Der Wissenschaftsfonds.

Doktoratskolleg
Particles and Interactions

REvolver

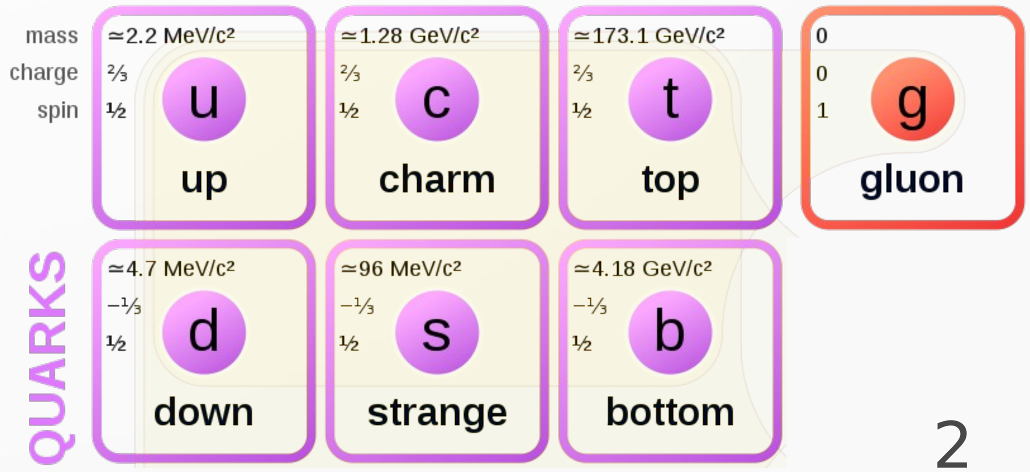
A fast code for automated running and matching of couplings and masses in QCD

Christopher Lepenik
in collaboration with André Hoang and Vicent Mateu

Mini Workshop on Quark Masses, KIT/Online, 2020-10-26

Preamble

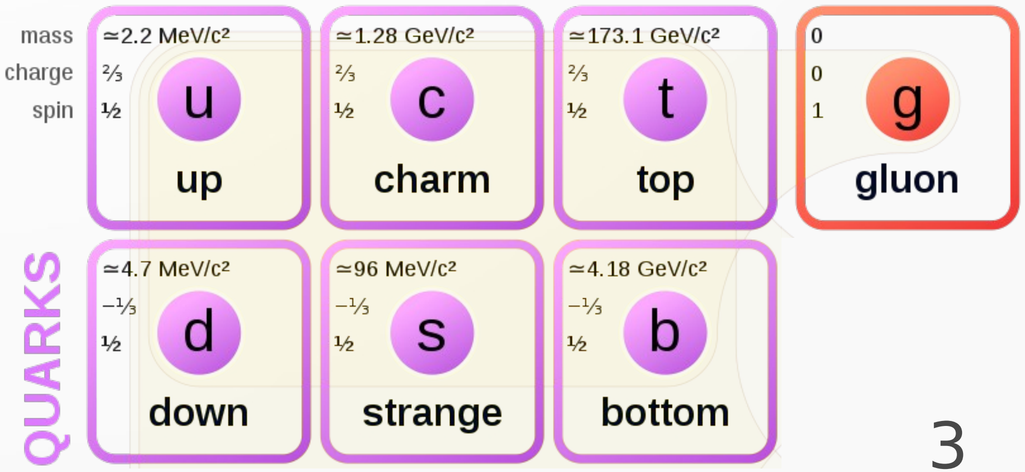
- REvolver is a code written in C++ (+MathLink +Python) to
  - Run quark masses (RGE)
    - Automatic mass decoupling at thresholds
  - Convert quark masses between renormalization schemes
    - Using R-Evolution to resum large logs
    - Including light flavor effects
  - Run QCD coupling (RGE)
    - Automatic mass decoupling



Preamble

- REvolver is a code written in C++ (+MathLink +Python) to
  - Run quark masses (RGE)
    - Automatic mass decoupling at thresholds
  - Convert quark masses between renormalization schemes
    - Using R-Evolution to resum large logs
    - Including light flavor effects
  - Run QCD coupling (RGE)
    - Automatic mass decoupling

→ To be released very soon
(beta available on request)



