for $\tilde{h}(u)$ reads:

$$\begin{bmatrix} f\big(z(u)\big) \\ 0 \end{bmatrix} = \left(\left(\frac{du}{dz}\right)\frac{d}{du} + A\right)\begin{bmatrix} \tilde{h}_1(u) \\ \tilde{h}_2(u) \end{bmatrix} \tag{16}$$

# Chebyshev pseudospectral method

$p_1, p_3$ - Chebyshev polynomials of order $N-1$ approximating $\tilde{h}_{1,3}$.

$$u_j = \cos\left(\frac{j\pi}{N}\right), \quad j \in \{0, ..., N-1\} \qquad \text{interpolation points,} \tag{17}$$

$$p_{1,3}(u_j) = \tilde{h}_{1,3}(u_j), \qquad f(z_j) = f\left(z(u_j)\right), \tag{18}$$

$$\frac{d}{du} p(u_j) = \sum_{k=0}^{N-1} D_{jk} p(u_k) \approx \frac{d}{du} \tilde{h}(u_j). \tag{19}$$

$D$ - Chebyshev differentiation matrix.

# Chebyshev pseudospectral method

Find the approximate solution by discretizing the system:

Let $z_j = z(u_j)$,

$$\begin{cases} z_j^{\nu} f\left(z_j\right) & = & \left(\frac{du}{dz}\right) \sum_{jk} p_1(u_k) - q z_j p_3(u_j), \\ 0 & = & z_j \left(\frac{du}{dz}\right) \sum_{jk} p_3(u_k) - 2\nu p_3(u_j) + q p_1(u_j). \end{cases} \tag{20}$$

# Chebyshev pseudospectral method

Find the approximate solution by discretizing the system:

Let $z_j = z(u_j)$,

$$\begin{cases} z_j^{\nu} f(z_j) & = \left(\frac{du}{dz}\right) \sum_{jk} p_1(u_k) - q z_j p_3(u_j), \\ 0 & = z_j \left(\frac{du}{dz}\right) \sum_{jk} p_3(u_k) - 2\nu p_3(u_j) + q p_1(u_j). \end{cases} \tag{20}$$

Let:

$$\vec{F} = \left[z_0^{\nu} f(z_0), ..., z_{N-1}^{\nu} f(z_{N-1}), 0, ...., 0\right]^T, \tag{21}$$

$$\vec{P} = \left[p_1(u_0), ..., p_1(u_{N-1}), p_3(u_0), ..., p_3(u_{N-1})\right]^T, \tag{22}$$

$$B\vec{P} = \vec{F}. \tag{23}$$

# Chebyshev pseudospectral method

$\longrightarrow 2N$ equations.

$\longrightarrow$ Need to find appropriate variable transformation (so that one can interpolate the integrand and the solution well on the resulting grid).

M. Diehl, R. Nagar, F. J. Tackmann, *ChiliPDF: Chebyshev Interpolation for Parton Distributions*, [arXiv:2112.09703] – efficient use of Chebyshev grids, many kinds of variable transformation implemented.

# Technical remarks

For $\sim 50$ nodes the best accuracy is obtained when splitting the integral into 2 parts:

$$\int_0^\infty J_\nu(qz) f(z)\, dz = \int_0^{z_1} J_\nu(qz) f(z)\, dz + \int_{z_1}^\infty J_\nu(qz) f(z)\, dz. \tag{24}$$

# Technical remarks

For $\sim 50$ nodes the best accuracy is obtained when splitting the integral into 2 parts:

$$\int_0^\infty J_\nu(qz) f(z)\, dz = \int_0^{z_1} J_\nu(qz) f(z)\, dz + \int_{z_1}^\infty J_\nu(qz) f(z)\, dz. \tag{24}$$

- $qz_1$ smaller then the first zero of $J_\nu \implies$ can integrate on the first subinterval using the Clenshaw-Curtis quadrature.

# Technical remarks

For $\sim 50$ nodes the best accuracy is obtained when splitting the integral into 2 parts:

$$\int_0^\infty J_\nu(qz) f(z)\, dz = \int_0^{z_1} J_\nu(qz) f(z)\, dz + \int_{z_1}^\infty J_\nu(qz) f(z)\, dz. \tag{24}$$

- $qz_1$ smaller then the first zero of $J_\nu \implies$ can integrate on the first subinterval using the Clenshaw-Curtis quadrature.
- larger $qz_1 \implies$ construct the matrix $B$ needed to solve (20).

