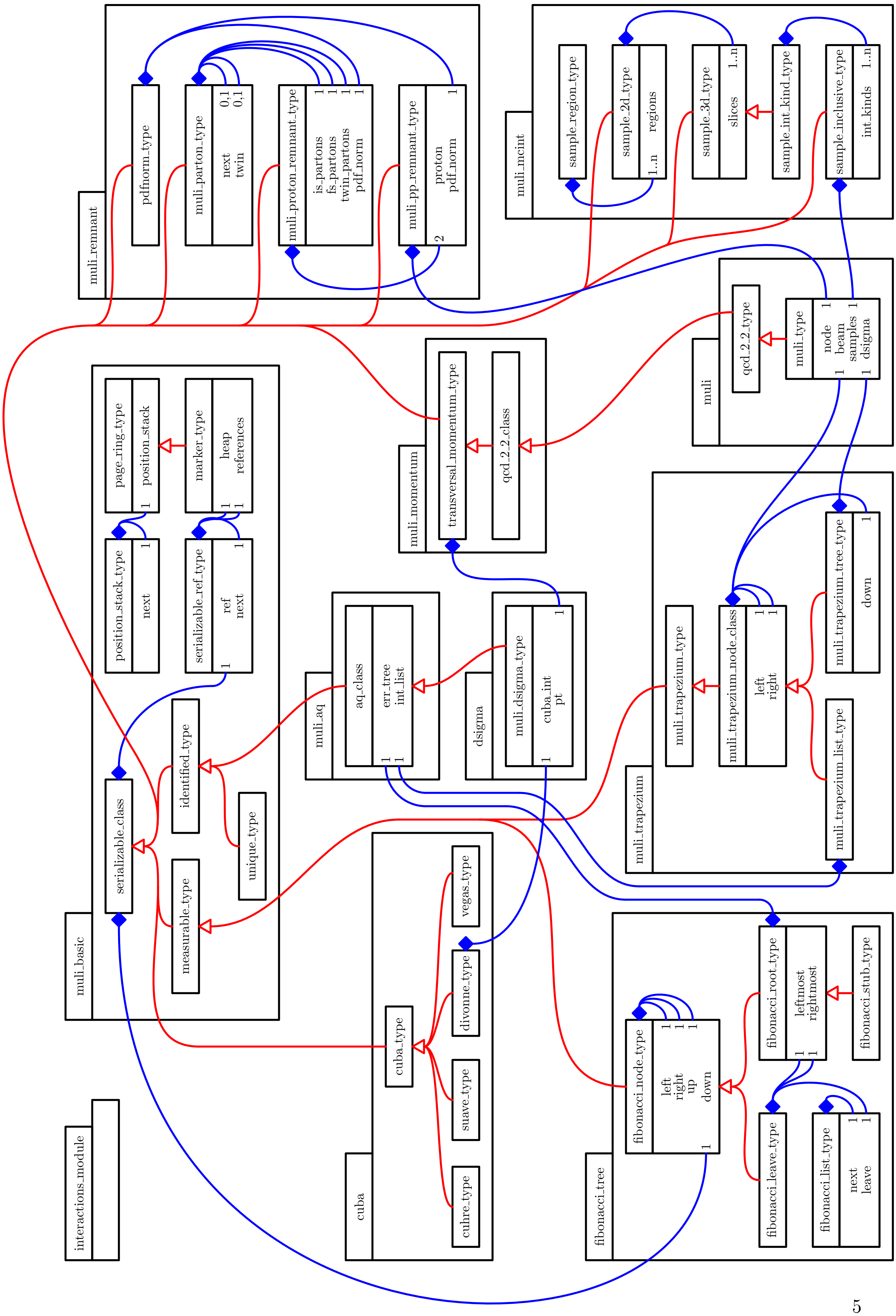
Inhaltsverzeichnis

Teil I

Allgemeines



muli_remnant
pdfnorm_type
muli_parton_type
next
twin
0,1
0,1
muli_proton_remnant_type
is_partons
fs_partons
twin_partons
pdf_norm
1
1
1
1
muli_pp_remnant_type
proton
pdf_norm
2
1
muli_mcint
sample_region_type
sample_2d_type
regions
1..n
sample_3d_type
slices
1..n
sample_int_kind_type
sample_inclusive_type
int_kinds
1..n
muli
qcd_2_2_type
muli_type
node
beam
samples
dsigma
1
1
1
1
muli_momentum
transversal_momentum_type
qcd_2_2_class
muli_basic
page_ring_type
position_stack
1
position_stack_type
next
1
marker_type
heap
references
1
1
serializable_ref_type
ref
next
1
1
serializable_class
identified_type
unique_type
measurable_type
muli_aq
aq_class
err_tree
int_list
1
1
dsigma
muli_dsigma_type
cuba_int
pt
1
1
muli_trapezium
muli_trapezium_type
muli_trapezium_node_class
left
right
1
1
muli_trapezium_tree_type
down
1
muli_trapezium_list_type
interactions_module
cuba
cuba_type
vegas_type
divonne_type
suave_type
cuhre_type
fibonacci_tree
fibonacci_node_type
left
right
up
down
1
1
1
1
fibonacci_root_type
leftmost
rightmost
1
1
fibonacci_stub_type
fibonacci_leave_type
fibonacci_list_type
next
leave
1
1

1 Nomenklatur

1.1 n-te Wechselwirkung

Durch den Algorithmus werden iterativ harte, partonische, treelevel, QCD $2 \to 2$ Wechselwirkungen mit absteigenden Wechselwirkungsskalen $p_\perp^{(n)}$ generiert. Variablen, die nach jeder harten Wechselwirkung einen neuen Wert erhalten, führen die Ordnungszahl n der aktuellen Wechselwirkung hochgestellt in runden Klammern, um Verwechslungen mit Potenzen zu vermeiden.

Diese Ordnungszahl wird mit k bezeichnet, wenn sie sich nicht auf die aktuelle Wechselwirkung bezieht, sondern Summations- oder Produktindex über alle bisherigen Wechselwirkungen ist. Die Ordnungszahl der letzten Wechselwirkung wird mit N notiert. Bevor eine harte Wechselwirkung stattfindet, ist die Ordnungszahl gleich Null und kann weggelassen werden.

1.2 Impulse

Die Viererimpulse der Remnants sind $P_1^{(n)}$ bzw. $P_2^{(n)}$ für das erste bzw. das zweite Proton. Die Viererimpulse der Partonen sind $\hat{p}_1^{(n)}$ bzw. $\hat{p}_2^{(n)}$ für das erste bzw. das zweite Proton. Die Viererimpulse der Teilchen im Endzustand der partonischen Wechselwirkung sind für MulI nicht von Bedeutung.

Die Kinematik eines partonischen Ereignisses wird vollständig durch das kartesische Quadrupel $p_{\text{cart}}^{(n)} = \left[x_1^{(n)}, x_2^{(n)}, p_\perp^{(n)}; s^{(n)}\right]$ bzw. das hyperbolische Quadrupel $p_{\text{hyp}}^{(n)} = \left[h_1^{(n)}, h_2^{(n)}, h_3^{(n)}; s^{(n)}\right]$ definiert. Die entsprechende Koordinatentransformation ist in (**??**) angegeben. Es wird nicht zwischen der Bjorken-Scaling-Variable x und dem Impulsanteil mit $xP = \hat{p}$ unterschieden. Wir nehmen an, dass die kinetische Energie der Protonen viel größer als die Ruhemasse der Protonen ist. In diesem Grenzwert stimmen beide Variablen überein.

1.3 Flavor

Quarks und Gluonen der n-ten harten Wechselwirkung haben Flavorindizes $a^{(n)}, b^{(n)}, c^{(n)}, d^{(n)}$, wobei a das Flavor des Partons aus dem Remnant 1 und b das Flavor des Partons aus dem Remnant 2 ist. Wenn nicht anders angegeben, wird das LHAPDF-Schema verwendet, mit $[\overline{t}, \overline{b}, \overline{c}, \overline{s}, \overline{u}, \overline{d}, g, d, u, s, c, b, t] = [-6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6]$.

1.4 Strukturfunktionen

Strukturfunktionen $f(x, \mu)$ sind in diesem Dokument synonym zu Flavor-Strukturfunktionen, die Impuls-Strukturfunktionen ergeben sich dann aus $xf(x, \mu)$. Im Gegensatz dazu liefert evolvePDF aus LHAPDF die Impulsstrukturfunktion. MulI hat diesbezüglich die gleiche Konvention wie die Builtin-PDFs aus WHIZARD, welche ebenfalls Flavor-PDFs liefern.

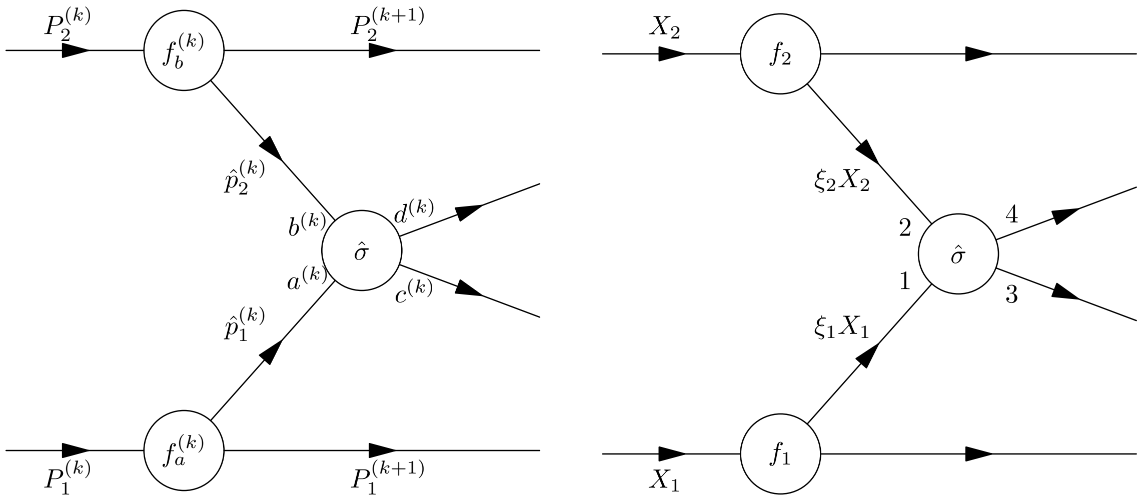


Abbildung 1.1: Links: Impulsvariablen P_x der Remnants, Impulsvariablen $\hat{p}_x$ der Partonen und Flavorindizes $a^{(k)}, b^{(k)}, c^{(k)}, d^{(k)}$ der Partonen in der k-ten Iteration des Multiple Interactions Algorithmus. Rechts: Die Prozeduren zur Generierung der Ereignisse kennen üblicherweise nicht die Ordnungszahl k nur einen Teil der Impulsinformation, nämlich die hadronischen Impulsanteile X mit $P^{(k)} = XP$ und die partonischen Impulsanteile ξ mit $\hat{p}^{(k)} = \xi P^{(k)} = xP$. Anstatt der Flavorindizes $a^{(k)}, b^{(k)}, c^{(k)}, d^{(k)}$ ist die festgelegte Position in dem Flavorquadrupel eingetragen.

Da beide Remnants eine verscheidene Historie haben können, sind im Allgemeinen auch beide Remnant-Strukturfunktionen verschieden. Die Zugehörigkeit wird durch den Flavorindex a für das erste und b für das zweite Proton notiert. Wir verzichten auf die Indizierung der Flavorindizes, also $a^{(n)} \to a$, da die Strukturfunktionen bereits Ordnungsindizes (n) haben. Wir erhalten $f_a^{(n)}\left(x_1^{(n)}, \mu_F^{(n)}\right)$ für das erste Proton und $f_b^{(n)}\left(x_2^{(n)}, \mu_F^{(n)}\right)$ für das zweite Proton.

Derzeit sind im Modul muli_remnant Proton-Strukturfunktionen fest implementiert, es können also ohne Eingriff in den Code z.B. keine Proton-Antiproton Streuungen generiert werden. Allerdings sollte die Verallgemeinerung auf Hadronen mit maximal zwei verschiedenen Valenzquarks kaum Probleme bereiten, da die Infrastruktur des Moduls nicht geändert werden muss.

1.5 Wirkungsquerschnitte

Wie die Strukturfunktionen ändern sich auch die Wirkungsquerschnitte σ mit jeder Iteration. Allerdings können wir die Abhängigkeit komplett in die Änderung der invarianten Masse $s^{(n)}$ der n-ten Iteration absorbieren. Wir notieren also keinen Ordnungsindex (n), sondern fügen die invariante Masse als Parameter durch ein Semikolon getrennt in die Liste der Argumente ein. Wir erhalten für den hadronischen Wirkungsquerschnitt σ

$$\sigma_{ab\to cd}^{(n)}\left(x_1^{(n)}, x_2^{(n)}, p_\perp^{(n)}\right) = \sigma_{ab\to cd}\left(x_1^{(n)}, x_2^{(n)}, p_\perp^{(n)}; s^{(n)}\right). \tag{1.1}$$

Der hadronische Wirkungsquerschnitt σ bezieht sich auf das Streuereignis, wie in Abbildung **??** dargestellt. Der partonische Wirkungsquerschnitt hingegen ist $\widehat{\sigma}$.

1.6 Übersicht

$$s = s^{(1)} = P_1 \cdot P_2 \tag{1.2}$$

$$s^{(n)} = P_1^{(n)} \cdot P_2^{(n)} \tag{1.3}$$

$$\hat{p}_1^{(n)} = P_1^{(n)} x_1^{(n)} \tag{1.4}$$

$$P_1^{(n+1)} = P_1^{(n)} \big(1 - x_1^{(n)}\big) \tag{1.5}$$

$$X^{(n)} = \prod_{k=1}^{n} \Big(1 - x^{(k)}\Big) \tag{1.6}$$

$$P_1^{(n+1)} = X_1^{(n)} P^{(1)} \tag{1.7}$$

$$p_\perp = \frac{\hat{t}\hat{u}}{\hat{s}} \tag{1.8}$$

2 Der Algorithmus

Der Algorithmus ist in meiner Dissertation in Kapitel 5 bereits dokumentiert, deswegen werde ich hier nicht alle Aspekte wiederholen. In der Dissertation wird allerdings nicht sorgfältig zwischen fertigen und geplanten Eigenschaften getrennt, deswegen gebe ich hier einen groben Überblick über den aktuellen Stand.

MulI wird derzeit ausschließlich von dem shower_interface aus dem interleaved Branch aus dem schmidtboschmann Verzeichnis des WHIZARD-Repositories aufgerufen. Es ist noch kein MulI-Code in den WHIZARD-Core übertragen worden, stattdessen sind alle relevanten Daten in einem erweiterten Datentyp muli_type gekapselt. muli_type stellt ebenfalls eine vollständige Schnittstelle bereit, um MPI zu generieren und Remnant-PDFs abzurufen. Die derzeit verwendeten Methoden dieser Schnittstelle sind in Tabelle **??** aufgeführt.

2.1 Stratified Sampling

Die Wahrscheinlichkeit dafür, dass die Wechselwirkung aus dem Stratum $\{\alpha, \beta\}$ mit der größten Skala $p_\perp \leq p_\perp^{(n-1)}$ bei der Skala $p_\perp^{(n-1)}$ stattfindet, ist durch

$$\mathcal{P}^{(n)}_{\mathrm{next},a,b}\left(p_\perp^{(n)}; p_\perp^{(n-1)}, s^{(n)}\right) := \exp\left[W_a^{(n)} W_b^{(n)}\left[\mathcal{S}_{\alpha\beta}\left(p_\perp^{(n)}; s^{(n)}\right) - \mathcal{S}_{\alpha\beta}\left(p_\perp^{(n-1)}; s^{(n)}\right)\right]\right] \tag{2.1}$$

gegeben. $s^{(n)}$ ist die invariante Masse des Remnant-Remnant-Systems, $W_\alpha^{(n)}$ und $W_\beta^{(n)}$ sind die Wichtungsfaktoren des Stratums α bzw. β und $\mathcal{S}_{\alpha\beta}$ ist das Stammstratum mit

$$\mathcal{S}_{\alpha\beta}\left(p_\perp^{(n)}; s^{(n)}\right) := \int_{p_\perp^{\max}}^{p_\perp^{(n)}} \mathrm{d}\, p_\perp \overline{S}_{\alpha\beta}\left(p_\perp; s^{(n)}\right). \tag{2.2}$$

Generischer Name	Spezifischer Name
muli_type%initialize	muli_initialize
muli_type%restart	muli_restart
muli_type%finalize	muli_finalize
muli_type%apply_initial_interaction	muli_apply_initial_interaction
muli_type%generate_gev2_pt2	muli_generate_gev2_pt2
muli_type%generate_partons	muli_generate_partons
qcd_2_2_type%get_color_correlations	qcd_2_2_get_color_correlations
muli_type%replace_parton	muli_replace_parton
muli_type%get_parton_pdf	muli_get_parton_pdf
muli_type%get_momentum_pdf	muli_get_momentum_pdf

Tabelle 2.1: Die Methoden der MulI Schnittstelle für den Interleaved-Algorithmus.

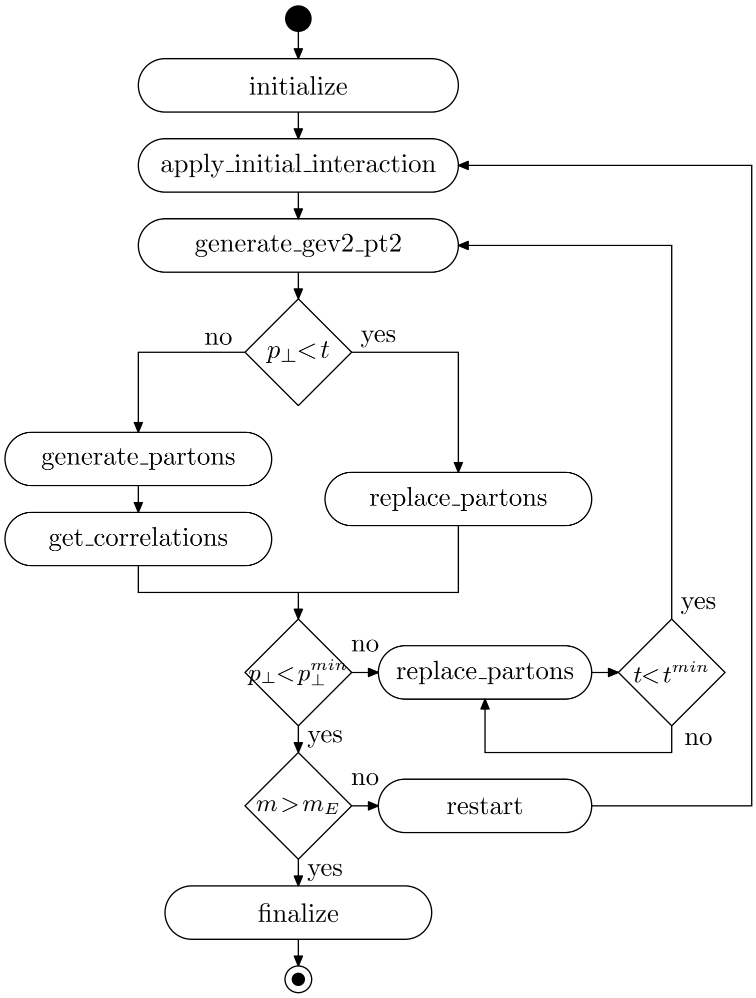


Abbildung 2.1: Flussdiagramm des Interleaved-Algorithmus. Eingetragen sind ausschließlich die Aufrufe der MulI-Schnittstelle, mit Ausnahme von get_parton_pdf und get_momentum_pdf, da diese nur für Interna des Partonshowers relevant sind und das Diagramm unnötig kompliziert machen würden. Die Generierung der Showerskalen und -Teilchen ist hier nicht dargestellt. Parallel zu generate_gev2_pt2 wird von dem ISR-Modul eine Showerskala t generiert und unmittelbar vor replace_parton wird von dem ISR-Modul ein neues Showerteilchen generiert, dass durch replace_parton in die Beschreibung des Remnants aufgenommen wird. m_E ist die Zahl der zu generierenden Events.

Stratum	Name	Partonen
S_1	Gluon	g
S_2	See	$\{q^S : \forall q\}$
S_3	Valenz-Down	d^V
S_4	Valenz-Up	u^V

(a)

Stratum	Name	Partonen
S_1	Gluon	g
S_2	See	$\{q^S : \forall q\}$
S_3	Valenz-Down	d^V
S_4	Valenz-Up	u^V
S_5	Quasivalenz	$\{q^Q : \forall q\}$

(b)

Tabelle 2.2: (a): Strati zur Berechnung der nächsten Skala. (b) Strati für die Wichtungsfaktoren in (**??**)

Das Stammstratum ist demnach eine negative Stammfunktion des integrierten Stratums $\overline{S}_{\alpha\beta}$ mit $p_\perp^{\max} = s/4$ und

$$\overline{S}_{\alpha\beta}\left(p_\perp; s^{(n)}\right) := \int_{x_{\min}}^{1} \mathrm{d}\,x_1 \int_{x_{\min}}^{1} \mathrm{d}\,x_2\; S_{\alpha\beta}\left(x_1, x_2, p_\perp; s^{(n)}\right). \tag{2.3}$$

Schließlich sind die Branchingstrati $S_{\alpha\beta}$ mit

$$S_{\alpha\beta} := \frac{1}{\sigma_{\mathrm{nd}}} \sum_{k\in S_a} \sum_{l\in S_b} \sum_{m,n} \frac{\partial^3 \sigma_{kl\to mn}\left(x_1, x_2, p_\perp; s\right)}{\partial x_1\; \partial x_2\; \partial p_\perp} \tag{2.4}$$

als bedingte Wahrscheinlichkeit dafür definiert, dass eine hadronische Wechselwirkung aus dem Stratum $\{\alpha, \beta\}$ stattfindet, gegeben dass eine nicht-diffraktive hadronische Wechselwirkung stattfindet. σ_{nd} ist der totale, nicht-diffraktive Wirkungsquerschnitt und einer der freien Parameter des MPI-Modells. Die einfachen Strati S_a sind in Tabelle **??** dargestellt.

In Tabelle (**??**) sind zwei Varianten angegeben. In der ersten fehlen offensichtlich die Quasivalezquarks. Diese werden zwar vollkommen korrekt in den Remnant-Strukturfunktionen berücksichtigt, werden aber für die eigentliche Generierung der MPI aus technischen Gründen komplett ignoriert. In der Dissertation ist in Kapitel 5.2 ein Vorschlag gemacht, wie Quasivalenzquarks mitgenommen werden können. Diese Umsetzung bedeutet aber einen erheblichen Eingriff in den Quellcode.

In muli_interactions%valid_processes ist für jedes Feynmandiagramm in der fünften Komponente valid_processes(5,:) die Nummer des Stratums eingetragen, zu dem das Diagramm gehört.

Die einfachen Strati S_α heißen im Quellcode pdf_int_kind. Sie sind in muli_interactions%pdf_int_kind_gluon und folgende definiert. Entsprechend sind die doppelten Strati $\{\alpha, \beta\}$ in muli_interactions%double_pdf_kinds hinterlegt.

Wir bestimmen die nächste Skala des Stratums $\{\alpha, \beta\}$ über

$$\widehat{p}_{\perp,a,b}^{(n)} = \mathcal{S}_{\alpha\beta}^{-1}\left(\,\cdot\,; s^{(n)}\right)\left(\zeta_{\alpha\beta}^{(n)}\right) \tag{2.5}$$

mit

$$\zeta_{\alpha\beta}^{(n)} := \frac{\ln(z_{\alpha\beta}^{(n)})}{W_a^{(n)} W_b^{(n)}} + \mathcal{S}_{\alpha\beta}\left(p_\perp^{(n-1)}; s^{(n)}\right) = \mathcal{S}_{\alpha\beta}\left(p_\perp^{(n)}; s^{(n)}\right) \tag{2.6}$$

und

$$z_{\alpha\beta}^{(n)} \in (0,1], \quad \text{zufällig und gleichverteilt.} \tag{2.7}$$

Durch einsetzen von $s^{(n)}$ in $\mathcal{S}_{\alpha\beta}$ erhalten wir eine einstellige, umkehrbare Funktion $\mathcal{S}_{\alpha\beta}\left(\,\cdot\,; s^{(n)}\right)$. Somit ist $\mathcal{S}_{\alpha\beta}^{-1}\left(\,\cdot\,; s^{(n)}\right)\left(\zeta_{\alpha\beta}^{(n)}\right)$ eben diese Umkehrfunktion, ausgewertet bei $\zeta_{\alpha\beta}^{(n)}$.

Der größte Wert von $\widehat{p}_{\perp,a,b}^{(n)}$ unter allen Strati ist die neue Skala $\widehat{p}_{\perp}^{(n)}$, das Stratum mit der größten Skala ist das neue Stratum $\{\alpha^{(n)}, \beta^{(n)}\}$

> Die Stammstrati in (**??**) hängen offensichtlich von der aktuellen hadronischen invarianten Masse $s^{(n)}$ ab. Derzeit werden die $\mathcal{S}_{\alpha\beta}$ aber als eindimensionale Funktionen in muli_type%dsigma ohne s-Abhängigkeit gespeichert. Geht man zu einer zweidimensionalen Dastellung über, dann kommt man zu den Performance- und Speicherproblemen, die in der Dissertation in Kapitel 5.2 beschrieben sind. Dynamische Werte von $s^{(n)}$ sind also nicht durch einen trivialen Patch implementierbar. Das Problem wird teilweise dadurch entschärft, dass die Skala später auf die invariante Masse normiert wird. Quotienten $p_\perp/s$ werden also korrekt behandelt, nur durch Faktoren von s ohne $p_\perp$ werden die Matrixelemente inkonsistent.

2.2 Importance Sampling

Das Stratum $\{\alpha, \beta\}$ sowie die Skala $p_\perp$ liegen fest. Wie auch in der Dissertation unterdrücken wir hier die Indizes $^{(n)}$, da hier nur Werte aus der aktuellen Iteration vorkommen. Wir generieren die hyperbolischen Impulsanteile h_1, h_2, indem wir die Gleichung

$$z s \overline{S}_{ab}\left(p_\perp(h_3, s^{(n)}), s^{(n)}\right) \overset{?}{<} H_{ab}\left(h_1, h_2, h_3, s^{(n)}\right) \tag{2.8}$$

mit

$$h_1, h_2, z \in (0,1], \quad \text{zufällig und gleichverteilt} \tag{2.9}$$

solange mit neu generierten h_1, h_2, z auswerten, bis sie erfüllt ist. H ist eine regularisierte Form der divergenten Branchingstrati S, mit

$$H_{ab}\left(h_1, h_2, h_3, s^{(n)}\right) = S_{ab}\left(x_1(h_1,h_2), x_2(h_1,h_2), p_\perp(h_1,h_2,s^{(n)}), s^{(n)}\right)\left(\frac{\partial h_1}{\partial x_1}\frac{\partial h_2}{\partial x_2} - \frac{\partial h_1}{\partial x_2}\frac{\partial h_2}{\partial x_1}\right) \tag{2.10}$$

und

$$h_1 := \frac{x_1 x_2 - h_3}{(1-h_3)^{1/4}} \tag{2.11}$$

$$h_2 := \frac{1 + (x_2^2 - x_1^2)^{1/3}}{2} \tag{2.12}$$

$$h_3 := \frac{4 p_\perp}{\widehat{s}^{(n)}} \tag{2.13}$$

$$x_1 = \sqrt{\sqrt{(h_1^4(1-h_3)+h_3)^2 + (4(h_2-1/2)^3)^2} - 4(h_2-1/2)^3} \tag{2.14}$$

$$x_2 = \sqrt{\sqrt{(h_1^4(1-h_3)+h_3)^2 + (4(h_2-1/2)^3)^2} + 4(h_2-1/2)^3} \tag{2.15}$$

$$p_\perp = \frac{h_3 \widehat{s}^{(n)}}{4}. \tag{2.16}$$

$\overline{S}_{ab}$ ist der Mittelwert von H_{ab}, gemittelt über h_1 und h_2. Mit dem willkürlichen reellen Faktor s wird $s\overline{S}_{ab}$ damit zu einer, von $p_\perp$ abhängigen, Majorante von H_{ab}.

Um H noch weiter zu glätten, wird in muli_type%samples eine Einteilung des $\{h_1, h_2, h_3\}$-Einheitsquaders und Unterquader abgelegt. Dabei wird der Quader zuerst so in h_3-Richtung in endlich viele Scheiben geschnitten, dass das Integral über H in jeder Scheibe etwa gleich ist. Anschließend wird jede Scheibe simultan in h_1 und h_2 so in Unterquader zerlegt, dass das Integral über jedes dieser Unterquader etwa gleich groß ist und die Varianz in jedem Unterquader klein wird. Jeder Unterquader hat einen Index q_i und einen Flächeninhalt in der $h_1 - h_2$-Ebene von a_i.

Weiterhin werden alle Feynmandiagramme, die in dem Stratus $\{\alpha, \beta\}$ enthalten sind, mit einem Wichtungsfaktor d_j versehen. Die tatsächliche Vorgehensweise zur Generierung der Impulse und der Flavor ist dann wie folgt:

1. Es wird ein Diagramm mit der Wahrscheinlichkeit $W_j / \sum_k W_k$ gewählt.

2. Es wird zufällig und gleichverteilt ein Quader q_i mit der Fläche a_i aus derjenigen Scheibe gewählt, die h_3 enthält.

3. Es werden zufällig und gleichverteilt reelle Zahlen h_1, h_2 und z aus dem Einheitsintervall gewählt

4. Es wird
$$z s W_j \overline{S}_{ab} \left(p_\perp(h_3, s^{(n)}), s^{(n)} \right) \overset{?}{<} a_i H_{ab} \left(h_1, h_2, h_3, s^{(n)} \right) \tag{2.17}$$
ausgewertet und bei 1 begonnen, bis die Ungleichung erfüllt ist.

5. Aus h_1 und h_2 ergeben sich die Impulsanteile x_1 und x_2, aus dem Diagramm j ergeben sich die Flavor a, b, c und d.

2.3 Remnants

Abweichend von der Dissertation (Gl. 4.49) werden hier fünf verschiedene Wichtungsfaktoren für die vier Strati {Gluon, See, Valenz-down, Valenz-up, Quasivalenz} zugelassen:

$$\begin{aligned} 1 = {} & \\ W_G^{(n)} \int_{\xi_{\min}^{(n)}}^{1} \mathrm{d}\,\xi^{(n)} f_g(\xi^{(n)}, Q^2) + {} & \\ + W_S^{(n)} \sum_q \int_{\xi_{\min}^{(n)}}^{1} \mathrm{d}\,\xi^{(n)} f_{q^S}(\xi^{(n)}, Q^2) + {} & \\ + \sum_q W_{q^V}^{(n)} \frac{N_{q^V}^{(n)}}{N_{q^V}^{(0)}} \int_{\xi_{\min}^{(n)}}^{1} \mathrm{d}\,\xi^{(n)} f_q^v(\xi^{(n)}, Q^2) + {} & \\ + W_Q^{(n)} \sum_q \int_{\xi_{\min}^{(n)}}^{1} \mathrm{d}\,\xi^{(n)} f_q^Q(\xi^{(n)}, Q^2) & \end{aligned} \tag{2.18}$$

Da wir nur eine Gleichung für vier Wichtungsfaktoren haben, müssen wir weitere Beziehungen festlegen. Durch den Parameter muli_remnant%remnant_weight_model wird entschieden, welche Wichtungsfaktoren auf Eins gesetzt werden. Die jeweils anderen werden gleich gesetzt. Für das Quadrupel $[W_G, W_S, W_{d^V}, W_{u^V}, W_Q]$ erhalten wir:

In (**??**) eingesetzt kann w eindeutig bestimmt werden.

Für remnant_weight_model=0 ist (**??**) nicht lösbar, es ist also streng genommen kein gültiges Wichtungsmodell. Stattdessen wird dadurch die Gewichtung deaktiviert.

remnant_weight_model	$[W_G, W_S, W_d, W_u, W_Q]$
0	$[1,\ 1,\ 1,\ 1,\ 1\ ]$
1	$[w, w, w, w, w]$
2	$[w, w, 1,\ 1,\ 1\ ]$
3	$[1,\ 1,\ w, w, w]$
4	$[1,\ w, 1,\ 1,\ w]$

Tabelle 2.3: remnant_weight_models

2.4 Programmfluss

Die einzeilnen Methoden sind ausführlich in muli beschrieben. Wir geben hier nur eine kurze Übersicht an:

- initialize

  Der Monte-Carlo-Generator von MulI und die Datenstruktur der Proton-Remnants werden initialisiert.

- apply_initial_interaction

  Eine von WHIZARD generierte harte Wechselwirkung wird an MulI übergeben. Die Remnants werden entsprechend angepasst.

- generate_gev2_pt2

  Mittels (**??**) wird eine Skala $\widehat{p}_{\perp,a,b}^{(n)}$ und ein Stratum $\{\alpha^{(n)}, \beta^{(n)}\}$ generiert.

- generate_partons

  Mittels (**??**) werden die Impulsanteile x_1 und x_2 und die Flavor a, b, c und d generiert. Außerdem wird eine interne Darstellung der Farbflüsse generiert.

- get_correlations

  Die interne Darstellung der Farbflüsse wird in der vom shower_interface gewünschten Form von Farbkorrelationen ausgegeben.

- replace_parton

  Der ISR-Algorithmus hat ein Branching eines aktiven Showerteilchens generiert. Dieses Showerteilchen ist fortan kein aktives Teilchen mehr, sondern ein inneres Teilchen der perturbativen Wechselwirkung. Entsprechend muss es durch replace_parton in der Beschreibung des Remnants durch das neue aktive Teilchen ersetzt werden. Siehe Abbildung **??**

- restart

  Es werden einige interne Variablen zurückgesetzt, wie z.B. das "finishedFlag. Außerdem werden die Remnants zurückgesetzt.