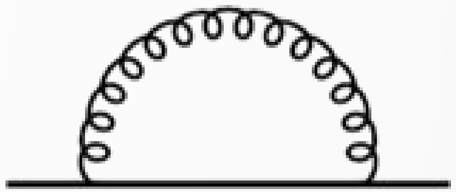
Outline

- Motivation
  - Quark Mass Renormalization Schemes
    - Mass Schemes & Conversions: Massless lighter flavors
    - Mass Schemes & Conversions: Massive lighter flavors
- Operating Principle of REvolver
- Live Demo, Documentation Overview

Motivation

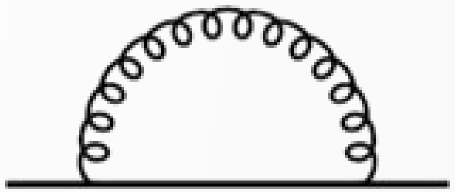
Quark Mass Schemes
Massless lighter Flavors

Quark Mass Schemes



- Quark masses:
  - Important parameters for SM predictions
  - Confinement → Quark masses not physical observables
    - Parameter in QCD action

Quark Mass Schemes



- Quark masses:
  - Important parameters for SM predictions
  - Confinement → Quark masses not physical observables
    - Parameter in QCD action
  - Quark self energy UV divergent → Needs to be absorbed into mass
    - Additional finite contributions – renormalization scheme dependent
    - Can choose appropriate mass scheme depending on application.
      → Need precise methods to convert between schemes

Mass Scheme Examples

$$m_Q^{\text{pole}} - \overline{m}_Q = \overline{m}_Q \sum_{n=1}^{\infty} a_n^{\overline{\text{MS}}} \left(\frac{\alpha_s^{(n_\ell+1)}(\overline{m}_Q)}{4\pi} \right)^n$$

[$n_\ell \cdots$ # massless quark flavors]

- $\overline{\text{MS}}$ mass:
  - Running mass
  - Analogous to $\alpha_s(\mu)$: Absorbs only UV $1/\varepsilon$ poles from self-energy
  - Intrinsic physical scale: $\overline{m}_Q$
  - Standard scheme for most high energy applications
  - Only physically relevant for $\mu \gtrsim \overline{m}_Q$
    - Lower scales: Virtual heavy quark effects should be integrated out

Mass Scheme Examples

$$m_Q^{\mathrm{1S}} - m_Q^{\mathrm{pole}} = M_B \sum_{n=1}^{\infty} c_n \left(\frac{\alpha_s^{(n_\ell)}(M_B)}{4\pi} \right)^n$$

- 1S mass:
  - Example of low scale mass
  - Defined as half of heavy quarkonium spin triplet ground state mass
    → Observable, well defined mass scheme
  - Intrinsic physical scale: Inverse Bohr radius $M_B \equiv C_F \alpha_s m_Q^{\mathrm{pole}}$

Mass Scheme Examples

$$m_Q^{1S} - m_Q^{\text{pole}} = M_B \sum_{n=1}^{\infty} c_n \left(\frac{\alpha_s^{(n_\ell)}(M_B)}{4\pi} \right)^n$$

- 1S mass:
  - Example of low scale mass
  - Defined as half of heavy quarkonium spin triplet ground state mass
    → Observable, well defined mass scheme
  - Intrinsic physical scale: Inverse Bohr radius $M_B \equiv C_F \alpha_s m_Q^{\text{pole}}$

- Many more well defined mass schemes: PS, RS, kinetic, jet, ...

MSR Scheme

$$m_Q^{\text{pole}} - m_Q^{\text{MSR}}(R) = R \sum_{n=1}^{\infty} a_n \left(\frac{\alpha_s^{(n_\ell)}(R)}{4\pi} \right)^n$$

$$\left[a_n = a_n^{\overline{\text{MS}}}(n_h = 0) \right]$$

- MSR mass:
  - Natural extension of $\overline{\text{MS}}$-mass for scales $\ll m_Q$
  - Defined directly from pole-$\overline{\text{MS}}$ relation
    - Heavy DOF integrated out → matching to $\overline{\text{MS}}$
  - Intrinsic physical scale: Adjustable momentum cut-off R

MSR Scheme

- Inherits cleanness and good infrared properties from $\overline{\mathrm{MS}}$
  - Low-scale short-distance mass with direct relation to self-energy diagrams

MSR Scheme

- Inherits cleanness and good infrared properties from $\overline{\mathrm{MS}}$
  - Low-scale short-distance mass with direct relation to self-energy diagrams

