Efficient numerical methods for solving evolution equations in QCD

Valerio Bertone
IRFU, CEA, Université Paris-Saclay

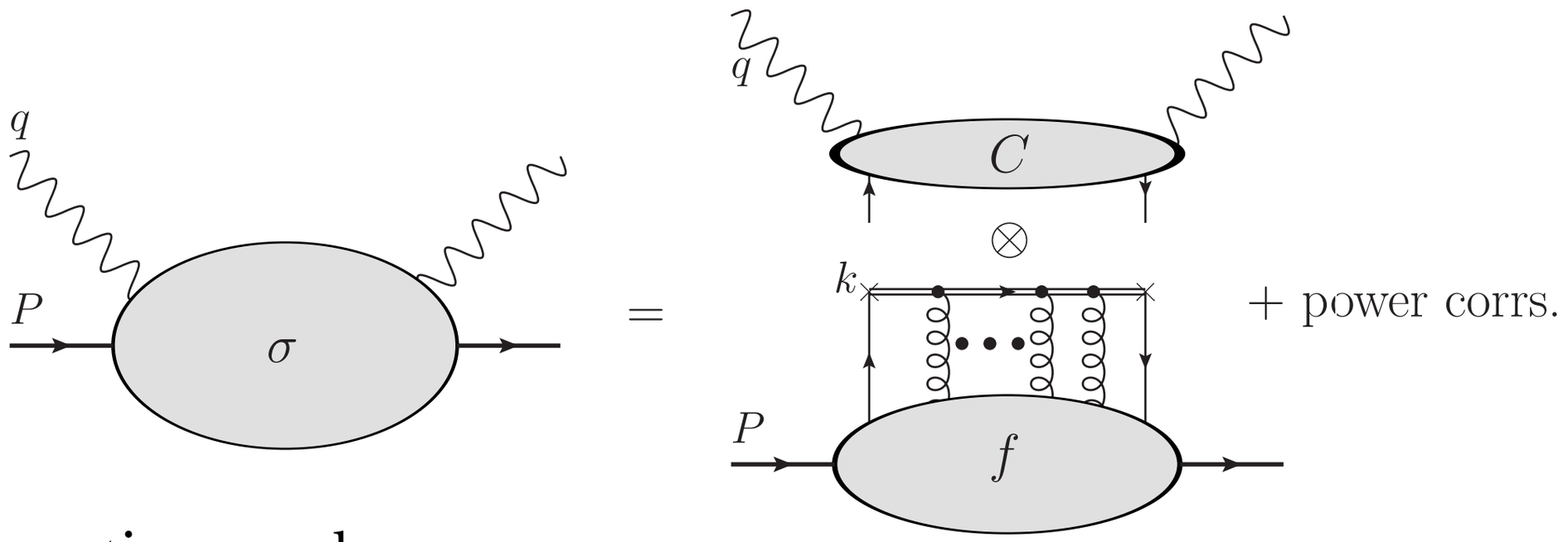
June 7, 2022, Progress in algorithms and numerical tools for QCD

Outline

- Collinear factorisation:
  - definition of parton distributions and their evolution.
- Lagrange interpolation:
  - basic concepts,
  - a showcase test,
  - (aside: derivation and integration).
- Evolution on the grid:
  - technicalities,
  - numerical results.

Collinear factorisation in QCD

- Typical examples where *collinear* factorisation in QCD applies are the cross section for deep-inelastic scattering (cut diagram) and deeply-virtual Compton scattering (uncut diagram) at large $Q^2 = -q^2$:



- The cross section reads:

$$d\sigma = C \otimes f^{(0)} + \mathcal{O}(\Lambda^n_{\mathrm{QCD}}/Q^n)$$

- C is the "partonic" cross section:
  - computable in perturbation theory (coll. divergences may need to be subtracted),
- $f^{(0)}$ is the *bare* partonic distribution:
  - **non-perturbative**, *i.e.* perturbation theory inapplicable,
  - UV divergent: needs to be **renormalised**!

Renormalisation and evolution

- Renormalisation of the a partonic distributions proceeds as usual:

$$f^{(0)}(\epsilon) = Z(\mu, \epsilon) \otimes f(\mu)$$

- so that:

$$\frac{df(\mu)}{d\ln\mu} = P(\mu) \otimes f(\mu)$$

- with:

$$P(\mu) = -\lim_{\epsilon\to 0} \frac{d\ln Z(\mu,\epsilon)}{d\ln\mu}$$

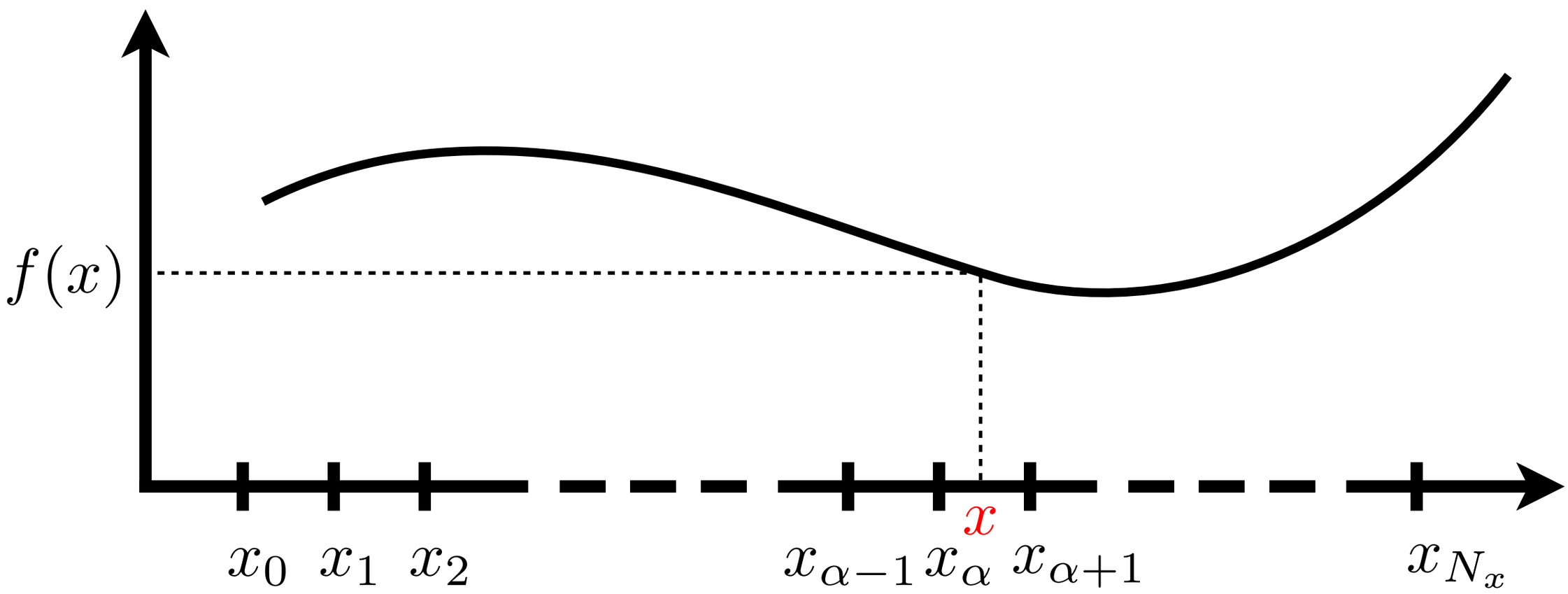
- P is finite and computable in perturbation theory (splitting function).
- Prominent example: the DGLAP equations that in their full glory read:

$$\frac{df_i(x,\mu)}{d\ln\mu^2} = \sum_j \int_x^1 \frac{dy}{y} P_{ij}\left(y, \alpha_s(\mu)\right) f_j\left(\frac{y}{x}, \mu\right) \qquad i,j = g,q$$

- A set of **coupled integro-differential equations**.
- The focus of this talk is the **efficient numerical solution** of equations of this type.

Lagrange interpolation

- Goal: interpolating a 1D function knowing its values on a grid.



- "Moving" polynomial interpolation of degree k:

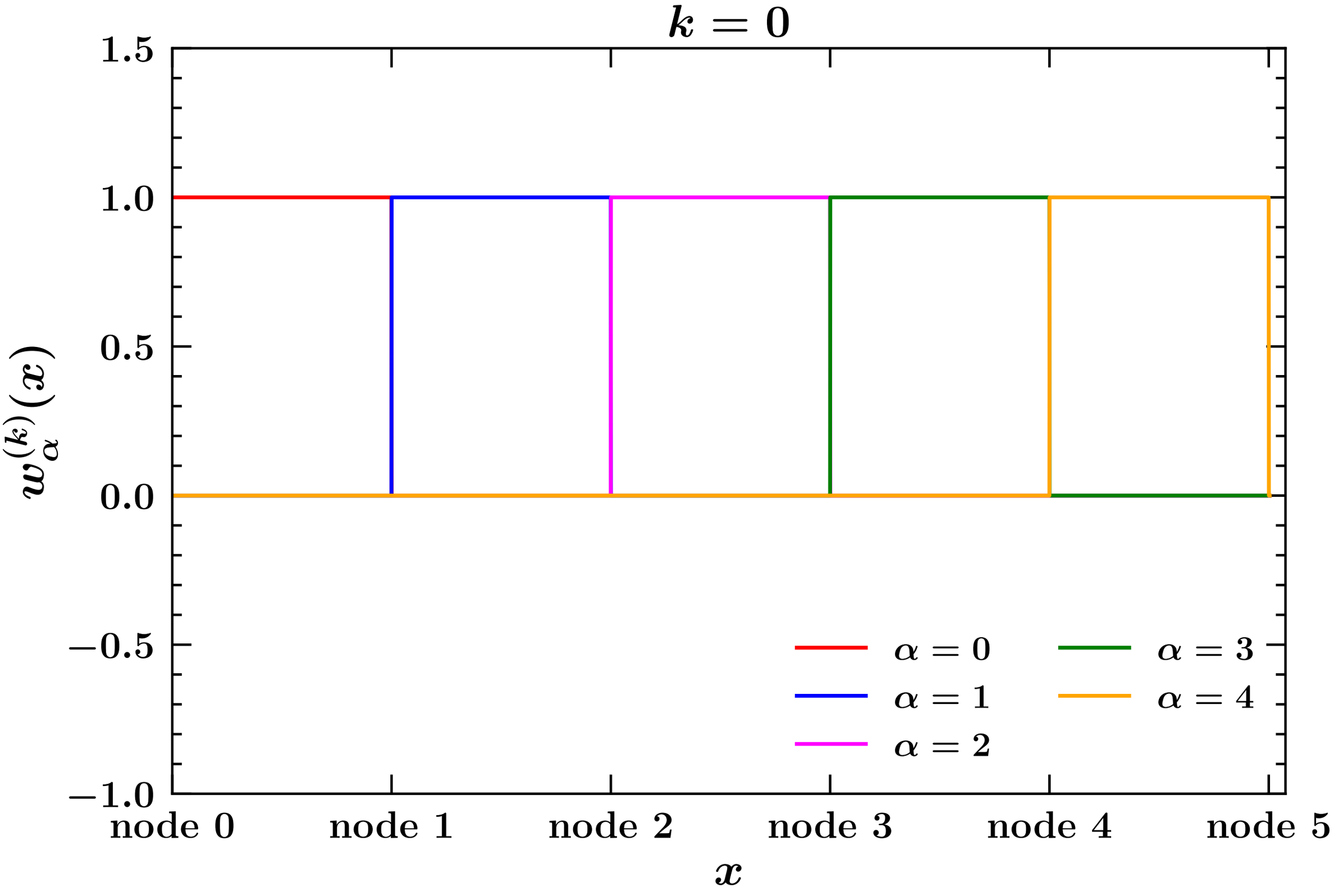
$$f(x) = \sum_{\alpha=0}^{N_x} w_\alpha^{(k)}(x) f(x_\alpha) + \mathcal{O}\left(\frac{f^{(k+1)}(x)}{(k+1)!} h^{k+1} \right)$$

- assuming that $x_\alpha < x \leq x_{\alpha+1}$:

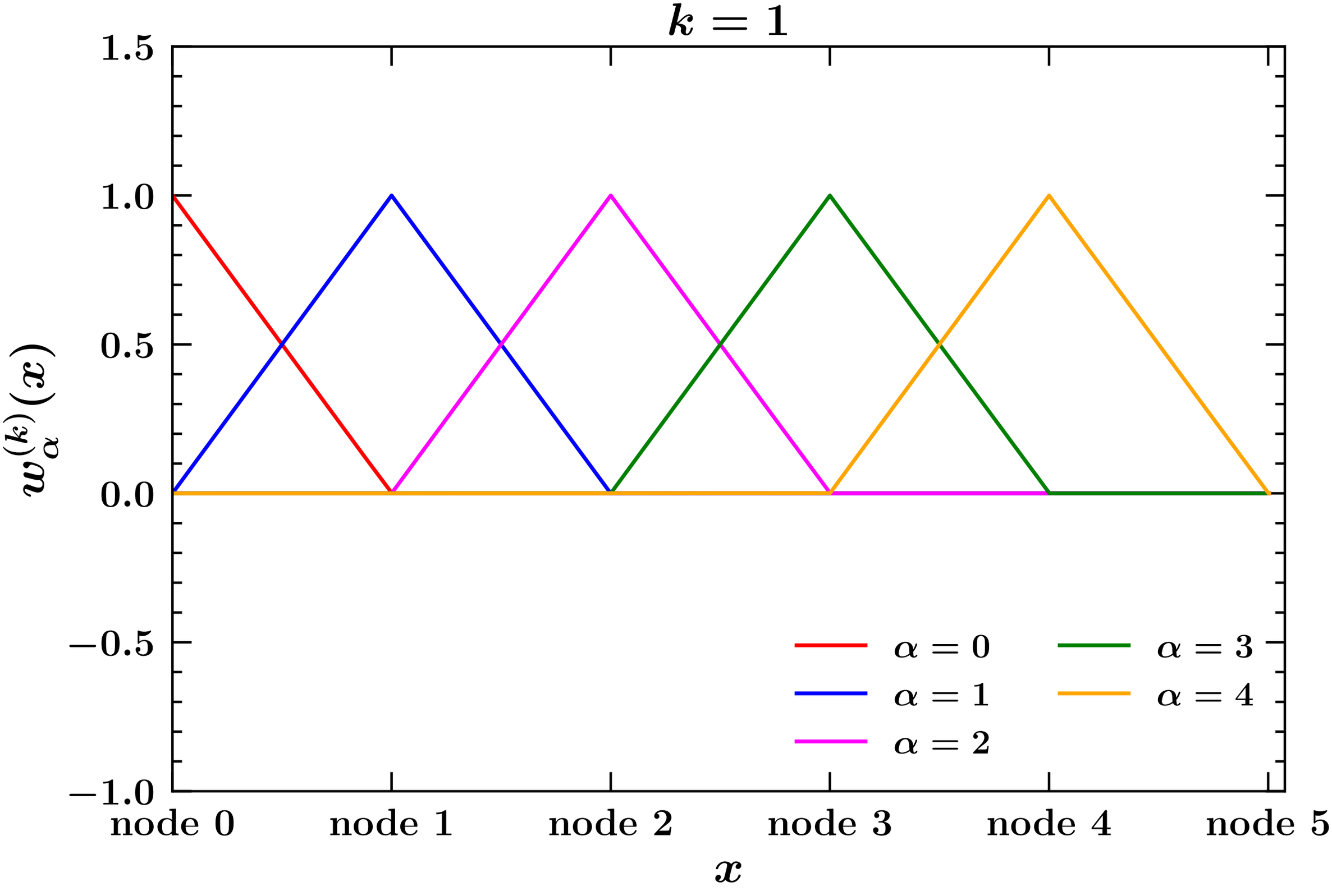
$$w_\alpha^{(k)}(x) = \sum_{i=0, i\leq\alpha}^{k} \theta(x - x_{\alpha-i}) \theta(x_{\alpha-i+1} - x) \prod_{m=0, m\neq i}^{k} \frac{x - x_{\alpha-i+m}}{x_\alpha - x_{\alpha-i+m}}$$