# Technical remarks

For $\sim 50$ nodes the best accuracy is obtained when splitting the integral into 2 parts:

$$\int_0^\infty J_\nu(qz) f(z)\, dz = \int_0^{z_1} J_\nu(qz) f(z)\, dz + \int_{z_1}^\infty J_\nu(qz) f(z)\, dz. \tag{24}$$

- $qz_1$ smaller then the first zero of $J_\nu \implies$ can integrate on the first subinterval using the Clenshaw-Curtis quadrature.
- larger $qz_1 \implies$ construct the matrix $B$ needed to solve (20).
  - First, compute the LU decomposition of $B$.

# Technical remarks

For $\sim 50$ nodes the best accuracy is obtained when splitting the integral into 2 parts:

$$\int_0^\infty J_\nu(qz) f(z)\, dz = \int_0^{z_1} J_\nu(qz) f(z)\, dz + \int_{z_1}^\infty J_\nu(qz) f(z)\, dz. \tag{24}$$

- $qz_1$ smaller then the first zero of $J_\nu \implies$ can integrate on the first subinterval using the Clenshaw-Curtis quadrature.
- larger $qz_1 \implies$ construct the matrix $B$ needed to solve (20).
  - First, compute the LU decomposition of $B$.
  - If $B$ is ill-conditioned (zero or small elements on diagonal), use the singular value decomposition (SVD) of $B$.

# Benchmarking the precision

Compare with precision obtained using optimised Ogata quadrature: Kang:2019ctl [arXiv:1906.05949v2].

# Benchmarking the precision

Compare with precision obtained using optimised Ogata quadrature: Kang:2019ctl [arXiv:1906.05949v2].

"Toy function" used in the cited work:

$$W(z) = \frac{1}{z}\left(\frac{\beta z}{\sigma^2}\right)^{\beta^2/\sigma^2} \exp\left(-\frac{z\beta}{\sigma^2}\right), \tag{25}$$

$Q^{-1} = \frac{\beta}{\beta^2-\sigma^2}$ – maximum of $W \to$ mimicks the inverse of the hard scale.

# Benchmarking the precision

$W(z)$ including LO evolution effects at low $z$ and various Ansaetze for large $z$ behavior:

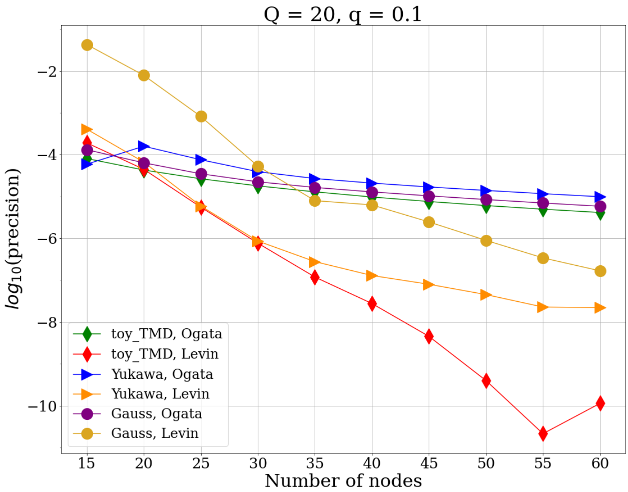
$$W(z) = \exp\Big(S(\mu_z, Q)\Big) \times \Big[F(z)\Big]^2 \tag{26}$$

$\exp\Big(S(\mu_z, Q)\Big)$ - Sudakov factor with $z$-dependent renormalization scale $\mu_z \propto 1/z$.