- Lighter massive quarks can be incorporated systematically via matching → flavor-number dependent RG evolution
  → Later

MSR Scheme

- Possible interpretation of R:
  - MSR mass contains self-energy corrections only for scales larger than R

MSR Scheme

- Possible interpretation of R:
  - MSR mass contains self-energy corrections only for scales larger than R

- Pole mass: Formal limit $R \to 0$
  - Absorbs all contributions from quark self energy
    → sensitivity to non-perturbative regions
  - Suffers from $\mathcal{O}(\Lambda_{\mathrm{QCD}})$ renormalon in QCD

$$m_Q^{\mathrm{pole}} - m_Q^{\mathrm{MSR}}(R) = R \sum_{n=1}^{\infty} a_n \left(\frac{\alpha_s^{(n_\ell)}(R)}{4\pi} \right)^n$$

Mass Scheme Conversion

- FOPT: Common scale μ and flavor number n_f have to be used for all α_s → renormalon cancellation
  - Potentially large logs when converting mass schemes with different characteristic scales

Mass Scheme Conversion

- FOPT: Common scale μ and flavor number n_f have to be used for all α_s → renormalon cancellation
  - Potentially large logs when converting mass schemes with different characteristic scales

- Solution: MSR / R-Evolution
  - Utilize freely adjustable intrinsic scale R
  - As intermediate step – RG running between scales

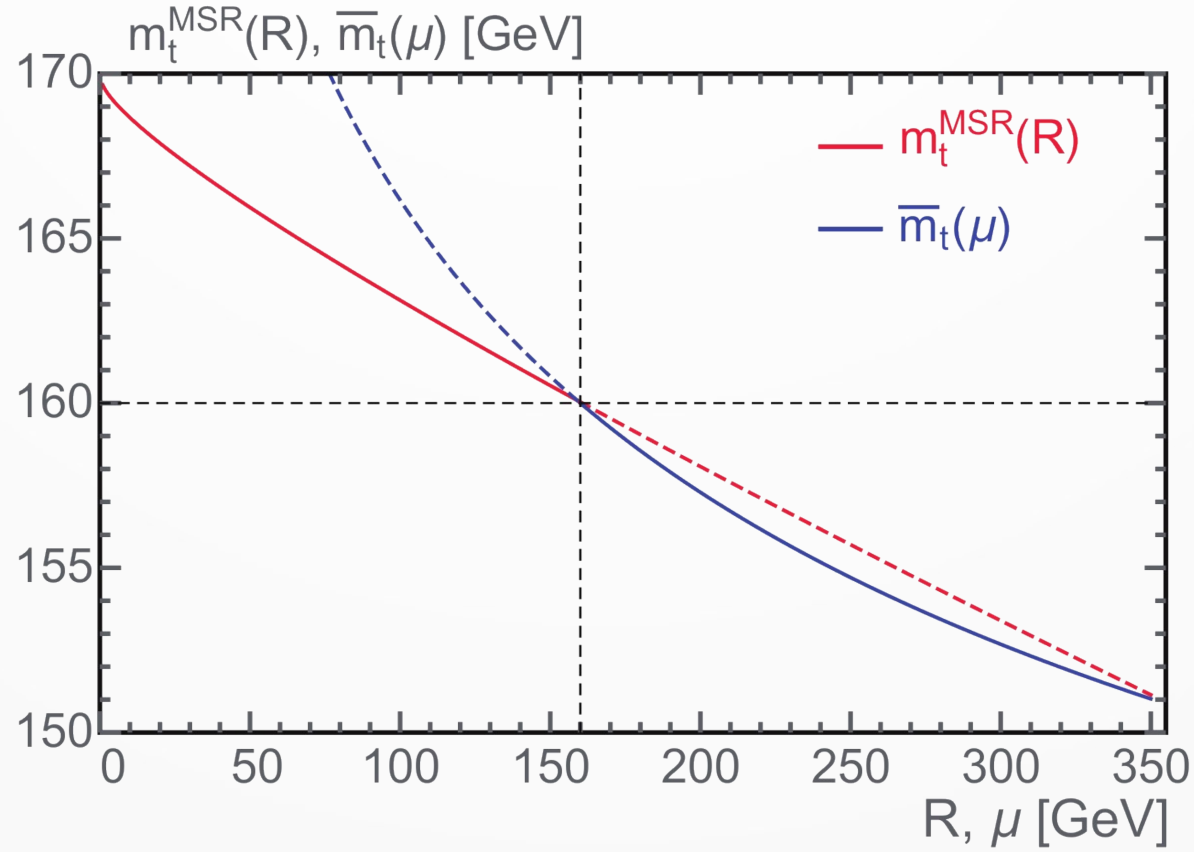
R-Evolution

$$R\frac{\mathrm{d}m_Q^{\mathrm{MSR}}(R)}{\mathrm{d}R} = -R\gamma^R(\alpha_s(R)) = -R\sum_{n=0}^{\infty}\gamma_n^R\left(\frac{\alpha_s(R)}{4\pi}\right)^{n+1}$$