Lagrange polynomials

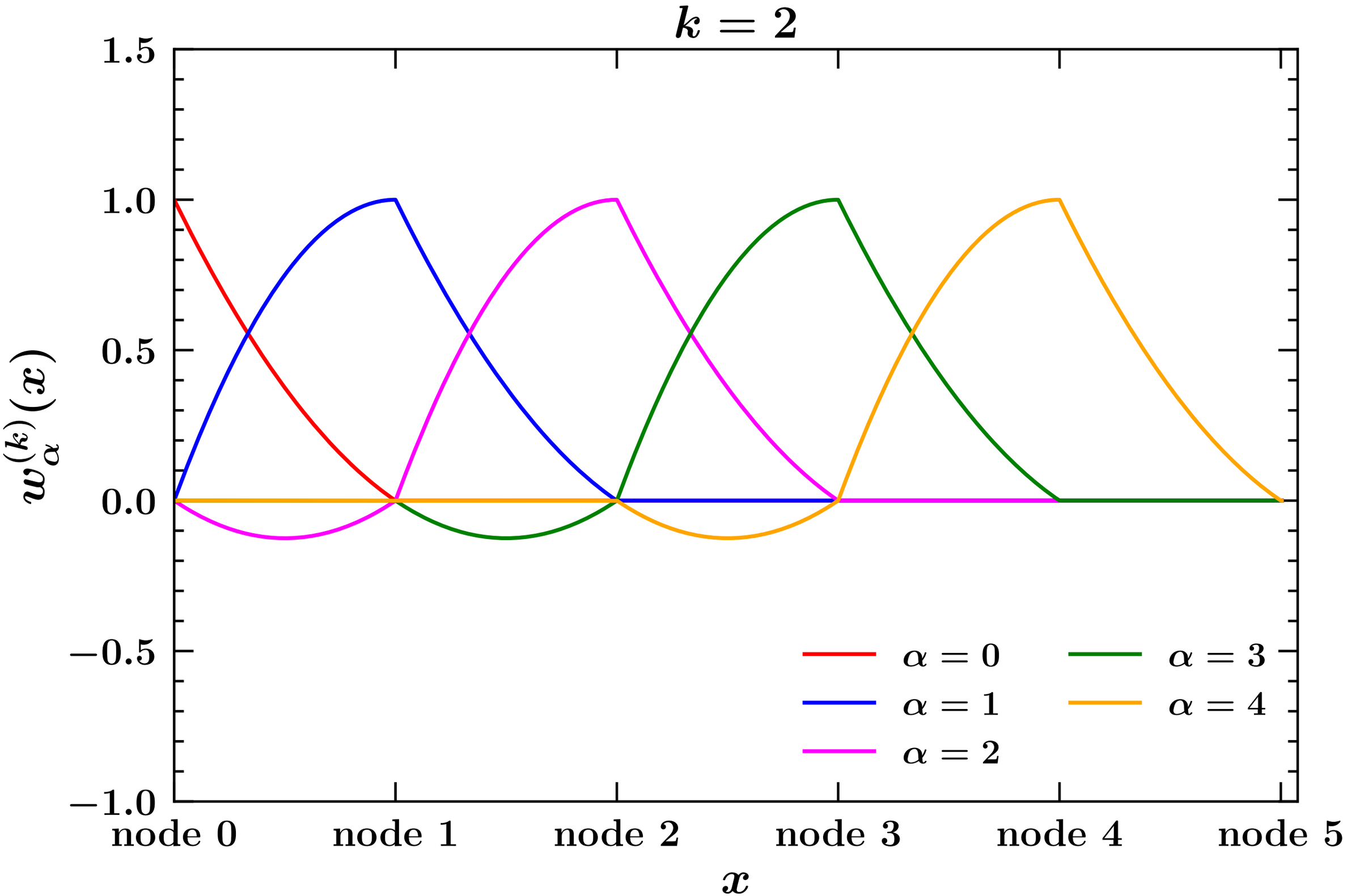
The interpolating functions



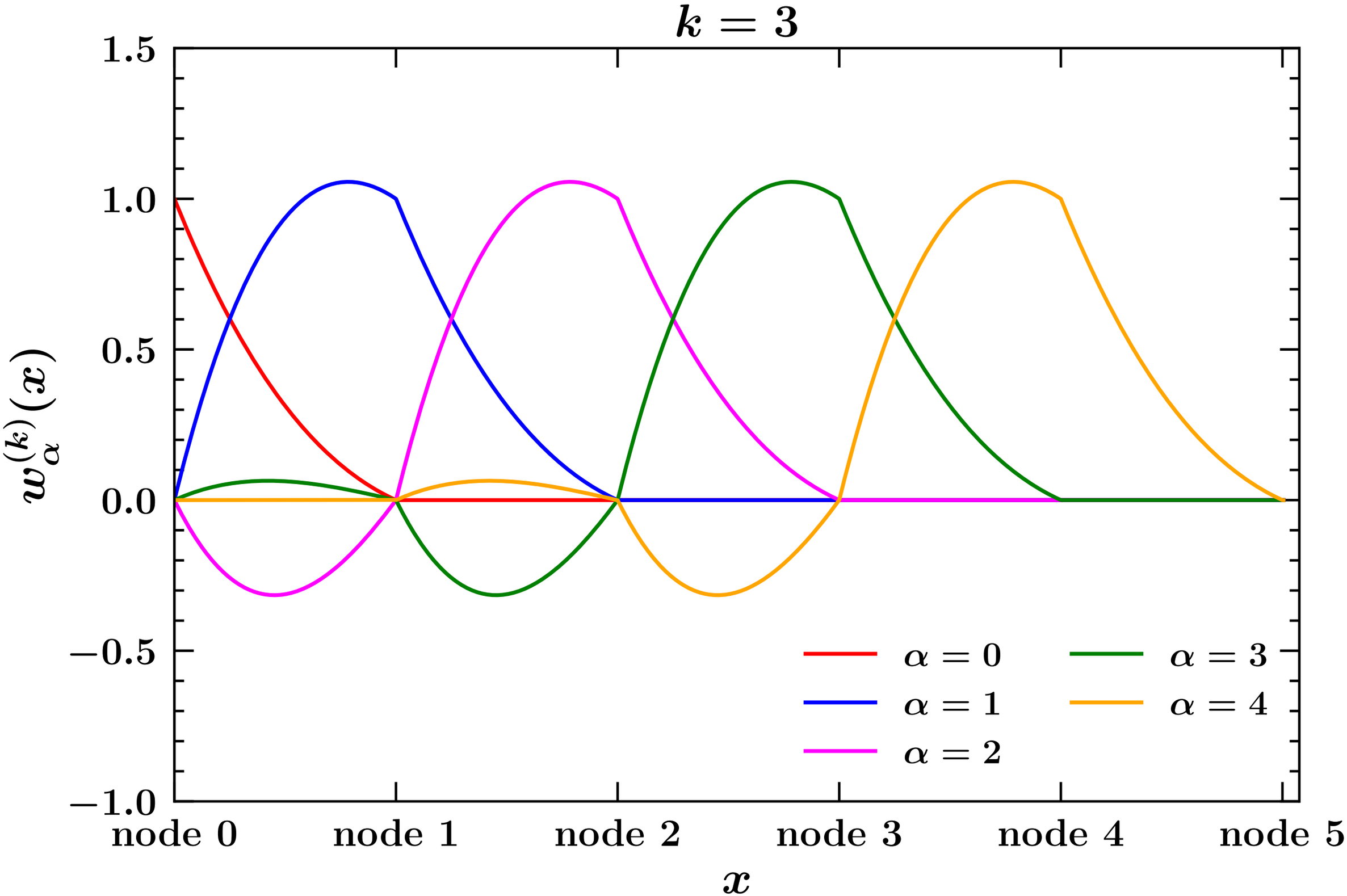
The interpolating functions



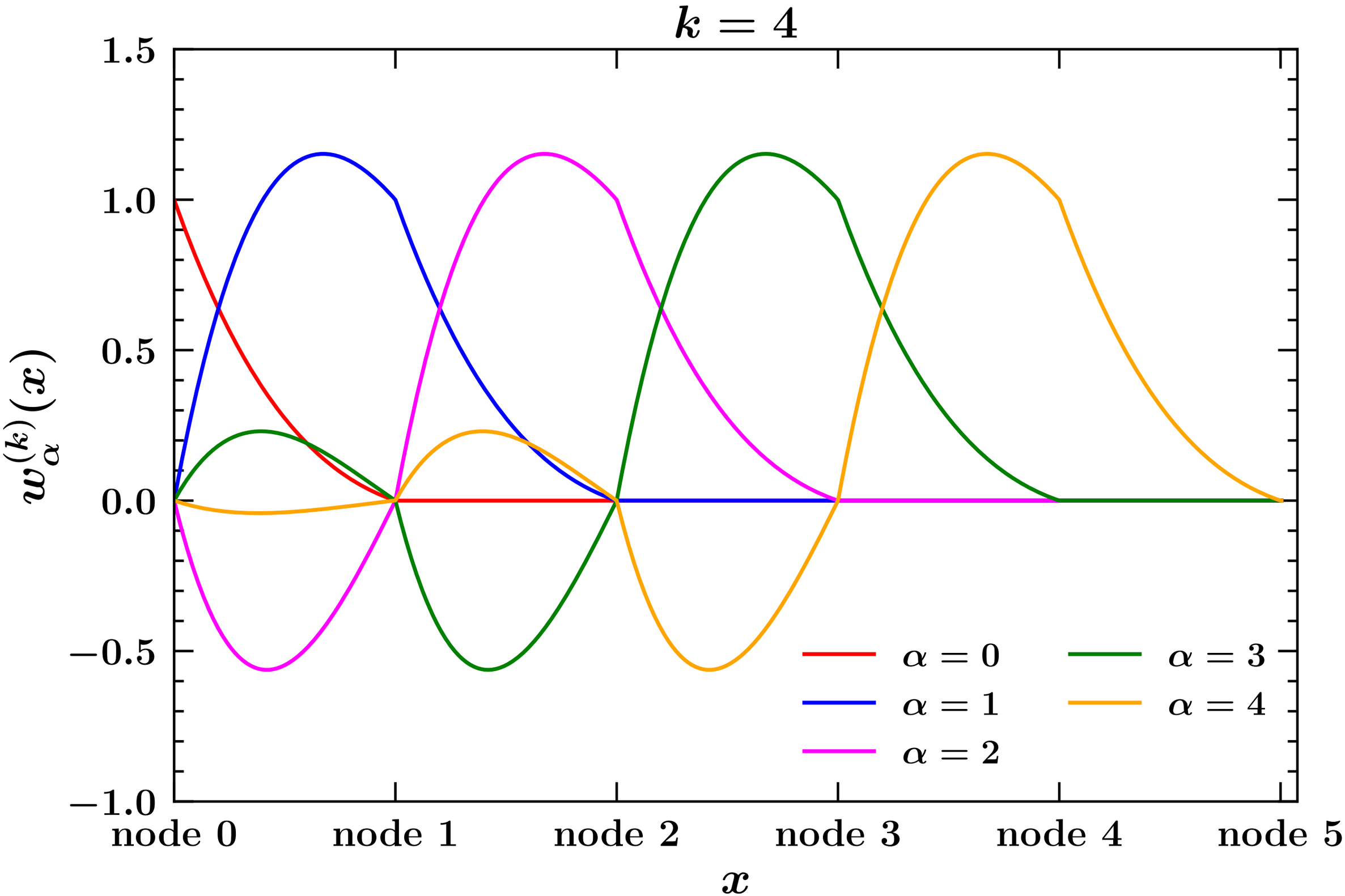
The interpolating functions



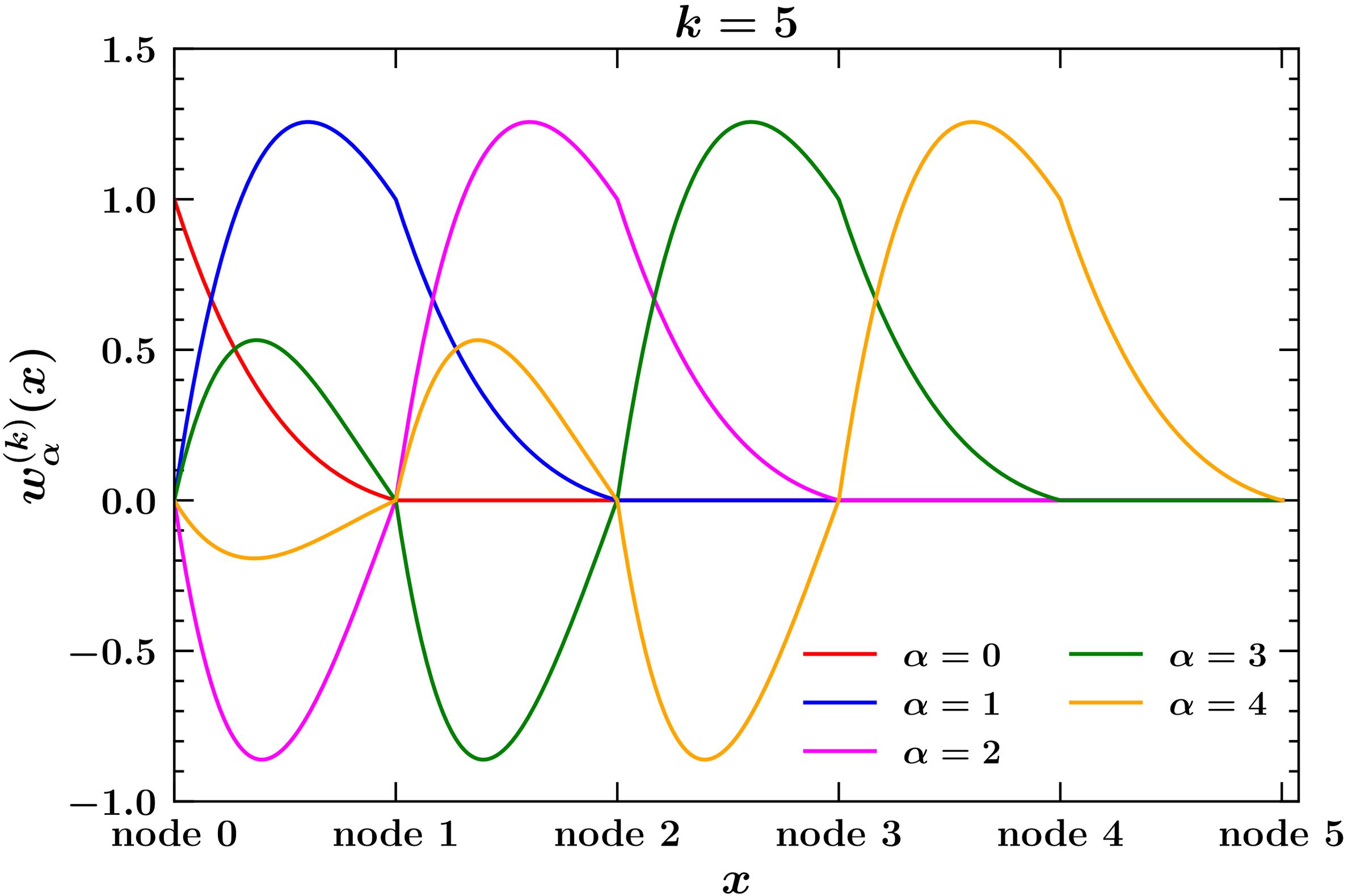
The interpolating functions



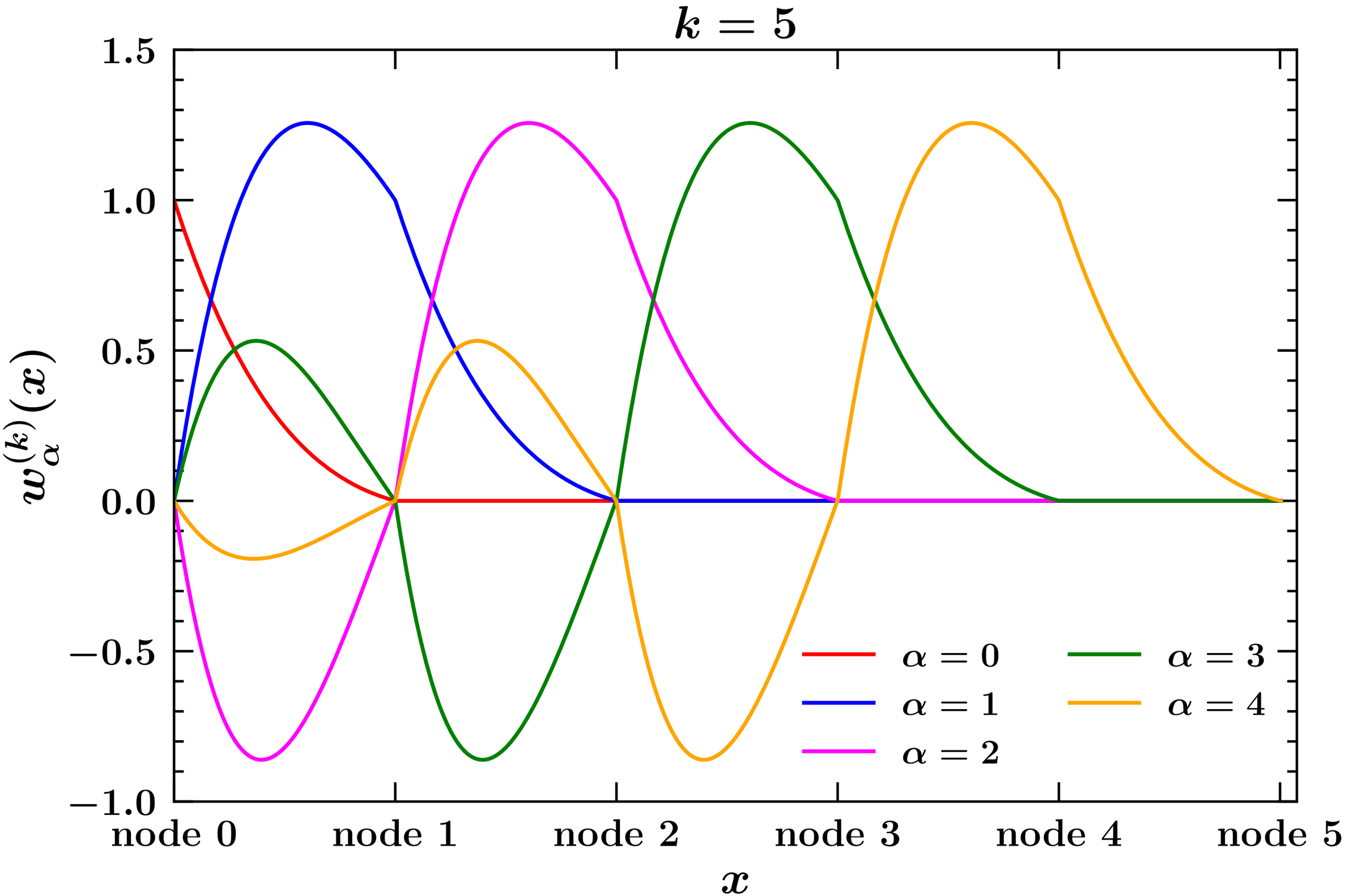
The interpolating functions



The interpolating functions



The interpolating functions