- finalize

  Der Monte-Carlo-Generator von MulI, die vorgenerierten Wirkungsquerschnitte und die Remnants werden dealllociert.

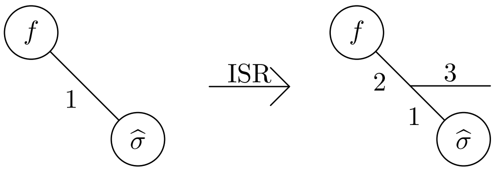


Abbildung 2.2: Ersetzung des aktiven ISR-Partons durch den ISR-Algorithmus. Links: Durch die härteste WW wurde ein Parton aus dem Hadron entfert. Dieses Parton ist ein „aktives“ Parton, da ISR-Branchings für dieses Teilchen generiert werden. Alle Teilchen, die jemals generiert werden, bekommen eine eindeutige Nummer, hier die Nummer 1. Die Eigenschaften des Remants und des Teichens #1 müssen in der Summe die Eigenschaften des Protons ergeben. Rechts: Durch den ISR-Algorithmus wurde ein Branching der Teilches #1 erzeugt. Das Mutterparton hat die Nummer 2 bekommen, das andere Tochterparton hat die Nummer 3 bekommen. Jetzt müssen die Eigenschaften des Remnants und des Teilchens #2 in der Summe die Eigenschaften des Protons ergeben. In diesem Sinne müssen wir das Teilchen #1 wieder „zurücklegen“ und das Teilchen #2 „herausnehmen“.

Teil II

Module

3 Modul muli

Dieses Modul dient als Interface für den Interleaved Algorithmus. Es stellt dem Algorithmus Methoden zur Initialisierung und zum Garbage-Collecting, Methoden zur Generierung von $p_\perp^{(n+1)}$, und zur Generierung einer vollständigen Wechselwirkung, Methoden zum Austauschen von aktiven Partonen, sowie Methoden zu den Remnant-Strukturfunktionen zur Verfügung.

Alle Parameter des MulI-Algorithmus sowie der komplette aktuelle Zusatand der Remnants, einschließlich der aktiven Shower-Partonen, sind als Komponenten des erweiterten Types muli_type definiert. Der Interleaved-Algorithmus muss also lediglich eine Instanz der Klasse muli_type definieren und deren public Type-Bound-Procedures aufrufen.

Auf lange Sicht ist es geplant, die Strukturfunktionen in den WHIZARD-Core aufzunehmen. Dann sollte der Core die Funktionalität dieses Moduls weitgehend übernehmen. Zum einen wird dieses Modul dann obsolet, zum anderen müssen die Interfaces des Cores erweitert werden. Der Hauptgrund, warum das noch nicht geschehen ist, ist aber, dass der Core noch nicht Objekt-Orientiert ist. Es kann also keine Instanz von muli_type an sf_initialize übergeben werden. Es muss also entweder die Konfiguration dieses Moduls in Form von Modulkomponenten bereitgestellt werden, oder der Core muss lernen, eine Instanzen einer abstrakten PDF-Klasse entgegenzunehmen. Im letzteren Fall muss muli_type nur diese abstrakte Klasse erweitern.

3.1 Abhängigkeiten

```
use muli_dsigma
use muli_mcint
use muli_remnant
```

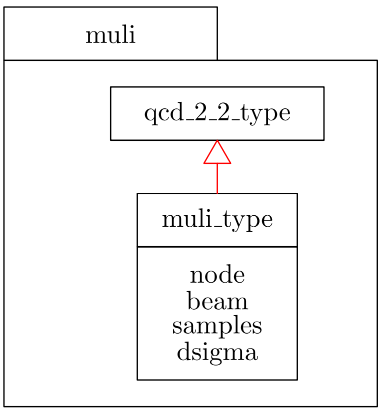


Abbildung 3.1: Klassendiagramm des Moduls muli

3.2 Parameter

Für Diagnose, Testzwecke und Konsistenzprüfungen können die Remnants deaktiviert werden. Wenn auf .false. gesetzt, läuft der Algorithmus trotz Mehrfachwechselwirkungen immer mit den original Proton-PDFs

```
logical,parameter::muli_default_modify_pdfs=.true.muli_default_modify_pdfs
```

Da MulI noch nicht in den Core eingebunden ist, hat MulI keinen Zugriff auf die PDF Eigenschaften, so wie sie in den Sindarin-Files eingestellt werden können. Deswegen müssen die Proton-PDFs noch per Hand als Parameter festgelegt werden.

```
integer,parameter::muli_default_lhapdf_member=0muli_default_lhapdf_member
character(*),parameter::muli_default_lhapdf_file=&
"cteq6ll.LHpdf"muli_default_lhapdf_file
```

3.3 Derived Types

3.3.1 qcd_2_2_type

Dieser Datentyp abstrahiert die interne Darstellung einer QCD-2 $\rightarrow$ 2 Wechselwirkungen und stellt Methoden mit traditionellen Namen zur Verfügung. Da verschiedene Module auf die Eigenschaften der Wechselwirkung zugreifen müssen, aber nur in dem Modul muli alle notwendigen Daten zur Verfügung stehen, ist in dem Modul muli_momentum eine abstrakte qcd_2_2_class definiert, die allen Modulen zur verfügung steht. qcd_2_2_type erweitert diese Klasse und implementiert die vorgeschriebenen Methoden.

```
type,extends(qcd_2_2_class)::qcd_2_2_type
```

Komponenten

```
private
```

Alle gültigen Kombinationen der vier Partonflavor sind in der Modulkomponente muli_interactions%valid_processes durchnummeriert. process_id gibt diese Nummer wieder und legt damit alle Flavor fest.

```
integer::process_id=-1
```

Alle Kombinationen aus Gluon, Seequark, Valenz-Up-Quark und Valenz-Down-Quark der beiden Partonen im Eingangszustand sind in der Modulkomponente muli_interactions%double_pdf_kinds des Moduls muli_interactions durchnummeriert. integrand_id gibt diese Nummer wieder und legt damit fest, ob z.B. ein Up-Quark im Eingangszustand ein Seequark oder ein Valenzquark ist. Diese Nummer ist gleichzeitig die Ordnungsnummer des Integrationsstratums $\{\alpha, \beta\}$ für die Generierung der Wechselwirkungsskala in muli_type%generate_next_scale.

```
integer::integrand_id=-1
```

Jedes Parton, dass an einer Wechselwirkung teilnimmt, bekommt eine eindeutige Nummer. Das schließt die Partonen des ISR-Algorithmus mit ein. Diese Nummern sind wichtig, wenn der ISR-Algorithmus ein Teilchen aus der Liste der Teilchen im Eingangszustand (aka aktive Partonen) entfernt, das MulI vorher in diese Liste aufgenommen hat. Über die Parton IDs können diese zugeordnet und in muli_type%replace_parton konsistent eliminiert werden. parton_ids enthält die Nummern der beiden Teilchen im direkten Eingangszustand der MPI Wechselwirkung, also ohne Showerbranchings.

```
integer,dimension(2)::parton_ids=[0,0]
```

Farbflüsse werden intern als Permutation dargestellt, die die Enden von Farblinien auf deren Angfänge abbildet. Eine 2 an dritter Stelle in flow bedeutet, dass eine Farbflusslinie in der zweiten Position beginnt und in der dritten Position endet. Die Positionen 1,2,3,4 entsprechen den Flavorindizes a,b,c,d in **??**. In muli_type%generate_flow werden diese Farbflüsse generiert.

```
integer,dimension(4)::flow=[0,0,0,0]
```

momentum_fractions enthält die dynamischen Impulsvariablen $[\xi_1, \xi_2, p_\perp]$, die die Kinematik einer Wechselwirkungen beschreiben, in kartesischen Koordinaten. hyperbolic_fractions enthält dieselben Impulsvariablen in hyperbolischen Koordinaten $[h_1, h_2, h_3]$. Die Koordinatentransformation ist in (**??**) angegeben.

```
real(kind=double),dimension(3)::momentum_fractions=[-1D0,-1D0,-1D0]
real(kind=double),dimension(3)::hyperbolic_fractions=[-1D0,-1D0,-1D0]
```

Methoden

```
!Überschriebene serializable_class Methoden
procedure,public::write_to_marker=>qcd_2_2_write_to_marker
procedure,public::read_from_marker=>qcd_2_2_read_from_marker
procedure,public::print_to_unit=>qcd_2_2_print_to_unit
procedure,public,nopass::get_type=>qcd_2_2_get_type
!Überschriebene qcd_2_2_class Methoden
procedure,public::get_process_id=>qcd_2_2_get_process_id
procedure,public::get_integrand_id=>qcd_2_2_get_integrand_id
procedure,public::get_diagram_kind=>qcd_2_2_get_diagram_kind
procedure,public::get_lha_flavors=>qcd_2_2_get_lha_flavors
procedure,public::get_pdg_flavors=>qcd_2_2_get_pdg_flavors
procedure,public::get_parton_id=>qcd_2_2_get_parton_id
procedure,public::get_parton_kinds=>qcd_2_2_get_parton_kinds
procedure,public::get_pdf_int_kinds=>qcd_2_2_get_pdf_int_kinds
procedure,public::get_momentum_boost=>qcd_2_2_get_momentum_boost
procedure,public::get_remnant_momentum_fractions=>qcd_2_2_get_remnant_momentum_fractions
procedure,public::get_total_momentum_fractions=>qcd_2_2_get_total_momentum_fractions
!Originäre qcd_2_2_type Methoden
procedure,public::get_color_flow=>qcd_2_2_get_color_flow
procedure,public::get_diagram_color_kind=>qcd_2_2_get_diagram_color_kind
procedure,public::get_io_kind=>qcd_2_2_get_io_kind
procedure,public::get_hyperbolic_fractions=>qcd_2_2_get_hyperbolic_fractions
procedure,public::get_color_correlations=>qcd_2_2_get_color_correlations
procedure,public::qcd_2_2_initialize
generic,public::initialize=>qcd_2_2_initialize
```

3.3.2 muli_type

Der Datentyp muli_type ist eine Sammlung von allen Daten, die für Generierung von mehrfachen Wechselwirkungen relevant sind. Da Instanzen von muli_type keine Zeiger enthalten, mit Ausnahme von Zeigen auf Komponenten von sich selbst, und nicht auf dynamische Modulkomponenten zugreifen, können beliebig viele Instanzen von muli_type erzeugt und parallel verwendet werden. Bloß kann LHAPDF nicht parallel aufgerufen werden.

muli_type erweitert qcd_2_2_type, damit *ist* eine Instanz von muli_type die aktuelle Wechselwirkung. Die aktuellen Zustände der Remnants sind hingegen Komponenten von muli_type, sodass andere Varianten von Remnants umgesetzt werden können, ohne diese Modul (entsprechend später den WHIZARD-Core) verändern zu müssen

Neben den aktuellen Zuständen der Wechselwirkung und der Remnants enthält muli_type auch einen Importance-Sampler für die Impulsanteile $[\xi_1, \xi_2]$. Dieser ist derzeit nicht auf dem Stand der Dissertation, wo ein gemeinsames Sampling für alle Strati beschrieben wird. Dieses gemeinsame Sampling wäre dann so generisch, dass es nicht mehr eine Komponente von muli_type sein sollte. Dann könnten mehrere Instancen von muli_type dasselbe Sampling verwenden. Derzeit ist das Sampling zweifach redundant, nämlich wird für jedes Stratum für jede Instanz von muli_type ein eigener Sampler allociert. Bei dem aktuellen Stand ist deswegen davon abzuraten, MulI zu parallelisieren.

```
type,extends(qcd_2_2_type)::muli_type
```

Komponenten

```
private
   !Untere Grenze für die Wechselwirkungsskala
   real(kind=double)::GeV2_scale_cutoff
   !Sollen die Strukturfunktionen im Laufe des Algorithmus verändert werden?
   logical::modify_pdfs=muli_default_modify_pdfs
   !Lag die letzte Wechselwirkungsskala unter GeV2_scale_cutoff?
   logical::finished=.false.
   !Wieviel Zeit haben die einzelnen Teile des Algorithmus benötigt?
   real(kind=double)::init_time=0D0
   real(kind=double)::pt_time=0D0
   real(kind=double)::partons_time=0D0
   real(kind=double)::confirm_time=0D0
   !Sind die Monte-Carlo-Generatoren bereit zur Generierung von MPI?
   logical::initialized=.false.
   !Wurde eine härteste Wechselwirkung vorgegeben?
   logical::initial_interaction_given=.false.
   !Der Mittelwert des hadronischen Wirkungsquerschnitts bei der aktuellen Skala.
   !Der Monte-Carlo-Generator für [x_1,x_2] verwendet ein vielfaches dieses Werts
   !als obere Schranke für den Wirkungsquerschnitt.
   real(kind=double)::mean=1D0
   !Die integrierten Wirkungsquerschnitte aller Strati bei der aktuellen Skala.
   !Der Monte-Carlo-Generator für die nächste Skala verwendet diese Integrale anstelle
   !der Skala als Startwerte.
   real(kind=double),dimension(0:16)::start_integrals=&
        [0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0]
   !Der Zufallsgenerator für diese Instanz.
```

```
type(tao_random_state)::tao_rnd
!Die Wirkungsquerschnitte und deren Stammfunktionen aller Strati
type(muli_trapezium_tree_type)::dsigma
!Importance-Sampler für alle Strati.
type(sample_inclusive_type)::samples
!Die Remnants der beiden Protonen.
type(pp_remnant_type)::beam
!Ein interner Zeiger auf ein Segment der Stammfunktion dsigma, dass die
!aktuelle Skala umfasst.
class(muli_trapezium_node_class),pointer::node=>null()
end type muli_type
```

Methoden

```
contains
  !Überschriebene serializable_class Methoden
  procedure,public::write_to_marker=>muli_write_to_marker
  procedure,public::read_from_marker=>muli_read_from_marker
  procedure,public::print_to_unit=>muli_print_to_unit
  procedure,public,nopass::get_type=>muli_get_type
  !Originäre muli_type Methoden
  ! init / final
  procedure,public::muli_initialize
  procedure,public::apply_initial_interaction=>muli_apply_initial_interaction
  procedure,public::finalize=>muli_finalize
  procedure,public::stop_trainer=>muli_stop_trainer
  procedure,public::reset_timer=>muli_reset_timer
  procedure,public::restart=>muli_restart
  generic,public::  initialize=>muli_initialize
  ! status query
  procedure,public::is_initialized=>muli_is_initialized
  procedure,public::is_initial_interaction_given=>muli_is_initial_interaction_given
  procedure,public::is_finished=>muli_is_finished
  ! user interface
  procedure,public::enable_remnant_pdf=>muli_enable_remnant_pdf
  procedure,public::disable_remnant_pdf=>muli_disable_remnant_pdf
  procedure,public::generate_gev2_pt2=>muli_generate_gev2_pt2
  procedure,public::generate_partons=>muli_generate_partons
  procedure,public::generate_flow=>muli_generate_flow
  procedure,public::replace_parton=>muli_replace_parton
  procedure,public::get_parton_pdf=>muli_get_parton_pdf
  procedure,public::get_momentum_pdf=>muli_get_momentum_pdf
  procedure,public::print_timer=>muli_print_timer
  procedure,public::generate_samples=>muli_generate_samples
  ! beam test
  procedure,public::fake_interaction=>muli_fake_interaction
  ! private procedures
  procedure,private::generate_next_scale=>muli_generate_next_scale
  procedure,private::confirm=>muli_confirm
```

3.4 Implementierung der Prozeduren

3.4.1 Methoden für qcd_2_2_type

Überschriebene serializable_class Methoden

qcd_2_2_write_to_marker ↑

```
subroutine qcd_2_2_write_to_marker(this,marker,status)
    class(qcd_2_2_type),intent(in)::this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    call marker%mark_begin("qcd_2_2_type")
    call transversal_momentum_write_to_marker(this,marker,status)
    call marker%mark("process_id",this%process_id)
    call marker%mark("integrand_id",this%integrand_id)
    call marker%mark("momentum_fractions",this%momentum_fractions)
    call marker%mark("hyperbolic_fractions",this%hyperbolic_fractions)
    call marker%mark_end("qcd_2_2_type")
  end subroutine qcd_2_2_write_to_marker
```

qcd_2_2_read_from_marker ↑

```
  subroutine qcd_2_2_read_from_marker(this,marker,status)
    class(qcd_2_2_type),intent(out)::this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    call marker%pick_begin("qcd_2_2_type",status=status)
    call transversal_momentum_read_from_marker(this,marker,status)
    call marker%pick("process_id",this%process_id,status)
    call marker%pick("integrand_id",this%integrand_id,status)
    call marker%pick("momentum_fractions",this%momentum_fractions,status)
    call marker%pick("hyperbolic_fractions",this%hyperbolic_fractions,status)
    call marker%pick_end("qcd_2_2_type",status=status)
  end subroutine qcd_2_2_read_from_marker
```

qcd_2_2_print_to_unit ↑

```
  subroutine qcd_2_2_print_to_unit(this,unit,parents,components,peers)
    class(qcd_2_2_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    integer,dimension(2,4)::flow
    integer::index
    if(parents>zero)&
    call transversal_momentum_print_to_unit(this,unit,parents-1,components,peers)
    write(unit,'("Components of qcd_2_2_type:")')
    write(unit,'("Process id is:       ",I3)')this%get_process_id()
    write(unit,'("Integrand id is:     ",I3)')this%get_integrand_id()
    if(this%get_integrand_id()>0)then
       write(unit,'("LHA Flavors are:     ",4(I3))')this%get_lha_flavors()
       write(unit,'("PDG Flavors are:     ",4(I3))')this%get_pdg_flavors()
```

```
      write(unit,'("Parton kinds are:    ",2(I3))')this%get_parton_kinds()
      write(unit,'("PDF int kinds are:   ",2(I3))')this%get_pdf_int_kinds()
      write(unit,'("Diagram kind is:     ",2(I3))')this%get_diagram_kind()
    end if
    call this%get_color_correlations(1,index,flow)
    write(unit,'("Color Permutations:  ",4(I0))')this%flow
    write(unit,'("Color Connections:")')
    write(unit,'("(",I0,",",I0,")+(",I0,",",I0,")->(",I0,",",I0,")+(",I0,",",I0,")")')flow
    write(unit,'("Evolution scale is:  ",E14.7)')this%get_unit2_scale()
    write(unit,'("Momentum boost is:   ",E14.7)')this%get_momentum_boost()
    write(unit,'("Remant momentum fractions are: ",2(E14.7))')&
    this%get_remnant_momentum_fractions()
    write(unit,'("Total momentum fractions are:  ",2(E14.7))')&
    this%get_total_momentum_fractions()
  end subroutine qcd_2_2_print_to_unit
```

qcd_2_2_get_type ↑

```
  pure subroutine qcd_2_2_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="qcd_2_2_type")
  end subroutine qcd_2_2_get_type
```

Überschriebene qcd_2_2_class Methoden

qcd_2_2_get_process_id ↑

```
  elemental function qcd_2_2_get_process_id(this) result(id)
    class(qcd_2_2_type),intent(in)::this
    integer::id
    id=this%process_id
  end function qcd_2_2_get_process_id
```

qcd_2_2_get_integrand_id ↑

```
  elemental function qcd_2_2_get_integrand_id(this) result(id)
    class(qcd_2_2_type),intent(in)::this
    integer::id
    id=this%integrand_id
  end function qcd_2_2_get_integrand_id
```

qcd_2_2_get_lha_flavors ↑

```
  pure function qcd_2_2_get_lha_flavors(this) result(lha)
    class(qcd_2_2_type),intent(in)::this
    integer,dimension(4)::lha
    lha=valid_processes(1:4,this%process_id)
  end function qcd_2_2_get_lha_flavors
```

qcd_2_2_get_pdg_flavors ↑

```
pure function qcd_2_2_get_pdg_flavors(this) result(pdg)
  class(qcd_2_2_type),intent(in)::this
  integer,dimension(4)::pdg
  pdg=this%get_lha_flavors()
  where(pdg==0) pdg=21
end function qcd_2_2_get_pdg_flavors
```

qcd_2_2_get_pdf_int_kinds ↑

```
pure function qcd_2_2_get_pdf_int_kinds(this) result(kinds)
  class(qcd_2_2_type),intent(in)::this
  integer,dimension(2)::kinds
  kinds=double_pdf_kinds(1:2,this%integrand_id)
end function qcd_2_2_get_pdf_int_kinds
```

qcd_2_2_get_parton_id ↑

```
elemental function qcd_2_2_get_parton_id(this,n) result(id)
  class(qcd_2_2_type),intent(in)::this
  integer,intent(in)::n
  integer::id
  id=this%parton_ids(n)
end function qcd_2_2_get_parton_id
```

qcd_2_2_get_parton_kinds ↑

```
pure function qcd_2_2_get_parton_kinds(this) result(kinds)
  class(qcd_2_2_type),intent(in)::this
  integer,dimension(2)::kinds
  kinds=this%get_pdf_int_kinds()
  kinds(1)=parton_kind_of_int_kind(kinds(1))
  kinds(2)=parton_kind_of_int_kind(kinds(2))
end function qcd_2_2_get_parton_kinds
```

qcd_2_2_get_io_kind ↑

```
elemental function qcd_2_2_get_io_kind(this) result(kind)
  class(qcd_2_2_type),intent(in)::this
  integer::kind
  kind=valid_processes(5,this%process_id)
end function qcd_2_2_get_io_kind
```

qcd_2_2_get_diagram_kind ↑

```
elemental function qcd_2_2_get_diagram_kind(this) result(kind)
  class(qcd_2_2_type),intent(in)::this
  integer::kind
  kind=valid_processes(6,this%process_id)
end function qcd_2_2_get_diagram_kind
```

Originäre qcd_2_2_type Methoden

qcd_2_2_get_diagram_color_kind ↑

This is one more hack. Before merging into the interleaved algorithm, muli has only cared for summed cross sections, but not for specific color flows. So two different diagrams with equal cross sections were summed up to diagram kind 1.
Now muli also generates color flows, so we must devide diagram kind 1 into diagram color kind 0 and diagram color kind 1.

```
elemental function qcd_2_2_get_diagram_color_kind(this) result(kind)
  class(qcd_2_2_type),intent(in)::this
  integer::kind
  kind=valid_processes(6,this%process_id)
  if(kind==1)then
     if(product(valid_processes(1:2,this%process_id))>0)then
        kind=0
     end if
  end if
end function qcd_2_2_get_diagram_color_kind
```

qcd_2_2_get_momentum_boost ↑

Noch nicht implementiert

```
elemental function qcd_2_2_get_momentum_boost(this) result(boost)
  class(qcd_2_2_type),intent(in)::this
  real(kind=double)::boost
  boost=-1D0
end function qcd_2_2_get_momentum_boost
```

qcd_2_2_get_hyperbolic_fractions ↑

```
pure function qcd_2_2_get_hyperbolic_fractions(this) result(fractions)
  class(qcd_2_2_type),intent(in)::this
  real(kind=double),dimension(3)::fractions
  fractions=this%hyperbolic_fractions
end function qcd_2_2_get_hyperbolic_fractions
```

qcd_2_2_get_remnant_momentum_fractions ↑

```
pure function qcd_2_2_get_remnant_momentum_fractions(this) result(fractions)
  class(qcd_2_2_type),intent(in)::this
  real(kind=double),dimension(2)::fractions
  fractions=this%momentum_fractions(1:2)
end function qcd_2_2_get_remnant_momentum_fractions
```

qcd_2_2_get_total_momentum_fractions ↑

Noch nicht implementiert

```
pure function qcd_2_2_get_total_momentum_fractions(this) result(fractions)
  class(qcd_2_2_type),intent(in)::this
  real(kind=double),dimension(2)::fractions
  fractions=[-1D0,-1D0]
end function qcd_2_2_get_total_momentum_fractions
```

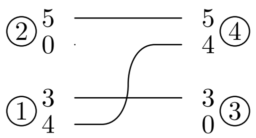
qcd_2_2_get_color_flow ↑

```
pure function qcd_2_2_get_color_flow(this) result(flow)
  class(qcd_2_2_type),intent(in)::this
  integer,dimension(4)::flow
  flow=this%flow
end function qcd_2_2_get_color_flow
```

qcd_2_2_get_color_correlations ↑

Diese Methode generiert Farbflussdiagramme aus der internen Darstellung mittels Permutationen, wie sie in muli_type%generate_flow generiert werden. Der Interleaved Algorithmus numeriert alle Farbflüsse, deswegen nehmen wir die aktuelle Anzahl von Farbflüssen mit start_index entgegen und liefern die neue Anzahl mit final_index zurück. Die Ordnungszahlen der neu generierten Farblinien laufen dann von start_index+1 bis einschließlich final_index.

flow liefert schießlich das Farbflussdiagramm selbst zurück. Das Format von flow sieht vor, dass der zweite Index die Position des Partons im Diagramm wie in **??** mit $[1,2,3,4] \to [a,b,c,d]$ beschreibt. Die beiden Stellen flow[:,a] beinhalten die Ordnungszahlen für eine eventuelle Farblinie bzw. eine eventuelle Antifarblinie oder 0 für keine Farblinie.



$$= [4,0,1,2] \to \begin{bmatrix} [3,4] \\ [5,0] \\ [3,0] \\ [5,4] \end{bmatrix}$$

Die eingekreisten Zahlen sind die Positionen in flow, die nicht eingekreisten Zahlen sind die Inizes der Farblinien.

```
subroutine qcd_2_2_get_color_correlations(this,start_index,final_index,flow)
  class(qcd_2_2_type),intent(in)::this
  integer,intent(in)::start_index
  integer,intent(out)::final_index
  integer,dimension(2,4),intent(out)::flow
  integer::pos,f_end,f_beginning
  final_index=start_index
  !we set all flows to zero. zero means no connection.
  flow=reshape([0,0,0,0,0,0,0,0],[2,4])
  !look at all four possible ends of color lines.
  do f_end=1,4
     !the beginning of this potential line is stored in flow. zero means no line.
     f_beginning=this%flow(f_end)
     !is there a line beginning at f_beginning and ending at f_end?
     if(f_beginning>0)then
        !yes it is. we get a new number for this new line
        final_index=final_index+1
        !is this line beginning in the initial state?
```

```
        if(f_beginning<3)then
           !yes it is. lets connect the color entry of f_begin.
           flow(1,f_beginning)=final_index
        else
           !no, it's the final state. lets connect the anticolor entry of f_begin.
           flow(2,f_beginning)=final_index
        end if
        !is this line ending in the final state?
        if(f_end>2)then
           !yes it is. lets connect the color entry of f_end.
           flow(1,f_end)=final_index
        else
           !no, it's the initial state. lets connect the anticolor entry of f_end.
           flow(2,f_end)=final_index
        end if
     end if
  end do
end subroutine qcd_2_2_get_color_correlations
```

qcd_2_2_initialize ↑

Gewöhnliche Initialisierung aller Komponenten.

```
subroutine qcd_2_2_initialize(this,gev2_s,process_id,integrand_id,parton_ids,flow,hyp,cart)
  class(qcd_2_2_type),intent(out)::this
  real(kind=double),intent(in)::gev2_s
  integer,intent(in)::process_id,integrand_id
  integer,dimension(2),intent(in)::parton_ids
  integer,dimension(4),intent(in)::flow
  real(kind=double),dimension(3),intent(in)::hyp
  real(kind=double),dimension(3),intent(in),optional::cart
  !Generischer Aufruf von transversal_momentum_initialize(this,gev2_s).
  call this%initialize(gev2_s)
  this%process_id=process_id
  this%integrand_id=integrand_id
  this%parton_ids=parton_ids
  this%flow=flow
  this%hyperbolic_fractions=hyp
  if(present(cart))then
     this%momentum_fractions=cart
  else
     this%momentum_fractions=h_to_c_param(hyp)
  end if
end subroutine qcd_2_2_initialize
```

3.4.2 Methoden für muli_type

Überschriebene serializable_class Methoden

muli_write_to_marker ↑

```
subroutine muli_write_to_marker(this,marker,status)
  class(muli_type),intent(in) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  call marker%mark_begin("muli_type")
  call qcd_2_2_write_to_marker(this,marker,status)
  call marker%mark("modify_pdfs",this%modify_pdfs)
  call marker%mark("initialized",this%initialized)
  call marker%mark("initial_interaction_given",this%initial_interaction_given)
  call marker%mark("finished",this%finished)
  call marker%mark("init_time",this%init_time)
  call marker%mark("pt_time",this%pt_time)
  call marker%mark("partons_time",this%partons_time)
  call marker%mark("confirm_time",this%confirm_time)
  call marker%mark_instance(this%dsigma,"dsigma")
  call marker%mark_instance(this%samples,"samples")
  call marker%mark_instance(this%beam,"beam")
  call marker%mark_end("muli_type")
end subroutine muli_write_to_marker
```

muli_read_from_marker ↑

```
subroutine muli_read_from_marker(this,marker,status)
  class(muli_type),intent(out) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  call marker%pick_begin("muli_type",status=status)
  call qcd_2_2_read_from_marker(this,marker,status)
  call marker%pick("modify_pdfs",this%modify_pdfs,status)
  call marker%pick("initialized",this%initialized,status)
  call marker%pick("initial_interaction_given",this%initial_interaction_given,status)
  call marker%pick("finished",this%finished,status)
  call marker%pick("init_time",this%init_time,status)
  call marker%pick("pt_time",this%pt_time,status)
  call marker%pick("partons_time",this%partons_time,status)
  call marker%pick("confirm_time",this%confirm_time,status)
  call marker%pick_instance("dsigma",this%dsigma,status=status)
  call marker%pick_instance("samples",this%samples,status=status)
  call marker%pick_instance("beam",this%beam,status=status)
  call marker%pick_end("muli_type",status)
end subroutine muli_read_from_marker
```

muli_print_to_unit ↑

```
subroutine muli_print_to_unit(this,unit,parents,components,peers)
  class(muli_type),intent(in)::this
  integer,intent(in)::unit
  integer(kind=dik),intent(in)::parents,components,peers
  if(parents>0)call qcd_2_2_print_to_unit(this,unit,parents-1,components,peers)
  write(unit,fmt="(a)")"Components of muli_type :"
  write(unit,'("Model Parameters:")')
  write(unit,'("GeV2_scale_cutoff : ",E20.10)')this%GeV2_scale_cutoff
```

```
    write(unit,'("Modify PDF         : ",L1)')this%modify_pdfs
    write(unit,'("PT Chain Status:")')
    write(unit,'("Initialized        : ",L1)')this%initialized
    write(unit,'("initial_interaction_given: ",L1)')this%initial_interaction_given
    write(unit,'("Finished           : ",L1)')this%finished
    write(unit,'("Exceeded           : ",L1)')this%exceeded
    write(unit,'("Generator Internals:")')
    write(unit,'("Mean Value         : ",E20.10)')this%mean
    if(components>zero)then
       write(unit,'("Start Integrals   : ",16(E20.10))')this%start_integrals(1:16)
       write(unit,'("dsigma Component:")')
       call this%dsigma%print_to_unit(unit,parents,components-1,peers)
       write(unit,'("samples Component:")')
       call this%samples%print_to_unit(unit,parents,components-1,peers)
       write(unit,'("beam Component:")')
       call this%beam%print_to_unit(unit,parents,components-1,peers)
    else
       write(unit,'("Skipping Derived-Type Components.")')
    end if
  end subroutine muli_print_to_unit
```

muli_get_type ↑

```
  pure subroutine muli_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="muli_type")
  end subroutine muli_get_type
```

Originäre muli_type Methoden

muli_apply_initial_interaction ↑

MulI kann die härteste Wechselwirkung selbst generieren oder eine bereits generiete Wechselwirkung auf die Remnants übertragen. Mit dieser Methode wird eine extern (üblicherweise durch WHIZARD) generierte Wechselwirkung übertragen.

Vor jedem Aufruf dieser Methode muss *this* durch muli_initialize initialisiert werden. Es sollten zwischen der Initialisierung und diesem Aufruf keine Wechselwirkungen generiert werden.

gev2_s	invariante Masse des hadronischen Systems vor der Wechselwirkung in GeV^2
x1	longitudinaler Impulsanteil des ersten Partons
x2	longitudinaler Impulsanteil des zweiten Partons
pdg_f1	Flavor des ersten Partons im PDG-Schema
pdg_f2	Flavor des zweiten Partons im PDG-Schema
n1	Ordnungszahl des ersten Partons
n2	Ordnungszahl des zweiten Partons

```
  subroutine muli_apply_initial_interaction(this,&
       gev2_s,&
       x1,&
       x2,&
       pdg_f1,&
```

```
      pdg_f2,&
      n1,&
      n2)
   class(muli_type),intent(inout)::this
   real(kind=double),intent(in)::Gev2_s,x1,x2
   integer,intent(in)::pdg_f1,pdg_f2,n1,n2
   real(kind=double)::rnd1,rnd2,time
   if(this%initialized)then
      !Timer Start für Benchmarkzwecke.
      call cpu_time(time)
      this%init_time=this%init_time-time
      !Einige Informationen für Debuggingzwecke.
      print *,"muli_apply_initial_interaction:"
      print *,"gev2_s=",gev2_s
      print *,"x1=",x1
      print *,"x2=",x2
      print *,"pdg_f1=",pdg_f1
      print *,"pdg_f2=",pdg_f2
      print *,"n1=",n1
      print *,"n2=",n2
!Aufgrund eines Bugs in gfortran 4.6 konnte ich die tao_state variable nicht an
!andere Module weitergeben und habe stattdessen vorgenerierte Zufallszahlen weitergegeben.
      call tao_random_number(this%tao_rnd,rnd1)
      call tao_random_number(this%tao_rnd,rnd2)
      !Timer Stop für Benchmarkzwecke.
      call cpu_time(time)
      this%init_time=this%init_time+time
      !Die nächste Zeile ist der eigentliche Aufruf für das Anpassen der Remnants.
      !Alles andere in dieser Methode ist Wrapper-Overhead.
```

Muli hat $p_\perp$ als Ordnungsparameter, WHIZARD generiert diese Valiable aber nicht. $p_\perp$ lässt sich auch nicht eindeutig aus den generierten Variablen ermitteln. Es ließe sich bestenfalls eine Verteilung von $p_\perp$ in Abhängigkeit der bekannten Variablen angeben. Hier verwenden wir einen einfacheren Weg und setzen die Obere Schranke $p_\perp \leq \hat{s}/4 = s x_1 x_2/4$ für $p_\perp$ ein.

```
      call this%beam%apply_initial_interaction(sqrt(gev2_s),x1,x2,pdg_f1,pdg_f2,n1,n2,&
          sqrt(gev2_s)*x1*x2/2D0,&
          rnd1,rnd2)
      this%initial_interaction_given=.true.
      !Das Program wir sofort beendet, wenn diese Methode uninitialisiert aufgerufen
      !wird. Das kann kein Bug sein, sondern muss eine falsche Verwendung dieser
      !Schnittstelle sein.
   else
      print *,"muli_apply_initial_interaction: call muli_initialize first. STOP"
      STOP
   end if
 end subroutine muli_apply_initial_interaction
```

muli_initialize ↑

Diese Methode initialisiert eine Instanz vom Typ muli_type.