- RGE in IR scale R, relating MSR masses at different scales
  - Sums systematically renormalon series and large logs
  - Linear dependence on R

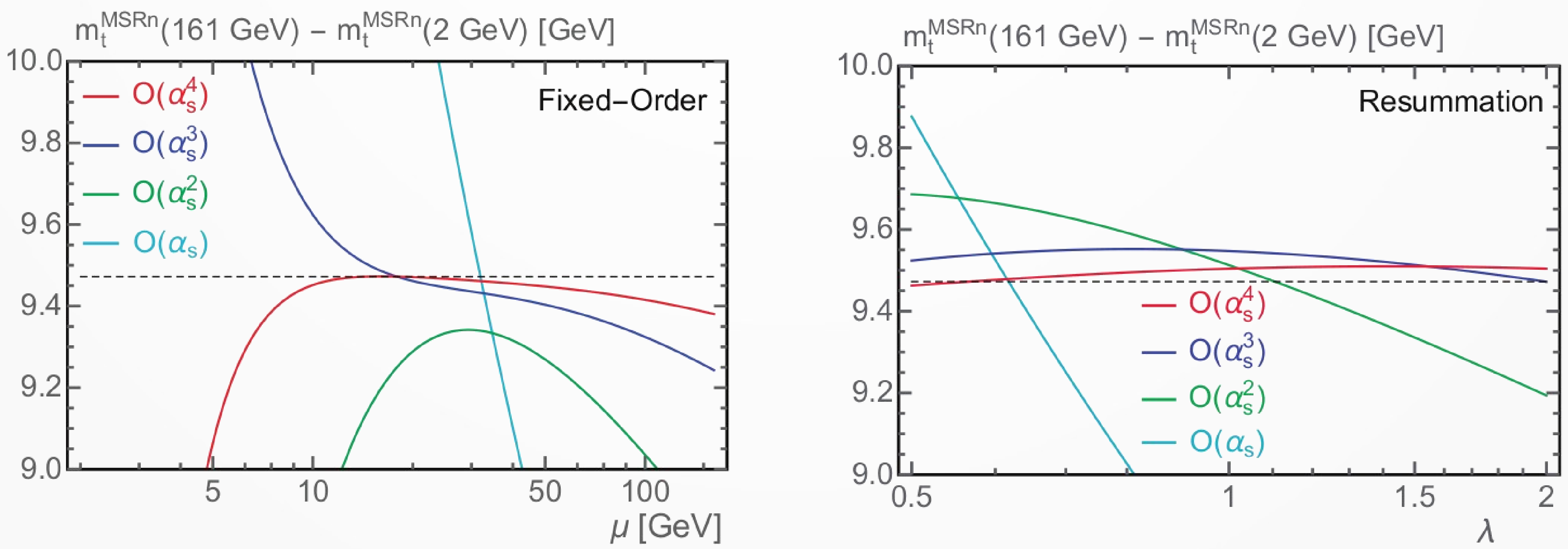
R-Evolution

- MS-scheme running: Large scale ($\overline{\mathrm{MS}}$) and Low scale (MSR)