- At any interpolation degree, the interpolating functions are such that $w_\alpha^{(k)}(x_\beta) = \delta_{\alpha\beta}$.

Interpolation test

- Interpolate the function:

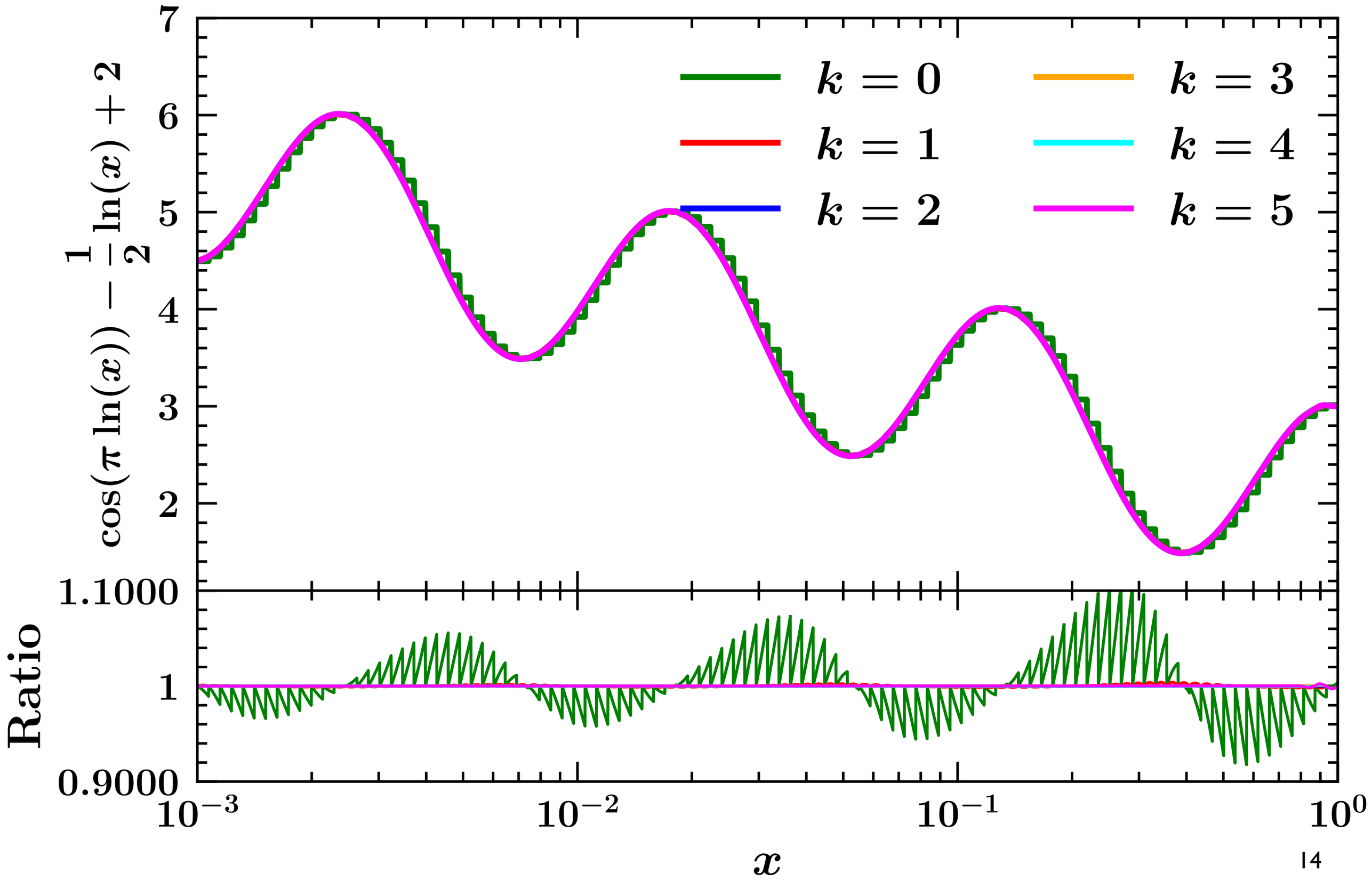
$$f(x) = \cos(\pi \ln(x)) - \frac{1}{2}\ln(x) + 2$$

- on grid with 100 nodes logarithmically distributed between 10^{-3} and 1.

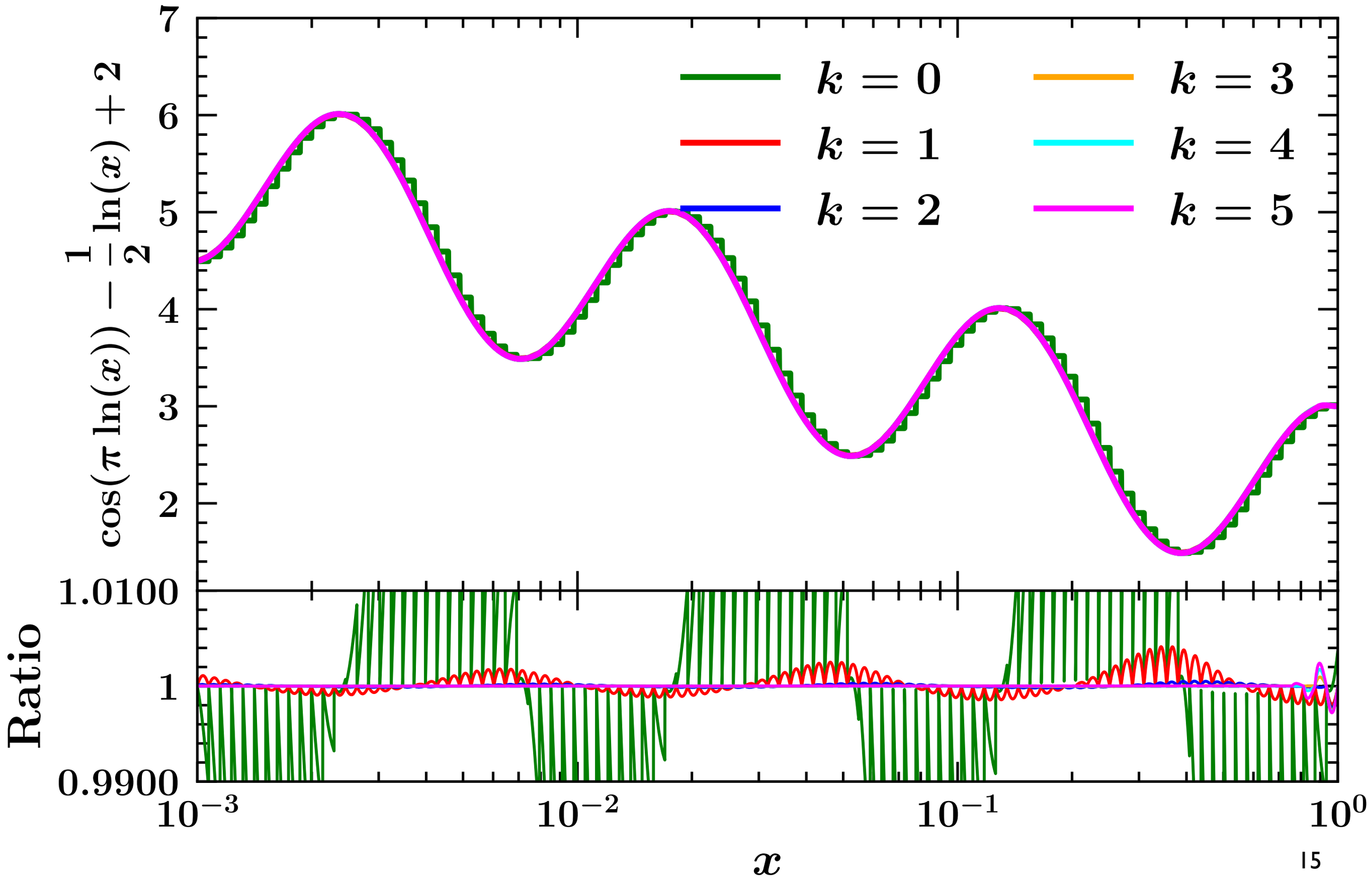
- Test different degrees of the Lagrange interpolants from 0 to 5:

- check the accuracy against the original function.