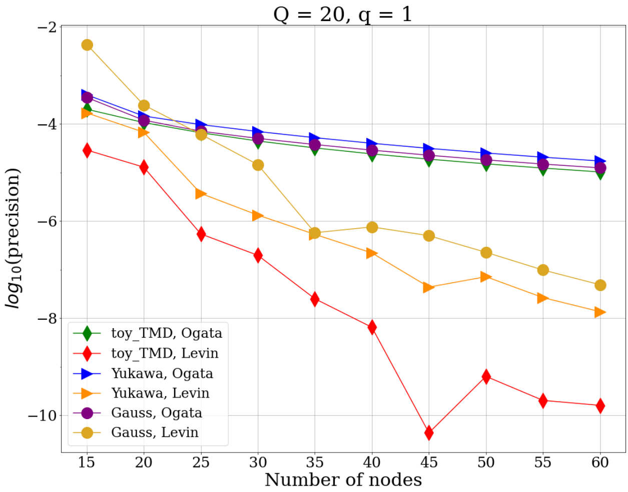
$F(z)$ - Ansatz for large-$z$ behavior of TMD:

- Gaussian behavior from Bacchetta et al., [arXiv:1703.10157],
- Exponential form from Scimemi and Vladimirov, [arXiv:1706.01473].

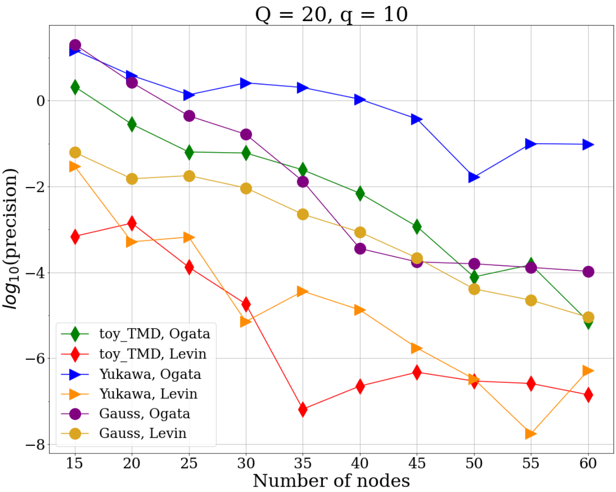
# Benchmarking the precision



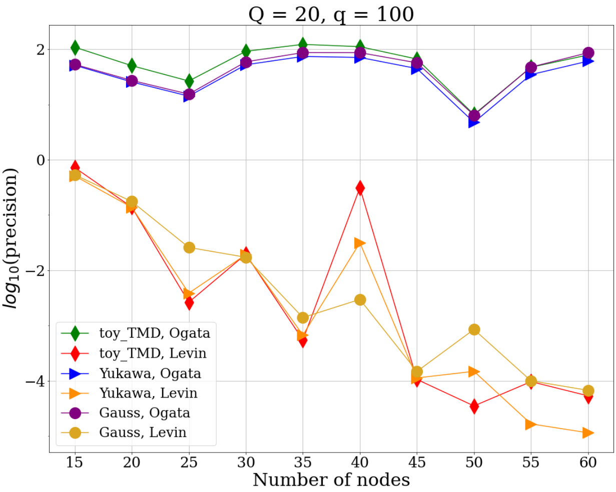
# Benchmarking the precision



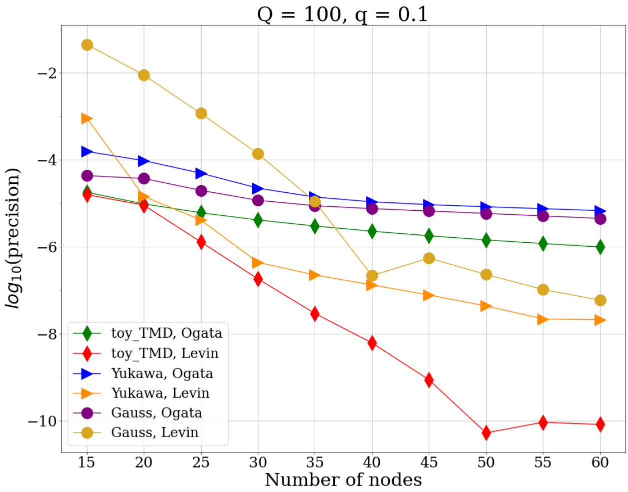
# Benchmarking the precision



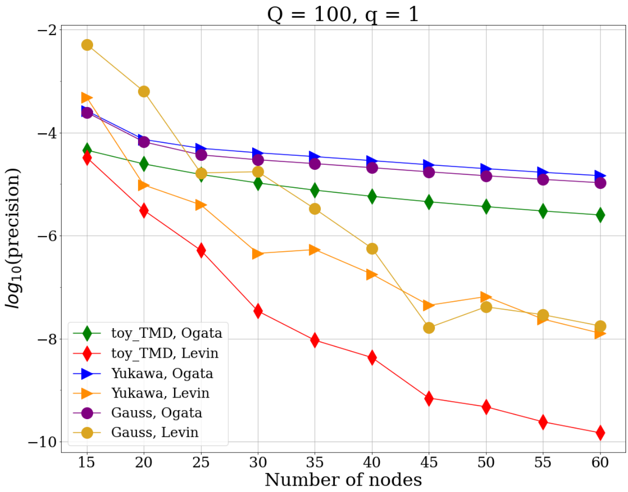
# Benchmarking the precision



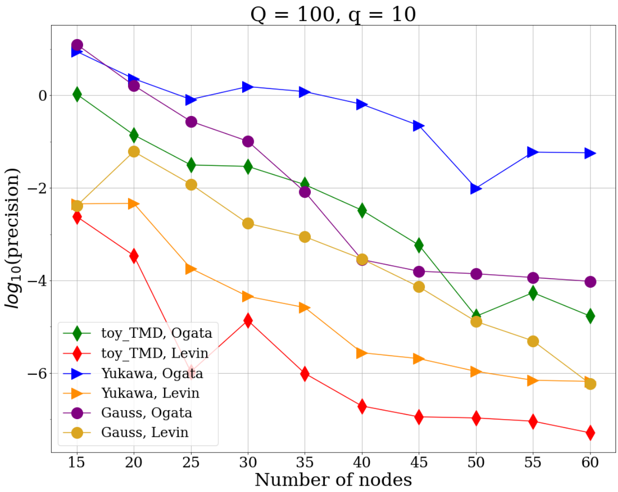
# Benchmarking the precision



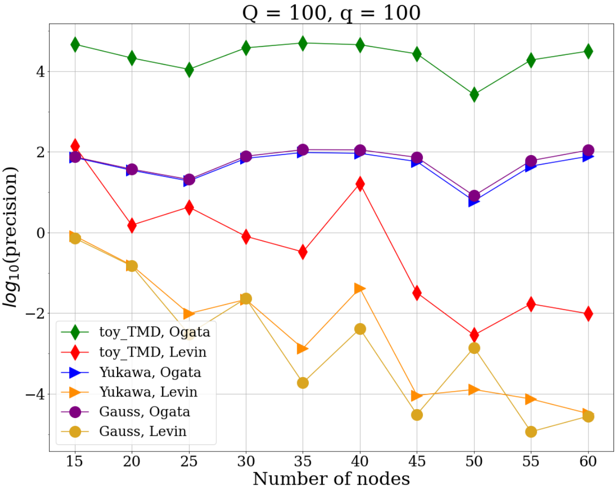
# Benchmarking the precision



# Benchmarking the precision

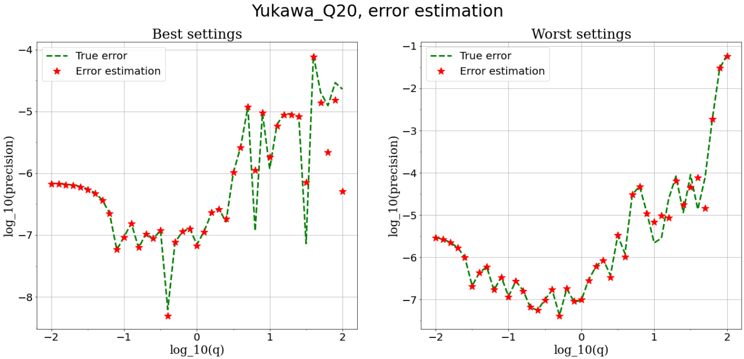


# Benchmarking the precision

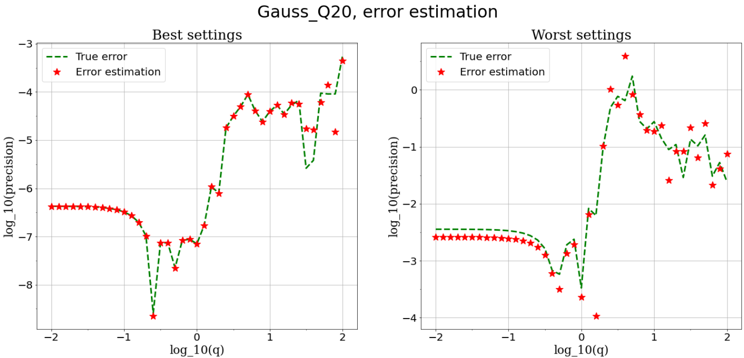