R-Evolution

- Fixed order vs. R-Evolution




R|EVOLUTION
The First 2000 Years of Computing

Motivation

Quark Mass Schemes

Massive lighter Flavors

Lighter Massive Flavors

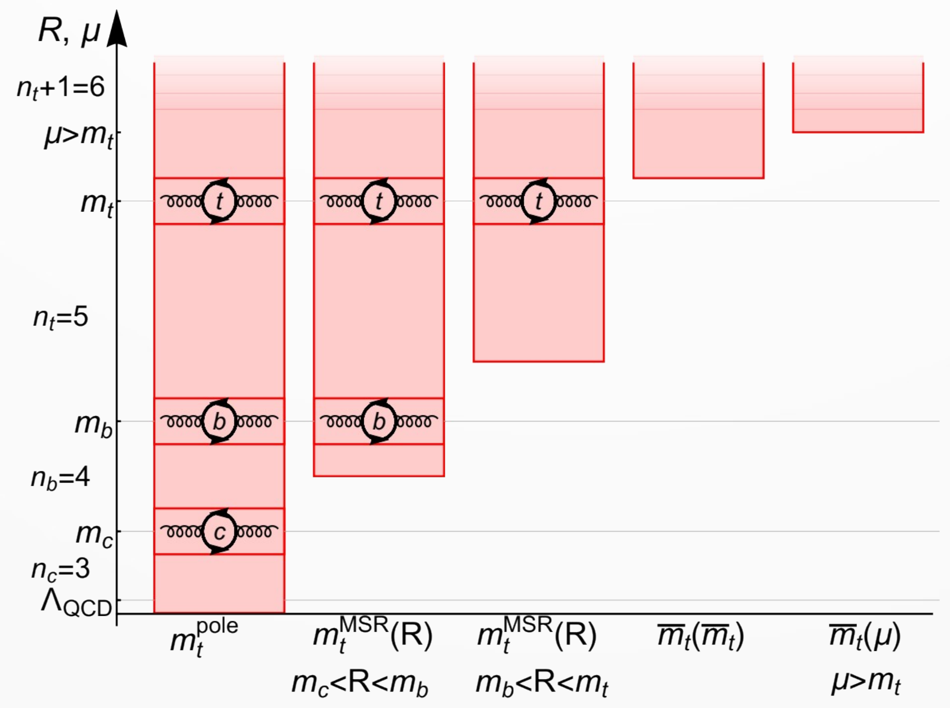
- Many applications: Lighter massive quarks set massless
  - Needed in high precision calculations

- Corrections known for some schemes

Lighter Massive Flavors

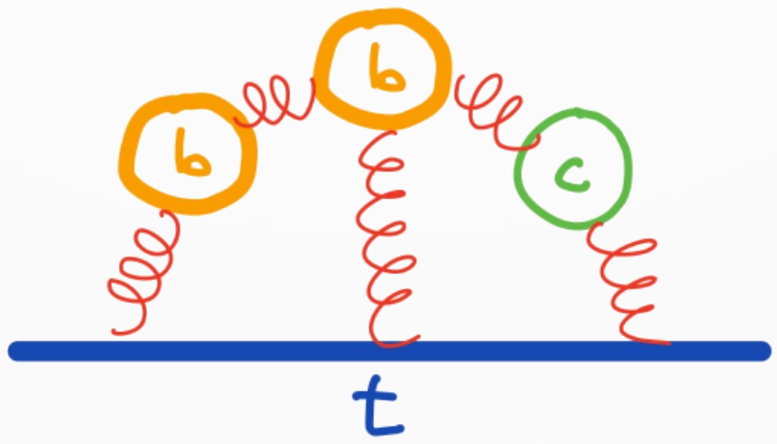
- Many applications: Lighter massive quarks set massless
  - Needed in high precision calculations

- Corrections known for some schemes
- MSR scheme: Systematic treatment via flavor matching



Lighter Massive Flavors

- Scheme conversions & light massive flavors
  - Impact at higher orders
    → Massive virtual quark loops act as IR-cutoff
    → Change renormalon structure



Lighter Massive Flavors

- Scheme conversions & light massive flavors
  - Impact at higher orders
    → Massive virtual quark loops act as IR-cutoff
    → Change renormalon structure

- Consistency is crucial
  - when converting to/from pole mass
  - when converting between schemes

REvolver

Features

REvolver Features

- All implemented in REvolver:
  - MSR scheme as low-scale extension of $\overline{\mathrm{MS}}$ → “MS-scheme”
    - R-Evolution
    - Flavor matching

REvolver Features

- All implemented in REvolver:
  - MSR scheme as low-scale extension of $\overline{\text{MS}}$ → “MS-scheme”
    - R-Evolution
    - Flavor matching
  - Mass conversion to/from MS scheme
    - using R-Evolution to resum logs
    - including light massive flavor contributions (if known)
    - Various schemes supported: MS, 1S, PS, RS, kinetic, pole, …

REvolver Features

- All implemented in REvolver:
  - MSR scheme as low-scale extension of $\overline{\mathrm{MS}}$ → “MS-scheme”
    - R-Evolution
    - Flavor matching
  - Mass conversion to/from MS scheme
    - using R-Evolution to resum logs
    - including light massive flavor contributions (if known)
    - Various schemes supported: MS, 1S, PS, RS, kinetic, pole, ...
- Highest available orders implemented for running/matching
  - 5 loop α_s, 5/4 loop $\overline{\mathrm{MS}}$/MSR, ...