GeV2_scale_cutoff	Skala in GeV^2, bei der der Algorithmus beendet wird.
GeV2_s	invariante Masse des hadronischen Systems in GeV^2
muli_dir	vollständiger Unix-Pfad zu dem Verzeichnis, in dem MulI-Daten liegen.
random_seed	

Diese Methode sollte für jede Instanz nur einmal aufgerufen werden. Der Hauptzweck ist, den Monte-Carlo-Generator zu initialisieren, nicht etwa die Remnants zu initialisieren. Da hierfür einige Zeiger allociert werden. ist es ratsam, die Instanz mit muli_finalize aufzuräumen, wenn man sie nicht mehr benötigt.

```
  subroutine muli_initialize(this,&
       GeV2_scale_cutoff,&
       gev2_s,&
       muli_dir,&
       random_seed)
    class(muli_type),intent(out)::this
    real(kind=double),intent(in)::gev2_s,GeV2_scale_cutoff
    character(*),intent(in)::muli_dir
    integer,intent(in),optional::random_seed
    real(kind=double)::time
    logical::exist
    type(muli_dsigma_type)::dsigma_aq
    character(3)::lhapdf_member_c
    !Timer Start für Benchmarkzwecke.
    call cpu_time(time)
    this%init_time=this%init_time-time
    !Einige Informationen für Debuggingzwecke.
    print *,"muli_initialize: The MULI modules are still not fully populated, so MULI might &
         &generate some dummy values instead of real Monte Carlo generated interactions."
    print *,"Given Parameters:"
    print *,"GeV2_scale_cutoff=",GeV2_scale_cutoff
    print *,"muli_dir=",muli_dir
    print *,"lhapdf_dir=",""
    print *,"lhapdf_file=",muli_default_lhapdf_file
    print *,"lhapdf_member=",muli_default_lhapdf_member
    print *,""
    !p⊥ wird auf die invariante Masse normiert.
    call transversal_momentum_initialize(this,gev2_s)
    !Die Remnants werden initialisiert.
    call this%beam%initialize(&
         muli_dir,&
         lhapdf_dir="",&
         lhapdf_file=muli_default_lhapdf_file,&
         lhapdf_member=muli_default_lhapdf_member)
    this%GeV2_scale_cutoff=GeV2_scale_cutoff
    if(present(random_seed))then
       call tao_random_create(this%tao_rnd,random_seed)
    else
       call tao_random_create(this%tao_rnd,1)
    end if
!Wir durchsuchen muli_dir nach vorgenerierten hadronischen Wirkungsquerschnitten.
```

```
!Dafür wird eine Zeichenkette generiert, die den Namen der LHAPDF-Datei enthält.
!Zusätzlich wird aus der lhapdf_member Variable eine Zeichenkette lhapdf_member_c
!generiert, denn in jeder xml-Datei können mehrere Wirkungsquerschnitte für
!verschiedene lhapdf_member liegen.
```

Ich habe noch nie mehrere Member in einer Datei verwendet. Höchstwahrscheinlich funktioniert es dann auch nicht.

```
    print *,"looking for previously generated root function..."
    call integer_with_leading_zeros(muli_default_lhapdf_member,3,lhapdf_member_c)
    inquire(file=muli_dir//"/dsigma_"//muli_default_lhapdf_file//".xml",exist=exist)
    if(exist)then
!Wir haben eine xml Datei zu der richtigen LHAPDF-Datei gefunden. Jetzt
!deserialisieren wir die Wirkungsquerschnitte zu dem gewünschten lhapdf_member.
       print *,"found. Starting deserialization..."
       call this%dsigma%deserialize(&
            name="dsigma_"//muli_default_lhapdf_file//"_"//lhapdf_member_c,&
            file=muli_dir//"/dsigma_"//muli_default_lhapdf_file//".xml")
       print *,"done. Starting generation of plots..."
!Einige Plots für Debuggingzwecke.
       call this%dsigma%gnuplot(muli_dir)
       print *,"done."
    else
!Es wurden keine passenden hadronischen Wirkungsquerschnitte gefunden. Es werden
!welche generiert und in muli_dir geschrieben. dsigma_aq ist nur für die Generierung
!der Wirkungsquerschnitte relevant, aber nicht für die Generierung der Ereignisse.
!dsigma_aq wird nur zu Debugging-Zwecken in muli_dir geschrieben. Die Serialisierung
!von dsigma_aq kann also jederzeit gefahrlos herausgenommen werden.
       print *,"No root function found. Starting generation of root function..."
       call dsigma_aq%generate(GeV2_scale_cutoff,gev2_s,this%dsigma)
       print *,"done. Starting serialization of root function..."
       call this%dsigma%serialize(&
            name="dsigma_"//muli_default_lhapdf_file//"_"//lhapdf_member_c,&
            file=muli_dir//"/dsigma_"//muli_default_lhapdf_file//".xml")
       print *,"done. Starting serialization of generator..."
       call dsigma_aq%serialize(&
            name="dsigma_aq_"//muli_default_lhapdf_file//"_"//lhapdf_member_c,&
            file=muli_dir//"/dsigma_aq_"//muli_default_lhapdf_file//".xml")
       print *,"done. Starting generation of plots..."
       call this%dsigma%gnuplot(muli_dir)
       print *,"done."
    end if
!Es wird noch nach Daten für das Importance-Sampling gesucht. Ohne diese kann MulI
!in seltenen Fällen unendlich langsam werden, wörtlich!
    print *,""
    print *,"looking for previously generated samples..."
    inquire(file=muli_dir//"/samples.xml",exist=exist)
    if(exist)then
       print *,"found. Starting deserialization..."
       call this%samples%deserialize("samples",muli_dir//"/samples.xml")
    else
```

```
      print *,"No samples found. Starting with default initialization."
      call this%samples%initialize(4,int_sizes_all,int_all,1D-2)
    end if
    !Jetzt wird MulI startklar gemacht.
    call this%restart()
    this%initialized=.true.
    !Timer Stopp für Benchmarkzwecke.
    call cpu_time(time)
    this%init_time=this%init_time+time
  end subroutine muli_initialize
```

muli_finalize ↑

```
  subroutine muli_finalize(this)
    class(muli_type),intent(inout)::this
    print *,"muli_finalize"
    nullify(this%node)
    call this%dsigma%finalize()
    call this%samples%finalize()
    call this%beam%finalize()
  end subroutine muli_finalize
```

muli_stop_trainer ↑

Trainer ist ein Bertiebsmodus von MulI, in dem das Importance-Sampling in jedem Schritt verfeinert wird. Das Sampling ist darauf optimiert, dass es schnell (also mit wenigen Ereignissen) den MCG beschleunigt, aber nicht darauf optimiert, bei vielen Daten den MCG perfekt zu beschleunigen. Deshalb konvergiert die Geschwindigkeit des MCG gegen eine Grenze und der Trainer-Modus kann abgestellt werden.

Das hat in der NAG Variante von MulI bereits funktioniert, allerdings musste ich für die gcc-Kompatiblität so tief in den Sampler eingreifen, dass ich ihn komplett neu geschrieben habe. Den Nicht-Trainer-Modus habe ich in der gcc-Version noch nicht umgesetzt, entsprechend ist diese Methode nur ein Dummy für ein nicht-vorhandenes Feature.

```
  subroutine muli_stop_trainer(this)
    class(muli_type),intent(inout)::this
    print *,"muli_stop_trainer: DUMMY!"
  end subroutine muli_stop_trainer
```

muli_reset_timer ↑

```
  subroutine muli_reset_timer(this)
    class(muli_type),intent(inout)::this
    this%init_time=0D0
    this%pt_time=0D0
    this%partons_time=0D0
    this%confirm_time=0D0
  end subroutine muli_reset_timer
```

muli_restart ↑

Wenn mehrere Ereignisse (Ein Ereignis ist eine Hadron-Hadron-Streuung inklusive ISR und MPI) in einem Programmaufruf generiert werden sollen, dann ist diese Methode und eventuell muli_apply_initial_interaction vor jedem neuen Ereignis aufzurufen, aber nicht muli_initialize.

```
  subroutine muli_restart(this)
    class(muli_type),intent(inout)::this
!this%node wird auf das letzte Blatt aus der Binärbaumdarstellung für die hadronischen
!Wirkungsquerschnitte gesetzt. Es enthält dann die Wirkungsquerschnitte bei der Startskala
!für p⊥.
call this%dsigma%get_rightmost(this%node)
!Die Remnants werden zurückgesetzt.
    call this%beam%reset()
```

Offensichtlich ist der $p_\perp$-Generator jetzt noch nicht fertig, es können also weitere Werte für $p_\perp$ generiert werden. Alle anderen Komponenten werden auf ungültige Werte gesetzt, damit das Programm tendentiell eher abstürtzt als mit willkürlichen Zahlen zu rechnen, falls etwas schiefgeht, wie z.B. eine Nachfrage nach der aktuellen Wechselwirkung, obwohl noch keine generiert wurde. start_integrals ist widerum eine gültige Initialisierung, denn die Integrale von der aktuellen (= der maximalen) Skala bis zur maximalen Skala sind offensichtlich gleich Null.

```
    this%finished=.false.
    this%process_id=-1
    this%integrand_id=-1
    this%momentum_fractions=[-1D0,-1D0,1D0]
    this%hyperbolic_fractions=[-1D0,-1D0,1D0]
    this%start_integrals=&
    [0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0]
  end subroutine muli_restart
```

muli_is_initialized ↑

```
  elemental function muli_is_initialized(this) result(res)
    logical::res
    class(muli_type),intent(in) :: this
    res=this%initialized
  end function muli_is_initialized
```

muli_is_initial_interaction_given ↑

```
  elemental function muli_is_initial_interaction_given(this) result(res)
    logical::res
    class(muli_type),intent(in) :: this
    res=this%initial_interaction_given
  end function muli_is_initial_interaction_given
```

muli_is_finished ↑

```
  elemental function muli_is_finished(this) result(res)
    logical::res
    class(muli_type),intent(in) :: this
    res=this%finished
  end function muli_is_finished
```

muli_enable_remnant_pdf ↑

Nur für Debugging-Zwecke

```
subroutine muli_enable_remnant_pdf(this)
  class(muli_type),intent(inout)::this
  this%modify_pdfs=.true.
end subroutine muli_enable_remnant_pdf
```

muli_disable_remnant_pdf ↑

Nur für Debugging-Zwecke

```
subroutine muli_disable_remnant_pdf(this)
  class(muli_type),intent(inout)::this
  this%modify_pdfs=.false.
end subroutine muli_disable_remnant_pdf
```

muli_generate_gev2_pt2 ↑

Wrapper für die Generierung der nächsten Skala $p_\perp$, die eigentliche Arbeit wird in muli_type%generate_next_scale gemacht. Diese Wrapper-Methode nimmt eine beliebige Start-Skala für $p_\perp$ entgegen, liefert einen Kandidaten für $p_\perp$ zurück und misst die Zeit, die die CPU dafür benötigt hat.

```
subroutine muli_generate_gev2_pt2(this,gev2_start_scale,gev2_new_scale)
  class(muli_type),intent(inout)::this
  real(kind=double),intent(in)::gev2_start_scale
  real(kind=double),intent(out)::gev2_new_scale
  real(kind=double)::time
  !Timer Start für Benchmark-Zwecke
  call cpu_time(time)
  this%pt_time=this%pt_time-time
  !Die aktuelle Skala wird auf den angegebenen Wert gesetzt
  call this%set_gev2_scale(gev2_start_scale)
  !Mit muli_trapezium_type%approx_integral wird die Stammfunktionen S_αβ an
  !dieser Skala ausgewertet.
  this%start_integrals=this%node%approx_integral(this%get_unit_scale())
  !Eine neue Wechselwirkungsskala wird MC-generiert.
  call this%generate_next_scale()
  !Die neue Skala wird zurückgegeben.
  gev2_new_scale=this%get_gev2_scale()
  !Timer Stopp
  call cpu_time(time)
  this%pt_time=this%pt_time+time
end subroutine muli_generate_gev2_pt2
```

muli_generate_flow ↑

Generierung einer internen Darstellung eines Farbflussdiagramms für das aktuelle Feynmandiagramm.

```
subroutine muli_generate_flow(this)
  class(muli_type),intent(inout)::this
  integer::rnd
  integer::m,n
  logical,dimension(3)::t
  integer,dimension(4)::tmp_flow
```

```
!we initialize with zeros. a zero means no line ends here.
this%flow=[0,0,0,0]
!we randomly pick a color flow
call tao_random_number(this%tao_rnd,rnd)
!the third position of muli_flow_stats is the sum of all flow wheights of
!stratum diagram_kind. so we generate a random number 0 <= m < sum(weights)
m=modulo(rnd,muli_flow_stats(3,this%get_diagram_color_kind()))
!lets visit all color flows of stratum diagram_kind.
!the first and second position of muli_flow_stats
!tell us the index of the first and the last valid color flow.
do n=muli_flow_stats(1,this%get_diagram_color_kind()),&
muli_flow_stats(2,this%get_diagram_color_kind())
   !now we remove the weight of flow n from our random number.
   m=m-muli_flows(0,n)
   !this is how we pick a flow.
   if(m<0)then
      !the actual flow
      this%flow=muli_flows(1:4,n)
      exit
   end if
end do
!the diagram kind contains a primitive diagram and all diagramms which can
!be deriven by
!(1) global charge conjugation
!(2) permutation of the initial state particles
!(3) permutation of the final state particles
!lets see, what transformations we have got in our actual interaction.
t=muli_get_state_transformations(this%get_diagram_color_kind(),this%get_lha_flavors())
!now we have to apply these transformations to our flow.
!(1) means: swap beginning and end of a line. flow is a permutation that maps
!ends to their beginnings, so we apply flow to itself:
if(t(1))then
   tmp_flow=this%flow
   this%flow=[0,0,0,0]
   do n=1,4
      if(tmp_flow(n)>0)this%flow(tmp_flow(n))=n
   end do
end if
if(t(2))then
   !we swap the particles 1 and 2
   tmp_flow(1)=this%flow(2)
   tmp_flow(2)=this%flow(1)
   tmp_flow(3:4)=this%flow(3:4)
   !we swap the beginnings assigned to particle 1 and 2
   where(tmp_flow==1)
      this%flow=2
   elsewhere(tmp_flow==2)
      this%flow=1
   elsewhere
      this%flow=tmp_flow
   end where
```

```
    end if
    if(t(3))then
       !we swap the particles 3 and 4
       tmp_flow(1:2)=this%flow(1:2)
       tmp_flow(3)=this%flow(4)
       tmp_flow(4)=this%flow(3)
       !we swap the beginnings assigned to particle 3 and 4
       where(tmp_flow==3)
          this%flow=4
       elsewhere(tmp_flow==4)
          this%flow=3
       elsewhere
          this%flow=tmp_flow
       end where
    end if
  end subroutine muli_generate_flow
```

muli_generate_partons ↑

Generierung der Partonimpulsanteile $[\xi_1, \xi_2]$ sowie der Partonflavor $[a, b, c, d]$, siehe Abschnitt **??**.

Im Wesentlichen ist dies ein Wrapper für sample_inclusive_type%mcgenerate_hit und muli_type%generate_flow

n1, n2: Identifikationsnummern der Partonen. Das shower_interface kümmert sich um die Durchnummerierung der Partonen, deswegen nimmt muli_generate_partons diese Nummern entgegen, statt sie zu erzeugen.

x_proton_1,x_proton_2: Longitudinale Impulsanteile der Partonen, bezogen auf die ursprünglichen Protonimpulse, nicht auf die aktuellen Remnantimpulse.

pdg_f1,pdg_f2,pdg_f3,pdg_f4: Die Flavor a,b,c,d der Partonen im PDG-Schema.

```
  subroutine muli_generate_partons(this,n1,n2,x_proton_1,x_proton_2,pdg_f1,pdg_f2,pdg_f3,pdg_f4)
    class(muli_type),intent(inout)::this
    integer,intent(in)::n1,n2
    real(kind=double),intent(out)::x_proton_1,x_proton_2
    integer,intent(out)::pdg_f1,pdg_f2,pdg_f3,pdg_f4
    integer,dimension(4)::pdg_f
    real(kind=double)::time
    !print *,"muli_generate_partons: n1=",n1," n2=",n2
    this%parton_ids(1)=n1
    this%parton_ids(2)=n2
    call cpu_time(time)
    this%partons_time=this%partons_time-time
```

Mittels muli_trapezium_type%approx_value_n wird der Mittelwert $\overline{S}_{\alpha\beta}$ ausgewertet. Anschließend werden mittels sample_inclusive_type%mcgenerate_hit die Nummer *process_id* des Feymandiagramms und die Impulsanteile *momentum_fractions* der Partonen generiert.

```
    this%mean=this%node%approx_value_n(this%get_unit_scale(),this%integrand_id)
    call sample_inclusive_mcgenerate_hit(&
         this%samples,&
         this%get_unit2_scale(),&
         this%mean,&
         this%integrand_id,&
```

```
        this%tao_rnd,&
        this%process_id,&
        this%momentum_fractions)
```

Mittels muli_type%generate_flow wird ein Farbflussdiagramm generiert.

```
    call this%generate_flow()
```

Üblichwerweise (Wenn ich nicht debugge) werden die Remnante mittels pp_remnant_type%apply_interaction die Remnants über die neue Wechselwirkung in Kenntnis gesetzt.

```
    if(this%modify_pdfs)then
       call cpu_time(time)
       this%partons_time=this%partons_time+time
       this%confirm_time=this%confirm_time-time
       call this%beam%apply_interaction(this)
       call cpu_time(time)
       this%confirm_time=this%confirm_time+time
       this%partons_time=this%partons_time-time
    end if
    x_proton_1=this%momentum_fractions(1)
    x_proton_2=this%momentum_fractions(2)
    pdg_f=this%get_pdg_flavors()
    pdg_f1=pdg_f(1)
    pdg_f2=pdg_f(2)
    pdg_f3=pdg_f(3)
    pdg_f4=pdg_f(4)
    call cpu_time(time)
    this%partons_time=this%partons_time-time
    call qcd_2_2_print_to_unit(this,output_unit,100_dik,100_dik,100_dik)
  end subroutine muli_generate_partons
```

muli_replace_parton ↑

Wrapper für die eigentliche Routine pp_remnant_type%replace_parton

```
  subroutine muli_replace_parton(this,proton_id,old_id,new_id,pdg_f,x_proton,gev_scale)
    class(muli_type),intent(inout)::this
    integer,intent(in)::proton_id,old_id,new_id,pdg_f
    real(kind=double),intent(in)::x_proton,gev_scale
    !print *,"muli_replace_parton(",proton_id,old_id,new_id,pdg_f,x_proton,gev_scale,")"
    if(proton_id==1.or.proton_id==2)then
       call this%beam%replace_parton(proton_id,old_id,new_id,pdg_f,x_proton,gev_scale)
    else
       print *,"muli_replace_parton: proton_id must be 1 or 2, but ",proton_id," was given."
       STOP
    end if
  end subroutine muli_replace_parton
```

muli_get_momentum_pdf ↑

Wrapper für die eigentliche Routine pp_remnant_type%momentum_pdf

```
function muli_get_momentum_pdf(this,x_proton,gev2_scale,n,pdg_f) result(pdf)
  real(kind=double)::pdf
  class(muli_type),intent(in)::this
  real(kind=double),intent(in)::x_proton,gev2_scale
  integer,intent(in)::n,pdg_f
  call this%beam%momentum_pdf(x_proton,gev2_scale,n,pdg_f,pdf)
end function muli_get_momentum_pdf
```

muli_get_parton_pdf ↑

Wrapper für die eigentliche Routine pp_remnant_type%parton_pdf

```
function muli_get_parton_pdf(this,x_proton,gev2_scale,n,pdg_f) result(pdf)
  real(kind=double)::pdf
  class(muli_type),intent(in)::this
  real(kind=double),intent(in)::x_proton,gev2_scale
  integer,intent(in)::n,pdg_f
  call this%beam%parton_pdf(x_proton,gev2_scale,n,pdg_f,pdf)
end function muli_get_parton_pdf
```

muli_print_timer ↑

```
subroutine muli_print_timer(this)
  class(muli_type),intent(in) :: this
  print('("Init time:    ",E20.10)'),this%init_time
  print('("PT gen time:  ",E20.10)'),this%pt_time
  print('("Partons time: ",E20.10)'),this%partons_time
  print('("Confirm time: ",E20.10)'),this%confirm_time
  print('("Overall time: ",E20.10)'),&
  this%init_time+this%pt_time+this%partons_time+this%confirm_time
end subroutine muli_print_timer
```

muli_generate_next_scale ↑

Hier wird die nächste Wechselwirkungsskala $h_3^{(n+1)}$ generiert. Für jedes Stratum $\{\alpha,\beta\}$ wird die Unterfunktion generate_single_pts aufgerufen, um einen Wert $h_{3\alpha\beta}^{(n+1)}$ zu generieren. muli_generate_next_scale setzt dann $h_3^{(n+1)} = \max(h_{3\alpha\beta}^{(n+1)})$. Intern werden normierte Skalen $h_3 = \frac{4p_\perp}{s}$=pts (pt normiert auf s) verwendet.

```
subroutine muli_generate_next_scale(this,integrand_kind)
  class(muli_type),intent(inout)::this
  integer,intent(in),optional::integrand_kind
  real(kind=double)::pts,tmp_pts,rnd
  integer::tmp_int_kind
  class(muli_trapezium_node_class),pointer::tmp_node
    pts=-1D0
```

Das optionale Argument integrand_kind wird nur für interne Testzwecke verwendet, man sollte es gefahrlos samt der nachfolgenden Konstruktion entfernen können. integrand_kind ist ein Stratum $\{\alpha,\beta\}$, das vorgegeben werden kann.

```
    if(present(integrand_kind))then
       call tao_random_number(this%tao_rnd,rnd)
       call generate_single_pts(&
```

```
                integrand_kind,&
                this%start_integrals(integrand_kind),&
                this%beam%get_pdf_int_weights(double_pdf_kinds(1:2,integrand_kind)),&
                rnd,&
                this%dsigma,&
                pts,&
                this%node)
       else
```

Das ist der vorgesehene Weg, hier werden alle Strati mit tmp_int_kind durchlaufen. mit pp_remnant_type%get_pdf_int_weights werden die Wichtungsfaktoren $[W_\alpha, W_\beta]$ angefordert. in muli_interactions%double_pdf_kinds ist eine Abbildung $[1..16] \to \{\alpha, \beta\}$ definiert.

Nach jedem Aufruf von generate_single_pts wird überprüft, ob tmp_pts=$h_{3\alpha\beta}^{(n+1)}$ größer als der bisher größte Wert ist. Wenn ja, wird $h_3^{(n+1)} = h_{3\alpha\beta}^{(n+1)}$ gesetzt.

generate_single_pts liefert $h_{3\alpha\beta} = -1$ zurück, wenn $p_{\perp\alpha\beta} < p_\perp^{\min}$. Es reicht am Schluss also aus, wenn wir nachsehen, ob $h_3 > 0$. Wenn nicht, dann ist die Skala $p_\perp$ am unteren Ende angekommen und es werden keine weiteren MPI generiert.

tmp_node zeigt auf das Blatt der approximierten Wirkungsquerschnitte muli_type%dsigma, das $p_{\perp\alpha\beta}$ enthält. Dieses Blatt wird später noch benötigt, wenn aus dem hier generierten Kandidaten eine tatsächliche Wechselwirkung generiert wird.

```
          do tmp_int_kind=1,16
             call tao_random_number(this%tao_rnd,rnd)
             call generate_single_pts(&
                  tmp_int_kind,&
                  this%start_integrals(tmp_int_kind),&
                  this%beam%get_pdf_int_weights(double_pdf_kinds(1:2,tmp_int_kind)),&
                  rnd,&
                  this%dsigma,&
                  tmp_pts,&
                  tmp_node)
             if(tmp_pts>pts)then
                pts=tmp_pts
                this%integrand_id=tmp_int_kind
                this%node=>tmp_node
             end if
          end do
       end if
       if(pts>0)then
          call this%set_unit_scale(pts)
       else
          this%finished=.true.
       end if
   contains
```

Siehe Abschnitt **??**, (**??**)-(**??**) mit weight $\to W_\alpha W_\beta$, rnd $\to z$ und arg $\to \zeta$. Wenn $W_\alpha W_\beta = 0$, dann wird keine Skala generiert, weil mindestens einer der beiden beteiligte n Beiträge zur Strukturfunktion gleich Null ist.

Mit muli_trapezium_tree_type%find_decreasing wird muli_type%dsigma nach dem Blatt durchsucht, dessen Bildmenge von $\mathcal{S}_{\alpha\beta}$ den Wert ζ enthält. Wenn der Funktionswert l_integral von $\mathcal{S}_{\alpha\beta}$ an der unteren Intervallgrenze kleiner als ζ ist, dann liegt ζ tatsächlich nicht in der Bildmenge von $\mathcal{S}_{\alpha\beta}$. Damit ist klar, dass der gesuchte Wert von $p_\perp < p_\perp^{\min}$ ist und das Ergebnis wird auf den Wert -1 gesetzt. Andernfalls wird mittels muli_trapezium_type%approx_position_by_integral die Umkehrfunktion $\mathcal{S}_{\alpha\beta}^{-1}$ ausgewertet.

```
    subroutine generate_single_pts(int_kind,start_int,weight,rnd,int_tree,pts,node)
      integer,intent(in)::int_kind
      real(kind=double),intent(in)::start_int,weight,rnd
      type(muli_trapezium_tree_type),intent(in)::int_tree
      real(kind=double),intent(out)::pts
      class(muli_trapezium_node_class),pointer,intent(out)::node
      real(kind=double)::arg
      if(weight>0D0)then
         arg=start_int-log(rnd)/weight
         call int_tree%find_decreasing(arg,int_kind,node)
         if(node%get_l_integral(int_kind)>arg)then
            pts=node%approx_position_by_integral(int_kind,arg)
         else
            pts=-1D0
         end if
      else
         pts=-1D0
      end if
    end subroutine generate_single_pts
  end subroutine muli_generate_next_scale
```

muli_confirm ↑

Wird nur für Debuggingzwecke in Zusammenhang mit muli_type%generate_samples verwendet.

```
  subroutine muli_confirm(this)
    class(muli_type),intent(inout) :: this
    this%mean=this%node%approx_value_n(this%get_unit_scale(),this%integrand_id)
    this%start_integrals=this%node%approx_integral(this%get_unit_scale())
  end subroutine muli_confirm
```

muli_generate_samples ↑

Ein Generator für die Zerlegung des $\{h_1, h_2, h_3\}$-Einheitsquaders, siehe Abschnitt **??**. Diese Methode wird nur für Debugginzwecke verwendet.

```
  subroutine muli_generate_samples(this,n_total,n_print,integrand_kind,muli_dir,analyse)
    class(muli_type),intent(inout)::this
    integer(kind=dik),intent(in)::n_total,n_print
    integer,intent(in)::integrand_kind
    character(*),intent(in)::muli_dir
    logical,intent(in)::analyse
    integer(kind=dik)::n_inner

    class(muli_trapezium_node_class),pointer::start_node=>null()
    class(muli_trapezium_node_class),pointer,save::s_node=>null()
    class(muli_trapezium_node_class),pointer,save::node=>null()
```

```
character(2)::prefix
integer,save::t_slice,t_region,t_proc,t_subproc,t_max_n=0
integer(kind=dik)::n_t,n_p,n_m
integer::n,m,u,unit=0
integer(kind=dik)::n_tries=0
integer(kind=dik)::n_hits=0
integer(kind=dik)::n_over=0
integer(kind=dik)::n_miss=0
real(kind=double),save,dimension(3)::cart_hit
integer,save,dimension(4)::t_i_rnd
real(kind=double),dimension(16)::d_rnd
real(kind=double),save::t_area,t_dddsigma,t_rnd,t_weight,t_arg
real(kind=double)::mean=0D0
real(kind=double)::time=0D0
real(kind=double)::timepa=0D0
real(kind=double)::timept=0D0
real(kind=double)::timet=0D0
real(kind=double)::pts,s_pts=1D0
real(kind=double)::pts2=1D0
real(kind=double)::rnd
logical::running
character(3)::num
integer::success=-1
integer::chain_length=0
integer::int_kind
integer::process_id
real(kind=double),dimension(0:16)::integral
call this%print_parents()
n_tries=one
n_inner=n_total/n_print
n_t=zero
print:do while(n_t<n_total)
   call cpu_time(time)
   timet=-time
   n_p=zero
   inner:do while(n_p<n_print)
      chain_length=0
      call this%restart()
      this%integrand_id=integrand_kind
      call cpu_time(time)
      timept=timept-time
      call this%generate_next_scale(integrand_kind)
      call cpu_time(time)
      timept=timept+time
      chain:do while(.not.this%is_finished())
         chain_length=chain_length+1
         n_p=n_p+1
         call this%confirm()
         call cpu_time(time)
         timepa=timepa-time
```

```
          !             print *,this%get_unit2_scale()
          call sample_inclusive_mcgenerate_hit(&
               this%samples,&
               this%get_unit2_scale(),&
               this%mean,&
               this%integrand_id,&
               this%tao_rnd,&
               this%process_id,&
               this%momentum_fractions)
          call cpu_time(time)
          timepa=timepa+time
          timept=timept-time
          call this%generate_next_scale(integrand_kind)
          call cpu_time(time)
          timept=timept+time
       end do chain
    end do inner
    n_t=n_t+n_p
    call this%samples%sum_up()
    call cpu_time(time)
    timet=timet+time
    print *,n_t,"/",n_total
    print *,"time: ",timet
    print *,"pt time: ",timept
    print *,"pa time: ",timepa
    print *,this%samples%n_tries_sum,this%samples%n_hits_sum,this%samples%n_over_sum
    if(this%samples%n_hits_sum>0)then
       print *,(this%samples%n_hits_sum*10000)/this%samples%n_tries_sum,&
            (this%samples%n_over_sum*10000)/this%samples%n_hits_sum
    else
       print *,"no hits"
    end if
 end do print
 call integer_with_leading_zeros(integrand_kind,2,prefix)
 if(analyse)then
    call this%samples%int_kinds(integrand_kind)%analyse(muli_dir,prefix//"_")
    call this%samples%int_kinds(integrand_kind)%serialize(&
         "sample_int_kind_"//prefix,&
         muli_dir//"/sample_int_kind/"//prefix//".xml")
 end if
 call this%samples%int_kinds(integrand_kind)%serialize(&
      "sample_int_kind_"//prefix,&
      muli_dir//"/sample_int_kind/"//prefix//".xml")
end subroutine muli_generate_samples
```

muli_fake_interaction ↑

Wird ebenfalls zu Debuggingzwecken verwendet. So können Wechselwirkungen ohne Verwendung von WHIZARD oder das shower_interface auf die Remnants übertragen werden.

```
subroutine muli_fake_interaction(this,GeV2_scale,x1,x2,process_id,integrand_id,n1,n2,flow)
  class(muli_type),intent(inout)::this
```

```
    real(kind=double),intent(in)::Gev2_scale,x1,x2
    integer,intent(in)::process_id,integrand_id,n1,n2
    integer,dimension(4),intent(in),optional::flow
    call this%set_gev2_scale(Gev2_scale)
    this%process_id=process_id
    this%integrand_id=integrand_id
    this%parton_ids=[n1,n2]
    if(present(flow))then
       this%flow=flow
    else
       this%flow=[0,0,0,0]
    end if
    this%momentum_fractions=[x1,x2,this%get_unit2_scale()]
    call this%beam%apply_interaction(this)
    call this%beam%print_all()
  end subroutine muli_fake_interaction

end module muli
```

4 Modul muli_momentum

4.1 Abhängigkeiten

```
use muli_basic
```

4.2 Derived Types

4.2.1 transversal_momentum_type

Dieser Datentyp abstrahiert den Entwicklungsparameter $p_\perp$. Intern wird $p_\perp$ durch die Komponente `momentum` dargestellt. Dieser enthält die fünf Einträge $\left[s, \sqrt{p_\perp}, p_\perp, \sqrt{4 * p_\perp / s}, 4 * p_\perp / s\right]$. Für jeden Eintrag werden get und set Methoden bereitgestellt. Wenn statt eines Werts von $p_\perp$ eine Instanz vom Typ `transversal_momentum_type` übergeben wird, werden Fehler durch falsche Einheiten vermieden.

Nach MulI-Konvention wird die Einheit immer vor den Namen der Variable gestellt, also GeV_scale für $\sqrt{p_\perp}$ und GeV2_scale für $p_\perp$ usw. Hier wird bei dimensionslosen Größen $\sim \sqrt{p_\perp}$ das Prefix unit und bei dimensionslosen Größen $\sim p_\perp$ das prefix unit2 vorangestellt, damit immer klar ist, was gemeint ist.