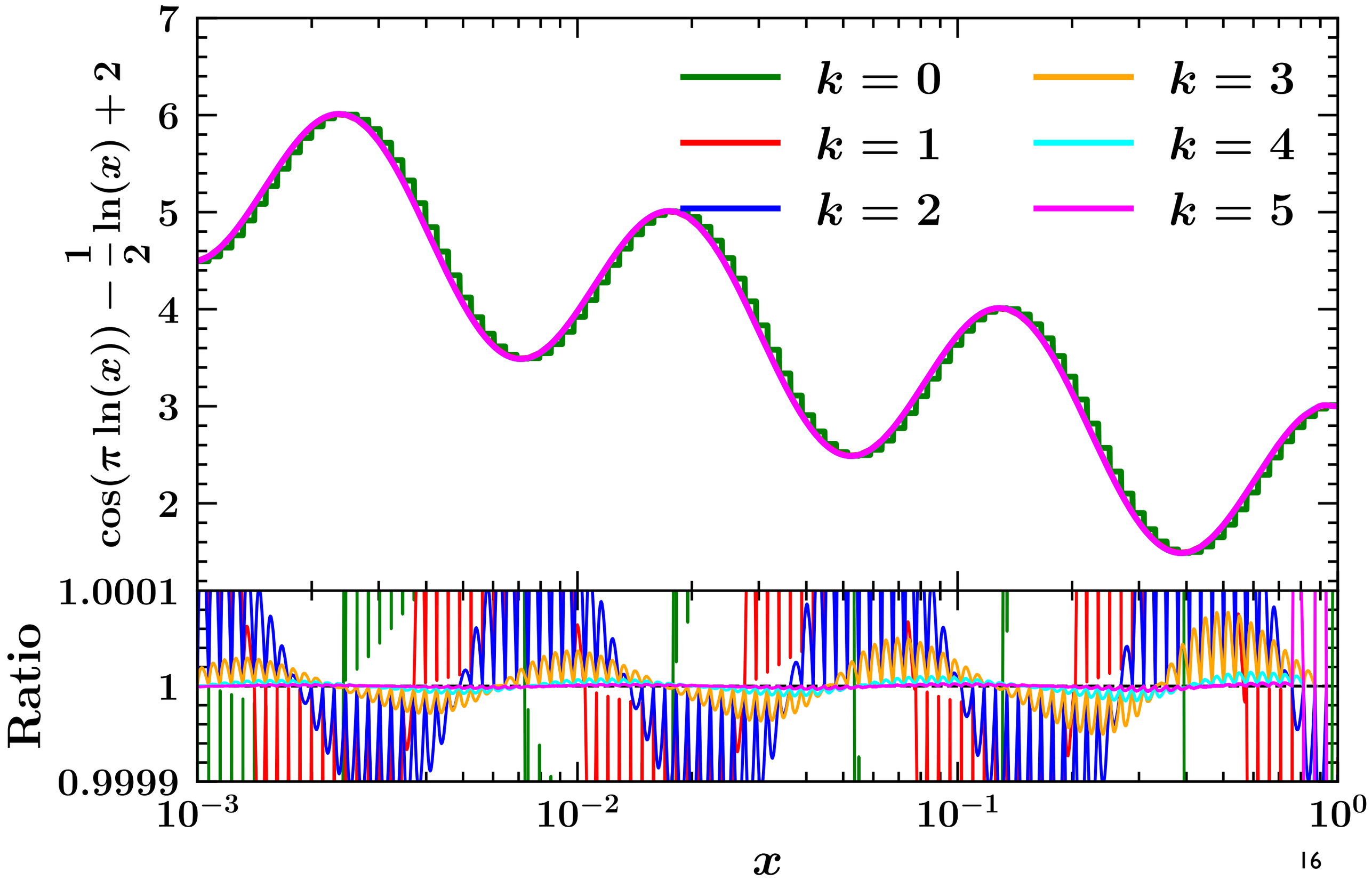
Interpolation test



Interpolation test



Interpolation test



Aside: deriving and integrating

- If:

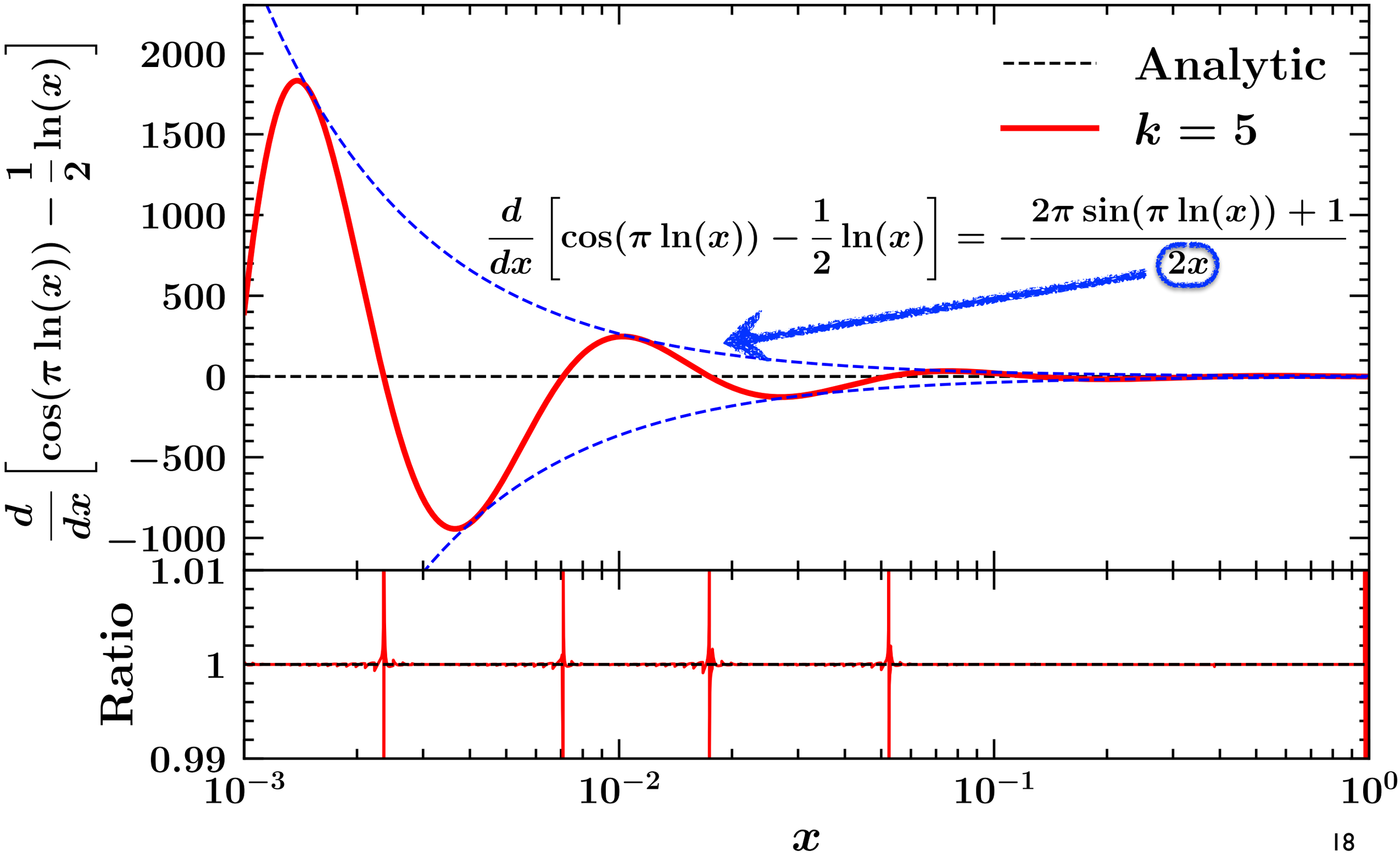
$$f(x) = \sum_{\alpha=1}^{N_x} w_\alpha^{(k)}(x) f(x_\alpha)$$

- then:

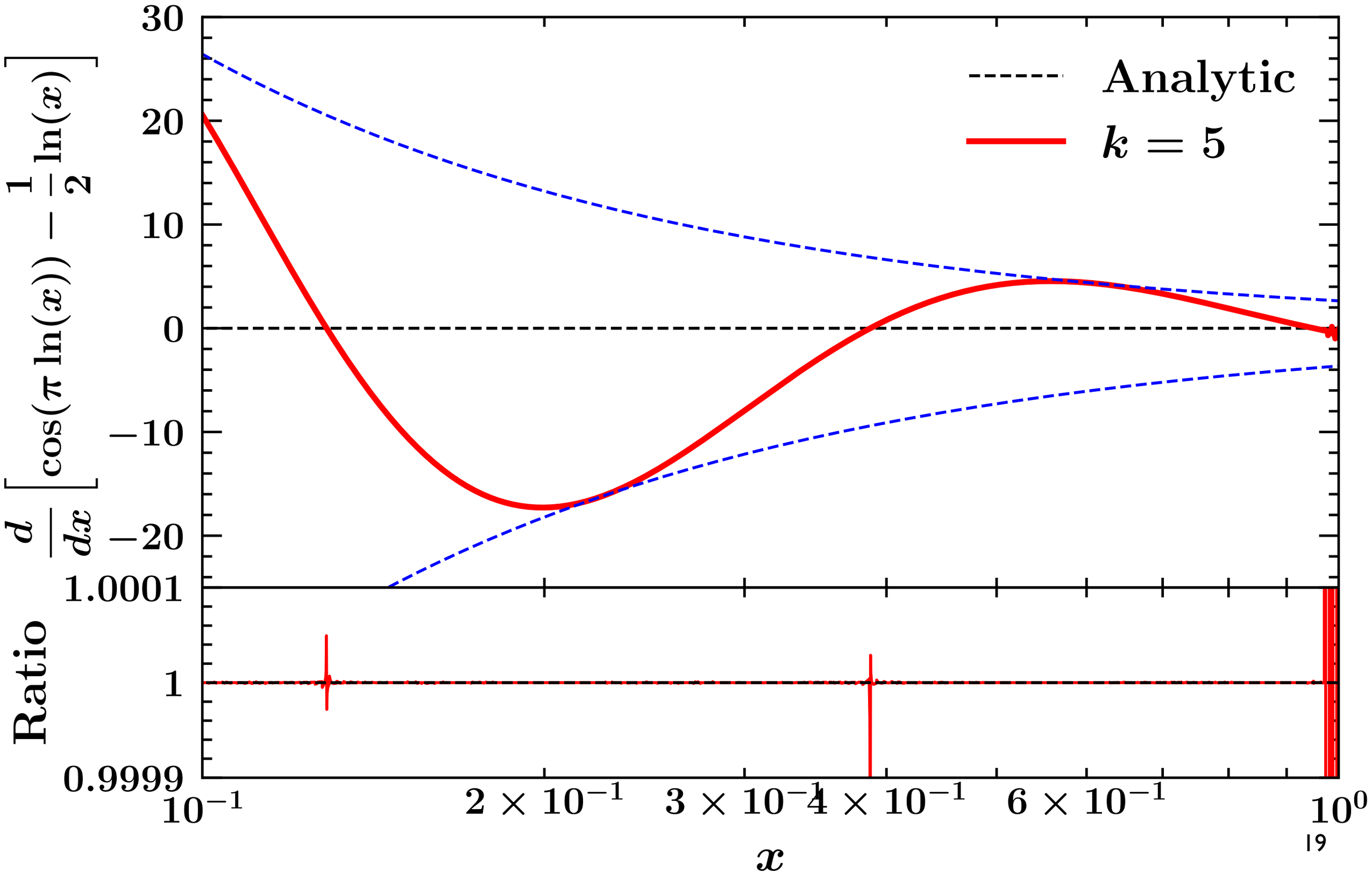
$$\frac{df}{dx} = \sum_{\alpha=1}^{N_x} \frac{dw_\alpha^{(k)}}{dx} f(x_\alpha) \quad \text{and} \quad \int_a^b f(x)\,dx = \sum_{\alpha=1}^{N_x} \left[\int_a^b w_\alpha^{(k)}(x) dx \right] f(x_\alpha)$$

- The derivative and the indefinite integral of the interpolation functions can be computed **analytically** for any interpolation degree.
- This enables one, not only to interpolate the function *f*, but also to compute **derivatives** and **integrals** by knowing its values on the grid.
- Potential drawbacks:
  - interpolation functions not smooth at the nodes, derivatives not defined there.
  - No direct control on the integration accuracy.