REvolver

Operating Principle

Operating Principle of REvolver

$$n_f^{\text{tot}}, \alpha_s^{(n_\alpha)}(\mu_\alpha), \{m_i^{(n_{m_i})}(R_i)\}, \text{optional parameters}$$

Operating Principle of REvolver

$$n_f^{\text{tot}}, \alpha_s^{(n_\alpha)}(\mu_\alpha), \{m_i^{(n_{m_i})}(R_i)\}, \text{optional parameters}$$

Core

$n_f^{\text{tot}},\ n_f^{m=0}$

$m_0^{(n_0)}(\overline{m}_0),\ m_0^{(n_1)}(\overline{m}_1),\ \ldots$

$m_1^{(n_0)}(\overline{m}_0),\ m_1^{(n_1)}(\overline{m}_1),\ \ldots$

$\vdots$

$\alpha_s^{(n_0)}(\overline{m}_0),\ \alpha_s^{(n_1)}(\overline{m}_1),\ \ldots$

optional parameters

$\Lambda_{\text{QCD}}^{(n_0)},\ \ldots$

perturbative coefficients, ...

Operating Principle of REvolver

$n_f^{\mathrm{tot}}, \alpha_s^{(n_\alpha)}(\mu_\alpha), \{m_i^{(n_{m_i})}(R_i)\}$, optional parameters

Request quantity via member function

Core

$n_f^{\mathrm{tot}},\ n_f^{m=0}$

$m_0^{(n_0)}(\overline{m}_0),\ m_0^{(n_1)}(\overline{m}_1),\ \ldots$

$m_1^{(n_0)}(\overline{m}_0),\ m_1^{(n_1)}(\overline{m}_1),\ \ldots$

$\vdots$

$\alpha_s^{(n_0)}(\overline{m}_0),\ \alpha_s^{(n_1)}(\overline{m}_1),\ \ldots$

optional parameters

$\Lambda_{\mathrm{QCD}}^{(n_0)},\ \ldots$

perturbative coefficients, ...

Operating Principle of REvolver

$$n_f^{\text{tot}}, \alpha_s^{(n_\alpha)}(\mu_\alpha), \{m_i^{(n_{m_i})}(R_i)\}, \text{optional parameters}$$

Request quantity via member function

Core

n_f^{tot}, $n_f^{m=0}$

$m_0^{(n_0)}(\overline{m}_0)$, $m_0^{(n_1)}(\overline{m}_1)$, $\ldots$

$m_1^{(n_0)}(\overline{m}_0)$, $m_1^{(n_1)}(\overline{m}_1)$, $\ldots$

$\vdots$

$\alpha_s^{(n_0)}(\overline{m}_0)$, $\alpha_s^{(n_1)}(\overline{m}_1)$, $\ldots$

optional parameters

$\Lambda_{\text{QCD}}^{(n_0)}$, $\ldots$

perturbative coefficients, ...

$\alpha_s^{(n_f)}(\mu)$

$m_i^{(n_f)}(\mu)$

$m_i^{\text{PS}}(\mu_f)$

$\vdots$

Operating Principle of REvolver

- Three ways to use the code:
  - directly via C++ library
    - Fast
    - Most suitable for extensive, automated tasks
    - Interaction with other C++ libraries
  - via WSTP/MathLink in Mathematica
    - User friendly
    - Most suitable for interactive tasks
    - Interaction with Mathematica features
  - via Python interface
    - Scripts
    - Interactive execution in Jupyter Notebooks

Operating Principle of REvolver

- Three ways to use the code:

  - directly via C++ library
    - Fast
    - Most suitable for extensive, automated tasks
    - Interaction with other C++ libraries
  - via WSTP/MathLink in Mathematica
    - User friendly
    - Most suitable for interactive tasks
    - Interaction with Mathematica features
  - via Python interface
    - Scripts
    - Interactive execution in Jupyter Notebooks

REvolver

Live Demo
and
Documentation Overview