# Error estimation

Estimate the error by comparing with results for grid with twice as many points.



# Error estimation



# Summary

- Need to find good grid settings (subgrid splitting, variable transformation).
- Can use the Levin's method on a fixed grid in $z$-space (independently on $q$!)
- Much better precision for higher $q$.
- Can handle integration on intervals different than $(0, \infty)$, e.g. when integrating with a lower cut-off $z_{\text{min.}}$.
- Computation of the relevant matrix decomposition allows to quickly compute integrals involving $J_\nu$, $J_{\nu-1}$ and $J_{\nu+1}$.

# Backup: speeding up the computation

- $n$-point grid $\implies$ a system of $2n$ linear equations.

# Backup: speeding up the computation

- $n$-point grid $\implies$ a system of $2n$ linear equations.
- The most costly part: computation of the LU (possibly also SV) decomposition $\propto n^3$ operations.

# Backup: speeding up the computation

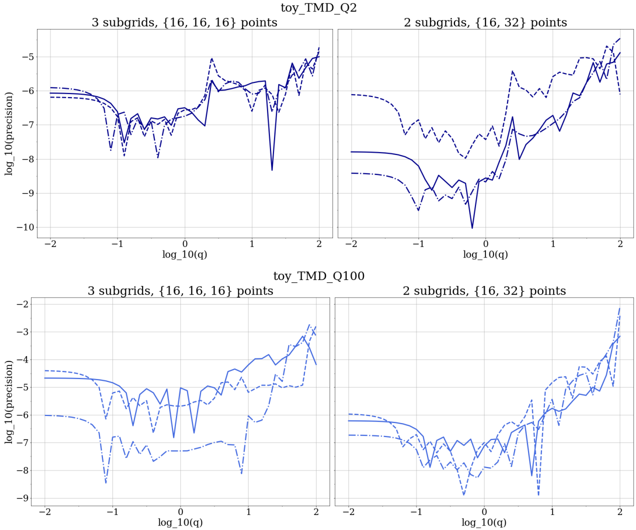
- $n$-point grid $\implies$ a system of $2n$ linear equations.
- The most costly part: computation of the LU (possibly also SV) decomposition $\propto n^3$ operations.
- Can use 3 subgrid with $\{16, 16, 16\}$ points instead of $\{16, 32\}$.

# Backup: speeding up the computation

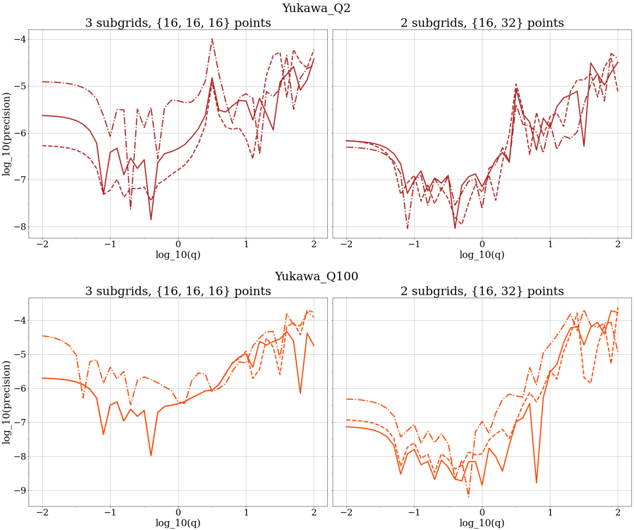
- $n$-point grid $\implies$ a system of $2n$ linear equations.
- The most costly part: computation of the LU (possibly also SV) decomposition $\propto n^3$ operations.
- Can use 3 subgrid with $\{16, 16, 16\}$ points instead of $\{16, 32\}$.