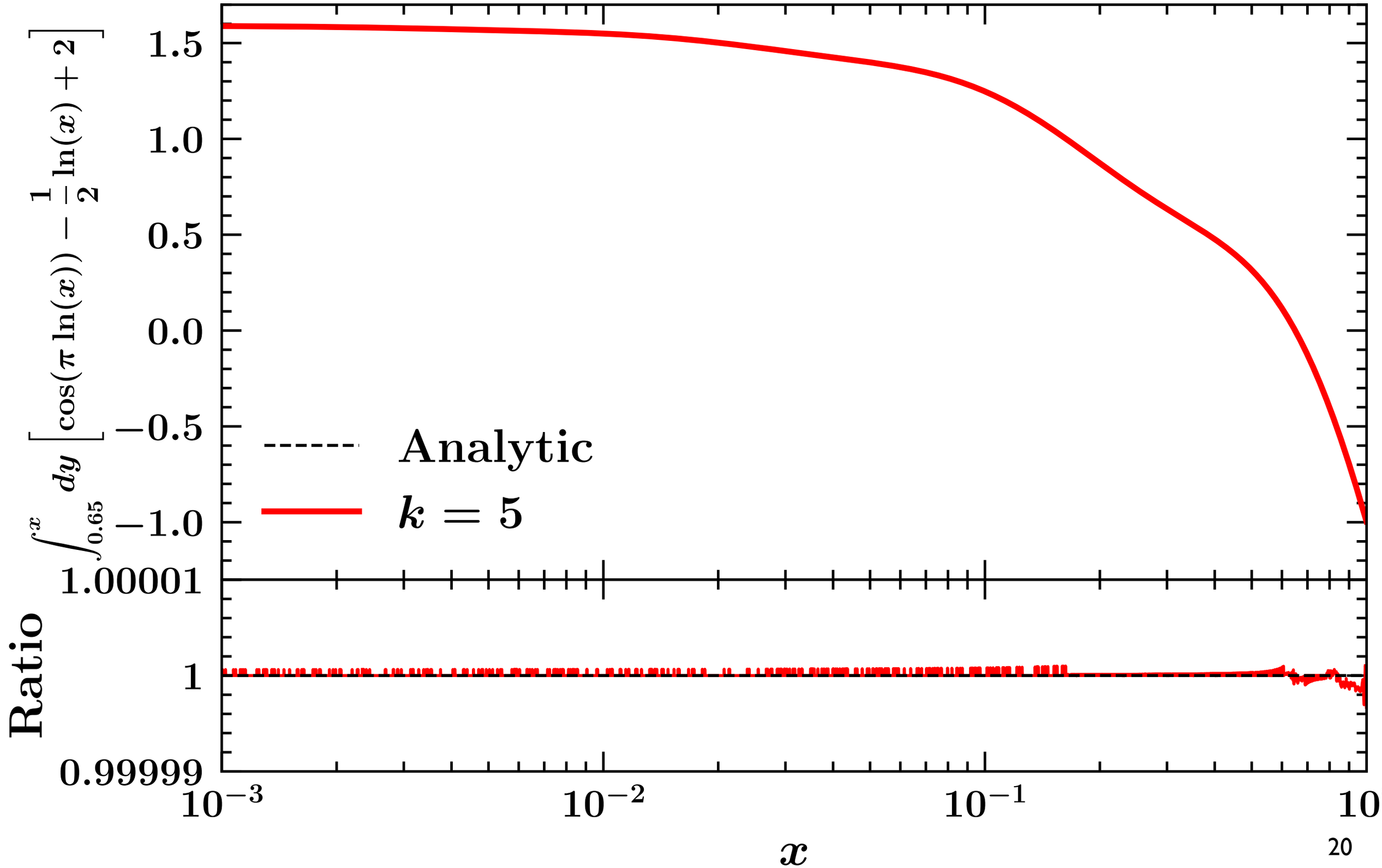
Aside: deriving and integrating



Aside: deriving and integrating



Aside: deriving and integrating



Solving the DGLAP on the grid

- The r.h.s. of the DGLAP equations requires computing integrals of the following kind:

$$I(x,\mu) = \int_x^1 \frac{dy}{y} P(y,\alpha_s(\mu)) f\left(\frac{x}{y},\mu\right)$$

- Using the interpolation formula for f (the parton distribution):

$$I(x_\beta,\mu) = \sum_{\alpha=0}^{N_x} \underbrace{\left[\int_{x_\beta}^1 \frac{dy}{y} P(y,\alpha_s(\mu)) w_\alpha^{(k)}\left(\frac{x_\beta}{y}\right)\right]}_{\Gamma_{\beta\alpha}(\mu)} f(x_\alpha,\mu)$$

- The $\Gamma_{\beta\alpha}$ don't depend on f and thus can be **precomputed** and **stored**.

- More compactly:

$$I_\beta(\mu) = \sum_{\alpha=0}^{N_x} \Gamma_{\beta\alpha}(\mu) f_\alpha(\mu) \quad \Rightarrow \quad \mathbf{I}(\mu) = \mathbf{\Gamma}(\mu)\cdot\mathbf{f}(\mu)$$

- This allows us to compute I on each point of the grid.

Solving the DGLAP on the grid

- The DGLAP equations reduce to a set of coupled **linear ordinary differential equations** (ODEs) that in vectorial notation read:

$$\frac{d\mathbf{f}(\mu)}{d\ln\mu^2} = \mathbf{\Gamma}(\mu)\cdot\mathbf{f}(\mu) \equiv \mathbf{F}(\mu,\mathbf{f})$$

- The functions to be determined are the partonic distribution f as a function of μ on each node of the grid knowing $\mathbf{f}(\mu_0)$ on the grid.
- Many algorithms to solve ODEs exist. A particularly popular choice is the **4th order Runge-Kutta** (RK4):

$$\begin{aligned}
\mathbf{f}_{n+1} &= \mathbf{f}_n + \tfrac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) + \mathcal{O}(h^5)\\
\mathbf{k}_1 &= h\mathbf{F}(\mu_n, \mathbf{f}_n)\\
\mathbf{k}_2 &= h\mathbf{F}(\mu_n + \tfrac{h}{2}, \mathbf{f}_n + \tfrac{\mathbf{k}_1}{2})\\
\mathbf{k}_3 &= h\mathbf{F}(\mu_n + \tfrac{h}{2}, \mathbf{f}_n + \tfrac{\mathbf{k}_2}{2})\\
\mathbf{k}_3 &= h\mathbf{F}(\mu_n + h, \mathbf{f}_n + \mathbf{k}_3)
\end{aligned}$$

- Tabulate $\mathbf{f}$ on a **grid in μ** that can successively interpolated.
- Final result: $f(x,\mu)$ is know on a 2D grid in x and μ.

Interpolation on a log grid

- In principle, one needs to compute N_x^2 integrals $\Gamma_{\beta\alpha}$.
- However, if one uses a logarithmically distributed grid:

$$x_{\alpha+1} = e^{\delta x} x_\alpha \qquad\qquad \delta x = \text{constant}$$

- and the Lagrange polynomials are expressed in **powers of** $\ln(x)$:

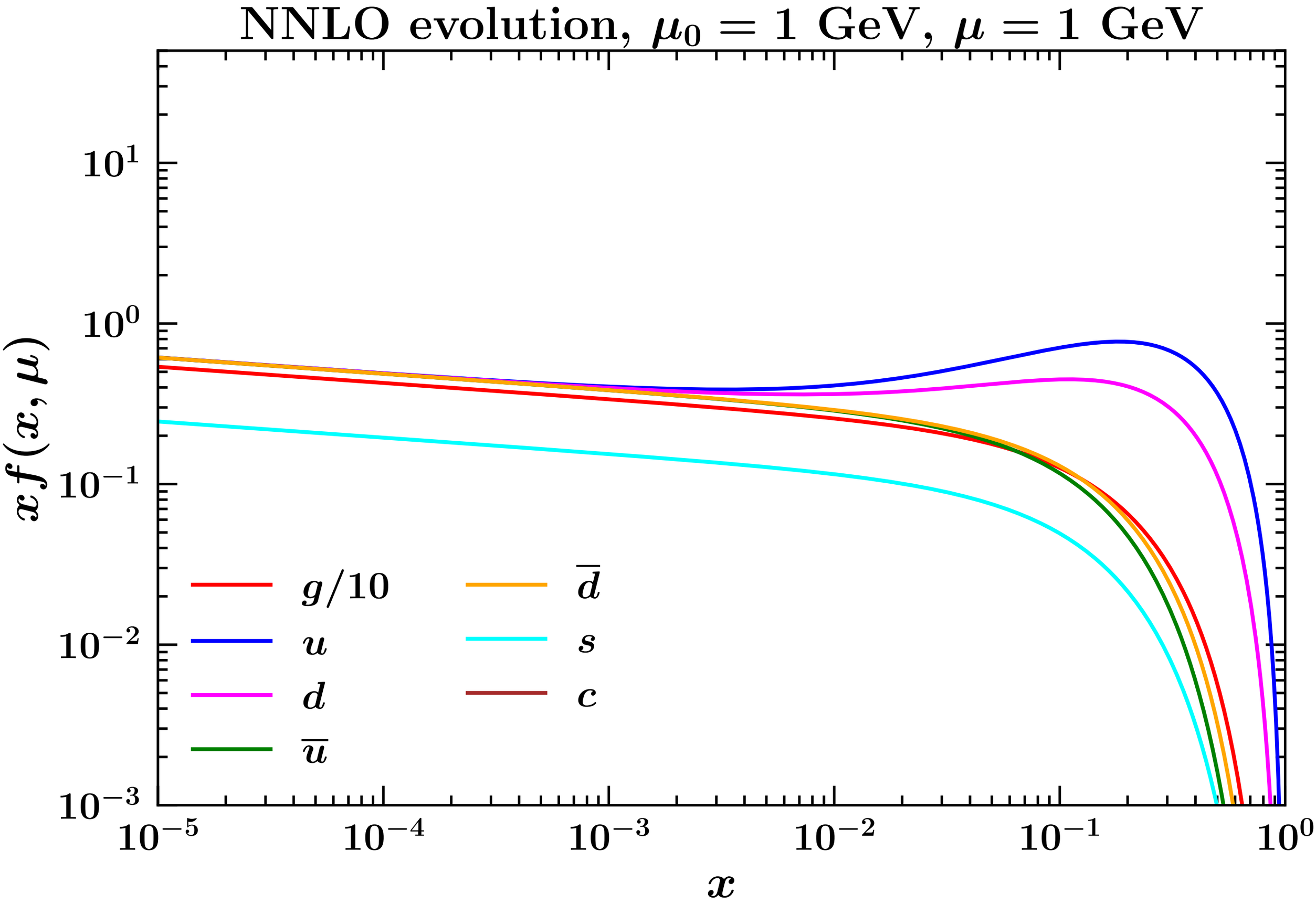
$$w_\alpha^{(k)}(x) = \sum_{i=0, i\le\alpha}^{k} \theta(x - x_{\alpha-i})\theta(x_{\alpha-i+1} - x) \prod_{m=0, m\neq i}^{k} \frac{\ln x - \ln x_{\alpha-i+m}}{\ln x_\alpha - \ln x_{\alpha-i+m}}$$

- one finds:

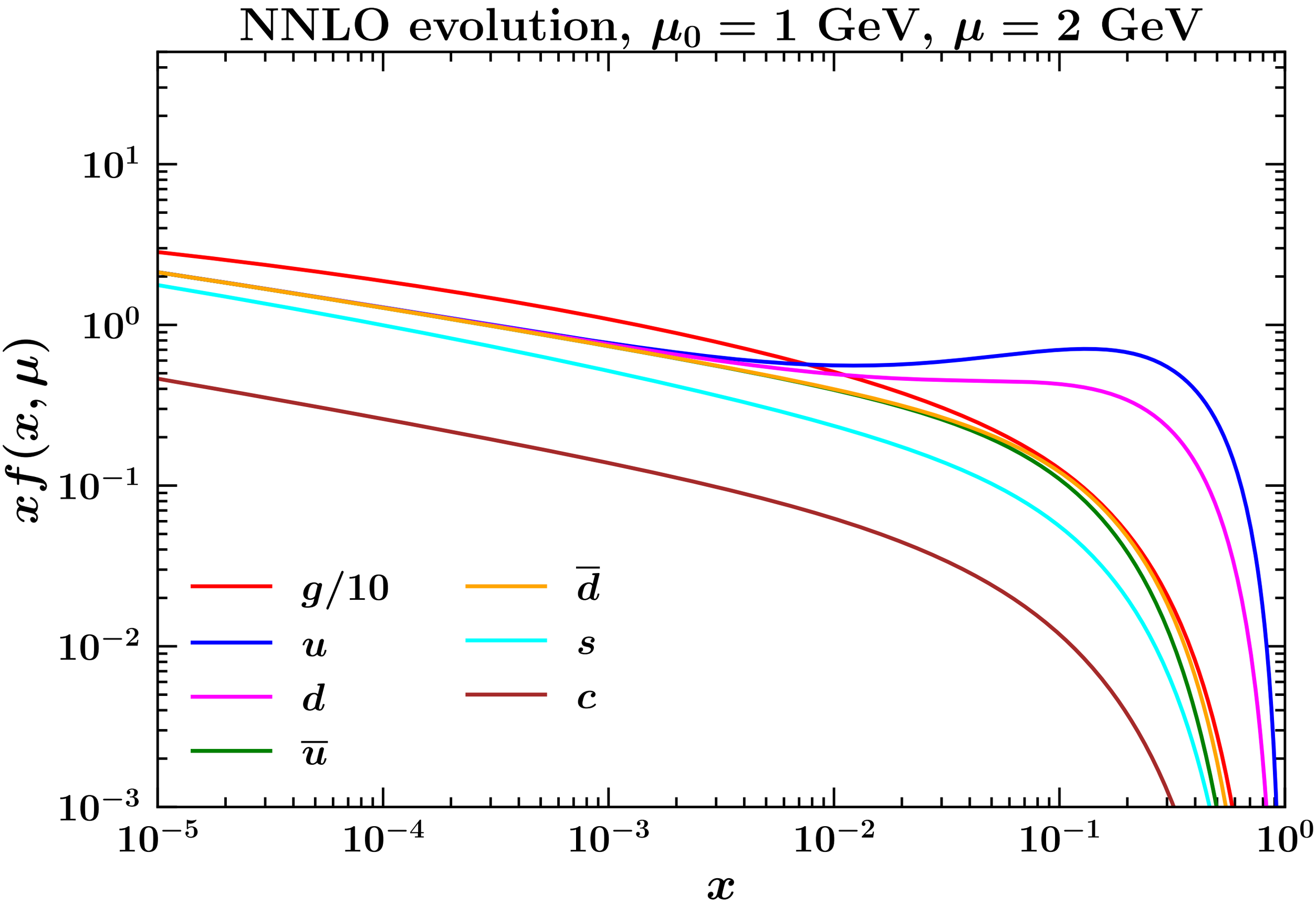
$$\Gamma_{\beta\alpha} = \begin{cases} \Gamma_{0,\alpha-\beta} & \alpha \ge \beta \\ 0 & \alpha < \beta \end{cases} \qquad \Gamma_{\beta\alpha} = \begin{pmatrix} a_0 & a_1 & a_2 & \cdots & a_{N_x} \\ 0 & a_0 & a_1 & \cdots & a_{N_x-1} \\ 0 & 0 & a_0 & \cdots & a_{N_x-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_0 \end{pmatrix}$$