`MaxScale` ist der größtmögliche Wert für $p_\perp$, mit $p_\perp^{\max} = s/2$.

Die invariante Masse s wird hier `initial_cme` genannt. Letzteres ist die invariante Masse des Proton-Proton-Systems vor der ersten (WHIZARD) Wechselwirkung. Das spiegelt die Tatsache wider, dass dynamische Energieen der Remnants noch nicht implementiert sind. Dieser Datentyp ist der richtige Ort, um die aktuelle CME zu speichern. Da alle anderen Module auf die get-Methoden dieses Moduls zurückgreifen, sollte es ausreichen, die interne Darstellung hier anzupassen.

```
implicit none
  type,extends(serializable_class)::transversal_momentum_type
     private
     real(kind=drk),dimension(0:4)::momentum=[0D0,0D0,0D0,0D0,0D0]
   contains
     !Überschriebene serializable_class Methoden
     procedure,public::write_to_marker=>transversal_momentum_write_to_marker
     procedure,public::read_from_marker=>transversal_momentum_read_from_marker
     procedure,public::print_to_unit=>transversal_momentum_print_to_unit
     procedure,public,nopass::get_type=>transversal_momentum_get_type
     !Originäre transversal_momentum_type Methodentransversal_momentum_type
     procedure,public::get_gev_initial_cme=>transversal_momentum_get_gev_initial_cme
     procedure,public::get_gev_max_scale=>transversal_momentum_get_gev_max_scale
     procedure,public::get_gev2_max_scale=>transversal_momentum_get_gev2_max_scale
     procedure,public::get_gev_scale=>transversal_momentum_get_gev_scale
```

```
    procedure,public::get_gev2_scale=>transversal_momentum_get_gev2_scale
    procedure,public::get_unit_scale=>transversal_momentum_get_unit_scale
    procedure,public::get_unit2_scale=>transversal_momentum_get_unit2_scale
    procedure,public::set_gev_initial_cme=>transversal_momentum_set_gev_initial_cme
    procedure,public::set_gev_max_scale=>transversal_momentum_set_gev_max_scale
    procedure,public::set_gev2_max_scale=>transversal_momentum_set_gev2_max_scale
    procedure,public::set_gev_scale=>transversal_momentum_set_gev_scale
    procedure,public::set_gev2_scale=>transversal_momentum_set_gev2_scale
    procedure,public::set_unit_scale=>transversal_momentum_set_unit_scale
    procedure,public::set_unit2_scale=>transversal_momentum_set_unit2_scale
    procedure,public::transversal_momentum_initialize
    generic,public::initialize=>transversal_momentum_initialize
 end type transversal_momentum_type
```

4.2.2 qcd_2_2_class

Abstrakte Klasse, die eine QCD-2→2-Wechselwirkung abstrahiert. pp_remnant_type greift auf Eigenschaften einer solchen Wechselwirkung zurück, allerdings werden die Methoden in dem Modul muli implementiert, auf das muli_remnant keinen Zugriff hat. Zwar könnte man dieses Problem durch eine andere Hierarchie von Modulen lösen, aber ich nehme an, dass dieses Problem wieder auftaucht, wenn die Remnants als WHIZARD-Strukturfunktionen implementiert werden. Deswegen habe ich diese Lösung gewählt.

```
 type,Extendstransversal_momentum_type,abstract::qcd_2_2_class
  contains
    procedure(qcd_get_int),deferred::get_process_id
    procedure(qcd_get_int),deferred::get_integrand_id
    procedure(qcd_get_int),deferred::get_diagram_kind
    procedure(qcd_get_int_4),deferred::get_lha_flavors
    procedure(qcd_get_int_4),deferred::get_pdg_flavors
    procedure(qcd_get_int_by_int),deferred::get_parton_id
    procedure(qcd_get_int_2),deferred::get_parton_kinds
    procedure(qcd_get_int_2),deferred::get_pdf_int_kinds
    procedure(qcd_get_drk),deferred::get_momentum_boost
    procedure(qcd_get_drk_2),deferred::get_remnant_momentum_fractions
    procedure(qcd_get_drk_2),deferred::get_total_momentum_fractions
 end type qcd_2_2_class
```

4.3 Interfaces

```
   abstract interface
    subroutine qcd_none(this)
      import qcd_2_2_class
      class(qcd_2_2_class),target,intent(in)::this
    end subroutine qcd_none
    elemental function qcd_get_drk(this)
      use muli_basic, only: drk
      import qcd_2_2_class
      class(qcd_2_2_class),intent(in)::this
```

```
      real(kind=drk)::qcd_get_drk
    end function qcd_get_drk
    pure function qcd_get_drk_2(this)
      use muli_basic, only: drk
      import qcd_2_2_class
      class(qcd_2_2_class),intent(in)::this
      real(kind=drk),dimension(2)::qcd_get_drk_2
    end function qcd_get_drk_2
    pure function qcd_get_drk_3(this)
      use muli_basic, only: drk
      import qcd_2_2_class
      class(qcd_2_2_class),intent(in)::this
      real(kind=drk),dimension(3)::qcd_get_drk_3
    end function qcd_get_drk_3
    elemental function qcd_get_int(this)
      use muli_basic, only: drk
      import qcd_2_2_class
      class(qcd_2_2_class),intent(in)::this
      integer::qcd_get_int
    end function qcd_get_int
    elemental function qcd_get_int_by_int(this,n)
      use muli_basic, only: drk
      import qcd_2_2_class
      class(qcd_2_2_class),intent(in)::this
      integer,intent(in)::n
      integer::qcd_get_int_by_int
    end function qcd_get_int_by_int
    pure function qcd_get_int_2(this)
      use muli_basic, only: drk
      import qcd_2_2_class
      class(qcd_2_2_class),intent(in)::this
      integer,dimension(2)::qcd_get_int_2
    end function qcd_get_int_2
    pure function qcd_get_int_4(this)
      use muli_basic, only: drk
      import qcd_2_2_class
      class(qcd_2_2_class),intent(in)::this
      integer,dimension(4)::qcd_get_int_4
    end function qcd_get_int_4
  end interface
```

4.4 Implementierung der Prozeduren

4.4.1 Methoden für transversal_momentum_type

Überschriebene serializable_class Methoden

transversal_momentum_write_to_marker ↑

```
  subroutine transversal_momentum_write_to_marker(this,marker,status)
    class(transversal_momentum_type),intent(in)::this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    call marker%mark_begin("transversal_momentum_type")
    call marker%mark("gev_momenta",this%momentum(0:1))
    call marker%mark_end("transversal_momentum_type")
  end subroutine transversal_momentum_write_to_marker
```

transversal_momentum_read_from_marker ↑

```
  subroutine transversal_momentum_read_from_marker(this,marker,status)
    class(transversal_momentum_type),intent(out)::this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    call marker%pick_begin("transversal_momentum_type",status=status)
    call marker%pick("gev_momenta",this%momentum(0:1),status)
    this%momentum(2:4)=[&
         this%momentum(1)**2,&
         this%momentum(1)/this%momentum(0),&
         (this%momentum(1)/this%momentum(0))**2]
    call marker%pick_end("transversal_momentum_type",status=status)
  end subroutine transversal_momentum_read_from_marker
```

transversal_momentum_print_to_unit ↑

```
  subroutine transversal_momentum_print_to_unit(this,unit,parents,components,peers)
    class(transversal_momentum_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    write(unit,'("Components of transversal_momentum_type:")')
    write(unit,fmt='("Actual energy scale:")')
    write(unit,fmt='("Max scale (MeV)    :",E20.10)')this%momentum(0)
    write(unit,fmt='("Scale (MeV)        :",E20.10)')this%momentum(1)
    write(unit,fmt='("Scale^2 (MeV^2)    :",E20.10)')this%momentum(2)
    write(unit,fmt='("Scale normalized   :",E20.10)')this%momentum(3)
    write(unit,fmt='("Scale^2 normalized:",E20.10)')this%momentum(4)
  end subroutine transversal_momentum_print_to_unit
```

transversal_momentum_get_type ↑

```
  pure subroutine transversal_momentum_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="transversal_momentum_type")
  end subroutine transversal_momentum_get_type
```

Originäre transversal_momentum_type Methoden

transversal_momentum_get_gev_initial_cme ↑

```
elemental function transversal_momentum_get_gev_initial_cme(this) result(scale)
  class(transversal_momentum_type),intent(in)::this
  real(kind=drk)::scale
  scale=this%momentum(0)*2D0
end function transversal_momentum_get_gev_initial_cme
```

transversal_momentum_get_gev_max_scale ↑

transversal_momentum_get_gev_max_scale ↑

```
elemental function transversal_momentum_get_gev_max_scale(this) result(scale)
  class(transversal_momentum_type),intent(in)::this
  real(kind=drk)::scale
  scale=this%momentum(0)
end function transversal_momentum_get_gev_max_scale
```

transversal_momentum_get_gev2_max_scale ↑

```
elemental function transversal_momentum_get_gev2_max_scale(this) result(scale)
  class(transversal_momentum_type),intent(in)::this
  real(kind=drk)::scale
  scale=this%momentum(0)**2
end function transversal_momentum_get_gev2_max_scale
```

transversal_momentum_get_gev_scale ↑

```
elemental function transversal_momentum_get_gev_scale(this) result(scale)
  class(transversal_momentum_type),intent(in)::this
  real(kind=drk)::scale
  scale=this%momentum(1)
end function transversal_momentum_get_gev_scale
```

transversal_momentum_get_gev2_scale ↑

```
elemental function transversal_momentum_get_gev2_scale(this) result(scale)
  class(transversal_momentum_type),intent(in)::this
  real(kind=drk)::scale
  scale=this%momentum(2)
end function transversal_momentum_get_gev2_scale
```

transversal_momentum_get_unit_scale ↑

```
elemental function transversal_momentum_get_unit_scale(this) result(scale)
  class(transversal_momentum_type),intent(in)::this
  real(kind=drk)::scale
  scale=this%momentum(3)
end function transversal_momentum_get_unit_scale
```

transversal_momentum_get_unit2_scale ↑

```
elemental function transversal_momentum_get_unit2_scale(this) result(scale)
  class(transversal_momentum_type),intent(in)::this
  real(kind=drk)::scale
  scale=this%momentum(4)
end function transversal_momentum_get_unit2_scale
```

transversal_momentum_set_gev_initial_cme ↑

```
subroutine transversal_momentum_set_gev_initial_cme(this,new_gev_initial_cme)
  class(transversal_momentum_type),intent(inout)::this
  real(kind=drk),intent(in) :: new_gev_initial_cme
  this%momentum(0) = new_gev_initial_cme/2D0
  this%momentum(3) = this%momentum(1)/this%momentum(0)
  this%momentum(4) = this%momentum(3)**2
end subroutine transversal_momentum_set_gev_initial_cme
```

transversal_momentum_set_gev_max_scale ↑

```
subroutine transversal_momentum_set_gev_max_scale(this,new_gev_max_scale)
  class(transversal_momentum_type),intent(inout)::this
  real(kind=drk),intent(in) :: new_gev_max_scale
  this%momentum(0) = new_gev_max_scale
  this%momentum(3) = this%momentum(1)/this%momentum(0)
  this%momentum(4) = this%momentum(3)**2
end subroutine transversal_momentum_set_gev_max_scale
```

transversal_momentum_set_gev2_max_scale ↑

```
subroutine transversal_momentum_set_gev2_max_scale(this,new_gev2_max_scale)
  class(transversal_momentum_type),intent(inout)::this
  real(kind=drk),intent(in) :: new_gev2_max_scale
  this%momentum(0) = sqrt(new_gev2_max_scale)
  this%momentum(3) = this%momentum(1)/this%momentum(0)
  this%momentum(4) = this%momentum(3)**2
end subroutine transversal_momentum_set_gev2_max_scale
```

transversal_momentum_set_gev_scale ↑

```
subroutine transversal_momentum_set_gev_scale(this,new_gev_scale)
  class(transversal_momentum_type),intent(inout)::this
  real(kind=drk),intent(in) :: new_gev_scale
  this%momentum(1) = new_gev_scale
  this%momentum(2) = new_gev_scale**2
  this%momentum(3) = new_gev_scale/this%momentum(0)
  this%momentum(4) = this%momentum(3)**2
end subroutine transversal_momentum_set_gev_scale
```

transversal_momentum_set_gev2_scale ↑

```
subroutine transversal_momentum_set_gev2_scale(this,new_gev2_scale)
  class(transversal_momentum_type),intent(inout)::this
  real(kind=drk),intent(in) :: new_gev2_scale
  this%momentum(1) = sqrt(new_gev2_scale)
  this%momentum(2) = new_gev2_scale
  this%momentum(3) = this%momentum(1)/this%momentum(0)
  this%momentum(4) = this%momentum(3)**2
end subroutine transversal_momentum_set_gev2_scale
```

transversal_momentum_set_unit_scale ↑

```
subroutine transversal_momentum_set_unit_scale(this,new_unit_scale)
  class(transversal_momentum_type),intent(inout)::this
  real(kind=drk),intent(in) :: new_unit_scale
  this%momentum(1) = new_unit_scale*this%momentum(0)
  this%momentum(2) = this%momentum(1)**2
  this%momentum(3) = new_unit_scale
  this%momentum(4) = this%momentum(3)**2
end subroutine transversal_momentum_set_unit_scale
```

transversal_momentum_set_unit2_scale ↑

```
subroutine transversal_momentum_set_unit2_scale(this,new_unit2_scale)
  class(transversal_momentum_type),intent(inout)::this
  real(kind=drk),intent(in) :: new_unit2_scale
  this%momentum(3) = sqrt(new_unit2_scale)
  this%momentum(4) = new_unit2_scale
  this%momentum(1) = this%momentum(3)*this%momentum(0)
  this%momentum(2) = this%momentum(1)**2
end subroutine transversal_momentum_set_unit2_scale
```

transversal_momentum_initialize ↑

```
subroutine transversal_momentum_initialize(this,gev2_s)
  class(transversal_momentum_type),intent(out)::this
  real(kind=drk),intent(in)::gev2_s
  real(kind=drk)::gev_s
  gev_s=sqrt(gev2_s)
  this%momentum=[gev_s/2D0,gev_s/2D0,gev2_s/4D0,1D0,1D0]
end subroutine transversal_momentum_initialize
```

5 Modul muli_remnant

5.1 Allgemeines

5.1.1 Zweck

Das Modul muli_remnant enthält die vollständige Beschreibung der Remnants. Bislang sind die ursprünglichen Hadronen fest auf Protonen implementiert, allerdings sind Verallgemeinerungen auf größere Klassen teilweise vorbereitet.

5.1.2 Voraussetzungen

Nicht in diesem Modul enthalten sind:

- Die ursprünglichen Strukturfunktionen der ungestörten Hadronen

  Diese werden für die Definition der Remnant-PDFs verwendet. Verschiedene PDF-Sets können daher auch zu verschiedene Remnant-PDFs führen. Im Moment werden die LHAPDF-Bibliotheken verwendet. muli_remnant verlässt sich darauf, dass diese bereits initialisiert sind und initialisiert sie nicht selbst.

  In pp_remnant_type%initialize werden zwar alle notwendigen LHAPDF Informationen, also Unix-Verzeichnis, Dateiname und Member, entgegengenommen. Diese werden aber nur verwendet, um die passenden integrierten PDFs pp_remnant_type%pdf_norm zu deserialisieren.

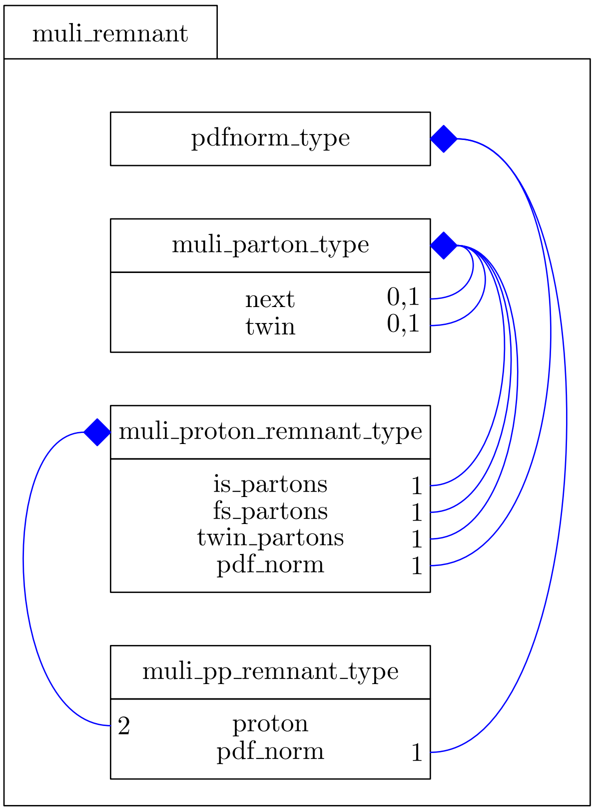
  Es wird das Modul pdf_builtin eingebunden, aber im aktuellen Status nicht verwendet.

- muli_interactions

5.1.3 Schnittstelle

Die meisten Typen und Prozeduren in diesem Modul sind nur für interne Zwecke vorgesehen. Der erweiterte Datentyp muli_pp_remnant_type wurde eigens implementiert, um als Schnittstelle nach außen zu dienen. Er enthält zwei Instanzen vom Typ muli_proton_remnant_type und abstrahiert auf diese Weise die einzelnen Hadron-Remnants. Deshalb sollte es auch ohne Änderung der Schnittstelle möglich sein, andere Hadron-Remnants zu realisieren. muli_pp_remnant_type enthält Wrapper-Methoden für alle momentan benötigten Informationen. Von muli_type werden verwendet:

- pp_remnant_type%initialize
- pp_remnant_type%finalize
- pp_remnant_type%reset
- pp_remnant_type%replace_parton
- pp_remnant_type%apply_interaction



- pp_remnant_type%apply_initial_interaction
- pp_remnant_type%momentum_pdf
- pp_remnant_type%parton_pdf
- pp_remnant_type%get_pdf_int_weights

Zusammen mit den überladenen Standardmethoden der Klasse serializable_class

- pp_remnant_type%write_to_marker
- pp_remnant_type%read_from_marker
- pp_remnant_type%print_to_unit
- pp_remnant_type%get_type

ergeben diese Methoden eine vollständige Schnittstelle für die Remnants. Konsequenterweise sollten alle Modul-Komponenten, alle Modul-Prozeduren und alle bis auf die erwähnten Type-Bound-Prozeduren als privat deklariert werden. Derzeit sind die Prozeduren nur deshalb public, um das Debugging zu erleichtern.

5.1.4 Datentypen

Alle Datentypen sind Erweiterungen von serializable_class.

- pdfnorm_type:

  Für die Berechnung der Wichtungsfaktoren W_k für die einzelnen Beiträge $f_k(x, \mu_F)$ zur Remnant-Strukturfunktion ist nicht die volle Information $f_k(x, \mu_F)$ notwendig. Es reicht aus, den Impulsmittelwert $\langle f_k(\mu_F) \rangle = \int_{x=x_0}^{1} x f_k(x, \mu_F)$ zu kennen. pdfnorm_type liefert eben diesen Impulsmittelwert. Außerdem enthält pdfnorm_type die Summe über alle Impulsmittelwerte. Idealerweise sollte diese Summe geich Eins sein. Damit aber Abweichungen von Eins die Summenregel (**??**) nicht beeinflussen, werden alle Impulserwartungswerte auf die Summe normiert, daher auch der Name pdfnorm_type[1].

- parton_type:

  Für die Berechnung der Wichtungsfaktoren ist es weiterhin notwendig zu wissen, welche Partonen bereits aus dem Remnant entnommen wurden. parton_type ist eine Liste von eben diesen Partonen. Für Seequarks ist es außerdem wichtig, die Eigenschaften des Splittingpartners aus dem $g \to q_S q_Q$ Gluonsplitting zu kennen. parton_type enthält folglich einen Zeiger parton_type%twin auf das jeweils andere Quark eines solchen Splittings

- proton_remnant_type:

  Container für alle Eigenschaften eines Proton-Remnants und für spezifische Methoden zur Bestimmung der Wichtungsfaktoren für Proton-Remnants.

- pp_remnant_type:

  Abstrahierung der beiden Remnants und Schnittstelle für das Modul.

5.2 Abhängigkeiten

```
use,intrinsic::iso_fortran_env
use pdf_builtin !NODEP!
use tao_random_numbers !NODEP!
use muli_basic
use muli_interactions
use muli_momentum
```

5.3 Parameter

muli_remnant%nx und muli_remnant%nq sind Parameter fur die Approximation von $\langle f_k(\mu) \rangle$. nx ist die Zahl der Stützstellen für x und nq ist die Zahl der Stützstellen für $\mu_F = Q$, bei denen $f_k(x, \mu_F)$ dafür ausgewertet wird. Da über x integriert wird, nehmen die Werte für x keinen Speicher in Anspruch, sondern nur CPU-Zeit. Die Werte für μ_F werden hingegen gespeichert.

muli_remnant%remnant_weight_model und muli_remnant%gluon_exp sind Modellparameter für die Behandlung der Remnants. remnant_weight_model gibt an, wie die Wichtungsfaktoren durch die einzige Bedingung (**??**) ausgedrückt werden. In Tabelle **??** ist eine Übersicht angegeben. In proton_remnant_type%calculate_weight werden die Wichtungsfaktoren bestimmt.

[1] hängt stark von den Integrationsparametern ab, bei exakterer Integration scheint der Fehler beliebig klein zu werden, dadurch wird die CPU-Last aber zu groß. Renormierung ist eine deutlich billigere Methode. In den aktuellen Einstellung mit einer gut angepassten Verteilung von 10^7 Stützstellen ist die Abweichung mit $< 10^{-4}$ eigentlich irrelevant, dafür dauert es einige Minuten, bis alle Integrationen fertig sind.

gluon_exp ist ein Parameter für die Approximation der Quasivalenzquarkbeiträge, siehe remnant_gluon_pdf_a

```
implicit none
integer,parameter::nx=10000000
integer,parameter::nq=60
integer,public::remnant_weight_model=2
integer::gluon_exp=4
```

5.4 Derived Types

5.4.1 pdfnorm_type

pdfnorm_type approximiert die Impulserwartungswerte $\langle f_k(\mu)\rangle = \int_{x=x_0}^{1} x f_k(x,\mu)$ aller Beiträge k zur Strukturfunktion. pdfnorm_type hat nur zwei neue Methoden, scan und get_norm. scan Wertet $f_k(x,\mu)$ an allen $nx \otimes nq$ Stellen für alle 13 Einträge der LHAPDF-Sets aus und bestimmt daraus die Werte

$$\text{pdf\_norm}(\mu) = \left[\langle f_\Sigma(\mu)\rangle\,, \frac{\langle f_g(\mu)\rangle}{\langle f_\Sigma(\mu)\rangle}, \frac{\langle f_s(\mu)\rangle}{\langle f_\Sigma(\mu)\rangle}, \frac{\langle f_{d^v}(\mu)\rangle}{\langle f_\Sigma(\mu)\rangle}, \frac{\langle f_{u^v}(\mu)\rangle}{\langle f_\Sigma(\mu)\rangle}\right] \tag{5.1}$$

für nq verschiedene Werte von μ_F. $\langle f_\Sigma(\mu)\rangle$ bezeichnet die Summe über die Impulsmittelwerte aller 13 Einträge der LHAPDF-Sets. pdf_int$(k,\mu_F) = \langle f_k(\mu_F)\rangle\,, k =$LHAPDF-Flavor, ist ein Zwischenschritt und wird eigentlich nicht benötigt, sobald pdf_norm bestimmt ist. pdf_int kann also gefahrlos von der Liste der Komponenten entfernt werden. Es wird nur zu Debugging-Zwecken mitgeführt.

pdfnorm_type%qmin, pdfnorm_type%qmax und pdfnorm_type%dq legen die Abbildung $j = 1..nq \to \mu_j$ fest, wobei j der zweite Index von pdf_int bzw. pdf_norm ist. Allerdings tragen sie die etwas unübliche Einheit $\sqrt{GeV}$. Es gilt:

$$\mu_j = \left(qmin + j\ dq\right)^2 \tag{5.2}$$

```
type,extends(serializable_class)::pdfnorm_type
   real(kind=double)::qmin,qmax,dq
   real(kind=double),dimension(-6:6,0:nq)::pdf_int
   real(kind=double),dimension(0:4,0:nq)::pdf_norm
 contains
   !Überschriebene serializable_class Methoden
   procedure::write_to_marker=>pdfnorm_write_to_marker
   procedure::read_from_marker=>pdfnorm_read_from_marker
   procedure::print_to_unit=>pdfnorm_print_to_unit
   procedure,nopass::get_type=>pdfnorm_get_type
   procedure,nopass::verify_type=>pdfnorm_verify_type
   !Originäre pdfnorm_type Methoden
   procedure::scan=>pdfnorm_scan
   procedure::get_norm=>pdfnorm_get_norm
end type pdfnorm_type
```

5.4.2 parton_type

parton_type ist eine Liste von Partonen. Jedes Parton bekommt eine eindeutige parton_type%id, die mit der id des ISR-Algorithmus übereinstimmt. So können sich ISR und MPI miteinander verständigen, mit welchem Parton etwas geschieht.

parton_type%lha_flavor ist das Flavor des Partons im LHA-Schema

> Handelt es sich nicht um ein Quasivalenzquark, dann ist parton_type%momentum der Impulsanteil ξ, bezogen auf den aktuellen Remnantimpuls, des Partons. Bei einem Quasivalenzquark hingegen ist parton_type%momentum der ungewichtete Impulserwartungswert des Quasivalenzquarks.

parton_type%twin ist nur von Bedeutung, wenn das Parton ein Seequark oder ein Quasivalenzquark ist. Dann ist twin der Spittingpartner des vorangegangenen Gluonsplittings.

parton_type%next ist das nächste Parton in der Liste.

```
type,Extendsserializable_class::parton_type
   private
   integer::id=-1
   integer::lha_flavor
   real(kind=double)::momentum=-1D0
   class(parton_type),pointer::twin=>null()
   class(parton_type),pointer::next=>null()
 contains
   !Überschriebene serializable_class Methoden
   procedure::write_to_marker=>parton_write_to_marker
   procedure::read_from_marker=>parton_read_from_marker
   procedure::print_to_unit=>parton_print_to_unit
   procedure,nopass::get_type=>parton_get_type
   !Originäre parton_type Methoden
   procedure::unweighted_pdf=>twin_unweighted_pdf
   procedure::deallocate=>twin_deallocate
   procedure::push=>parton_push
   procedure::pop_by_id=>parton_pop_by_id
   procedure::pop_by_association=>parton_pop_by_association
   generic::pop=>pop_by_id,pop_by_association
end type parton_type
```

5.4.3 proton_remnant_type

proton_remnant_type enthält den aktuellen Status eines Proton-Remnants. Das sind

die Anzahl der jeweiligen Valenzquarks proton_remnant_type%valence_content,

die Anzahl aller Quasivalenzquarks proton_remnant_type%n_twins,

die aktuellen Wichtungsfaktoren proton_remnant_type%pdf_int_weight (siehe (**??**)),

den Remnantimpuls dividiert durch den ursprünglichen Protonimpuls proton_remnant_type%momentum_fraction,

die Summe der ungewichteten Impulsmittelwerte der Quasivalenzquarks proton_remnant_type%twin_norm,

die Liste der Quasivalenzquarks im Remnant proton_remnant_type%twin_partons, die Liste der aktiven Initial State Partonen proton_remnant_type%is_partons,

die Liste der aktiven Final State Partonen proton_remnant_type%fs_partons

sowie eine redundante Referenz auf die integrierten LHAPDFs proton_remnant_type%pdf_norm. Redundant bedeutet, dass mehrere Instanzen einen Zeiger auf dasselbe Ziel haben. Die Allokierung der pdf_norm wird einer Instanz des Datentyps pp_remnant_type durchgeführt. Nur diese Instanz sollte auch die Deallokierung durchführen.

```
type,extends(serializable_class)::proton_remnant_type
   integer,dimension(2)::valence_content=[1,2]
   integer::n_twins=0
   ![gluon,sea quark,valence down,valence up,twin]
   real(kind=drk),dimension(5)::pdf_int_weight=[1D0,1D0,1D0,1D0,0D0]
   real(kind=drk)::momentum_fraction=1D0
   real(kind=double)::twin_norm=0D0
   type(parton_type)::twin_partons
   type(parton_type)::is_partons
   type(parton_type)::fs_partons
   ! these pointers shall not be allocated, deallocated, serialized or
   ! deserialized explicitly.
   class(pdfnorm_type),pointer::pdf_norm=>null()
 contains
   !Überschriebene serializable_class Methoden
   procedure::write_to_marker=>proton_remnant_write_to_marker
   procedure::read_from_marker=>proton_remnant_read_from_marker
   procedure::print_to_unit=>proton_remnant_print_to_unit
   procedure,nopass::get_type=>proton_remnant_get_type
   !Originäre proton_remnant_type Methoden
   ! manipulating parton content
   procedure::remove_valence_quark=>proton_remnant_remove_valence_quark
   procedure::remove_sea_quark=>proton_remnant_remove_sea_quark
   procedure::remove_gluon=>proton_remnant_remove_gluon
   procedure::remove_valence_up_quark=>proton_remnant_remove_valence_up_quark
   procedure::remove_valence_down_quark=>proton_remnant_remove_valence_down_quark
   procedure::remove_twin=>proton_remnant_remove_twin
   ! getting pdf
   procedure::momentum_twin_pdf=>proton_remnant_momentum_twin_pdf
   procedure::momentum_twin_pdf_array=>proton_remnant_momentum_twin_pdf_array
   procedure::momentum_kind_pdf=>proton_remnant_momentum_kind_pdf
   procedure::momentum_flavor_pdf=>proton_remnant_momentum_flavor_pdf
   procedure::momentum_kind_pdf_array=>proton_remnant_momentum_kind_pdf_array
   procedure::momentum_flavor_pdf_array=>proton_remnant_momentum_flavor_pdf_array
   procedure::parton_twin_pdf=>proton_remnant_parton_twin_pdf
   procedure::parton_twin_pdf_array=>proton_remnant_parton_twin_pdf_array
   procedure::parton_kind_pdf=>proton_remnant_parton_kind_pdf
   procedure::parton_flavor_pdf=>proton_remnant_parton_flavor_pdf
   procedure::parton_kind_pdf_array=>proton_remnant_parton_kind_pdf_array
   procedure::parton_flavor_pdf_array=>proton_remnant_parton_flavor_pdf_array
   ! getting components
   procedure::get_pdf_int_weight=>proton_remnant_get_pdf_int_weight
   procedure::get_valence_down_weight=>proton_remnant_get_valence_down_weight
```

```
    procedure::get_valence_up_weight=>proton_remnant_get_valence_up_weight
    procedure::get_valence_weight=>proton_remnant_get_valence_weight
    procedure::get_gluon_weight=>proton_remnant_get_gluon_weight
    procedure::get_sea_weight=>proton_remnant_get_sea_weight
    procedure::get_twin_weight=>proton_remnant_get_twin_weight
    procedure::get_valence_content=>proton_remnant_get_valence_content
    procedure::get_momentum_fraction=>proton_remnant_get_momentum_fraction
    ! misc
    procedure::deallocate=>proton_remnant_deallocate
    procedure::initialize=>proton_remnant_initialize
    procedure::finalize=>proton_remnant_finalize
    procedure::apply_initial_splitting=>proton_remnant_apply_initial_splitting
    procedure::reset=>proton_remnant_reset
    ! private
    procedure,private::calculate_weight=>proton_remnant_calculate_weight
    procedure,private::push_is_parton=>proton_remnant_push_is_parton
    procedure,private::push_twin=>proton_remnant_push_twin
    procedure,private::calculate_twin_norm=>proton_remnant_calculate_twin_norm
    procedure,private::replace_is_parton=>proton_remnant_replace_is_parton
    ! plots
    procedure::gnuplot_momentum_kind_pdf_array=>proton_remnant_gnuplot_momentum_kind_pdf_array
  end type proton_remnant_type
```

5.4.4 pp_remnant_type

pp_remnant_type abstrahiert die einzelnen Hadron-Remnants und dient als Schnittstelle für das komplette Modul. Aus anderen Modulen heraus sollen keine anderen Methoden, als die hier definierten, aufgerufen werden. Deswegen hat pp_remnant_type als einziger Datentyp in diesem Modul eine Komponente „initialized“, so dass eine Warnung ausgegeben werden kann, wenn ein Zugriff auf nichtinitialisierte Komponenten versucht wird.