$\longrightarrow \sim 40\%$ faster computation, but slightly worse accuracy.

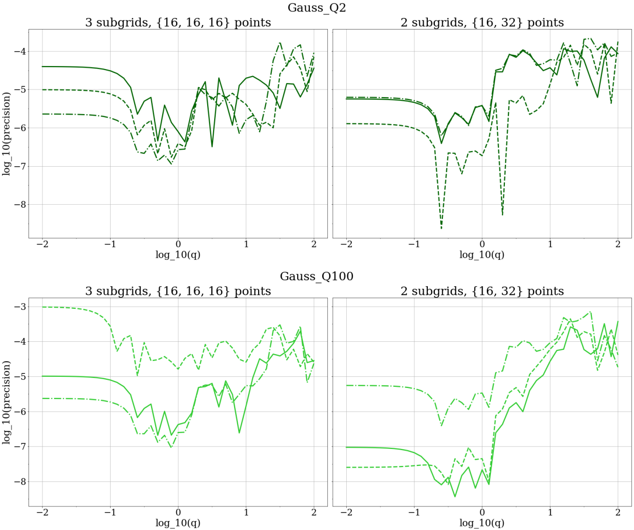
# Backup: 2 vs 3 subgrids



# Backup: 2 vs 3 subgrids



# Backup: 2 vs 3 subgrids



# Backup: LU decomposition of the matrix B

$$PB = LU \tag{27}$$

$$L = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{1,2} & 1 & 0 & \dots & 0 \\ & & \dots & & \\ l_{1,n-1} & l_{2,n-1} & \dots & \ddots & 0 \\ l_{1,n} & l_{2,n} & \dots & l_{n-1,n} & 1 \end{pmatrix} \quad U = \begin{pmatrix} u_{1,1} & u_{2,1} & u_{3,1} & \dots & u_{n,1} \\ 0 & u_{2,2} & u_{3,2} & \dots & u_{n,2} \\ & & \dots & & \\ 0 & 0 & \dots & \ddots & u_{n,n-1} \\ 0 & 0 & \dots & 0 & u_{n,n} \end{pmatrix} \tag{28}$$

$P$ - permutation matrix.

Can solve $B\vec{h} = \vec{g}$ using the backward substitution method.

# Backup: SV decomposition of the matrix B

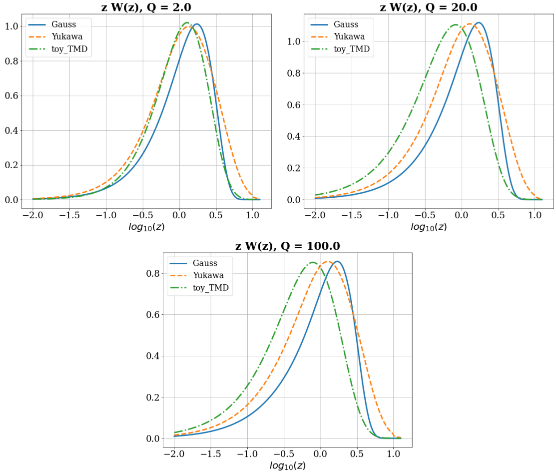
$$B = U\left[\mathrm{diag}(w_j)\right]V^T, \tag{29}$$

$U, V$ - orthogonal matrices, $w_j$ - singular values.

$$B^{-1} = V\left[\mathrm{diag}(1/w_j)\right]U^T. \tag{30}$$

$w_j = 0$ (or $|w_j| < \varepsilon$ - arbitrarily chosen small value) $\rightarrow$ replace $1/w_j$ by 0. Solution $h'$ obtained this way minimizes the error

$$\sum_j \left|\left(B\vec{h}' - \vec{g}\right)_j\right|. \tag{31}$$


z W(z), Q = 2.0
z W(z), Q = 20.0
z W(z), Q = 100.0
Gauss
Yukawa
toy_TMD
$log_{10}(z)$
1.0
0.8
0.6
0.4
0.2
0.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0