- The number of integrals reduces to N_x and so thus the computation time.
- Problem: a logarithmic grid becomes increasingly sparse at large x:
  - solution: concatenate grids with increasing density as one moves towards large x.

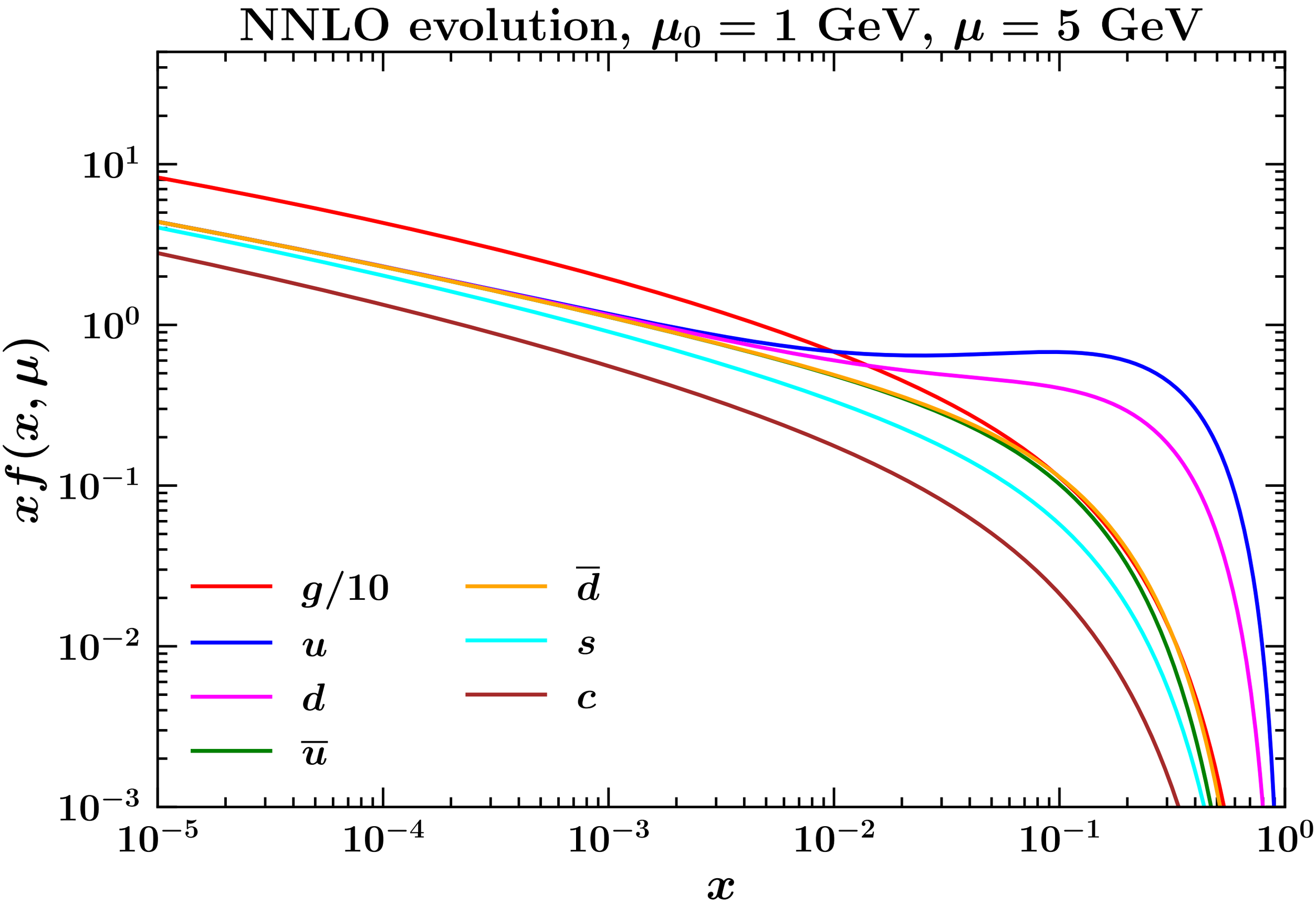
Numerical results



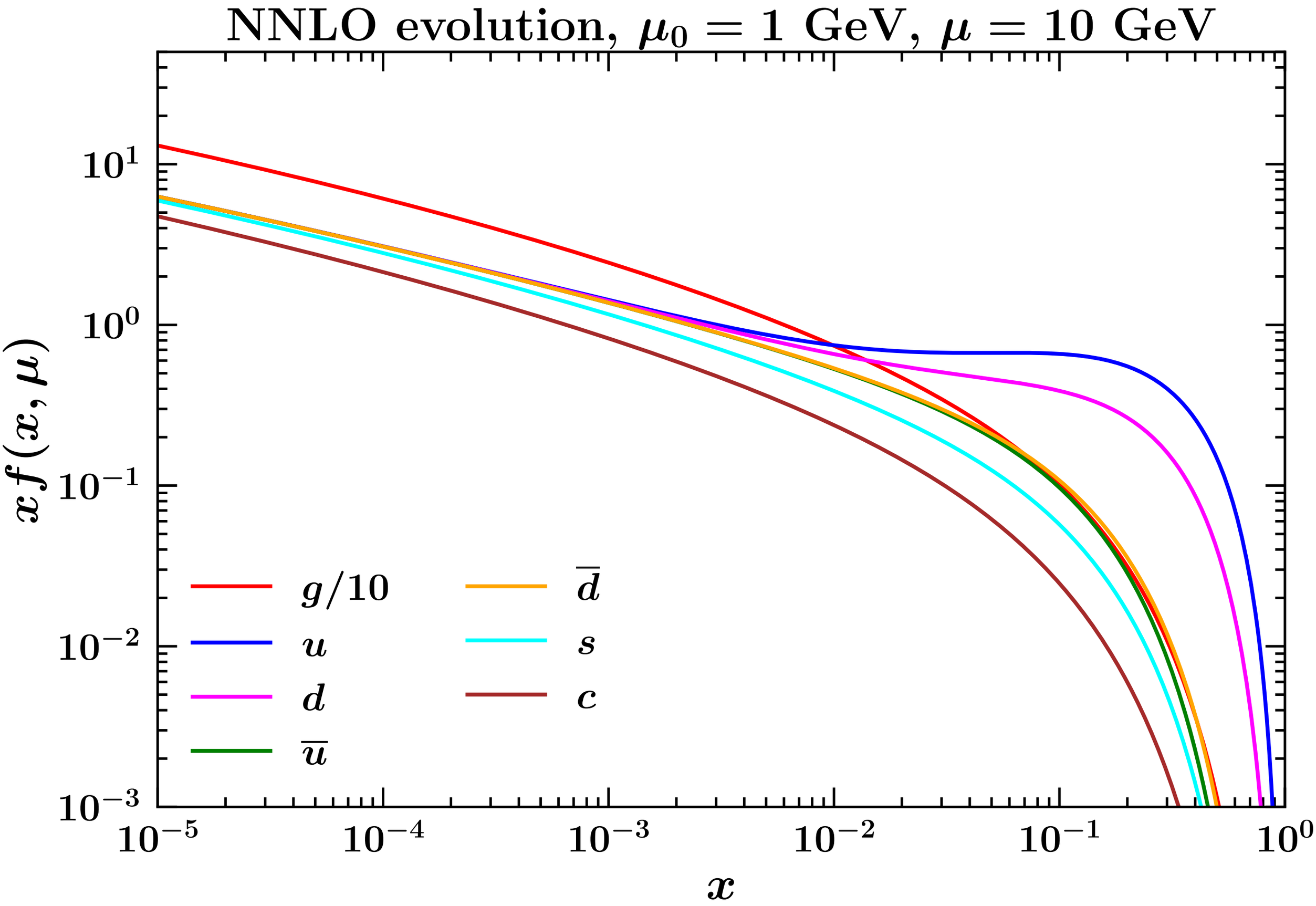
Numerical results



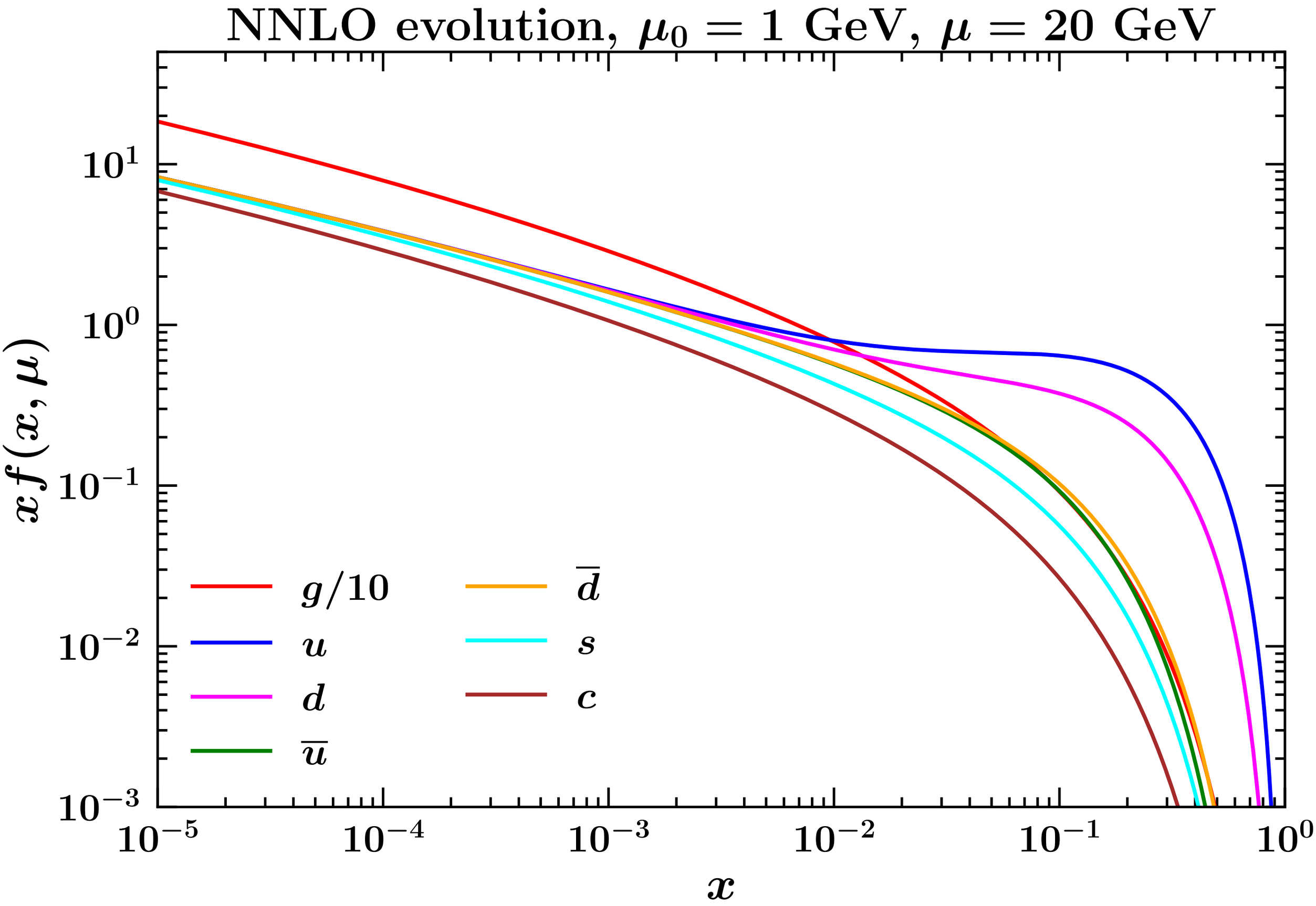
Numerical results



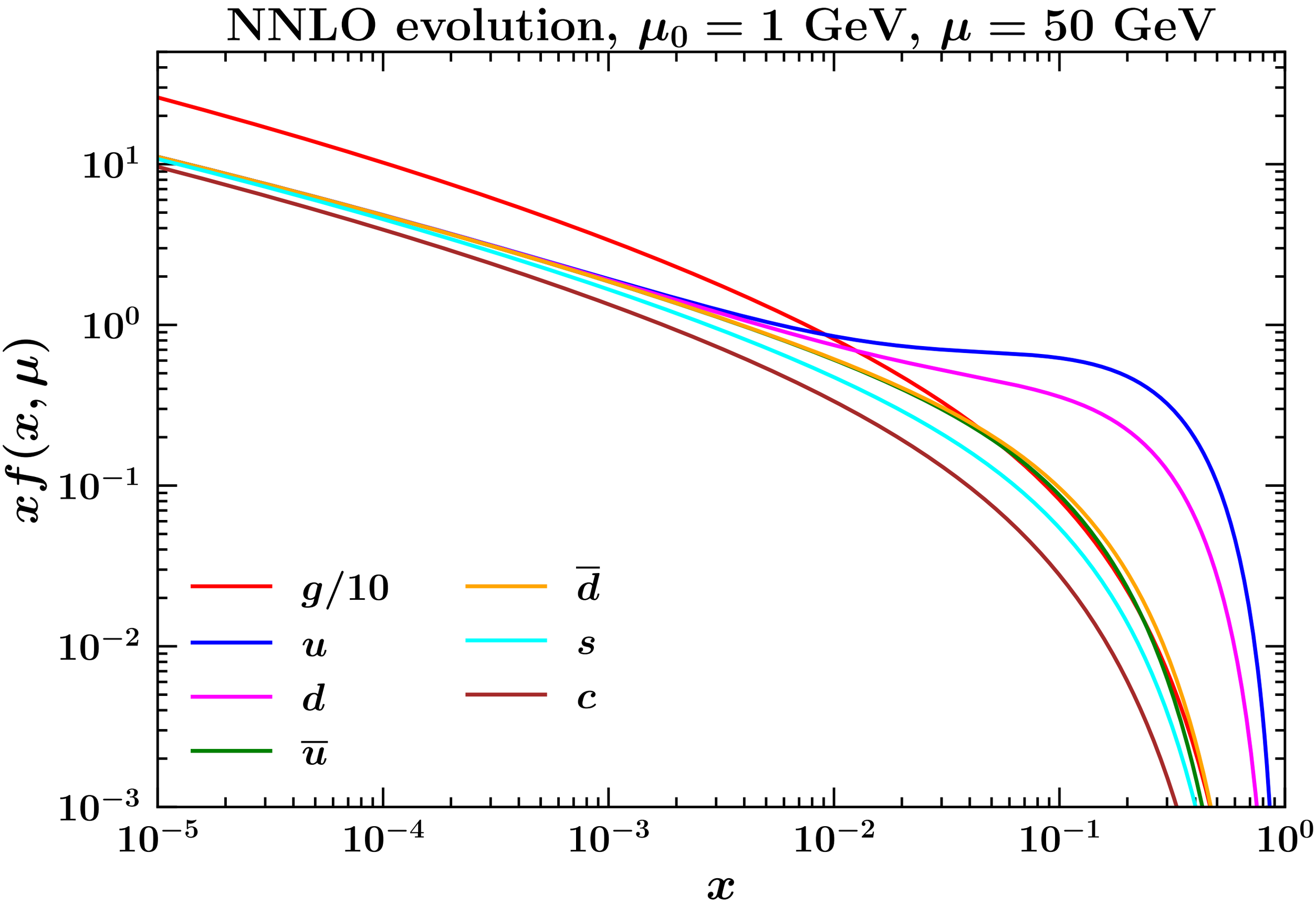
Numerical results



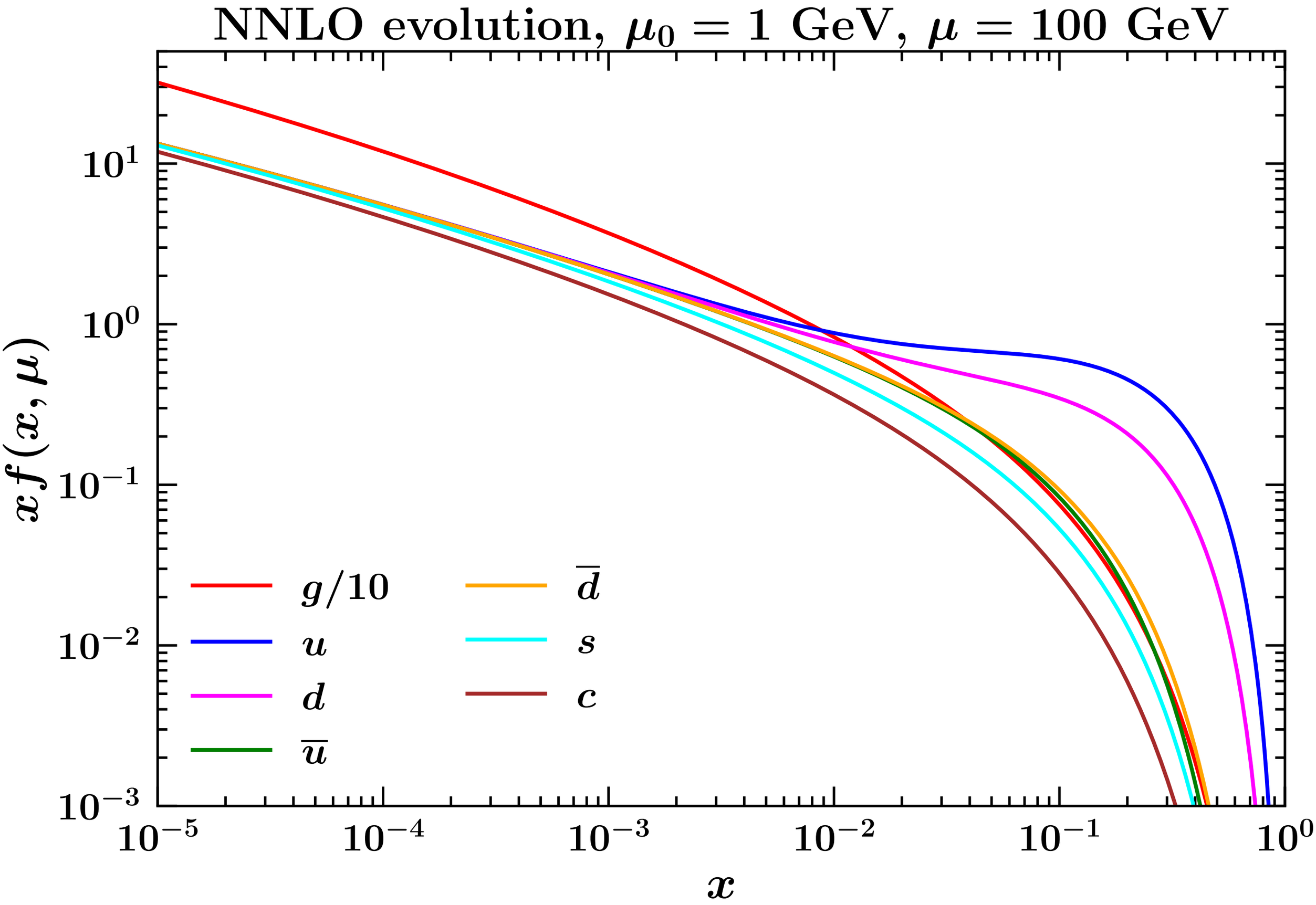
Numerical results



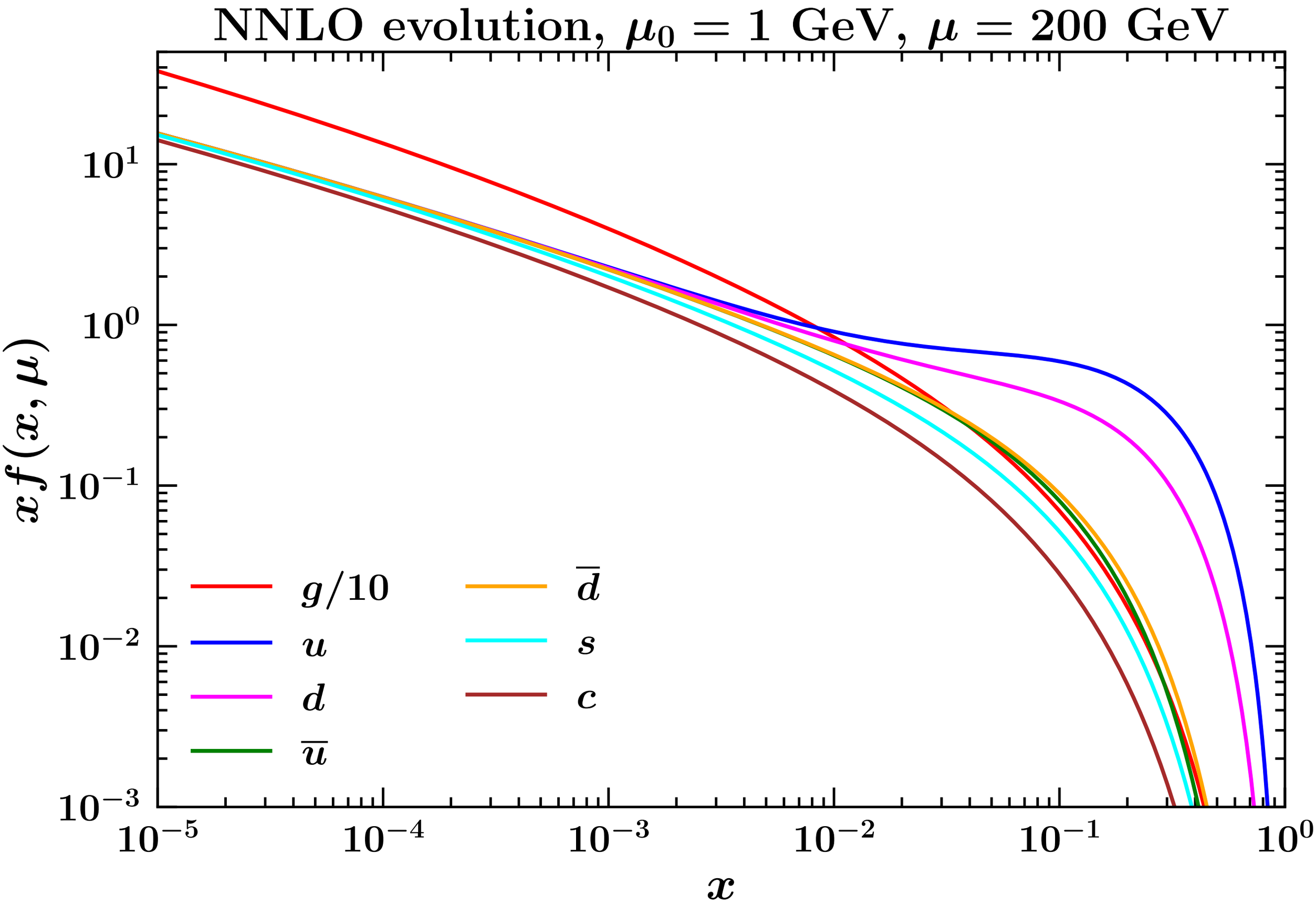
Numerical results



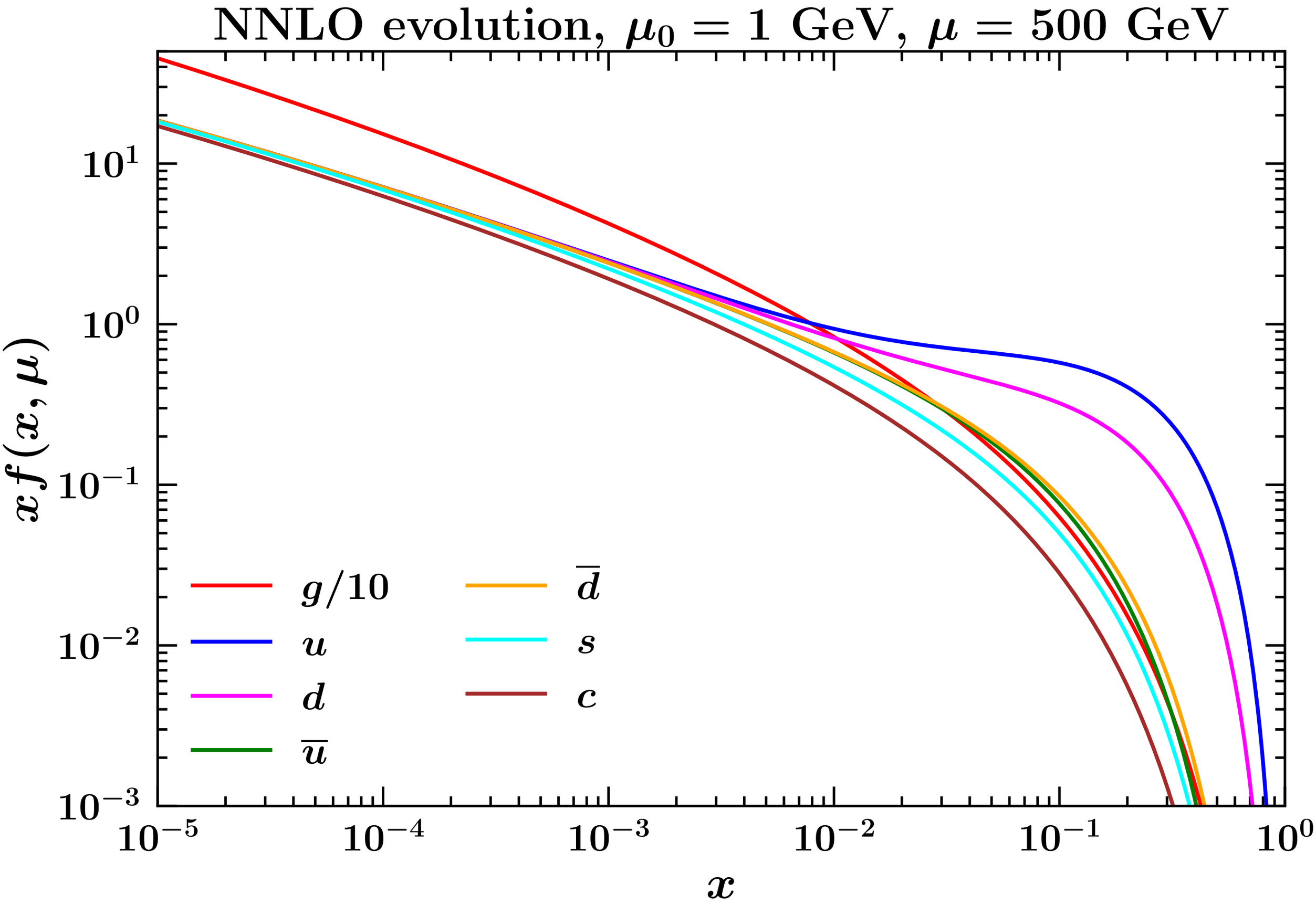
Numerical results



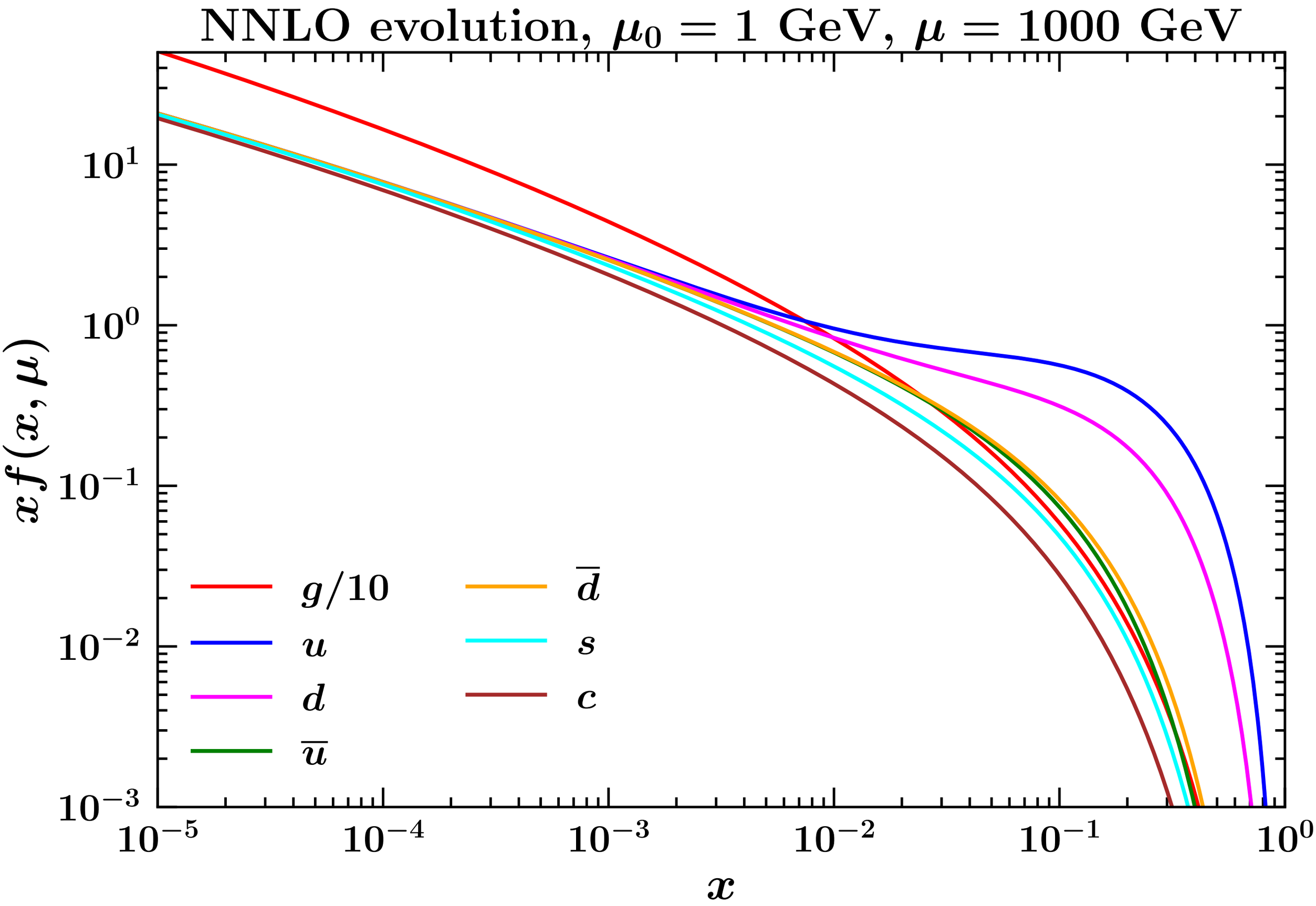
Numerical results



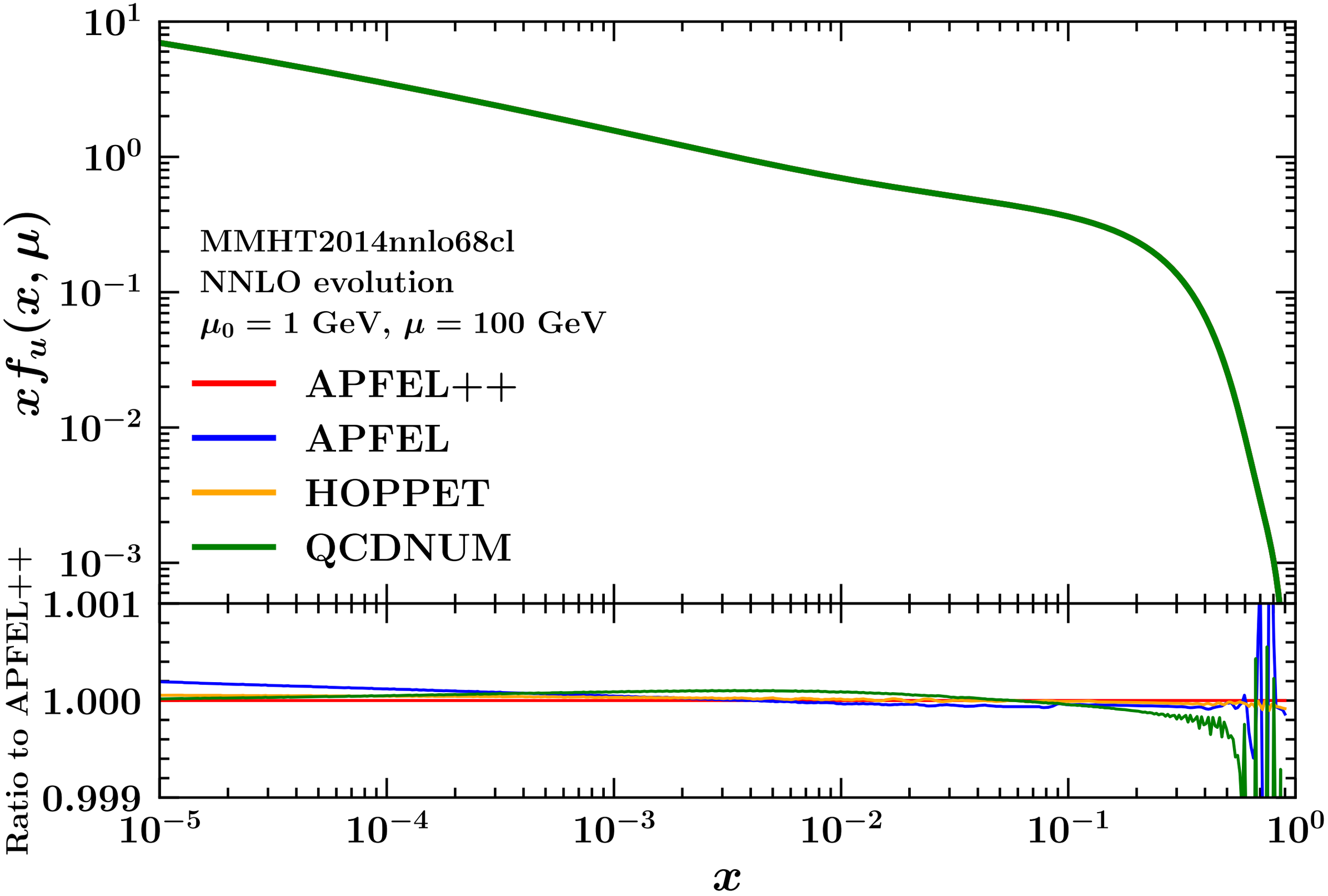
Numerical results



Numerical results



Comparison amongst codes

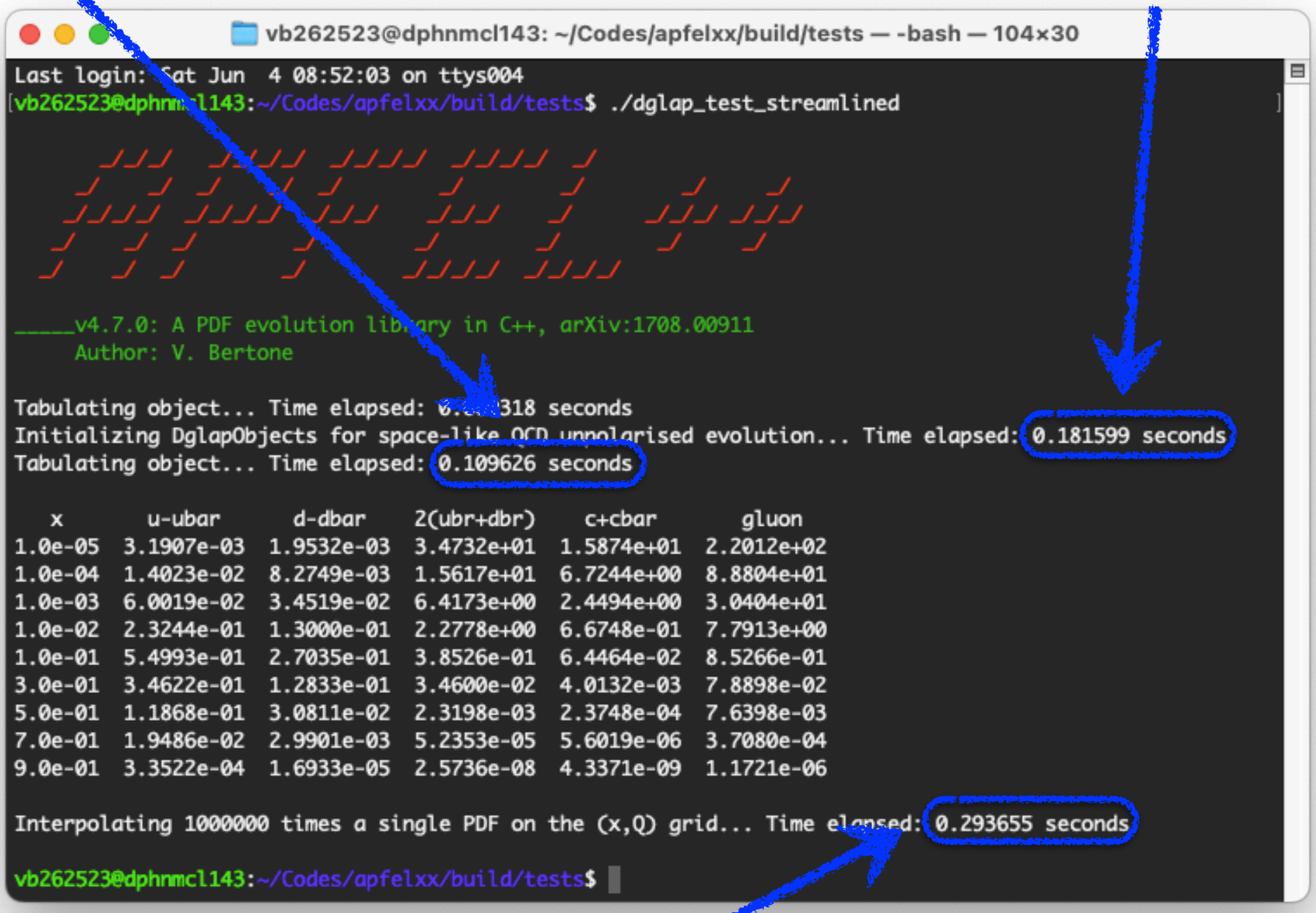


- Independent implementations agree well below the per-mil level!

Performance

Tabulation in μ
on a grid with 50 nodes,
interpolation degree 3

Computation of $\Gamma_{\beta\alpha}$ on a
x-space grid with ~250 nodes,
interpolation degree 3



Remarkable speed!

“Details” I didn’t discuss

- Computing the integrals $\Gamma_{\beta\alpha}$:
  - integrands are distributions that involve δ-functions and +-distributions.
- Addressing the problem of inaccuracies at large x caused by a logarithmically-spaced grid.
- The flavour dependence of parton distributions.
- The possible presence of heavy-quark thresholds along the evolution.
- The evolution equation for generalised parton distributions (GPDs):
  - complications due to skewed kinematics,
  - more complicated convolution integrals (principal-valued distributions and spurious divergences),
  - logarithmically-spaced grid does not help reduce the number of integrals.
- Alternative: the Mellin-moment approach.

Conclusions

- Evolution equations in QCD:
  - allow one to connect parton distributions at different scales,
  - when fitting parton distributions to data, we can parametrise them at one single scale and obtain them at any other scale,
  - fast evolution codes are required.
- Solving evolution equations in QCD is a numerically demanding task:
  - integro-differential equations in flavour space,
  - discretising the problem allows one reduce the problem to a set of coupled ODEs,
  - this however requires computing many integrals,
  - A smart choice of the grid and of the interpolation procedure allows one to lessen the numerical burden, finally reaching very high performance and accuracy.
- As of today, there exist several evolution codes whose mutual numerical agreement is remarkably good. Proof of the solidity of the procedure!