> pp_remnant_type%gev_cme_tot soll die aktuelle invariante Masse des hadronischen Systems zurückgeben. Dynamische invariante Massen sind aber noch nicht implementiert. pp_remnant_type%X ist die aktuelle invariante Masse dividiert durch die ursprüngliche invariante Masse des Proton-Proton-Systems, allerdings wird diese Variable (noch) nirgends verwendet.

pp_remnant_type%proton sind die beiden Proton-Remnants.

pp_remnant_type%pdfnorm_type sind die Impulsmittelwerte der PDFs.

```
  type,Extendsserializable_class::pp_remnant_type
    logical::initialized=.false.
    real(kind=double),private::gev_initial_cme = gev_cme_tot
    real(kind=double),private::X=1D0
    type(proton_remnant_type),dimension(2)::proton
    class(pdfnorm_type),pointer,private::pdf_norm
   contains
    !Überschriebene serializable_class Methoden
    procedure::write_to_marker=>pp_remnant_write_to_marker
    procedure::read_from_marker=>pp_remnant_read_from_marker
    procedure::print_to_unit=>pp_remnant_print_to_unit
```

```
    procedure,nopass::get_type=>pp_remnant_get_type
    !Originäre pp_remnant_type Methoden
    ! init /final
    procedure::initialize=>pp_remnant_initialize
    procedure::finalize=>pp_remnant_finalize
    procedure::reset=>pp_remnant_reset
    ! manipulating parton content
    procedure::apply_initial_interaction=>pp_remnant_apply_initial_interaction
    procedure::replace_parton=>pp_remnant_replace_parton
    procedure::apply_interaction=>pp_remnant_apply_interaction
    ! getting pdfs
    procedure::momentum_pdf=>pp_remnant_momentum_pdf
    procedure::parton_pdf=>pp_remnant_parton_pdf
    procedure::get_proton_remnant_momentum_fractions=>pp_remnant_get_proton_remnant_momentu
    procedure::get_remnant_parton_flavor_pdf_arrays=>pp_remnant_get_remnant_parton_flavor_p
    ! getting components
    procedure::get_pdf_int_weights=>pp_remnant_get_pdf_int_weights
    procedure::get_pdf_int_weight=>pp_remnant_get_pdf_int_weight
    procedure::set_pdf_weight=>pp_remnant_set_pdf_weight
    procedure::get_gev_initial_cme=>pp_remnant_get_gev_initial_cme
    procedure::get_gev_actual_cme=>pp_remnant_get_gev_actual_cme
    procedure::get_cme_fraction=>pp_remnant_get_cme_fraction
    procedure::get_proton_remnants=>pp_remnant_get_proton_remnants
  end type pp_remnant_type
```

5.5 Implementierung der Methoden

5.5.1 Methoden für pdfnorm_type

pdfnorm_write_to_marker ↑

```
  subroutine pdfnorm_write_to_marker(this,marker,status)
    class(pdfnorm_type),intent(in)::this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    call marker%mark_begin("pdfnorm_type")
    call marker%mark("qmin",this%qmin)
    call marker%mark("qmax",this%qmax)
    call marker%mark("dq",this%dq)
    call marker%mark("pdf_int",this%pdf_int)
    call marker%mark("pdf_norm",this%pdf_norm)
    call marker%mark_end("pdfnorm_type")
  end subroutine pdfnorm_write_to_marker
```

pdfnorm_read_from_marker ↑

```
  subroutine pdfnorm_read_from_marker(this,marker,status)
    class(pdfnorm_type),intent(out)::this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
```

```
    character(:),allocatable::name
    call marker%pick_begin("pdfnorm_type",status=status)
    call marker%pick("qmin",this%qmin,status)
    call marker%pick("qmax",this%qmax,status)
    call marker%pick("dq",this%dq,status)
    call marker%pick("pdf_int",this%pdf_int,status)
    call marker%pick("pdf_norm",this%pdf_norm,status)
    call marker%pick_end("pdfnorm_type",status=status)
  end subroutine pdfnorm_read_from_marker
```

pdfnorm_print_to_unit ↑

```
  recursive subroutine pdfnorm_print_to_unit(this,unit,parents,components,peers)
    class(pdfnorm_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    write(unit,'("Components of pdfnorm_type:")')
    write(unit,'("qmin:    ",F7.6)')this%qmin
    write(unit,'("qmax:    ",F7.6)')this%qmax
    write(unit,'("dq:      ",F7.6)')this%dq
    if(components>0)then
       write(unit,'("pdf_int:  ",13(F8.6," "))')this%pdf_int
       write(unit,'("pdf_norm: ",5(F8.6," "))')this%pdf_norm
    else
       write(unit,'("Skipping pdf_int")')
       write(unit,'("Skipping pdf_norm")')
    end if
  end subroutine pdfnorm_print_to_unit
```

pdfnorm_get_type ↑

```
  pure subroutine pdfnorm_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="pdfnorm_type")
  end subroutine pdfnorm_get_type
```

pdfnorm_verify_type ↑

```
  elemental logical function pdfnorm_verify_type(type) result(match)
    character(*),intent(in)::type
    match=type=="pdfnorm_type"
  end function pdfnorm_verify_type
```

pdfnorm_scan ↑

Für verschiedene Werte μ_j von μ_F integrieren wir über x und bekommen so eine Approximation von $\langle f_k(\mu)\rangle = \int_{x=x_0}^1 x f_k(x,\mu)$. In der Komponente this%pdf_int[k,j] werden die die Werte $\langle f_k(\mu_j)\rangle$ gespeichert, mit $k \in [-6:6] = [\bar{t}^S, \bar{b}^S, \bar{c}^S, \bar{s}^S, \bar{u}^S, \bar{d}^S, g, d^V, u^V, s^S, c^S, b^S, t^S]$. In der Komponente this%pdf_norm[m,j] werden normierte Summen dieser Integrale gspeichert.

$$this\%pdf\_norm[m,j] = \frac{\sum_{k\in I_m}\langle f_k(\mu_j)\rangle}{N} \tag{5.3}$$

mit

$$N = \sum_{\text{alle } I_m} \langle f_k(\mu_j) \rangle \tag{5.4}$$

und mit $I =$ [gluon,see,valenz-down,valenz-up], genauer:

$$I_1 = g \tag{5.5}$$
$$I_2 = \bar{t}^S, \bar{b}^S, \bar{c}^S, \bar{s}^S, \bar{u}^S, \bar{d}^S, d^S, u^S, s^S, c^S, b^S, t^S \tag{5.6}$$
$$I_3 = d^V \tag{5.7}$$
$$I_4 = u^V \tag{5.8}$$

Schließlich wird N in this%pdf_norm[0,j] gespeichert. Idealerweise sollte $N = 1$ sein, das wird auch bei der Berechnung der Wichtungsfaktoren der Remnant-PDFs in proton_remnant_type%calculate_weight explizit angenommen. Da diese Summenregel durch nummerische Fehler aber nicht exakt erfüllt ist, normieren wir die Beiträge auf deren Summe. Dadurch bekommen wir in proton_remnant_type%calculate_weight Wichtungsfaktorn, die kaum von dem nummerischen Fehler der Gleichung $N = 1$ abhängen.

Die Integration über x wird mit der Trapezregel durchgeführt. LHAPDF liefert bereits die momentum-pdfs xf, deswegen tritt der Faktor x hier nicht mehr auf. Die x-Werte sind wie auch die μ-Werte nicht äquidistant, sondern deren Abstand ist $\sim \sqrt{x}$ bzw. $\sim \sqrt{\mu}$. Da $x > 0$ und $\mu > 0$ treten keine Koordinaten-divergenzen auf.

```
subroutine pdfnorm_scan(this)
  class(pdfnorm_type),intent(out)::this
  integer::ix,iq
  real(kind=double)::xmin,xmax,dx
  real(kind=double)::q,q2min,q2max
  real(kind=double),dimension(-6:6)::f
  real(kind=double),dimension(0:2)::x
  call getxmin(0,xmin)
  call getxmax(0,xmax)
  call getq2min(0,q2min)
  call getq2max(0,q2max)
  this%qmin=sqrt(sqrt(q2min))
  this%qmax=sqrt(sqrt(q2max))
  this%dq=(this%qmax-this%qmin)/nq
  xmin=sqrt(xmin)
  xmax=sqrt(xmax)
  dx=(xmax-xmin)/nx
  do iq=0,nq
     print *,"iq=",iq,"/",nq
     q=(this%qmin+iq*this%dq)**2
     x(0)=xmin**2
     x(1)=(xmin+dx)**2
     call evolvePDF(x(0),q,f)
     !Valenzbeiträge
     f(1)=f(1)-f(-1)
     f(2)=f(2)-f(-2)
     !Trapezregel: linker Rand
     this%pdf_int(:,iq)=(x(1)-x(0))*f
     do ix=2,nx
        x(2)=(xmin+ix*dx)**2
```

```
        call evolvePDF(x(1),q,f)
        f(1)=f(1)-f(-1)
        f(2)=f(2)-f(-2)
        !Trapezregel: Die bekannte Form ergibt sich aus einer Umsummierung dieser Beiträge.
        this%pdf_int(:,iq)=this%pdf_int(:,iq)+f*(x(2)-x(0))
        !Die x-Werte werden nach links geschoben
        x(0)=x(1)
        x(1)=x(2)
      end do
      !Trapezregel: rechter Rand
      call evolvePDF(x(1),q,f)
      f(1)=f(1)-f(-1)
      f(2)=f(2)-f(-2)
      !Hier wird endlich durch 2 dividiert.
      this%pdf_int(:,iq)=(this%pdf_int(:,iq)+f*(x(1)-x(0)))/2D0
      !⟨f_u^v⟩
      this%pdf_norm(4,iq)=this%pdf_int(2,iq)
      !⟨f_d^v⟩
      this%pdf_norm(3,iq)=this%pdf_int(1,iq)
      !⟨f_u⟩
      this%pdf_int(2,iq)=this%pdf_int(2,iq)+this%pdf_int(-2,iq)
      !⟨f_d⟩
      this%pdf_int(1,iq)=this%pdf_int(1,iq)+this%pdf_int(-1,iq)
      !⟨f_g⟩
      this%pdf_norm(1,iq)=this%pdf_int(0,iq)
      !∑⟨f_q^s⟩
      this%pdf_norm(2,iq)=sum(this%pdf_int(-6:-1,iq))+sum(this%pdf_int(-2:-1,iq))+sum(this%pdf_int
      !∑_{alle Partonen}⟨f_k⟩
      this%pdf_norm(0,iq)=sum(this%pdf_int(:,iq))
      !Normierung auf pdf_norm(0,iq)
      this%pdf_norm(1,iq)=this%pdf_norm(1,iq)/this%pdf_norm(0,iq)
      this%pdf_norm(2,iq)=this%pdf_norm(2,iq)/this%pdf_norm(0,iq)
      this%pdf_norm(3,iq)=this%pdf_norm(3,iq)/this%pdf_norm(0,iq)
      this%pdf_norm(4,iq)=this%pdf_norm(4,iq)/this%pdf_norm(0,iq)
    end do
  end subroutine pdfnorm_scan
```

pdfnorm_get_norm ↑

Hier habe ich verschiedene Polynome zur Approximation der μ_F-Abhängigkeit probiert. Dim ist die Ordnung des Polynoms. Wie auch bei der x-Integration hat die Trapezregel, also dim=1, die besten Resultate gebracht.

```
  subroutine pdfnorm_get_norm(this,gev_q,dim,kind,norm)
    class(pdfnorm_type),intent(in)::this
    real(kind=double),intent(in)::gev_q
    integer,intent(in)::dim,kind
    real(kind=double),intent(out)::norm
    integer::iq
    real(kind=double)::x,q,z0,z1,z2,z3,z4
    norm=-1D0
    q=sqrt(gev_q)-this%qmin
```

```
  iq=floor(q/this%dq)
  x=q/this%dq-iq
  if(iq<0)then
     print *,"pdfnorm_getnorm: q < q_min ",gev_q,this%qmin**2
     norm=this%pdf_norm(kind,0)
  else
     if(iq>=nq)then
        print *,"pdfnorm_getnorm: q >= q_max ",gev_q,this%qmax**2
        norm=this%pdf_norm(kind,nq)
     else
        select case(dim)
        case(0)
           norm=this%pdf_norm(kind,iq)
        case(1)
           norm=this%pdf_norm(kind,iq)*(1D0-x)+this%pdf_norm(kind,iq+1)*x
        case(2)
           x=x+mod(iq,2)
           iq=iq-mod(iq,2)
           z0=this%pdf_norm(kind,iq)
           z1=this%pdf_norm(kind,iq+1)
           z2=this%pdf_norm(kind,iq+2)
           norm=((z0-2D0*z1+z2)*x-(3D0*z0-4D0*z1+z2))*x/2D0+z0
        case(3)
           x=x+mod(iq,3)
           iq=iq-mod(iq,3)
           z0=this%pdf_norm(kind,iq)
           z1=this%pdf_norm(kind,iq+1)
           z2=this%pdf_norm(kind,iq+2)
           z3=this%pdf_norm(kind,iq+3)
           norm=((-(z0-3*z1+3*z2-z3)*x+3*(2*z0-5*z1+4*z2-z3))*x-(11*z0-18*z1+9*z2-2*z3))*x/
        case(4)
           x=x+mod(iq,4)
           iq=iq-mod(iq,4)
           z0=this%pdf_norm(kind,iq)
           z1=this%pdf_norm(kind,iq+1)
           z2=this%pdf_norm(kind,iq+2)
           z3=this%pdf_norm(kind,iq+3)
           z4=this%pdf_norm(kind,iq+4)
           norm=(((((z0-4*z1+6*z2-4*z3+z4)*x&
                -2*(5*z0-18*z1+24*z2-14*z3+3*z4))*x&
                +(35*z0-104*z1+114*z2-56*z3+11*z4))*x&
                -2*(25*z0-48*z1+36*z2-16*z3+3*z4))*x)/24D0&
                +z0
        case default
           norm=this%pdf_norm(kind,iq)*(1D0-x)+this%pdf_norm(kind,iq+1)*x
        end select
        !           print *,iq,x,norm
     end if
  end if
end subroutine pdfnorm_get_norm
```

5.5.2 Methoden für parton_type

parton_write_to_marker ↑

```
subroutine parton_write_to_marker(this,marker,status)
  class(parton_type),intent(in)::this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  call marker%mark_begin("parton_type")
  call marker%mark("id",this%id)
  call marker%mark("lha",this%lha_flavor)
  call marker%mark("momentum",this%momentum)
  call marker%mark_end("parton_type")
end subroutine parton_write_to_marker
```

parton_read_from_marker ↑

```
subroutine parton_read_from_marker(this,marker,status)
  class(parton_type),intent(out)::this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  character(:),allocatable::name
  call marker%pick_begin("parton_type",status=status)
  call marker%pick("id",this%id,status)
  call marker%pick("lha",this%lha_flavor,status)
  call marker%pick("momentum",this%momentum,status)
  call marker%pick_end("parton_type",status=status)
end subroutine parton_read_from_marker
```

parton_print_to_unit ↑

```
recursive subroutine parton_print_to_unit(this,unit,parents,components,peers)
  class(parton_type),intent(in)::this
  integer,intent(in)::unit
  integer(kind=dik),intent(in)::parents,components,peers
  class(serializable_class),pointer::ser
  write(unit,'("Components of parton_type:")')
  write(unit,'("id:         ",I7)')this%id
  write(unit,'("lha flavor: ",I7)')this%lha_flavor
  write(unit,'("momentum:   ",F7.6)')this%momentum
  ser=>this%next
  call serialize_print_peer_pointer(ser,unit,parents,components,peers-one,"next")
  ser=>this%twin
  call serialize_print_comp_pointer(ser,unit,parents,components,peers-one,"twin")
end subroutine parton_print_to_unit
```

parton_get_type ↑

```
pure subroutine parton_get_type(type)
  character(:),allocatable,intent(out)::type
  allocate(type,source="parton_type")
end subroutine parton_get_type
```

twin_unweighted_pdf ↑

```
pure function twin_unweighted_pdf(this,momentum_fraction) result(pdf)
  !parton pdf
  class(parton_type),intent(in)::this
  real(kind=double),intent(in)::momentum_fraction
  real(kind=double)::pdf
  if(momentum_fraction+this%twin%momentum<1D0)then
     pdf=remnant_twin_pdf_p(momentum_fraction,this%twin%momentum,gluon_exp)
  else
     pdf=0D0
  end if
end function twin_unweighted_pdf
```

twin_deallocate ↑

```
recursive subroutine twin_deallocate(this)
  class(parton_type)::this
  if(associated(this%next))then
     call this%next%deallocate
     deallocate(this%next)
  end if
end subroutine twin_deallocate
```

parton_push ↑

```
subroutine parton_push(this,parton)
  class(parton_type),intent(inout)::this
  class(parton_type),intent(inout),pointer::parton
  parton%next=>this%next
  this%next=>parton
end subroutine parton_push
```

parton_pop_by_id ↑

```
subroutine parton_pop_by_id(this,id,parton)
  class(parton_type),target,intent(inout)::this
  integer,intent(in)::id
  class(parton_type),intent(out),pointer::parton
  class(parton_type),pointer::tmp_parton
  tmp_parton=>this
  do while(associated(tmp_parton%next))
     if(tmp_parton%next%id==id)exit
     tmp_parton=>tmp_parton%next
  end do
  !Noch wissen wir nicht, ob die Schleife erfolglos durchgelaufen ist.
  if(associated(tmp_parton%next))then
     !Erfolg: Das Parton wird aus der Liste entfernt.
     parton=>tmp_parton%next
     tmp_parton%next=>parton%next
     nullify(parton%next)
  else
     !Kein Erfolg: Das Dummyargument wird deassociiert.
```

```
      nullify(parton)
      print *,"parton_pop ",id,"NULL"
    end if
  end subroutine parton_pop_by_id
```

parton_pop_by_association ↑

```
  subroutine parton_pop_by_association(this,parton)
    class(parton_type),target,intent(inout)::this
    class(parton_type),intent(inout),target::parton
    class(parton_type),pointer::tmp_parton
    tmp_parton=>this
    do while(associated(tmp_parton%next))
      if(associated(tmp_parton%next,parton))exit
      tmp_parton=>tmp_parton%next
    end do
    !Noch wissen wir nicht, ob die Schleife erfolglos durchgelaufen ist.
    if(associated(tmp_parton%next))then
      !Erfolg: Das Parton wird aus der Liste entfernt.
      tmp_parton%next=>parton%next
      nullify(parton%next)
    else
      !Kein Erfolg
      print *,"parton_pop NULL"
    end if
  end subroutine parton_pop_by_association
```

5.5.3 Methoden für proton_remnant_type

! manipulating parton content

proton_remnant_remove_valence_quark ↑

```
  subroutine proton_remnant_remove_valence_quark(this,id,GeV_scale,momentum_fraction,lha_flavor)
    class(proton_remnant_type),intent(inout)::this
    integer,intent(in)::id
    real(kind=double),intent(in)::GeV_scale,momentum_fraction
    integer,intent(in)::lha_flavor  !d=1 u=2
    if(lha_flavor==1.or.lha_flavor==2)then
      !q ist die Anzahl der entsprechenden Valenzquarks.
      associate(q=>this%valence_content(lha_flavor))
        if(q>0)then
          q=q-1
          !Das Quark ist ab jetzt ein aktiven Shower-Teilchen.
          call this%push_is_parton(id,lha_flavor,momentum_fraction)
          !Der Remnant-Impuls wird um den Partonimpuls reduziert.
          this%momentum_fraction=this%momentum_fraction*(1D0-momentum_fraction)
          !Die Wichtugsfaktoren werden neu ausgewertet
          call this%calculate_weight(GeV_scale)
        else
          print('("proton_remnant_remove_valence_quark: Cannot remove parton ",I2,": &
```

```
                &There are no such partons left.")'),lha_flavor
           call this%print_all()
        end if
      end associate
    else
      print('("proton_remnant_remove_valence_quark: Cannot remove parton ",I2,": &
           &There are no such valence partons.")'),lha_flavor
    end if
  end subroutine proton_remnant_remove_valence_quark
```

proton_remnant_remove_valence_up_quark ↑

q ist die Zahl der Valenz-up-Quarks im Remnant. Wenn keine mehr da sind, dann ist etwas schief gelaufen, sonst wird die Zahl um eins reduziert. Das Parton, das aus dem Remnant entfernt wird, verschwindet natürlich nicht, sondern wird mit proton_remnant_type%push_is_parton in die perturbative Beschreibung aufgenommen. is_parton bedeutet Initial-State Parton, genauer gesagt aktives Initial-State Parton.

Selbstverständlich wird der Impuls des Remants aktualisiert und schließlich werden mit proton_remnant_type%calculate_weight die neuen Wichtungsfaktoren W_α bestimmt.

```
  subroutine proton_remnant_remove_valence_up_quark(this,id,GeV_scale,momentum_fraction)
    class(proton_remnant_type),intent(inout)::this
    integer,intent(in)::id
    real(kind=double),intent(in)::GeV_scale,momentum_fraction
    associate(q=>this%valence_content(lha_flavor_u))
      if(q>0)then
         q=q-1
         call this%push_is_parton(id,lha_flavor_u,momentum_fraction)
         this%momentum_fraction=this%momentum_fraction*(1D0-momentum_fraction)
         call this%calculate_weight(GeV_scale)
      else
         print('("proton_remnant_remove_valence_up_quark: Cannot remove parton ",I2,": &
              &There are no such partons left.")'),lha_flavor_u
         call this%print_all
      end if
    end associate
  end subroutine proton_remnant_remove_valence_up_quark
```

proton_remnant_remove_valence_down_quark ↑

Siehe proton_remnant_type%remove_valence_up_quark

```
  subroutine proton_remnant_remove_valence_down_quark(this,id,GeV_scale,momentum_fraction)
    class(proton_remnant_type),intent(inout)::this
    integer,intent(in)::id
    real(kind=double),intent(in)::GeV_scale,momentum_fraction
    associate(q=>this%valence_content(lha_flavor_d))
      if(q>0)then
         q=q-1
         call this%push_is_parton(id,lha_flavor_d,momentum_fraction)
         this%momentum_fraction=this%momentum_fraction*(1D0-momentum_fraction)
         call this%calculate_weight(GeV_scale)
      else
```

```
        print('("proton_remnant_remove_valence_down_quark: Cannot remove&
             & parton ",I2,": There are no such partons left.")')&
             &,lha_flavor_d
        call this%print_all
      end if
    end associate
  end subroutine proton_remnant_remove_valence_down_quark
```

proton_remnant_remove_sea_quark ↑

Es wird ein Seequark aus dem Remnant genommen und ein Quasivalenzquark hinzugefügt. Wir merken uns, welches Quasivalenzquark zu welchem Seequark gehört und nennen das jeweils andere twin. Mit proton_remnant_type%push_twin erzeugen wir sowohl das neue aktive ISR Seequark als auch das neue Quasivalenzquark.

```
  subroutine proton_remnant_remove_sea_quark(this,id,GeV_scale,momentum_fraction&
       &,lha_flavor)
    integer,intent(in)::id
    class(proton_remnant_type),intent(inout)::this
    real(kind=double),intent(in)::GeV_scale,momentum_fraction
    integer,intent(in)::lha_flavor
    if(lha_flavor>-6.and.lha_flavor<6.and.(lha_flavor.ne.0))then
       this%momentum_fraction=this%momentum_fraction*(1D0-momentum_fraction)
       call this%push_twin(id,lha_flavor,momentum_fraction,GeV_scale)
    end if
  end subroutine proton_remnant_remove_sea_quark
```

proton_remnant_remove_gluon ↑

Hier ist am wenigsten zu tun. Es wird nur der Remnant-Impuls aktualisiert und mit proton_remnant_type%push_is_parton ein neues aktives ISR-Gluon erzeugt.

```
  subroutine proton_remnant_remove_gluon(this,id,GeV_scale,momentum_fraction)
    class(proton_remnant_type),intent(inout)::this
    integer,intent(in)::id
    real(kind=double),intent(in)::GeV_scale,momentum_fraction
    this%momentum_fraction=this%momentum_fraction*(1D0-momentum_fraction)
    call this%push_is_parton(id,lha_flavor_g,momentum_fraction)
  end subroutine proton_remnant_remove_gluon
```

proton_remnant_remove_twin ↑

Ein Quasivalenzquark wird aus dem Remnant genommen und in die Liste der aktiven Showerteilchen aufgenommen.

```
  subroutine proton_remnant_remove_twin(this,id,GeV_scale)
    class(proton_remnant_type),intent(inout)::this
    integer,intent(in)::id
    real(kind=double),intent(in)::GeV_scale
    class(parton_type),pointer::twin
    call this%twin_partons%pop(id,twin)
    call this%fs_partons%push(twin)
    this%twin_norm=this%twin_norm-twin%momentum
    this%n_twins=this%n_twins-1
```

```
    call this%calculate_weight(GeV_scale)
  end subroutine proton_remnant_remove_twin
```

! getting pdf

proton_remnant_parton_twin_pdf ↑

Die Parton-PDFs aller Quasivalenzbeiträge zu dem angegebenen Flavor werden aufaddiert.

```
  subroutine proton_remnant_parton_twin_pdf(this,lha_flavor,momentum_fraction,pdf)
    class(proton_remnant_type),intent(in)::this
    integer,intent(in)::lha_flavor
    real(kind=double),intent(in)::momentum_fraction
    real(kind=double)::pdf
    class(parton_type),pointer::tmp_twin
    pdf=0D0
    tmp_twin=>this%twin_partons%next
    do while(associated(tmp_twin))
       if(tmp_twin%lha_flavor==lha_flavor)pdf=pdf+tmp_twin%unweighted_pdf(momentum_fraction)
       tmp_twin=>tmp_twin%next
    end do
    pdf=pdf*this%get_twin_weight()
  end subroutine proton_remnant_parton_twin_pdf
```

proton_remnant_parton_twin_pdf_array ↑

Aus der Liste proton_remnant_type%twin_partons wird in ein array von Parton-PDFs erzeugt. Jeder Eintrag in dem Dummy-Argument pdf entspricht einem Quasivalenzquark.

```
  subroutine proton_remnant_parton_twin_pdf_array(this,momentum_fraction,pdf)
    class(proton_remnant_type),intent(in)::this
    real(kind=double),intent(in)::momentum_fraction
    real(kind=double),dimension(this%n_twins),intent(out)::pdf
    class(parton_type),pointer::tmp_twin
    integer::l
    tmp_twin=>this%twin_partons%next
    l=0
    do while(associated(tmp_twin))
       l=l+1
       pdf(l)=tmp_twin%unweighted_pdf(momentum_fraction)*this%twin_norm
       tmp_twin=>tmp_twin%next
    end do
  end subroutine proton_remnant_parton_twin_pdf_array
```

proton_remnant_momentum_twin_pdf ↑

Die Momentum-PDFs aller Quasivalenzbeiträge zu dem angegebenen Flavor werden aufaddiert.

```
  subroutine proton_remnant_momentum_twin_pdf(this,lha_flavor,momentum_fraction,pdf)
    class(proton_remnant_type),intent(in)::this
    integer,intent(in)::lha_flavor
    real(kind=double),intent(in)::momentum_fraction
    real(kind=double),intent(out)::pdf
    call this%parton_twin_pdf(lha_flavor,momentum_fraction,pdf)
    pdf=pdf*momentum_fraction
  end subroutine proton_remnant_momentum_twin_pdf
```

proton_remnant_momentum_twin_pdf_array ↑

Aus der Liste proton_remnant_type%twin_partons wird in ein array von Momentum-PDFs erzeugt. Jeder Eintrag in dem Dummy-Argument pdf entspricht einem Quasivalenzquark.

```
subroutine proton_remnant_momentum_twin_pdf_array(this,momentum_fraction,pdf)
  class(proton_remnant_type),intent(in)::this
  real(kind=double),intent(in)::momentum_fraction
  real(kind=double),dimension(this%n_twins),intent(out)::pdf
  call this%parton_twin_pdf_array(momentum_fraction,pdf)
  pdf=pdf*momentum_fraction
end subroutine proton_remnant_momentum_twin_pdf_array
```

proton_remnant_momentum_kind_pdf ↑

Zu dem angegebenen Flavor wird die Momentum-Strukturfunktion nach See- (einschließlich Gluon-), Valenz- und Quasivalenzbeitrag aufgeschlüsselt.

```
subroutine proton_remnant_momentum_kind_pdf(this,GeV_scale,momentum_fraction&
     &,lha_flavor,valence_pdf,sea_pdf,twin_pdf)
  class(proton_remnant_type),intent(in)::this
  real(kind=double),intent(in)::GeV_scale,momentum_fraction
  integer,intent(in)::lha_flavor                     !g,u,d,etc.
  real(kind=double),intent(out)::valence_pdf,sea_pdf,twin_pdf
  real(kind=double),dimension(-6:6)::pdf_array
  call evolvePDF(momentum_fraction,GeV_scale,pdf_array)
  select case (lha_flavor)
  case(0) !gluon
     valence_pdf=0D0
     sea_pdf=pdf_array(0)
  case(1) !down
     valence_pdf=this%get_valence_down_weight()*(pdf_array(1)-pdf_array(-1))
     sea_pdf=pdf_array(-1)
  case(2) !up
     valence_pdf=this%get_valence_up_weight()*(pdf_array(2)-pdf_array(-2))
     sea_pdf=pdf_array(-2)
  case default
     valence_pdf=0D0
     sea_pdf=pdf_array(lha_flavor)
  end select
  sea_pdf=sea_pdf*this%get_sea_weight()
  call this%momentum_twin_pdf(lha_flavor,momentum_fraction,twin_pdf)
end subroutine proton_remnant_momentum_kind_pdf
```

proton_remnant_momentum_flavor_pdf ↑

Zu dem angegebenen Flavor wird die Momentum-Strukturfunktion zurückgegeben. (Summe über alle Beiträge mit diesem Flavor.)

```
subroutine proton_remnant_momentum_flavor_pdf(this,GeV_scale,momentum_fraction&
     &,lha_flavor,pdf)
  class(proton_remnant_type),intent(in)::this
  real(kind=double),intent(in)::GeV_scale,momentum_fraction
  integer,intent(in)::lha_flavor     !g,u,d,etc.
  real(kind=double),intent(out)::pdf
```

```
    real(kind=double)::valence_pdf,sea_pdf,twin_pdf
    call proton_remnant_momentum_kind_pdf(this,GeV_scale,momentum_fraction,lha_flavor&
         &,valence_pdf,sea_pdf,twin_pdf)
    pdf=valence_pdf+sea_pdf+twin_pdf
  end subroutine proton_remnant_momentum_flavor_pdf
```

proton_remnant_momentum_flavor_pdf_array ↑

Es wird ein array von Momentum-PDFs (Summe über alle Beiträge für jedes Flavor), aufgeschlüsselt nach Partonflavor zurückgegeben.

Es sind (noch) keine Quasivalenzquarks enthalten. Das ist aber nur eine Fleißübung, es gibt keinen technischen Hinderungsgrund.

```
  subroutine proton_remnant_momentum_flavor_pdf_array(this,GeV_scale,momentum_fraction&
       &,pdf)
    class(proton_remnant_type),intent(in)::this
    real(kind=double),intent(in)::GeV_scale,momentum_fraction
    real(kind=double),dimension(-6:6),intent(out)::pdf
    real(kind=double),dimension(2)::valence_pdf
    call this%momentum_kind_pdf_array(GeV_scale,momentum_fraction,valence_pdf,pdf)
    pdf(1:2)=pdf(1:2)+valence_pdf
    ! no twin yet
  end subroutine proton_remnant_momentum_flavor_pdf_array
```

proton_remnant_momentum_kind_pdf_array ↑

Es werden See- (einschließlich Gluon-) und Valenzbeiträge zur Momentum-PDF als separate arrays ausgegeben.

```
  subroutine proton_remnant_momentum_kind_pdf_array(this,GeV_scale,momentum_fraction&
       &,valence_pdf,sea_pdf)
    class(proton_remnant_type),intent(in)::this
    real(kind=double),intent(in)::GeV_scale,momentum_fraction
    real(kind=double),dimension(2),intent(out)::valence_pdf
    real(kind=double),dimension(-6:6),intent(out)::sea_pdf
    call evolvePDF(momentum_fraction,GeV_scale,sea_pdf)
    valence_pdf(1)=(sea_pdf(1)-sea_pdf(-1))*this%pdf_int_weight(pdf_int_kind_val_down)
    valence_pdf(2)=(sea_pdf(2)-sea_pdf(-2))*this%pdf_int_weight(pdf_int_kind_val_up)
    sea_pdf(1)=sea_pdf(-1)
    sea_pdf(2)=sea_pdf(-2)
    sea_pdf=sea_pdf*this%get_sea_weight()
    ! no twin yet
  end subroutine proton_remnant_momentum_kind_pdf_array
```

proton_remnant_parton_kind_pdf ↑

Zu dem angegebenen Flavor wird die Parton-Strukturfunktion nach See- (einschließlich Gluon-), Valenz- und Quasivalenzbeitrag aufgeschlüsselt.

```
  subroutine proton_remnant_parton_kind_pdf(this,GeV_scale,momentum_fraction&
       &,lha_flavor,valence_pdf,sea_pdf,twin_pdf)
    class(proton_remnant_type),intent(in)::this
    real(kind=double),intent(in)::GeV_scale,momentum_fraction
    integer,intent(in)::lha_flavor                      !g,u,d,etc.
```

```
    real(kind=double),intent(out)::valence_pdf,sea_pdf,twin_pdf
    call this%momentum_kind_pdf(GeV_scale,momentum_fraction,lha_flavor,valence_pdf&
         &,sea_pdf,twin_pdf)
    valence_pdf=valence_pdf/momentum_fraction
    sea_pdf=sea_pdf/momentum_fraction
    twin_pdf=twin_pdf/momentum_fraction
  end subroutine proton_remnant_parton_kind_pdf
```

proton_remnant_parton_flavor_pdf ↑

Zu dem angegebenen Flavor wird die Parton-Strukturfunktion zurückgegeben (Summe über alle Beiträge mit diesem Flavor).

```
  subroutine proton_remnant_parton_flavor_pdf(this,GeV_scale,momentum_fraction&
       &,lha_flavor,pdf)
    class(proton_remnant_type),intent(in)::this
    real(kind=double),intent(in)::GeV_scale,momentum_fraction
    integer,intent(in)::lha_flavor     !g,u,d,etc.
    real(kind=double),intent(out)::pdf
    call this%momentum_flavor_pdf(GeV_scale,momentum_fraction,lha_flavor,pdf)
    pdf=pdf/momentum_fraction
  end subroutine proton_remnant_parton_flavor_pdf
```

proton_remnant_parton_kind_pdf_array ↑

Es wird ein array von Parton-PDFs (Summe über alle Beiträge für jedes Flavor), aufgeschlüsselt nach Partonflavor zurückgegeben.

Es sind (noch) keine Quasivalenzquarks enthalten. Das ist aber nur eine Fleißübung, es gibt keinen technischen Hinderungsgrund.

```
  subroutine proton_remnant_parton_kind_pdf_array(this,GeV_scale,momentum_fraction&
       &,valence_pdf,sea_pdf)
    class(proton_remnant_type),intent(in)::this
    real(kind=double),intent(in)::GeV_scale,momentum_fraction
    real(kind=double),dimension(2),intent(out)::valence_pdf
    real(kind=double),dimension(-6:6),intent(out)::sea_pdf
    call evolvePDF(momentum_fraction,GeV_scale,sea_pdf)
    sea_pdf=sea_pdf/momentum_fraction
    valence_pdf(1)=(sea_pdf(1)-sea_pdf(-1))*this%valence_content(1)
    valence_pdf(2)=(sea_pdf(2)-sea_pdf(-2))*(this%valence_content(2)/2D0)
    sea_pdf(1)=sea_pdf(-1)
    sea_pdf(2)=sea_pdf(-2)
    valence_pdf=valence_pdf*this%get_valence_weight()
    sea_pdf=sea_pdf*this%get_sea_weight()
    ! no twin yet
  end subroutine proton_remnant_parton_kind_pdf_array
```

proton_remnant_parton_flavor_pdf_array ↑

Es werden See- (einschließlich Gluon-) und Valenzbeiträge zur Parton-PDF als separate arrays ausgegeben.

```
  subroutine proton_remnant_parton_flavor_pdf_array(this,GeV_scale,momentum_fraction&
       &,pdf)
    class(proton_remnant_type),intent(in)::this
    real(kind=double),intent(in)::GeV_scale,momentum_fraction
    real(kind=double),dimension(-6:6),intent(out)::pdf
    real(kind=double),dimension(2)::valence_pdf
    real(kind=double),dimension(-6:6)::twin_pdf
    print('("proton_remnant_flavor_pdf_array: Not yet implemented.")')
  end subroutine proton_remnant_parton_flavor_pdf_array
```

! getting components

proton_remnant_get_pdf_int_weight ↑

```
  pure function proton_remnant_get_pdf_int_weight(this) result(weight)
    class(proton_remnant_type),intent(in)::this
    real(kind=double),dimension(5)::weight
    weight=this%pdf_int_weight
  end function proton_remnant_get_pdf_int_weight
```

proton_remnant_get_valence_weight ↑

```
  pure function proton_remnant_get_valence_weight(this) result(weight)
    class(proton_remnant_type),intent(in)::this
    real(kind=double),dimension(2)::weight
    weight=this%pdf_int_weight(3:4)
  end function proton_remnant_get_valence_weight
```

proton_remnant_get_valence_down_weight ↑

```
  elemental function proton_remnant_get_valence_down_weight(this) result(weight)
    class(proton_remnant_type),intent(in)::this
    real(kind=double)::weight
    weight=this%pdf_int_weight(pdf_int_kind_val_down)
  end function proton_remnant_get_valence_down_weight
```

proton_remnant_get_valence_up_weight ↑

```
  elemental function proton_remnant_get_valence_up_weight(this) result(weight)
    class(proton_remnant_type),intent(in)::this
    real(kind=double)::weight
    weight=this%pdf_int_weight(pdf_int_kind_val_up)
  end function proton_remnant_get_valence_up_weight
```

proton_remnant_get_sea_weight ↑

```
  elemental function proton_remnant_get_sea_weight(this) result(weight)
    class(proton_remnant_type),intent(in)::this
    real(kind=double)::weight
    weight=this%pdf_int_weight(pdf_int_kind_sea)
  end function proton_remnant_get_sea_weight
```

proton_remnant_get_gluon_weight ↑

```
elemental function proton_remnant_get_gluon_weight(this) result(weight)
  class(proton_remnant_type),intent(in)::this
  real(kind=double)::weight
  weight=this%pdf_int_weight(pdf_int_kind_gluon)
end function proton_remnant_get_gluon_weight
```

proton_remnant_get_twin_weight ↑

```
elemental function proton_remnant_get_twin_weight(this) result(weight)
  class(proton_remnant_type),intent(in)::this
  real(kind=double)::weight
  weight=this%pdf_int_weight(pdf_int_kind_twin)
end function proton_remnant_get_twin_weight
```

proton_remnant_get_valence_content ↑

```
pure function proton_remnant_get_valence_content(this) result(valence)
  class(proton_remnant_type),intent(in)::this
  integer,dimension(2)::valence
  valence=this%valence_content
end function proton_remnant_get_valence_content
```

proton_remnant_get_momentum_fraction ↑

```
elemental function proton_remnant_get_momentum_fraction(this) result(momentum)
  class(proton_remnant_type),intent(in)::this
  real(kind=double)::momentum
  momentum=this%momentum_fraction
end function proton_remnant_get_momentum_fraction
```

! misc

proton_remnant_deallocate ↑

```
subroutine proton_remnant_deallocate(this)
  class(proton_remnant_type),intent(inout)::this
  call this%is_partons%deallocate
  call this%fs_partons%deallocate
  call this%twin_partons%deallocate
  this%twin_norm=0D0
  this%n_twins=0
end subroutine proton_remnant_deallocate
```

proton_remnant_initialize ↑

```
subroutine proton_remnant_initialize(this,pdf_norm)
  class(proton_remnant_type),intent(out)::this
  class(pdfnorm_type),target,intent(in)::pdf_norm
  this%pdf_norm=>pdf_norm
end subroutine proton_remnant_initialize
```

proton_remnant_finalize ↑

```
subroutine proton_remnant_finalize(this)
  class(proton_remnant_type),intent(inout)::this
  call this%deallocate()
  nullify(this%pdf_norm)
end subroutine proton_remnant_finalize
```

proton_remnant_apply_initial_splitting ↑

Es wird eine WHIZARD-Interaktion auf den Remnant übertragen. Im Falle eines Gluons im Eingangszustand wird einfach die Methode proton_remnant_type%remove_gluon aufgerufen. Im Falle eines Quarks muss noch entschieden werden, ob es sich um ein See- oder ein Valenzquark handelt.

Mit proton_remnant_type%parton_kind_pdf bekommen wir die Strukturfunktion nach See-, Valenz- und Quasivalenzanteil ($f_{q^S}, f_{q^V}, f_{q^Q}$) aufgeschlüsselt. Durch Vergleich des Verhältnisses $\frac{f_{q^V}}{f_{q^V}+f_{q^S}}$ mit der Zufallszahl *rnd* entscheiden wir, ob proton_remnant_type%remove_valence_up_quark bzw. proton_remnant_type%remove_valence_down_quark oder proton_remnant_type%remove_sea_quark aufgerufen wird.

```
subroutine proton_remnant_apply_initial_splitting(this,id,pdg_flavor,x,gev_scale,rnd)
  class(proton_remnant_type),intent(inout)::this
  integer,intent(in)::id,pdg_flavor
  real(kind=double),intent(in)::x,gev_scale,rnd
  real(kind=double)::valence_pdf,sea_pdf,twin_pdf
  select case(pdg_flavor)
  case(pdg_flavor_g)
     call this%remove_gluon(id,gev_scale,x)
  case(pdg_flavor_u)
     call this%parton_kind_pdf(gev_scale,x&
          &,pdg_flavor,valence_pdf,sea_pdf,twin_pdf)
     if(valence_pdf/(valence_pdf+sea_pdf)<rnd)then
        call this%remove_sea_quark(id,gev_scale,x,pdg_flavor)
     else
        call this%remove_valence_up_quark(id,gev_scale,x)
     end if
  case(pdg_flavor_d)
     call this%parton_kind_pdf(gev_scale,x&
          &,pdg_flavor,valence_pdf,sea_pdf,twin_pdf)
     if(valence_pdf/(valence_pdf+sea_pdf)<rnd)then
        call this%remove_sea_quark(id,gev_scale,x,pdg_flavor)
     else
        call this%remove_valence_down_quark(id,gev_scale,x)
     end if
  case default
     call this%remove_sea_quark(id,gev_scale,x,pdg_flavor)
  end select
  this%momentum_fraction=(1D0-x)
end subroutine proton_remnant_apply_initial_splitting
```

proton_remnant_reset ↑

```
subroutine proton_remnant_reset(this)
  class(proton_remnant_type),intent(inout)::this
  call this%deallocate()
```

```
    this%valence_content=[1,2]
    this%pdf_int_weight=[1D0,1D0,1D0,1D0,1D0]
    this%momentum_fraction=1D0
  end subroutine proton_remnant_reset
```

! private

proton_remnant_push_is_parton ↑

Es wird eine neue Instanz vom Typ parton_type allokiert und mit parton_type%push auf den Stapel proton_remnant_type%is_partons der aktiven ISR-Partonen gelegt.

```
  subroutine proton_remnant_push_is_parton(this,id,lha_flavor,momentum_fraction)
    class(proton_remnant_type),intent(inout)::this
    integer,intent(in)::id,lha_flavor
    real(kind=double),intent(in)::momentum_fraction
    class(parton_type),pointer::tmp_parton
    allocate(tmp_parton)
    tmp_parton%id=id
    tmp_parton%lha_flavor=lha_flavor
    tmp_parton%momentum=momentum_fraction
    call this%is_partons%push(tmp_parton)
  end subroutine proton_remnant_push_is_parton
```

proton_remnant_push_twin ↑

Ein Seequark wird aus dem Remnant entfernt, indem ein neues quark auf den Stapel proton_remnant_type%is_partons der aktiven ISR-Partonen gelegt und ein Quasivalenzquark(twin) in den Remnant aufgenommen wird. Die Quasivalenzquarks im Remnant werden durch den Stapel proton_remnant_type%twin_partons dargestelt. Das Quasivalenzquark bekommt eine negative ID, wodurch es als Quasivalenzquark ausgezeichnet wird. Beide bekommen einen Zeiger *twin*, der auf das jeweils andere zeigt.

Die Modulfunktion remnant_twin_momentum_4 liefert das Integral über die ungewichtete momentum-PDF des Quasivalenzquarks zurück. new_twin%momentum ist also der ungewichtete Impulserwartungswert des Quasivalenzquarks, während new_is%momentum der Impulsanteil ξ des Partons ist.

Mit parton_type%push werden die neuen Teichen auf die jeweiligen Stapel gelegt und mit proton_remnant_type%calculate_weight werden die neuen Wichtungsfaktoren ausgewertet.

```
  subroutine proton_remnant_push_twin(this,id,lha_flavor,momentum_fraction,gev_scale)
    class(proton_remnant_type),intent(inout)::this
    integer,intent(in)::id,lha_flavor !of IS parton
    real(kind=double),intent(in)::momentum_fraction !of IS parton
    real(kind=double),intent(in)::GeV_scale
    class(parton_type),pointer::new_is,new_twin
    real(kind=double)::norm
    !print *,"proton_remnant_push_twin",momentum_fraction
    allocate(new_is)
    allocate(new_twin)
    !IS initialization
    new_is%id=id
    new_is%lha_flavor=lha_flavor
```

```
    new_is%momentum=momentum_fraction
    new_is%twin=>new_twin
    !twin initialization
    new_twin%id=-id
    new_twin%lha_flavor=-lha_flavor
    new_twin%momentum=remnant_twin_momentum_4(momentum_fraction)
    new_twin%twin=>new_is
    !remnant update
    this%n_twins=this%n_twins+1
    this%twin_norm=this%twin_norm+new_twin%momentum
    call this%is_partons%push(new_is)
    call this%twin_partons%push(new_twin)
    call this%calculate_weight(GeV_scale)
  end subroutine proton_remnant_push_twin
```

proton_remnant_calculate_twin_norm ↑

Wenn proton_remnant_type%twin_partons Partonen enthält, dann wird die Summe der Impulsmittelwerte aller Partonen aus proton_remnant_type%twin_partons in der Komponente proton_remnant_type%twin_norm abgelegt. Sonst wird proton_remnant_type%twin_norm auf Null gesetzt.

```
  subroutine proton_remnant_calculate_twin_norm(this)
    class(proton_remnant_type),intent(inout)::this
    class(parton_type),pointer::twin
    integer::n
    if(associated(this%twin_partons%next))then
       this%twin_norm=0D0
       twin=>this%twin_partons%next
       do while(associated(twin))
          this%twin_norm=this%twin_norm+twin%momentum
          twin=>twin%next
       end do
    else
       this%twin_norm=0D0
    end if
  end subroutine proton_remnant_calculate_twin_norm
```

proton_remnant_replace_is_parton ↑

Der ISR-Algorithmus hat ein Splitting eines Teilchens mit der id *old_id* generiert, das zuvor aus dem Remnant entfert wurde. Jetzt wird das alte Remnant-Teilchen wieder in den Remnant zurückgelegt und das neue Teilchen mit der id *new_id* aus dem Remnant entfernt (siehe Abbildung **??**). In Abschnitt 5.4.3 meiner Dissertation wird noch einiges zu dieser Prozedur erläutert.

```
  subroutine proton_remnant_replace_is_parton&
    (this,&
    old_id,&
    new_id,&
    pdg_f,&
    x_proton,&
    gev_scale)
    class(proton_remnant_type),intent(inout)::this
    integer,intent(in)::old_id,new_id,pdg_f
```

```
real(kind=double),intent(in)::x_proton,gev_scale
class(parton_type),pointer::old_is_parton
integer::lha_flavor
real(kind=double)::momentum_fraction
momentum_fraction=x_proton/this%momentum_fraction()
!convert pdg flavor numbers to lha flavor numbers
if(pdg_f==pdg_flavor_g)then
   lha_flavor=lha_flavor_g
else
   lha_flavor=pdg_f
end if
!we remove the old initial state parton from initial state stack.
call this%is_partons%pop(old_id,old_is_parton)
!this check has no physical meaning, it's just a check for consistency.
if(associated(old_is_parton))then
   !do we emit a gluon?
   if(lha_flavor==old_is_parton%lha_flavor)then
      !has the old initial state parton been a sea quark?
      if(associated(old_is_parton%twin))then
         !the connection of the old is parton with it's twin was provisional.
         !We remove it now
         call this%twin_partons%pop(old_is_parton%twin)
         call this%fs_partons%push(old_is_parton%twin)
         this%n_twins=this%n_twins-1
         !and generate a new initial state parton - twin pair.
         call this%push_twin(new_id,lha_flavor,momentum_fraction,gev_scale)
      else
         !there is no twin, so we just insert the new initial state parton.
         call this%push_is_parton(new_id,lha_flavor,momentum_fraction)
      end if
   else
      !we emit a quark. is this a g->qqbar splitting?
      if(lha_flavor==lha_flavor_g)then
         !we insert the new initial state gloun.
         call this%push_is_parton(new_id,lha_flavor,momentum_fraction)
         !has the old initial state quark got a twin?
         if(associated(old_is_parton%twin))then
            !we assume that this twin is the second splitting particle. so the
            !twin becomes a final state particle now and must be removed from
            !the is stack.
            call this%remove_twin(-old_id,GeV_scale)
         else
            !the old initial state quark has been a valence quark.
            !what should we do now? is this splitting sensible at all?
            !we don't know but allow these splittings.
            !The most trivial treatment is to restore the former valence quark.
            this%valence_content(old_is_parton%lha_flavor)=&
                 this%valence_content(old_is_parton%lha_flavor)+1
         end if
      else
         !this is a q->qg splitting. the new initial state quark emits the
```

```
          !preceding initial state gluon. yeah, backward evolution is confusing!
          !the new initial state quark is not part of the proton remnant any longer.
          !how do we remove a quark from the remnant? we add a conjugated twin
          !parton and assume, that this twin is created in a not yet resolved
          !g->qqbar splitting.
          call this%push_twin(new_id,lha_flavor,momentum_fraction,gev_scale)
        end if
      end if
      !everything is done. what shall we do with the old initial state parton?
      !we don't need it any more but we store it anyway for future FSR extension.
      call this%fs_partons%push(old_is_parton)
      !the new initial state parton has taken away momentum, so we update the remnant
      !momentum fraction.
      this%momentum_fraction=&
      this%momentum_fraction*(1-momentum_fraction)/(1-old_is_parton%momentum)
    else
      !this indicates a bug.
      print *,"proton_remnant_replace_is_parton: parton #",old_id,&
      " not found on ISR stack."
      if(associated(this%is_partons%next))then
        print *,"actual content of isr stack:"
        call this%is_partons%next%print_peers()
      else
        print *,"isr stack is not associated."
      end if
      STOP
    end if
  end subroutine proton_remnant_replace_is_parton
```

proton_remnant_calculate_weight ↑

Die Wichtungsfaktoren $[W_G, W_S, W_{d^V}, W_{u^V}, W_Q]$ aus (??) werden bestimmt.

```
  subroutine proton_remnant_calculate_weight(this,GeV_scale)
    class(proton_remnant_type),intent(inout)::this
    real(kind=double),intent(in)::GeV_scale
    real(kind=double)::all,gluon,sea,vu,vd,valence,twin,weight
    !Die 1 aus (??)
    call this%pdf_norm%get_norm(GeV_scale,1,0,all)
    !Die Impulsmittelwerte
    call this%pdf_norm%get_norm(GeV_scale,1,pdf_int_kind_gluon,gluon)
    call this%pdf_norm%get_norm(GeV_scale,1,pdf_int_kind_sea,sea)
    call this%pdf_norm%get_norm(GeV_scale,1,pdf_int_kind_val_down,vd)
    call this%pdf_norm%get_norm(GeV_scale,1,pdf_int_kind_val_up,vu)
    !Wir multiplizieren die Valenzbeiträge mit dem Valenz-Inhalt-Faktor.
    valence=&
         vd*this%valence_content(lha_flavor_d)+&
         vu*this%valence_content(lha_flavor_u)/2D0
    !Die Quasivalenzquark-Beiträge werden auf die Summe aller LHAPDF-Mittelwerte normiert.
    !(siehe pdfnorm_type)
    twin=this%twin_norm/all
    !(siehe Tabelle ?? mit w→weight)
```

```
    select case(remnant_weight_model)
    case(0) ! no reweighting
       this%pdf_int_weight=[1D0,1D0,1D0,1D0,1D0]
    case(2) !pythia-like, only sea
       weight=(1D0-valence-twin)&
            &/(sea+gluon)
       this%pdf_int_weight=[weight,weight,1D0,1D0,1D0]
    case(3) !only valence and twin
       weight=(1D0-sea-gluon)&
            &/(valence+twin)
       this%pdf_int_weight=[1D0,1D0,weight,weight,weight]
    case(4) !only sea and twin
       weight=(1D0-valence)&
            &/(sea+gluon+twin)
       this%pdf_int_weight=[1D0,weight,1D0,1D0,weight]
    case default !equal weight
       weight=1D0/(valence+sea+gluon+twin)
       this%pdf_int_weight=[weight,weight,weight,weight,weight]
    end select
    this%pdf_int_weight(pdf_int_kind_val_down)=&
      this%pdf_int_weight(pdf_int_kind_val_down)*this%valence_content(1)
    this%pdf_int_weight(pdf_int_kind_val_up)=&
      this%pdf_int_weight(pdf_int_kind_val_up)*this%valence_content(2)*5D-1
  end subroutine proton_remnant_calculate_weight
```

Überschriebene serializable_class Methoden

proton_remnant_write_to_marker ↑

```
  subroutine proton_remnant_write_to_marker(this,marker,status)
    class(proton_remnant_type),intent(in)::this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    call marker%mark_begin("proton_remnant_type")
    call marker%mark("valence_content",this%valence_content)
    call marker%mark("momentum_fraction",this%momentum_fraction)
    call marker%mark("pdf_int_weight",this%pdf_int_weight)
    call marker%mark_end("proton_remnant_type")
  end subroutine proton_remnant_write_to_marker
```

proton_remnant_read_from_marker ↑

```
  subroutine proton_remnant_read_from_marker(this,marker,status)
    class(proton_remnant_type),intent(out)::this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    character(:),allocatable::name
    call marker%pick_begin("proton_remnant_type",status=status)
    call marker%pick("valence_content",this%valence_content,status)
    call marker%pick("momentum_fraction",this%momentum_fraction,status)
    call marker%pick("pdf_int_weight",this%pdf_int_weight,status)
```

```
    call marker%pick_end("proton_remnant_type",status=status)
  end subroutine proton_remnant_read_from_marker
```

proton_remnant_print_to_unit ↑

```
  subroutine proton_remnant_print_to_unit(this,unit,parents,components,peers)
    class(proton_remnant_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    write(unit,'("Components of proton_remnant_type:")')
    write(unit,'("Valence Content:           ",I1,":",I1)')this&
         &%valence_content
    write(unit,'("N Twins:                   ",I1)')this%n_twins
    write(unit,'("INT weights [g,s,d,u,t]    ",5(F7.3))')this%pdf_int_weight
    write(unit,'("Total Momentum Fraction:   ",F7.3)')this%momentum_fraction
    write(unit,'("Twin Norm:            ",F7.3)')this%twin_norm
  end subroutine proton_remnant_print_to_unit
```

proton_remnant_get_type ↑

```
  pure subroutine proton_remnant_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="proton_remnant_type")
  end subroutine proton_remnant_get_type
```

proton_remnant_gnuplot_momentum_kind_pdf_array ↑

Außerhalb dieser Prozedur müssen zwei Dateien mit formatiertem, sequentiellen Schreibzugriff geöffnet werden und die assoziierten units an momentum_unit und parton_unit übergeben werden. Dann wird $[x_j, \sum_k f_k(x_j,\mu), \{f_k(x_j,\mu)\}], j = 1..100$ nach parton_unit und $[x_j, \sum_k x_j f_k(x_j,\mu), x_j\{f_k(x_j,\mu)\}], j = 1..100$ nach momentum_unit geschrieben. k sind alle einzelnen Beiträge zur Strukturfunktion mit $k = [v^d, v^u, \overline{\{q\}}, g, \{q\}, \{Q\}]$, $\{q\}$ sind alle Flavor und $\{Q\}$ alle Quasivalenzquarks.

```
  subroutine proton_remnant_gnuplot_momentum_kind_pdf_array&
    (this,momentum_unit,parton_unit,GeV_scale)
    class(proton_remnant_type),intent(in)::this
    integer,intent(in)::momentum_unit,parton_unit
    real(kind=double),intent(in)::GeV_scale
    real(kind=double),dimension(2)::valence_pdf
    real(kind=double),dimension(-6:6)::sea_pdf
    real(kind=double),dimension(this%n_twins)::twin_pdf
    integer::x
    real(kind=double)::momentum_fraction
    do x=1,100
       momentum_fraction=x*1D-2
       call this%momentum_kind_pdf_array(GeV_scale,momentum_fraction&
            &,valence_pdf,sea_pdf)
       call this%momentum_twin_pdf_array(momentum_fraction,twin_pdf)
       write(momentum_unit,fmt=*)momentum_fraction,&
            sum(valence_pdf)+sum(sea_pdf)+sum(twin_pdf),&
            valence_pdf,&
            sea_pdf,&
            twin_pdf
```

```
      call this%parton_kind_pdf_array(GeV_scale,momentum_fraction&
           &,valence_pdf,sea_pdf)
      call this%parton_twin_pdf_array(momentum_fraction,twin_pdf)
      write(parton_unit,fmt=*)momentum_fraction,&
           sum(valence_pdf)+sum(sea_pdf)+sum(twin_pdf),&
           valence_pdf,&
           sea_pdf,&
           twin_pdf
   end do
 end subroutine proton_remnant_gnuplot_momentum_kind_pdf_array
```

5.5.4 Methoden für pp_remnant_type

pp_remnant_initialize ↑

Der Hauptzweck dieser Prozedur ist es, die Impulsmittelwerte $\langle f \rangle(\mu) = \int \mathrm{d}x\, x f(x,\mu)$ bereitzustellen. Im Verzeichnis muli_dir wird nach integrierten PDFs, passend zu dem verwendeten LHAPDF-Set, gesucht. Wenn sie existierten, werden sie deserialisiert, sonst werden sie neu generiert und serialisiert.

```
 subroutine pp_remnant_initialize(&
      this,&
      muli_dir,&
      lhapdf_dir,&
      lhapdf_file,&
      lhapdf_member)
   class(pp_remnant_type),intent(out)::this
   character(*),intent(in)::muli_dir,lhapdf_dir,lhapdf_file
   integer,intent(in)::lhapdf_member
   logical::exist
   allocate(this%pdf_norm)
   print *,"looking for previously generated pdf integrals..."
   inquire(file=muli_dir//"/pdf_norm_"//lhapdf_file//".xml",exist=exist)
   if(exist)then
      print *,"found. Starting deserialization..."
      call this%pdf_norm%deserialize(&
           name="pdf_norm_"//lhapdf_file,&
           file=muli_dir//"/pdf_norm_"//lhapdf_file//".xml")
      print *,"done."
   else
      print *,"No integrals found. Starting generation..."
      call this%pdf_norm%scan()
      print *,"done."
      call this%pdf_norm%serialize(&
           name="pdf_norm_"//lhapdf_file,&
           file=muli_dir//"/pdf_norm_"//lhapdf_file//".xml")
   end if
   call this%proton(1)%initialize(this%pdf_norm)
   call this%proton(2)%initialize(this%pdf_norm)
   this%initialized=.true.
 end subroutine pp_remnant_initialize
```

pp_remnant_finalize ↑

Die Impulsmittelwerte in pp_remnant_type%pdf_norm werden deallokiert und Zeiger darauf deassoziiert.

```
subroutine pp_remnant_finalize(this)
  class(pp_remnant_type),intent(inout)::this
  call this%proton(1)%finalize()
  call this%proton(2)%finalize()
  deallocate(this%pdf_norm)
end subroutine pp_remnant_finalize
```

pp_remnant_apply_initial_interaction ↑

Wrapper für proton_remnant_type%apply_initial_splitting

```
subroutine pp_remnant_apply_initial_interaction&
(this,gev_cme,x1,x2,pdg_f1,pdg_f2,n1,n2,gev_scale,rnd1,rnd2)
  class(pp_remnant_type),intent(inout)::this
  real(kind=double),intent(in)::gev_cme,x1,x2,gev_scale,rnd1,rnd2
  integer,intent(in)::pdg_f1,pdg_f2,n1,n2
  if(this%initialized)then
     call this%proton(1)%apply_initial_splitting(n1,pdg_f1,x1,gev_scale,rnd1)
     call this%proton(2)%apply_initial_splitting(n2,pdg_f2,x2,gev_scale,rnd2)
     this%X=(1D0-x1)*(1D0-x2)
     this%gev_initial_cme=gev_cme
  else
     print *,"pp_remnant_apply_initial_interaction:"
     print *,"Not yet initialized, call pp_remnant_initialize first!"
     stop
  end if
end subroutine pp_remnant_apply_initial_interaction
```

pp_remnant_replace_parton ↑

Wrapper für proton_remnant_type%replace_is_parton

```
subroutine pp_remnant_replace_parton(this,proton_id,old_id,new_id,pdg_f,x_proton,gev_scale)
  class(pp_remnant_type),intent(inout)::this
  integer,intent(in)::proton_id,old_id,new_id,pdg_f
  real(kind=double),intent(in)::x_proton,gev_scale
  call this%proton(proton_id)%replace_is_parton(old_id,new_id,pdg_f,x_proton,gev_scale)
end subroutine pp_remnant_replace_parton
```

pp_remnant_momentum_pdf ↑

```
subroutine pp_remnant_momentum_pdf(this,x_proton,gev2_scale,n,pdg_f,pdf)
  class(pp_remnant_type),intent(in)::this
  real(kind=double),intent(in)::x_proton,gev2_scale
  integer,intent(in)::n,pdg_f
  real(kind=double),intent(out)::pdf
  !Von welchem Remnant wollen wir die Momentum PDF haben?
  !Es muss das erste oder das zweite sein.
  if(n==1.or.n==2)then
     !Der Impulsanteil $x$ ist auf den Impuls des ungestörten Protons bezogen,
```

```
      !deswegen darf es nicht zwischen 0 und 1 sein, sondern nur zwischen 0 und $X$.
      if(x_proton<=this%proton(n)%momentum_fraction)then
         !momentum_flavor_pdf erwartet Flavor im PDG-Schema.
         !Das Gluon muss entsprechend konvertiert werden.
         if(pdg_f==pdg_flavor_g)then
            call this%proton(n)%momentum_flavor_pdf(&
                 sqrt(GeV2_scale),&
                 !momentum_flavor_pdf erwartet Impulsanteile, die auf die Remnantimpulse
                 !bezogen sind.
                 x_proton/this%proton(n)%momentum_fraction,&
                 lha_flavor_g,&
                 pdf&
                 )
         else
            call this%proton(n)%momentum_flavor_pdf(&
                 sqrt(GeV2_scale),x_proton/this%proton(n)%momentum_fraction,pdg_f,pdf&
                 )
         end if
         !Durch die Transformation des Arguments müssen auch die Funktionswerte
         !angepasst werden.
         pdf=pdf*this%proton(n)%momentum_fraction
      else
         pdf=0D0
      end if
   else
      print *,"pp_remnant_momentum_pdf: n must be either 1 or 2, but it is ",n
      stop
   end if
 end subroutine pp_remnant_momentum_pdf
```

pp_remnant_parton_pdf ↑

```
 subroutine pp_remnant_parton_pdf(this,x_proton,gev2_scale,n,pdg_f,pdf)
   class(pp_remnant_type),intent(in)::this
   real(kind=double),intent(in)::x_proton,gev2_scale
   integer,intent(in)::n,pdg_f
   real(kind=double),intent(out)::pdf
   if(n==1.or.n==2)then
      if(x_proton<=this%proton(n)%momentum_fraction)then
         if(pdg_f==pdg_flavor_g)then
            call this%proton(n)%parton_flavor_pdf(&
                 sqrt(GeV2_scale),&
                 x_proton*(1D0-this%proton(n)%momentum_fraction),&
                 lha_flavor_g,&
                 pdf&
                 )
         else
            call this%proton(n)%parton_flavor_pdf(&
                 sqrt(GeV2_scale),&
                 x_proton*(1D0-this%proton(n)%momentum_fraction),&
                 pdg_f,&
```

```
                pdf&
                )
         end if
         pdf=pdf/(1D0-this%proton(n)%momentum_fraction)
      else
         pdf=0D0
      end if
   else
      print *,"pp_remnant_parton_pdf: n must be either 1 or 2, but it is ",n
      stop
   end if
 end subroutine pp_remnant_parton_pdf
```

pp_remnant_apply_interaction ↑

Den Remnants wird mitgeteilt, dass eine Wechselwirkung stattgefunden hat. Alle Informationen über diese Wechselwirkung liegen in einer Instanz vom Typ muli_type. Um sie zu erreichen, wird ein Dummyargument der Klasse qcd_2_2_type, die von muli_type erweitert wird, deklariert.

Hier geschieht nichts, außer dass das Stratum $\{\alpha, \beta\}$ explizit als Ganzzahlen-Doublet in *int_k* abgelegt wird und für beide Remnants this%proton(1) und this%proton(2) die entsprechende Methoden aus der Menge {proton_remnant_type%remove_valence_down_quark, proton_remnant_type%remove_valence_up proton_remnant_type%remove_sea_quark, proton_remnant_type%remove_gluon} aufgerufen wird.

```
 subroutine pp_remnant_apply_interaction(this,qcd_2_2)
   class(pp_remnant_type),intent(inout)::this
   class(qcd_2_2_class),intent(in)::qcd_2_2
   integer,dimension(4)::lha_f
   integer,dimension(2)::int_k
   real(kind=double)::gev_pt
   real(kind=double),dimension(2)::mom_f
   integer::n
   mom_f=qcd_2_2%get_remnant_momentum_fractions()
   lha_f=qcd_2_2%get_lha_flavors()
   int_k=qcd_2_2%get_pdf_int_kinds()
   gev_pt=qcd_2_2%get_gev_scale()
   do n=1,2
      select case (int_k(n))
      case(pdf_int_kind_val_down)
         call this%proton(n)%remove_valence_down_quark(&
           qcd_2_2%get_parton_id(n),&
           gev_pt,&
           mom_f(n))
      case(pdf_int_kind_val_up)
         call this%proton(n)%remove_valence_up_quark(&
           qcd_2_2%get_parton_id(n),&
           gev_pt,&
           mom_f(n))
      case(pdf_int_kind_sea)
         call this%proton(n)%remove_sea_quark(&
         qcd_2_2%get_parton_id(n),&
         gev_pt,&
```

```
        mom_f(n),&
        lha_f(n))
      case(pdf_int_kind_gluon)
        call this%proton(n)%remove_gluon(&
        qcd_2_2%get_parton_id(n),&
        gev_pt,&
        mom_f(n))
      end select
    end do
    this%X=this%proton(1)%momentum_fraction*this%proton(2)%momentum_fraction
  end subroutine pp_remnant_apply_interaction
```

pp_remnant_reset ↑

```
  subroutine pp_remnant_reset(this)
    class(pp_remnant_type),intent(inout)::this
    call this%proton(1)%reset()
    call this%proton(2)%reset()
    this%X=1D0
  end subroutine pp_remnant_reset
```

pp_remnant_get_pdf_int_weights ↑

```
  pure function pp_remnant_get_pdf_int_weights(this,pdf_int_kinds) result(weight)
    class(pp_remnant_type),intent(in)::this
    real(kind=double)::weight
    integer,dimension(2),intent(in)::pdf_int_kinds ! pdf_int_kind
    weight=this%proton(1)%pdf_int_weight(pdf_int_kinds(1))&
          *this%proton(2)%pdf_int_weight(pdf_int_kinds(2))
  end function pp_remnant_get_pdf_int_weights
```

pp_remnant_get_pdf_int_weight ↑

```
  elemental function pp_remnant_get_pdf_int_weight(this,kind1,kind2) result(weight)
    class(pp_remnant_type),intent(in)::this
    real(kind=double)::weight
    integer,intent(in)::kind1,kind2 ! pdf_int_kind
    weight=this%proton(1)%pdf_int_weight(kind1)&
          *this%proton(2)%pdf_int_weight(kind2)
  end function pp_remnant_get_pdf_int_weight
```

pp_remnant_set_pdf_weight ↑

```
  subroutine pp_remnant_set_pdf_weight(this,weights)
    class(pp_remnant_type),intent(inout)::this
    real(kind=double),dimension(10),intent(in)::weights
    this%proton(1)%pdf_int_weight=weights(1:5)
    this%proton(2)%pdf_int_weight=weights(6:10)
  end subroutine pp_remnant_set_pdf_weight
```

pp_remnant_get_gev_initial_cme ↑

```
elemental function pp_remnant_get_gev_initial_cme(this) result(cme)
  class(pp_remnant_type),intent(in)::this
  real(kind=double)::cme
  cme=this%gev_initial_cme
end function pp_remnant_get_gev_initial_cme
```

pp_remnant_get_gev_actual_cme ↑

```
elemental function pp_remnant_get_gev_actual_cme(this) result(cme)
  class(pp_remnant_type),intent(in)::this
  real(kind=double)::cme
  cme=this%gev_initial_cme*this%X
end function pp_remnant_get_gev_actual_cme
```

pp_remnant_get_cme_fraction ↑

```
elemental function pp_remnant_get_cme_fraction(this) result(cme)
  class(pp_remnant_type),intent(in)::this
  real(kind=double)::cme
  cme=this%X
end function pp_remnant_get_cme_fraction
```

pp_remnant_get_proton_remnant_momentum_fractions ↑

```
pure function pp_remnant_get_proton_remnant_momentum_fractions(this) result(fractions)
  class(pp_remnant_type),intent(in)::this
  real(kind=double),dimension(2)::fractions
  fractions=[&
    this%proton(1)%get_momentum_fraction(),&
    this%proton(2)%get_momentum_fraction()]
end function pp_remnant_get_proton_remnant_momentum_fractions
```

pp_remnant_get_proton_remnants ↑

```
subroutine pp_remnant_get_proton_remnants(this,proton1,proton2)
  class(pp_remnant_type),target,intent(in)::this
  class(proton_remnant_type),intent(out),pointer::proton1,proton2
  proton1=>this%proton(1)
  proton2=>this%proton(2)
end subroutine pp_remnant_get_proton_remnants
```

pp_remnant_get_remnant_parton_flavor_pdf_arrays ↑

```
subroutine pp_remnant_get_remnant_parton_flavor_pdf_arrays&
  (this,GeV_scale,momentum1,momentum2,pdf1,pdf2)
  class(pp_remnant_type),intent(in)::this
  real(kind=double),intent(in)::GeV_scale,momentum1,momentum2
  real(kind=double),dimension(-6:6),intent(out)::pdf1,pdf2
  call this%proton(1)%parton_flavor_pdf_array(GeV_scale,momentum1,pdf1)
  call this%proton(2)%parton_flavor_pdf_array(GeV_scale,momentum2,pdf2)
end subroutine pp_remnant_get_remnant_parton_flavor_pdf_arrays
```

!overridden procedures

pp_remnant_write_to_marker ↑

```
subroutine pp_remnant_write_to_marker(this,marker,status)
  class(pp_remnant_type),intent(in)::this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  call marker%mark_begin("pp_remnant_type")
  call marker%mark("gev_initial_cme",this%gev_initial_cme)
  call marker%mark("X",this%X)
  call this%proton(1)%write_to_marker(marker,status)
  call this%proton(2)%write_to_marker(marker,status)
  call marker%mark_end("pp_remnant_type")
end subroutine pp_remnant_write_to_marker
```

pp_remnant_read_from_marker ↑

```
subroutine pp_remnant_read_from_marker(this,marker,status)
  class(pp_remnant_type),intent(out)::this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  character(:),allocatable::name
  call marker%pick_begin("pp_remnant_type",status=status)
  call marker%pick("gev_initial_cme",this%gev_initial_cme,status)
  call marker%pick("X",this%X,status)
  call this%proton(1)%read_from_marker(marker,status)
  call this%proton(2)%read_from_marker(marker,status)
  call marker%pick_end("pp_remnant_type",status=status)
end subroutine pp_remnant_read_from_marker
```

pp_remnant_print_to_unit ↑

```
subroutine pp_remnant_print_to_unit(this,unit,parents,components,peers)
  class(pp_remnant_type),intent(in)::this
  integer,intent(in)::unit
  integer(kind=dik),intent(in)::parents,components,peers
  write(unit,'("Components of pp_remnant_type:")')
  write(unit,'("Initial center of mass energy: ",F10.3)')this%gev_initial_cme
  write(unit,'("Actual center of mass energy:  ",F10.3)')this%get_gev_actual_cme()
  write(unit,'("Total Momentum Fraction is:    ",F10.3)')this%X
  if(components>0)then
     write(unit,'("Proton 1:")')
     call this%proton(1)%print_to_unit(unit,parents,components-1,peers)
     write(unit,'("Proton 2:")')
     call this%proton(2)%print_to_unit(unit,parents,components-1,peers)
  end if
end subroutine pp_remnant_print_to_unit
```

pp_remnant_get_type ↑

```
pure subroutine pp_remnant_get_type(type)
  character(:),allocatable,intent(out)::type
```

```
    allocate(type,source="pp_remnant_type")
  end subroutine pp_remnant_get_type
```

5.5.5 Sonstige Prozeduren

remnant_dglap_splitting_gqq

Der DGLAP-Splitting Kernel für ein $g \to q\bar{q}$ Splitting.

```
  pure function remnant_dglap_splitting_gqq(z) result(p)
    real(kind=double)::p
    real(kind=double),intent(in)::z
    p=(z**2+(1-z)**2)/2D0
  end function remnant_dglap_splitting_gqq
```

remnant_gluon_pdf_approx

Die Approximation der Gluon-Momentum-PDF. p ist Parameter der Approximation, üblicherweise wird er auf 4 gesetzt. Die Wahl von p wird in muli_remnant%gluon_exp festgelegt.

```
  pure function remnant_gluon_pdf_approx(x,p) result(g)
    real(kind=double)::g
    integer,intent(in)::p
    real(kind=double),intent(in)::x
    g=((1-x)**p)/x
  end function remnant_gluon_pdf_approx
```

remnant_norm_0

Der reziproke Normierungsfaktor der Quasivalenzverteilung für p=0. xs ist der Impulsanteil des Seequarks.

```
  pure function remnant_norm_0(xs) result(c0)
    real(kind=double)::c0
    real(kind=double),intent(in)::xs
    c0=6*xs/(2-xs*(3-3*xs+2*xs**2))
  end function remnant_norm_0
```

remnant_norm_1

Der reziproke Normierungsfaktor der Quasivalenzverteilung für p=1. xs ist der Impulsanteil des Seequarks.

```
  pure function remnant_norm_1(xs) result(c1)
    real(kind=double)::c1
    real(kind=double),intent(in)::xs
    c1=3*xs/(2-xs**2*(3-xs)+3*xs*log(xs))
  end function remnant_norm_1
```

remnant_norm_4

Der reziproke Normierungsfaktor der Quasivalenzverteilung für p=4. xs ist der Impulsanteil des Seequarks.

```
pure function remnant_norm_4(xs) result(c4)
  real(kind=double)::c4
  real(kind=double),intent(in)::xs
  real(kind=double)::y
  if((1D0-xs)>1D-3)then
     c4=3*xs/&
     (1 + 11*xs + 6*xs*log(xs) + 12*xs**3*log(xs) + 18*xs**2*log(xs)&
        + 9*xs**2 - 19*xs**3 - 2*xs**4)
  else
     y=1D0/(1D0-xs)
     c4=&
          &1130D0/11907D0&
          & -10D0 *y**5&
          & -40D0 *y**4/3D0&
          & -160D0*y**3/63D0&
          & +50D0 *y**2/189D0&
          & -565D0*y   /3969D0&
          & -186170D0*(1D0-xs)/2750517D0
  end if
end function remnant_norm_4
```

remnant_norm

Der reziproke Normierungsfaktor der Quasivalenzverteilung für p. xs ist der Impulsanteil des Seequarks.

```
pure function remnant_norm(xs,p) result(c)
  real(kind=double)::c
  real(kind=double),intent(in)::xs
  integer,intent(in)::p
  select case (p)
  case(0)
     c=remnant_norm_0(xs)
  case(1)
     c=remnant_norm_1(xs)
  case default
     c=remnant_norm_4(xs)
  end select
end function remnant_norm
```

remnant_twin_pdf_p

Der normierte, aber ungewichtete Quasivalenzbeitrag $f_{q^Q}(x,\overline{x})$ mit $xs = \overline{x}$.

```
pure function remnant_twin_pdf_p(x,xs,p) result(qc)
  real(kind=double)::qc
  real(kind=double),intent(in)::x,xs
  integer,intent(in)::p
  qc=remnant_norm(xs,p)*&
    remnant_gluon_pdf_approx(xs+x,p)*&
    remnant_dglap_splitting_gqq(xs/(xs+x))/(xs+x)
end function remnant_twin_pdf_p
```

remnant_twin_momentum_4

Der Impulsmittelwert des normierten, aber ungewichteten Quasivalenzbeitrags $\left\langle f_{q^Q}\right\rangle(\overline{x}) = \int \mathrm{d}x\, x f_{q^Q}(x,\overline{x})$ mit $xs = \overline{x}$.

```
elemental function remnant_twin_momentum_4(xs) result(p)
  real(kind=double)::p
  real(kind=double),intent(in)::xs
  if(xs<0.99D0)then
     p=(-9*(-1+xs)*xs*(1+xs)*(5+xs*(24+xs))+12*xs*(1+2*xs)*(1+2*xs*(5+2*xs))*Log(xs))/&
          (8*(1+2*xs)*((-1+xs)*(1+xs*(10+xs))-6*xs*(1+xs)*Log(xs)))
  else
     p=(1-xs)/6-(5*(-1+xs)**2)/63+(5*(-1+xs)**3)/216
  end if
end function remnant_twin_momentum_4
```

gnuplot_integrated_pdf

Zu Debuggingzwecken können die integrierten PDFs geplottet werden.

```
subroutine gnuplot_integrated_pdf(this,momentum_unit,parton_unit)
  class(proton_remnant_type),intent(in)::this
  integer,intent(in)::momentum_unit,parton_unit
  integer,parameter::x_grid=1000000
  integer,parameter::q_grid=100
  integer::n,m,mem
  real(kind=double)::x,q,dx,dq,overall_sum,xmin,xmax,q2min,q2max,qmin,qmax
  real(kind=double),dimension(-6:6)::sea_pdf,sea_momentum_pdf_sum,sea_parton_pdf_sum
  real(kind=double),dimension(2)::valence_pdf,valence_momentum_pdf_sum,valence_parton_pdf_s
  real(kind=double),allocatable,dimension(:)::twin_momentum_pdf_sum
  class(parton_type),pointer::tmp_twin
  mem=1
  call GetXmin(mem,xmin)
  call GetXmax(mem,xmax)
  call GetQ2max(mem,q2max)
  call GetQ2min(mem,q2min)
  qmin=sqrt(q2min)
  qmax=sqrt(q2max)
  print *,"qmin=",qmin,"GeV"
  print *,"qmax=",qmax,"GeV"
  dx=(xmax-xmin)/x_grid
  dq=(qmax-qmin)/q_grid
  q=qmin+dq/2D0
  tmp_twin=>this%twin_partons%next
  n=0
  if(this%n_twins>0)then
     allocate(twin_momentum_pdf_sum(this%n_twins))
     do while(associated(tmp_twin))
        n=n+1
        twin_momentum_pdf_sum(n)=tmp_twin%momentum
        tmp_twin=>tmp_twin%next
     end do
  end if
```

```
  do m=1,q_grid
     valence_momentum_pdf_sum=[0D0,0D0]
     valence_parton_pdf_sum=[0D0,0D0]
     sea_momentum_pdf_sum=[0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0]
     sea_parton_pdf_sum=[0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0]
     x=xmin+dx/2D0
     do n=1,x_grid
        call this%parton_kind_pdf_array(Q,x,valence_pdf,sea_pdf)
        valence_parton_pdf_sum=valence_parton_pdf_sum+valence_pdf
        sea_parton_pdf_sum=sea_parton_pdf_sum+sea_pdf
        call this%momentum_kind_pdf_array(Q,x,valence_pdf,sea_pdf)
        valence_momentum_pdf_sum=valence_momentum_pdf_sum+valence_pdf
        sea_momentum_pdf_sum=sea_momentum_pdf_sum+sea_pdf
        x=x+dx
     end do
     valence_parton_pdf_sum=valence_parton_pdf_sum*dx
     sea_parton_pdf_sum=sea_parton_pdf_sum*dx
     valence_momentum_pdf_sum=valence_momentum_pdf_sum*dx
     sea_momentum_pdf_sum=sea_momentum_pdf_sum*dx
     if(this%n_twins>0)then
        write(momentum_unit,fmt=*)q,&
             sum(valence_momentum_pdf_sum)&
              +sum(sea_momentum_pdf_sum)&
              +sum(twin_momentum_pdf_sum),&
             valence_momentum_pdf_sum,&
             sea_momentum_pdf_sum,&
             twin_momentum_pdf_sum
     else
        write(momentum_unit,fmt=*)q,&
             sum(valence_momentum_pdf_sum)+sum(sea_momentum_pdf_sum),&
             valence_momentum_pdf_sum,&
             sea_momentum_pdf_sum
     end if
     write(parton_unit,fmt=*)q,&
          sum(valence_parton_pdf_sum)+sum(sea_parton_pdf_sum),&
          valence_parton_pdf_sum,&
          sea_parton_pdf_sum
     q=q+dq
  end do
end subroutine gnuplot_integrated_pdf
```

6 Modul muli_dsigma

Hier wird eine Approximation der Stammstrati $\mathcal{S}(p_\perp)$ aus (??) bereitgestellt. Die Integrationen in (??) werden mit der externen Bibliothek libcuba ausgewertet, die verbleibende Integration in (??) wird mit dem muli-eigenen Modul muli_aq ausgewertet.

Zu Beginn hatte ich mit verschiedenen Darstellungen der Wirkungsquerschnitte und mit verschiedenen Integrationsparametern und verschiedenen Einteilungen in Strati experimentiert. Um Codevervielfältigung zu vermeiden hatte ich dann den Code für die Integration von den Wirkungsquerschnitten getrennt. Da die Wirkungsquerschnitte auf Parameter zugreifen müssen, die nicht fur alle Darstellungen gleich sind, konnte ich die Integraden nicht als Funktion an aq_class übergeben. Stattdessen habe ich mich entschieden, die verschiedenen Varianten durch Überladen der Methode evaluate zu erzeugen. So konnten die Erweiterung von aq_class komplett verschiedene Methoden zur Auswertung von (??), und dennoch dieselbe Quadratur für (??) verwenden. Heute ist nur noch eine einzige Erweiterung übrig, nämlich muli_dsigma_type in diesem Modul. Deswegen ist der Sinn zwischen der Aufteilung der Module muli_aq und muli_dsigma nicht mehr offensichtlich.

6.1 Abhängigkeiten

```
use muli_momentum
use muli_interactions
use muli_cuba
use muli_aq
```

6.2 Parameter

```
!Die Anzahl der Strati plus 1, für die Summe aller Strati.
integer,parameter,private::dim_f=17
```

6.3 Derived Types

6.3.1 muli_dsigma_type

Der Zweck von muli_dsigma_type liegt darin, die abstrakte Methode evaluate von aq_class zu implementieren und so einen Integradem für die nummerische Integration bereitzustellen. Weiterhin stellt aq_class die Methode muli_dsigma_type%generate zur Verfügung, um die Integration zu starten.

Für das setzten der Faktorisierungsskala wird eine eigene Instanz muli_dsigma_type%pt des Datentyps transversal_momentum_type verwendet. Eigen bedeutet, dass muli_dsigma_type%pt nicht mit der muli-Skala synchronisiert ist, denn diese Integration findet vor der Eventgenerierung mit MULI statt.

```
type,public,extends(aq_class) :: muli_dsigma_type
   private
   type(transversal_momentum_type)::ptpt
   type(cuba_divonne_type) :: cuba_intcuba_int
 contains
   !Überschriebene serializable_class Methoden
   procedure::write_to_marker=>muli_dsigma_write_to_marker
   procedure::read_from_marker=>muli_dsigma_read_from_marker
   procedure::print_to_unit=>muli_dsigma_print_to_unit
   procedure,nopass::get_type=>muli_dsigma_get_type
   !Originäre muli_dsigma_type Methoden
   procedure :: generate=>muli_dsigma_generate
   procedure :: evaluate=>muli_dsigma_evaluate
   procedure :: muli_dsigma_initialize
   generic   :: initialize=>muli_dsigma_initialize
end type muli_dsigma_type
```

6.4 Implementierung der Prozeduren

6.4.1 Methoden für muli_dsigma_type

muli_dsigma_write_to_marker ↑

```
subroutine muli_dsigma_write_to_marker(this,marker,status)
  class(muli_dsigma_type), intent(in) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik), intent(out) :: status
  ! local variables
  class(serializable_class),pointer::ser
  call marker%mark_begin("muli_dsigma_type")
  call aq_write_to_marker(this,marker,status)
  call this%cuba_int%serialize(marker,"cuba_int")
  call marker%mark_end("muli_dsigma_type")
end subroutine muli_dsigma_write_to_marker
```

muli_dsigma_read_from_marker ↑

```
subroutine muli_dsigma_read_from_marker(this,marker,status)
  class(muli_dsigma_type), intent(out) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik), intent(out) :: status
  ! local variables
  call marker%pick_begin("muli_dsigma_type",status=status)
  call aq_read_from_marker(this,marker,status)
  call this%cuba_int%deserialize("cuba_int",marker)
  call marker%pick_end("muli_dsigma_type",status)
end subroutine muli_dsigma_read_from_marker
```

muli_dsigma_print_to_unit ↑

```
subroutine muli_dsigma_print_to_unit(this,unit,parents,components,peers)
  class(muli_dsigma_type),intent(in)::this
  integer,intent(in)::unit
  integer(kind=dik),intent(in)::parents,components,peers
  integer::ite
  if(parents>0)call aq_print_to_unit(this,unit,parents-1,components,peers)
  write(unit,'("Components of muli_dsigma_type")')
  if(components>0)then
     write(unit,fmt=*)"Printing components of cuba_int:"
     call this%cuba_int%print_to_unit(unit,parents,components-1,peers)
  else
     write(unit,fmt=*)"Skipping components of cuba_int:"
  end if
end subroutine muli_dsigma_print_to_unit
```

muli_dsigma_get_type ↑

```
pure subroutine muli_dsigma_get_type(type)
  character(:),allocatable,intent(out)::type
  allocate(type,source="muli_dsigma_type")
end subroutine muli_dsigma_get_type
```

muli_dsigma_generate ↑

Initialisierung und Generierung der Stammstrati $\mathcal{S}$.

Man kann eine Start-Segmentierung des Integrationsbereichs angeben. Dadurch kann die Integration erheblich beschleunigt werden. Wir wählen eine Segmentierung in *initial_values* so, dass $\mu_j = \mu_0 \exp(j)$, solange $\mu_j < \sqrt{s}/2$ und nehmen als letzten Wert $\sqrt{s}/2$ hinzu.

```
subroutine muli_dsigma_generate(this,gev2_scale_cutoff,gev2_s,int_tree)
  class(muli_dsigma_type),intent(inout)::this
  real(kind=drk),intent(in)::gev2_scale_cutoff,gev2_s
  type(muli_trapezium_tree_type),intent(out)::int_tree
  real(kind=drk),dimension(ceiling(log(gev2_s/gev2_scale_cutoff)/2D0))::initial_values
  integer::n
  !Debugging
  print *,gev2_s/gev2_scale_cutoff,ceiling(log(gev2_s/gev2_scale_cutoff)/2D0)
  !Setzen der Start-Segmentierung
  initial_values(1)=sqrt(gev2_scale_cutoff/gev2_s)*2D0
  do n=2,size(initial_values)-1
     initial_values(n)=initial_values(n-1)*euler
  end do
  initial_values(n)=1D0
  !Debugging
  print *,initial_values
  !Wir geben dieser Instanz einen Namen und die Nummer 1.
  call identified_initialize(this,one,"dsigma")
  !Die Skala wird initialisiert.
  call this%pt%initialize(gev2_s)
  !Die Genauigkeit der Stammfunktion (??)
  this%abs_error_goal = 0D0
  this%rel_error_goal=scale(1D0,-12)!-12
```

```
    this%max_nodes=1000
    !Dimension und Genauigkeit der Integration (??)
    call this%cuba_int%set_common(&
         &dim_f=dim_f,&
         &dim_x=2,&
         &eps_rel=scale(this%rel_error_goal,-8),&!-8
         &flags = 0)
    !Die ungefähre Position der Maxima des Integranden
    call this%cuba_int%set_deferred&
      (xgiven_flat=[1D-2,5D-1+epsilon(1D0),1D-2,5D-1-epsilon(1D0)])
    print *,"muli_dsigma_generate:"
    print *,"Overall Error Goal: ",this%rel_error_goal
    !Wir initialisieren die Integration mit der Start-Segmentierung
    call this%init_error_tree(dim_f,initial_values)
    !Die eigentliche Integration
    call this%run()
    !Konvertierung der internen Darstellung mittels fibonacci_root_type
    !in ein bessere Darstellung mittels muli_trapezium_tree_type.
    call this%integrate(int_tree)
    !Aufräumen
    call this%err_tree%deallocate_all()
    deallocate(this%err_tree)
    nullify(this%int_list)
  end subroutine muli_dsigma_generate
```

muli_dsigma_evaluate ↑

Die Wahl der Integrationsroutine und der Darstellung der Wirkungsquerschnitte.

```
  subroutine muli_dsigma_evaluate(this,x,y)
    class(muli_dsigma_type),intent(inout) :: this
    real(kind=double), intent(in) :: x
    real(kind=double), intent(out),dimension(:):: y
    call this%pt%set_unit_scale(x)
    call this%cuba_int%integrate_userdata(&
         interactions_proton_proton_integrand_param_17_reg,this%pt)
    call this%cuba_int%get_integral_array(y)
  end subroutine muli_dsigma_evaluate
```

muli_dsigma_initialize ↑

```
    subroutine muli_dsigma_initialize(this,id,name,goal,max_nodes,dim,cuba_goal)
      class(muli_dsigma_type),intent(inout) :: this
      integer(kind=dik),intent(in)::id,max_nodes
      integer,intent(in)::dim
      character(*),intent(in)::name
      real(kind=double),intent(in)::goal,cuba_goal
      call identified_initialize(this,id,name)
      this%rel_error_goal = goal!1d-4
      this%max_nodes=max_nodes
      call this%cuba_int%set_common(&
           &dim_f=dim,&
           &dim_x=2,&
```

```
       &eps_rel=cuba_goal,&!1d-6
       &flags = 0)
  call this%cuba_int%set_deferred&
    (xgiven_flat=[1D-2,5D-1+epsilon(1D0),1D-2,5D-1-epsilon(1D0)])
  call this%init_error_tree(dim,(/8D-1/7D3,2D-3,1D-2,1D-1,1D0/))
  this%is_deferred_initialised = .true.
end subroutine muli_dsigma_initialize
```

7 Modul muli_aq

aq ist eine Abkürzung für adaptive Quadratur. Mit aq_class kann für eine beliebige Funktion $f : \mathbb{R} \to \mathbb{R}^n$ die Quasistammfunktion $F(x) = \int_x^1 f(y)\ \mathrm{d}y$ mittels adaptiver Quadratur ausgewertet und als Binärbaum von Segmenten $s_j = [x_{j-1}, x_j]$ gespeichert werden. Zu jedem Segment werden $f(x_j), F(x_j)$ und $\exp[-F(x_j)]$ gespeichert.

f wird mit der Trapezregel approximiert, entsprechend wird F durch Parabeln approximiert. Dennoch ist die Approximation von F nicht gleich Simpsons Regel, denn wir haben nur zwei Stützstellen x_{j-1} und x_j für jede Parabel.

Der Integrationsfehler *delta* δ' bzw. δ' ergibt sich bei der Spaltung des Segments s_j in zwei Untersegmente $s'_j = [x_{j-1}, y]$ und $s''_j = [y, x_j]$ durch die Differenz des alten und des neuen Integrals:

$$\delta' = \left| \frac{\big(f(y) - f_j(y)\big)\big(x_{j-1} - y\big)}{2} \right| \tag{7.1}$$

$$\delta'' = \left| \frac{\big(f(y) - f_j(y)\big)\big(x_j - y\big)}{2} \right| \tag{7.2}$$

mit der alten Approximation f_j des alten Segments j

$$f_j(y) = f(x_j) - y\ \frac{f(x_j) - f(x_{j-1})}{x_j - x_{j-1}}. \tag{7.3}$$

7.1 Abhängigkeiten

```
use muli_basic
use muli_cuba
use muli_trapezium
use muli_fibonacci_tree
```

7.2 Derived Types

7.2.1 aq_class

```
type,extends(identified_type),abstract :: aq_class
   ! private
   !Erweiterungen müssen durch is_deferred_initialised signalisieren, dass sie bereit sind.
   logical :: is_deferred_initialised = .false.
   !Ist aq_class%err_tree bereit?
   logical :: is_error_tree_initialised = .false.
   !Wurden die internen Fehlerziele bestimmt?
```

```
logical :: is_goal_set = .false.
!Ist alles bereit zur Integration?
logical :: is_initialised = .false.
!Wurde die Integration durchgeführt?
logical :: is_run = .false.
!Wurde das Fehlerziel erreicht?
logical :: is_goal_reached = .false.
!Wurde aq_class%err_tree nach aq_class%int_list konvertiert?
logical :: is_integrated = .false.
!Die aktuelle Anzahl von Segmenten
integer(kind=dik) :: n_nodes = 0
!Die maximale Anzahl von Segmenten
integer(kind=dik) :: max_nodes = 10000
!Die Dimension von f
integer :: dim_integral = 1
!Das gegebene absolute Fehlerziel
real(kind=double) :: abs_error_goal = 0D0
!Das gegebene relative Fehlerziel
real(kind=double) :: rel_error_goal = 0.1D0
!Das berechnete absolute Fehlerziel, basierend auf der aktuellen
!Schätzung des Integrals
real(kind=double) :: scaled_error_goal = 0.0D0
!Schätzung des Integrals F(x_min)
real(kind=double) :: integral = 1D0
!Aktueller absoluter Integrationsfehler
real(kind=double) :: integral_error = 0D0
!Integrationsintervall
real(kind=double),dimension(2) :: region = (/0D0,1D0/)
!Zu Debuggingzwecken wird die Historie des Integrationsfehlers gespeichert.
!Wenn die Historie oszilliert, dann ist der Fehler in f, also in der
!Cuba-Integration zu groß.
real(kind=double),dimension(:,:),allocatable :: convergence
!time stamps um die Performance des Allgorithmus zu überwachen.
real(kind=double) :: total_time = 0
real(kind=double) :: loop_time = 0
real(kind=double) :: int_time = 0
real(kind=double) :: cuba_time = 0
real(kind=double) :: init_time = 0
real(kind=double) :: cpu_time = 0

!These variables *must* be initialised before the main loop may be called.
!Additionaly the nodes and segments should be preprocessed by first_run
!before the main loop may be called.

!Das tatsächliche Fehlerziel.
real(kind=double) :: error_goal = 0D0
!Während der Integration werden die Segmente des Integranden nach ihrem
!Integrationsfehler sortiert in diesem Binärbaum gespeichert.
class(fibonacci_root_type),pointer ::err_tree=>null()
!Nach erfolgreicher Integration werden die Segmente nach dem Skalenparameter
!sortiert in dieser Liste gespeichert.
```

```
    class(muli_trapezium_list_type),pointer ::int_list=>null()
   contains
    !Überschriebene serializable_class Methoden
    procedure::write_to_marker=>aq_write_to_marker
    procedure::read_from_marker=>aq_read_from_marker
    procedure::print_to_unit=>aq_print_to_unit
    procedure,nopass::get_type=>aq_get_type
    procedure::deserialize_from_marker=>aq_deserialize_from_marker
    !Originäre aq_class Methoden
    procedure :: aq_initialize
    generic   ::initialize=>aq_initialize
    procedure ::print_times=>aq_print_times
    procedure ::write_convergence=>aq_write_convergence
    ! init/ de-init
    procedure ::reset=>aq_reset
    procedure ::dealloc_trees=>aq_dealloc_trees
    procedure ::finalize=>aq_dealloc_trees
    procedure ::init_error_tree=>aq_init_error_tree
    procedure ::set_rel_goal=>aq_set_rel_goal
    procedure ::set_abs_goal=>aq_set_abs_goal
    procedure ::set_goal=>aq_set_goal
    procedure ::check_init=>aq_check_init
    ! calculation
    procedure ::main_loop=>aq_main_loop
    procedure ::run=>aq_run
    procedure ::integrate=>aq_integrate
    ! deferred
    procedure(evaluate_if),deferred :: evaluate
  end type aq_class
```

7.3 Interfaces

```
  interface
    subroutine evaluate_if(this,x,y)
      use kinds!NODEP!
      import aq_class
      class(aq_class),intent(inout) :: this
      real(kind=double), intent(in) :: x
      real(kind=double), intent(out) ,dimension(:):: y
    end subroutine evaluate_if
```

7.4 Implementierung der Prozeduren

7.4.1 Methoden für aq_class

Überschriebene serializable_class Methoden

aq_write_to_marker ↑

```
subroutine aq_write_to_marker(this,marker,status)
  class(aq_class), intent(in) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  class(serializable_class),pointer::ser
  call marker%mark_begin("aq_class")
  call identified_write_to_marker(this,marker,status)
  call marker%mark("is_deferred_initialised",this&
       &%is_deferred_initialised)
  call marker%mark("is_error_tree_initialised",this&
       &%is_error_tree_initialised)
  call marker%mark("is_goal_set",this%is_goal_set)
  call marker%mark("is_initialised",this%is_initialised)
  call marker%mark("is_run",this%is_run)
  call marker%mark("is_goal_reached",this%is_goal_reached)
  call marker%mark("is_integrated",this%is_integrated)
  call marker%mark("n_nodes",this%n_nodes)
  call marker%mark("max_nodes",this%max_nodes)
  call marker%mark("dim_integral",this%dim_integral)
  call marker%mark("abs_error_goal",this%abs_error_goal)
  call marker%mark("rel_error_goal",this%rel_error_goal)
  call marker%mark("scaled_error_goal",this%scaled_error_goal)
  call marker%mark("error_goal",this%error_goal)
  call marker%mark("integral",this%integral)
  call marker%mark("integral_error",this%integral_error)
  call marker%mark("region",this%region(1:2))
  ser=>this%err_tree
  call marker%mark_pointer("err_tree",ser)
  ser=>this%int_list
  call marker%mark_pointer("int_list",ser)
  call marker%mark_end("aq_class")
end subroutine aq_write_to_marker
```

aq_read_from_marker ↑

```
subroutine aq_read_from_marker(this,marker,status)
  class(aq_class), intent(out) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  class(serializable_class),pointer::ser
  call marker%pick_begin("aq_class",status=status)
  call identified_read_from_marker(this,marker,status)
  call marker%pick("is_deferred_initialised",this%is_deferred_initialised&
       &,status)
  call marker%pick("is_error_tree_initialised",this&
       &%is_error_tree_initialised,status)
  call marker%pick("is_goal_set",this%is_goal_set,status)
  call marker%pick("is_initialised",this%is_initialised,status)
  call marker%pick("is_run",this%is_run,status)
  call marker%pick("is_goal_reached",this%is_goal_reached,status)
  call marker%pick("is_integrated",this%is_integrated,status)
```

```
    call marker%pick("n_nodes",this%n_nodes,status)
    call marker%pick("max_nodes",this%max_nodes,status)
    call marker%pick("dim_integral",this%dim_integral,status)
    call marker%pick("abs_error_goal",this%abs_error_goal,status)
    call marker%pick("rel_error_goal",this%rel_error_goal,status)
    call marker%pick("scaled_error_goal",this%scaled_error_goal,status)
    call marker%pick("error_goal",this%error_goal,status)
    call marker%pick("integral",this%integral,status)
    call marker%pick("integral_error",this%integral_error,status)
    call marker%pick("region",this%region(1:2),status)
    call marker%pick_pointer("err_tree",ser)
    if(associated(ser))then
       select type(ser)
       class is (fibonacci_root_type)
          this%err_tree=>ser
       class default
          nullify(this%err_tree)
       end select
    end if
    call marker%pick_pointer("int_list",ser)
    if(associated(ser))then
       select type(ser)
       class is (muli_trapezium_list_type)
          this%int_list=>ser
       class default
          nullify(this%int_list)
       end select
    end if
    call marker%pick_end("aq_class",status)
  end subroutine aq_read_from_marker
```

aq_print_to_unit ↑

```
  subroutine aq_print_to_unit(this,unit,parents,components,peers)
    class(aq_class),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    integer::ite
    class(serializable_class),pointer::ser
    if(parents>0)call identified_print_to_unit(this,unit,parents-1,components&
         &,peers)
    write(unit,'("Components of aq_class")')
    write(unit,'(a,L1)')"Deferred class initialised: ",this&
         &%is_deferred_initialised
    write(unit,'(a,L1)')"Error tree initialised:     ",this&
         &%is_error_tree_initialised
    write(unit,'(a,L1)')"Accuracy goal set:          ",this%is_goal_set
    write(unit,'(a,L1)')"Ready for run:              ",this%is_initialised
    write(unit,'(a,L1)')"Is run:                     ",this%is_run
    write(unit,'(a,L1)')"Accuracy goal reached:      ",this%is_goal_reached
    write(unit,'(a,L1)')"Integral calculated:        ",this%is_integrated
```

```
    write(unit,'(a,I10)')"Number of nodes:           ",this%n_nodes
    write(unit,'(a,I10)')"Maximal number of nodes:    ",this%max_nodes
    write(unit,'(a,I10)')"Dimension of integral:      ",this%dim_integral
    write(unit,'(a,E20.10)')"Given abs. error goal: ",this%abs_error_goal
    write(unit,'(a,E20.10)')"Given rel. error goal: ",this%rel_error_goal
    write(unit,'(a,E20.10)')"Guessed abs error goal:",this%scaled_error_goal
    write(unit,'(a,E20.10)')"Actual abs error goal: ",this%error_goal
    write(unit,'(a,E20.10)')"Integral              ",this%integral
    write(unit,'(a,E20.10)')"Estimated abs. error:  ",this%integral_error
    write(unit,'(a,E10.5,a,E10.5,a)')"Integration region =  (",this%region(1)&
         &," : ",this%region(2),")"
    ser=>this%err_tree
    call serialize_print_comp_pointer(ser,unit,parents,components,peers&
         &,"error tree")
    ser=>this%int_list
    call serialize_print_comp_pointer(ser,unit,parents,components,peers&
         &,"integral list")
  end subroutine aq_print_to_unit
```

aq_get_type ↑

```
  pure subroutine aq_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="aq_type")
  end subroutine aq_get_type
```

aq_deserialize_from_marker ↑

```
  subroutine aq_deserialize_from_marker(this,name,marker)
    class(aq_class),intent(out)::this
    character(*),intent(in)::name
    class(marker_type),intent(inout)::marker
    class(serializable_class),pointer::ser
    allocate(muli_trapezium_type::ser)
    call marker%push_reference(ser)
    allocate(fibonacci_root_type::ser)
    call marker%push_reference(ser)
    allocate(fibonacci_leave_type::ser)
    call marker%push_reference(ser)
    allocate(fibonacci_node_type::ser)
    call marker%push_reference(ser)
    call serializable_deserialize_from_marker(this,name,marker)
    call marker%pop_reference(ser)
    deallocate(ser)
    call marker%pop_reference(ser)
    deallocate(ser)
    call marker%pop_reference(ser)
    deallocate(ser)
    call marker%pop_reference(ser)
    deallocate(ser)
  end subroutine aq_deserialize_from_marker
```

Originäre aq_class Methoden

aq_initialize ↑

```
subroutine aq_initialize(this,id,name,goal,max_nodes,dim,init)
  class(aq_class),intent(out) :: this
  integer(kind=dik),intent(in)::id,max_nodes
  integer,intent(in)::dim
  character,intent(in)::name
  real(kind=double)::goal
  real(kind=double),dimension(:),intent(in)::init
  call identified_initialize(this,id,name)
  this%rel_error_goal = goal!1d-4
  this%max_nodes=max_nodes
  call this%init_error_tree(dim,init)
end subroutine aq_initialize
```

aq_print_times ↑

```
subroutine aq_print_times(this)
  class(aq_class),intent(in) :: this
  print '(a,E20.10)',"Initialization time:  ",this%init_time
  print '(a,E20.10)',"Main loop time:       ",this%loop_time
  print '(a,E20.10)',"Integration time:     ",this%int_time
  print '(a,E20.10)',"Overall run time:     ",this%total_time
  print '(a,E20.10)',"Cuba integration time:",this%cuba_time
end subroutine aq_print_times
```

aq_write_convergence ↑

```
subroutine aq_write_convergence(this,unit)
  class(aq_class),intent(in) :: this
  integer,intent(in)::unit
  integer,dimension(2)::s
  integer::node
  if(allocated(this%convergence))then
     s=shape(this%convergence)
     do node=1,s(2)
        write(unit,fmt=*)node,this%convergence(1:2,node)
     end do
  end if
end subroutine aq_write_convergence
```

! init/ de-init

aq_reset ↑

```
subroutine aq_reset(this)
  class(aq_class) :: this
  this%is_deferred_initialised = .false.
  this%is_error_tree_initialised = .false.
  this%is_goal_set = .false.
  this%is_initialised = .false.
```

```
    this%is_run = .false.
    this%is_goal_reached = .false.
    this%is_integrated = .false.
    this%n_nodes = 0
    this%max_nodes = 10000
    this%dim_integral=1
    this%abs_error_goal = 1D0
    this%rel_error_goal = 0.1D0
    this%scaled_error_goal = 0.0D0
    this%error_goal = 0.0D0
    this%integral = 0D0
    this%integral_error = 0D0
    this%region = (/0D0,1D0/)
    this%total_time = 0
    this%loop_time = 0
    this%int_time = 0
    this%init_time = 0
    call this%dealloc_trees()
  end subroutine aq_reset
```

aq_check_init ↑

```
  subroutine aq_check_init(this)
    class(aq_class) :: this
    this%is_initialised = this%is_error_tree_initialised .and. this%is_deferred_initialised
  end subroutine aq_check_init
```

aq_dealloc_trees ↑

```
  subroutine aq_dealloc_trees(this)
    class(aq_class) :: this
    if(associated(this%err_tree))then
       call this%err_tree%deallocate_all()
       deallocate(this%err_tree)
    end if
    if(associated(this%int_list))then
       call this%int_list%finalize()
       deallocate(this%int_list)
    end if
  end subroutine aq_dealloc_trees
```

aq_init_error_tree ↑

```
  subroutine aq_init_error_tree(this,dim_integral,x_array)
    class(aq_class) :: this
    !Wie viele Einträge hat der Rückgabewert von evaluate?
    integer,intent(in)::dim_integral
    !Eine geordnete Liste von Skalenparametern
    real(kind=double), dimension(:), intent(in) :: x_array
    !(x_j - x_{j-1})/2
    real(kind=double) :: center
    !Die Funktionswerte am linken Rand, in der Mitte und am rechten Rand des Intervalls.
```

```
real(kind=double), dimension(:),allocatable::l_val,c_val,r_val
!Jedes der gegebenen Intervalle wird in zwei Unterintervalle zerlegt, um eine
!Abschätzung des Integrationsfehlees zu bekommen. In left_node und right_node
!werden diese Intervalle gespeichert und in den Binärbaum eingefügt.
class(muli_trapezium_type),pointer :: left_node => null()
class(muli_trapezium_type),pointer :: right_node => null()
!Die Anzahl der gegebenen x-Werte und die Nummer des aktuellen x-Werts.
integer :: x_size,pos
!Timer Start
call cpu_time(this%init_time)
!Signalisieren, dass die Bäume in einem undefinierten Zustand sind.
this%is_initialised=.false.
this%integral=0D0
this%dim_integral=dim_integral
x_size = size(x_array)
if (x_size<2) then
     write (*,'("aq_init_error_tree: I need at least two real values")')
else
   !In der Null-Komponente wird die Summe aller anderen Einträge gespeichert.
   allocate(l_val(0:dim_integral-1))
   allocate(c_val(0:dim_integral-1))
   allocate(r_val(0:dim_integral-1))
   !Der Integrationsbereich wird festgelegt.
   this%region=(/x_array(1),x_array(x_size)/)
   if (x_size<3) then
      !Wir haben nur ein Startsegment, das sich über den gesamten Integrationsbereich
      !erstreckt. Wir Teilen in der Mitte, denn der Binärbaum
      !aq_class%error_tree muss mindestens zwei Blätter haben.
      center=(x_array(2)-x_array(1))/2D0
      !Wir fordern die Funktionswerte an.
      call this%evaluate(x_array(1),l_val)
      call this%evaluate(center,    c_val)
      call this%evaluate(x_array(2),r_val)
      !Wir erzeugen ein neues Segment [x_1,c].
      allocate(left_node)
      call left_node%initialize(&
           &dim=dim_integral,&
           &r_position=center,&
           &d_position=center-x_array(1))
      call left_node%set_r_value(c_val)
      call left_node%set_d_value(c_val-l_val)
      !Wir erzeugen ein neues Segment [c,x_2].
      allocate(right_node)
      call right_node%initialize(&
           &dim=dim_integral,&
           &r_position=x_array(2),&
           &d_position=x_array(2)-center)
      call right_node%set_r_value(r_val)
      call right_node%set_d_value(r_val-c_val)
   else
      !wir haben genügend x-Werte, um einen minimalen Baum
```

```
            !aq_class%error_tree mit zwei Blättern zu initialisieren.
            call this%evaluate(x_array(1),l_val)
            call this%evaluate(x_array(2),c_val)
            call this%evaluate(x_array(3),r_val)
            allocate(left_node)
            call left_node%initialize(&
                 &dim=dim_integral,&
                 &r_position=x_array(2),&
                 &d_position=x_array(2)-x_array(1))
            call left_node%set_r_value(c_val)
            call left_node%set_d_value(c_val-l_val)
            allocate(right_node)
            call right_node%initialize(&
                 &dim=dim_integral,&
                 &r_position=x_array(3),&
                 &d_position=x_array(3)-x_array(2))
            call right_node%set_r_value(r_val)
            call right_node%set_d_value(r_val-c_val)
         end if
         !Die beiden Startblätter des Baums werden bereitgemacht
         call left_node%update()
         call right_node%update()
         !Der Wert für das Integral über diese Blätter wird abgeschätzt.
         this%integral=sum(left_node%get_d_integral()+right_node%get_d_integral())
         if (.not. associated(this%err_tree)) then
            allocate(this%err_tree)
         end if
         !Debugging
         print *,left_node%measure()
         print *,right_node%measure()
         !Der Baum wird mit den beiden Blättern initialisiert.
         call this%err_tree%init_by_content(left_node,right_node)
         !Wenn wir noch mehr Segmente haben, dann werden sie in den Baum aufgenommen.
         if (x_size > 3) then
            do pos=4,x_size
               !Fortschrittsanzeige. Die Intagrationen können einige Minuten dauern.
               print *,"aq_init_error_tree",pos,"/",x_size
               !Wir merken uns den Funktionswert am rechen Rand des letzten Segments.
               !Das ist der neue linke Funktionswert des neuen Segments.
               l_val=right_node%get_r_value_array()
               !Wir forden den Funktionswert am rechten Rand des neuen Intervalls an.
               call this%evaluate(x_array(pos),r_val)
               !Ein Missbrauch der Variablen, c_val ist jetzt die Intervallänge.
               c_val=r_val-l_val
               allocate(right_node)
               call right_node%initialize(&
                    &dim=dim_integral,&
                    &r_position=x_array(pos),&
                    &d_position=x_array(pos)-x_array(pos-1))
               call right_node%set_r_value(r_val)
               call right_node%set_d_value(c_val)
```

```
          call right_node%update()
          call this%err_tree%push_by_content(right_node)
          !Das Gesamtintegral wird um das Integral über das neue Segment erhöht.
          this%integral=this%integral+sum(right_node%get_d_integral())
       end do
       !So viele Blätter hat der Baum jetzt.
       this%n_nodes = x_size
    end if
    !Der Baum ist wieder in einem definierten Zustand.
    this%is_error_tree_initialised=.true.
  end if
  !Da wir jetzt eine erste Abschätzung für das Integral haben, können wir
  !eine Abschätzung für das absolute Fehlerziel machen.
  call this%set_goal()
  !Damit ist alles Bereit für die adaptive Integration.
  this%is_initialised=.true.
  !Timer Stopp
  call cpu_time(this%cpu_time)
  this%init_time=this%cpu_time-this%init_time
  this%cuba_time=this%init_time
  !Debugging: Ab jetzt schreiben wir den aktuellen Integrationsfehler mit.
  allocate(this%convergence(2,this%n_nodes:this%max_nodes))
end subroutine aq_init_error_tree
```

aq_set_abs_goal ↑

```
subroutine aq_set_abs_goal(this,goal)
  class(aq_class) :: this
  real(kind=double) :: goal
  this%abs_error_goal = goal
  call this%set_goal
end subroutine aq_set_abs_goal
```

aq_set_rel_goal ↑

```
subroutine aq_set_rel_goal(this,goal)
  class(aq_class) :: this
  real(kind=double) :: goal
  this%rel_error_goal = goal
  call this%set_goal
end subroutine aq_set_rel_goal
```

aq_set_goal ↑

Die angegebenen Fehlerziele werden auf Konsistenz geprüft. Aus dem relativen Fehler wird mithilfe der aktuellen Abschätzung des Integrals ein absoluter Fehler scaled_error_goal berechnet. Das Minimum aus abs_error_goal und scaled_error_goal wird das tatsächliche absulute Fehlerziel error_goal.

```
subroutine aq_set_goal(this)
  class(aq_class) :: this
  this%scaled_error_goal = this%rel_error_goal*abs(this%integral)
  if ((this%scaled_error_goal==0D0).and.(this%abs_error_goal==0D0)) then
     this%is_goal_set = .false.
```

```
      this%error_goal = 0D0
   else
      if (this%scaled_error_goal == 0D0) then
         this%error_goal = this%abs_error_goal
      else
         if (this%abs_error_goal == 0D0) then
            this%error_goal = this%scaled_error_goal
         else
            this%error_goal = max(this%scaled_error_goal,this%abs_error_goal)
         end if
      end if
      if (this%error_goal > 0D0) then
         this%is_goal_set = .true.
      else
         this%is_goal_set = .false.
      end if
   end if
 end subroutine aq_set_goal
```

! calculation

aq_main_loop ↑

Die eigentliche adaptive Quadratur findet in dieser Prozedur statt.

```
 subroutine aq_main_loop(this)
   ! unsafe, when n_nodes < 4
   class(aq_class) :: this
   !Das Blatt mit dem größten Integrationsfehler
   class(fibonacci_leave_type), pointer :: rightmost

   class(measurable_class), pointer :: content
   class(muli_trapezium_type),pointer :: new_node
   !Wurde die maximale Anzahl von Blättern erreicht?
   logical :: limit = .false.
   !Die Stelle, bei der das Segment geteilt wird.
   real(kind=double) :: center
   !Der Funktionswert an dieser Stelle.
   real(kind=double),dimension(:),allocatable::c_val
   allocate(c_val(0:this%dim_integral-1))
   loop:do
      !Wir holen uns das Blatt mit den größten Integrationsfehler.
      call this%err_tree%pop_right(rightmost)
      !Wenn diese Bedingung erfüllt ist, dann ist auch der gesammte Fehler
      !kleiner als this%error_goal
      if (rightmost < this%error_goal/this%n_nodes) then
         this%is_goal_reached = .true.
         exit loop
      else
         !Wir holen uns das Integrationssegment aus dem Blatt.
         call rightmost%get_content(content)
         !Zugriff auf die speziellen Methoden von muli_trapezium_type
         select type (content)
```

```
        class is (muli_trapezium_type)
           !Fortschrittsanzeige
           print&
           ('("nodes: ",I5," error: ",E14.7," goal: ",E14.7," node at: ",E14.7,"-",E14.7)'),&
                this%n_nodes,&
                rightmost%measure()*this%n_nodes,&
                this%error_goal,&
                content%get_l_position(),&
                content%get_r_position()
           !Debugging: Wir schreiben den Forschritt in den Abbruchbedingung mit.
           this%convergence(1,this%n_nodes)=this%error_goal/this%n_nodes
           this%convergence(2,this%n_nodes)=rightmost%measure()
           !Wir wollen das Segment in der Mitte teilen.
           center = content%get_r_position()-content%get_d_position()/2D0
           call cpu_time(this%cpu_time)
           this%cuba_time=this%cuba_time-this%cpu_time
           !Wir fordern den Funktionswert in der Mitte des Segments an.
           call this%evaluate(center,c_val)
           call cpu_time(this%cpu_time)
           this%cuba_time=this%cuba_time+this%cpu_time
           !Wir teilen das Segment in zwei neue Segmente.
           !Siehe muli_trapezium_type%split
           call content%split(c_val,center,new_node)
           !content ist das rechte Segment und immer noch in rightmost enthalten.
           !Wir können also das Blatt rightmost wieder in den Baum einfügen.
           call this%err_tree%push_by_leave(rightmost)
           !Fur das linke Segment new_node muss noch ein neues Blatt erzeugt werden.
           call this%err_tree%push_by_content(new_node)
        end select
        this%n_nodes=this%n_nodes+1
        !Wenn die maximale Zahl von Blättern erreicht ist, dann müssen wir erfolglos aufhören.
        if (this%n_nodes > this%max_nodes) then
           limit = .true.
           exit loop
        end if
     end if
  end do loop
  !Ein Blatt halten wir noch in der Hand, wir legen es in den Baum zurück.
  call this%err_tree%push_by_leave(rightmost)
end subroutine aq_main_loop
```

aq_run ↑

Wrapper für aq_main_loop.

```
subroutine aq_run(this)
  class(aq_class) :: this
  call cpu_time(this%total_time)
  if (.not. this%is_error_tree_initialised) then
     call this%init_error_tree(this%dim_integral,this%region)
  end if
  this%is_run = .false.
```

```
    this%is_goal_reached = .false.
    call aq_main_loop(this)
    this%is_run = .true.
    call cpu_time(this%cpu_time)
    this%total_time=this%cpu_time-this%total_time
  end subroutine aq_run
```

aq_integrate ↑

Die eigentliche Integration ist schon fertig, aber die Integrationssegmente sind nach Integrationsfehler sortiert. Wir wollen jetzt einen Binärbaum erzeugen, in dem die Segmente nach den x-Werten sortiert sind.

```
  subroutine aq_integrate(this,int_tree)
    class(aq_class) :: this
    class(muli_trapezium_node_class),pointer :: node
    type(muli_trapezium_tree_type),intent(out)::int_tree
    real(kind=double) :: sum
    this%is_integrated=.false.
    this%integral_error=0D0
    if (this%is_run) then
    call cpu_time(this%int_time)
       !Umsortieren
       call fibonacci_tree_resort_and_convert_to_trapezium_list&
         (this%err_tree,this%int_list)
       !Die Integrale über die einzelnen Segmente aufaddieren
       call muli_trapezium_list_integrate(this%int_list,this%integral,this%integral_error)
       !Einen Baum aus der Liste machen
       call this%int_list%to_tree(int_tree)
       this%is_integrated=.true.
       call cpu_time(this%cpu_time)
       this%int_time=this%cpu_time-this%int_time
    end if
  end subroutine aq_integrate
```

7.4.2 Sonstige Prozeduren

fibonacci_tree_resort_and_convert_to_trapezium_list

```
  recursive subroutine fibonacci_tree_resort_and_convert_to_trapezium_list&
    (fib_tree,lin_list)
    !usually, the tree is sorted by the sum of errors.
    !now it shall be sorted by the right position.
    class(fibonacci_node_type),intent(in) :: fib_tree
    class(fibonacci_node_type),pointer :: leave
    class(muli_trapezium_list_type),pointer,intent(out) :: lin_list
    class(muli_trapezium_list_type),pointer :: left_list,right_list
    class(muli_trapezium_node_class),pointer :: left_node,right_node,last_node
    class(measurable_class),pointer :: content
    !When at least one branch of the tree is itself a tree, i.e. each branch has
    !got at least two leaves, then process each branch and merge the results.
    if (fib_tree%depth>1) then
```

```
   call fibonacci_tree_resort_and_convert_to_trapezium_list(fib_tree%left,left_list)
   call fibonacci_tree_resort_and_convert_to_trapezium_list(fib_tree%right,right_list)
   !Now we got two sortet lists.
   !Which one's leftmost node has got the lowest value of "r_position"?
   !That one shall be the beginning of the merged list "lin_list".
   if(left_list%is_left_of(right_list))then
      lin_list => left_list
      call left_list%get_right(left_node)
      right_node=>right_list
   else
      lin_list => right_list
      left_node=>left_list
      call right_list%get_right(right_node)
   end if
   last_node=>lin_list
   !Everything is prepared for the algorithm: lin_list is the beginning of the
   !sorted list, last_node is it's end. left_node and right_node are the leftmost
   !nodes of the remainders of left_list and right_list. The latter will get
   !stripped from left to right, until one of them ends.
   do while(associated(left_node).and.associated(right_node))
      if (left_node%is_left_of(right_node)) then
         call last_node%append(left_node)
         call last_node%get_right(last_node)
         call left_node%get_right(left_node)
      else
         call last_node%append(right_node)
         call last_node%get_right(last_node)
         call right_node%get_right(right_node)
      end if
   end do
   !Either left_list or right_list is completely merged into lin_list. The other
   !one gets appended to lin_list.
   if (associated(left_node)) then
      call last_node%append(left_node)
   else
      call last_node%append(right_node)
   end if
   !It's done.
else
   !The tree has got two leaves at most. Is it more than one?
   if (fib_tree%depth == 0) then
      !Here fib_tree is a single leave with an allocated "content" componet of
      !type muli_trapezium_type. If "content" is not type compatible with
      !muli_trapezium_type, then this whole conversion cannot succeed.
      !We allocate a new node of type muli_trapezium_list_type. This list does
      !not contain the content of fib_tree, it *IS* a copy of the content, for
      !muli_trapezium_list_type is an extension of muli_trapezium_type.
      select type (fib_tree)
      class is (fibonacci_leave_type)
         call fib_tree%get_content(content)
         select type (content)
```

```
            class is (muli_trapezium_type)
               call muli_trapezium_to_node(content,content%get_r_position(),list=lin_list)
            class default
               print *,"fibonacci_tree_resort_and_convert_to_trapezium_list: &
                    &Content of fibonacci_tree is not type compatible to &
                    &muli_trapezium_type"
            end select
         end select
      else
         !Each branch of fib_tree is a single leave. We could call this soubroutine
         !for each branch, but we do copy and paste for each branch instead.
         leave=>fib_tree%left
         select type (leave)
         class is (fibonacci_leave_type)
            call leave%get_content(content)
            select type (content)
            class is (muli_trapezium_type)
               call muli_trapezium_to_node(content,content%get_r_position(),list=left_list)
            class default
               print *,"fibonacci_tree_resort_and_convert_to_trapezium_list: &
                    &Content of fibonacci_tree is not type compatible to &
                    &muli_trapezium_type"
            end select
         end select
         leave=>fib_tree%right
         select type (leave)
         class is (fibonacci_leave_type)
            call leave%get_content(content)
            select type (content)
            class is (muli_trapezium_type)
               call muli_trapezium_to_node%
                 (content,content%get_r_position(),list=right_list)
            class default
               print *,"fibonacci_tree_resort_and_convert_to_trapezium_list: &
                    &Content of fibonacci_tree is not type compatible to &
                    &muli_trapezium_type"
            end select
         end select
         !Finally we append one list to the other, the lowest value of "r_position"
         !comes first.
         if (left_list%is_left_of(right_list)) then
            call left_list%append(right_list)
            lin_list=>left_list
         else
            call right_list%append(left_list)
            lin_list=>right_list
         end if
      end if
   end if
 end subroutine fibonacci_tree_resort_and_convert_to_trapezium_list
```

8 Modul muli_trapezium

In dem Modul muli_trapezium wird ein Datentyp muli_trapezium_type definiert, der eine affin-lineare Approximation von $\overline{S}_{\alpha\beta}(p_\perp;s)$, $\mathcal{S}_{\alpha\beta}(p_\perp;s)$, $exp[-\mathcal{S}_{\alpha\beta}(p_\perp;s)]$ und einen Integrationsfehler für $\mathcal{S}_{\alpha\beta}(p_\perp;s)$ für alle 16 Kombinationen von $\{\alpha,\beta\}$ für $\alpha,\beta \in \{g,s,v_d,v_u\}$ bereitstellt.

Weiterhin dir ein abstrakter Datentyp muli_trapezium_node_class definiert, der mehrere Segmente vom Typ muli_trapezium_type zusammenfasst und somit eine Approximation der Wirkungsquerschnitte entsprechend der Trapezregel bereitstellt. Daher kommt auch der name des Moduls und der Datentypen.

Für die Auswertung dieser Approximation wird ein Datentyp muli_trapezium_tree_type definiert, der Instanzen der Klasse muli_trapezium_node_class zu einem Binärbaum zusammenstellt. Die Blätter dieses Binärbaums sind jeweils mit ihren linken und rechten Nachbarblättern verbunden, sodass man wahlweise in logarithmischer Zeit ein Blatt von der Wurzel her suchen kann oder die Blätter, sortiert nach $p_\perp$ durchlaufen kann.

Für die Verbindungen unter den Blättern wird der Datentyp muli_trapezium_list_type definiert, der auch ohne Binärbaum als Liste verwendet wird. Jedes Blatt eines Baums ist eine Instanz vom Typ muli_trapezium_list_type.

Die Entscheidung für eine affin-lineare Approximation liegt an der Tatsache, dass höhere Polynome transzendente Funktionen im schlimmsten Fall so deutlich unterschätzen können, dass negative approximierte Funktionswerte bei positiven Funktionen auftreten können. In Abbildung **??** ist das illustriert.

8.1 Abhängigkeiten

```
use muli_basic
```

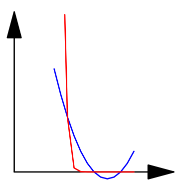


Abbildung 8.1: Illustration einer quadratischen Approximation von exp(-x)/x. Die Parabel (blau) wird negativ, obwohl die Funktion (rot) streng positiv ist. Mit der Trapezregel kann das nicht passieren.

8.2 Parameter

Für jedes Intervall $[x_l, x_r]$ der approximierten Funktionen $g(x)$ wird deren Funktionswert am rechten Rand r_value=$f(x_r)$ und die Differenz der Funktionswerte d_value=$f(x_r) - f(x_l)$ gespeichert. Entsprechend werden die Werte für die negative Stammfunktion (integral) und die daraus berechnete Wahscheinlichkeistfunktion (propability) gespeichert. Schließlich wird ein Integrationsfehler für die Stammfunktion gespeichert. In der Summe macht das value_dimension=7 Werte, die pro Intervall gespeichert werden müssen. Es wird ein array mit value_dimension Stellen allokiert, wobei die verschiedenen Werte an der Position *_index abgelegt werden.

```
implicit none
integer,private,parameter::value_dimension=7
integer,private,parameter::r_value_index=1
integer,private,parameter::d_value_index=2
integer,private,parameter::r_integral_index=3
integer,private,parameter::d_integral_index=4
integer,private,parameter::r_propability_index=5
integer,private,parameter::d_propability_index=6
integer,private,parameter::error_index=7
```

8.3 Derived Types

8.3.1 muli_trapezium_type

```
type,extends(measurable_class) :: muli_trapezium_type
```

dim ist die Dimension des Bildraums, also $\overline{S} : \mathbb{R} \rightarrow \mathbb{R}^{\dim}$. dim ergibt sich aus der Zahl der Strati (=16) plus eins für die Summe über alle Strati. Die Summe wird in der Null-ten Komponente des Funktionswerte-Arrays gespeichert, damit die Strati nach der üblichen Konvention $[1..n]$ indiziert werden.

r_position ist der obere Grenze des $p_\perp$-Intervalls, d_position ist die Intervallänge.

measure_comp ist eine Maßzahl für jedes Intervall, nach der sie in dem Binärbaum sortiert werden. Die Blätter des Baumes sind dann eine geordnete Liste mit aufsteigendem measure_comp. measure_comp wird auf r_position gesetzt.

values ist schließlich die Matrix der Funktionswerte für die verschiedenen Funktionen. Der erste, schnelle Index läuft über die Strati $\{0.. \dim -1\}$, der zweite, langsame Index läuft über die Menge der Funktionen {r_value,d_value,r_integral,d_integral,r_propability,d_propability,error}

```
    private
    integer::dim=0
    real(kind=double)::r_position=0D0
    real(kind=double)::d_position=0D0
    real(kind=double)::measure_comp=0D0
    real(kind=double),dimension(:,:),allocatable::values
  contains
    !Überschriebene serializable_class Methoden
    procedure ::write_to_marker=>muli_trapezium_write_to_marker
    procedure ::read_from_marker=>muli_trapezium_read_from_marker
    procedure ::print_to_unit=>muli_trapezium_print_to_unit
```

```
procedure,nopass ::get_type=>muli_trapezium_get_type
procedure,nopass ::verify_type=>muli_trapezium_verify_type
!Überschriebene measurable_class Methoden
procedure::measure=>muli_trapezium_measure
!Originäre muli_trapezium_type Methoden
! init/deinit
procedure::initialize=>muli_trapezium_initialize
! components
procedure,public::get_dimension=>muli_trapezium_get_dimension
procedure,public::get_l_position=>muli_trapezium_get_l_position
procedure,public::get_r_position=>muli_trapezium_get_r_position
procedure,public::get_d_position=>muli_trapezium_get_d_position
procedure,public::get_l_value_array=>muli_trapezium_get_l_value_array
procedure,public::get_l_value_element=>muli_trapezium_get_l_value_element
procedure,public::get_r_value_array=>muli_trapezium_get_r_value_array
procedure,public::get_r_value_element=>muli_trapezium_get_r_value_element
procedure,public::get_d_value_array=>muli_trapezium_get_d_value_array
procedure,public::get_d_value_element=>muli_trapezium_get_d_value_element
procedure,public::get_l_integral_array=>muli_trapezium_get_l_integral_array
procedure,public::get_l_integral_element=>muli_trapezium_get_l_integral_element
procedure,public::get_r_integral_array=>muli_trapezium_get_r_integral_array
procedure,public::get_r_integral_element=>muli_trapezium_get_r_integral_element
procedure,public::get_d_integral_array=>muli_trapezium_get_d_integral_array
procedure,public::get_d_integral_element=>muli_trapezium_get_d_integral_element
procedure,public::get_l_propability_element=>muli_trapezium_get_l_propability_element
procedure,public::get_l_propability_array=>muli_trapezium_get_l_propability_array
procedure,public::get_r_propability_element=>muli_trapezium_get_r_propability_element
procedure,public::get_r_propability_array=>muli_trapezium_get_r_propability_array
procedure,public::get_d_propability_element=>muli_trapezium_get_d_propability_element
procedure,public::get_d_propability_array=>muli_trapezium_get_d_propability_array
procedure,public::get_error=>muli_trapezium_get_error
procedure,public::get_error_sum=>muli_trapezium_get_error_sum
procedure,public::get_integral_sum=>muli_trapezium_get_integral_sum
generic,public::get_l_value=>get_l_value_array,get_l_value_element
generic,public::get_r_value=>get_r_value_array,get_r_value_element
generic,public::get_d_value=>get_d_value_array,get_d_value_element
generic,public::get_l_integral=>get_l_integral_array,get_l_integral_element
generic,public::get_r_integral=>get_r_integral_array,get_r_integral_element
generic,public::get_d_integral=>get_d_integral_array,get_d_integral_element
generic,public::get_l_propability=>get_l_propability_array,get_l_propability_element
generic,public::get_r_propability=>get_r_propability_array,get_r_propability_element
generic,public::get_d_propability=>get_d_propability_array,get_d_propability_element
! interpolations
procedure,public::get_value_at_position=>muli_trapezium_get_value_at_position
procedure::set_r_value=>muli_trapezium_set_r_value
procedure::set_d_value=>muli_trapezium_set_d_value
procedure::set_r_integral=>muli_trapezium_set_r_integral
procedure::set_d_integral=>muli_trapezium_set_d_integral
procedure::set_r_propability=>muli_trapezium_set_r_propability
procedure::set_d_propability=>muli_trapezium_set_d_propability
procedure::set_error=>muli_trapezium_set_error
```

```
    ! tests
    procedure,public::is_left_of=>muli_trapezium_is_left_of
    procedure,public::includes=>muli_trapezium_includes
    ! convert
    procedure ::to_node=>muli_trapezium_to_node
    procedure ::sum_up=>muli_trapezium_sum_up
    ! approximation
    procedure ::approx_value=>muli_trapezium_approx_value
    procedure ::approx_value_n=>muli_trapezium_approx_value_n
    procedure ::approx_integral=>muli_trapezium_approx_integral
    procedure ::approx_integral_n=>muli_trapezium_approx_integral_n
    procedure ::approx_propability=>muli_trapezium_approx_propability
    procedure ::approx_propability_n=>muli_trapezium_approx_propability_n
    procedure ::approx_position_by_integral=>muli_trapezium_approx_position_by_integral
    procedure ::split=>muli_trapezium_split
    procedure ::update=>muli_trapezium_update
  end type muli_trapezium_type
```

8.3.2 muli_trapezium_node_class

muli_trapezium_node_class ist eine abstake Klasse, die durch die Komonenten left und right wahlweise ein Binärbaum oder eine doppelt-verknüpften Liste sein kann. Welche dieser Ausprägungen vorliegt, hängt von dem Datentyp ab. Jede Instanz der Klasse muli_trapezium_node_class ist entweder vom Typ muli_trapezium_tree_type oder vom Typ muli_trapezium_list_type. Hier werden alle Methoden definiert, die eine Liste, und ein Baum gemein haben.

```
  type,Extendsmuli_trapezium_type,abstract :: muli_trapezium_node_class
```

Komponenten

```
    private
    class(muli_trapezium_node_class), pointer :: left => null()
    class(muli_trapezium_node_class), pointer :: right => null()
```

Methoden

```
   contains
!     private
    !Überschriebene measurable_class Methoden
    procedure,public ::deserialize_from_marker=>muli_trapezium_node_deserialize_from_marker
    !Originäre muli_trapezium_node_class Methoden
    procedure(muli_trapezium_append_interface),deferred,public::append
    procedure(muli_trapezium_final_interface),deferred,public :: finalize
    procedure,public ::nullify=>muli_trapezium_node_nullify
    procedure,public ::get_left=>muli_trapezium_node_get_left
    procedure,public ::get_right=>muli_trapezium_node_get_right
    procedure,public ::get_leftmost=>muli_trapezium_node_get_leftmost
    procedure,public ::get_rightmost=>muli_trapezium_node_get_rightmost
    procedure,public ::decide_by_value=>muli_trapezium_node_decide_by_value
    procedure,public ::decide_by_position=>muli_trapezium_node_decide_by_position
```

```
    procedure,public ::decide_decreasing=>muli_trapezium_node_decide_decreasing
    procedure,public :: muli_trapezium_node_to_tree
    procedure,private::untangle=>muli_trapezium_node_untangle
    procedure,public ::apply=>muli_trapezium_node_apply
    generic,public::decide=>decide_by_value,decide_by_position
  end type muli_trapezium_node_class
```

8.3.3 muli_trapezium_tree_type

muli_trapezium_node_class in der Ausprägung "Binärbaum".

```
  type,extends(muli_trapezium_node_class) :: muli_trapezium_tree_type
```

Komponenten down ist ein Zeiger auf das rechteste Blatt von dem linken Nachfolger. Da das Maß eines Blatts gleich r_position des Blatt ist, gibt down%measure() die obere Grenze des Intervalls des linken Unterbaums wieder. Bei einer Suche nach dem Blatt, das $p_\perp$ enthält, wird also per $p_\perp \overset{?}{<}$ down%measure() entschieden, ob wir nach links oder nach rechts absteigen.

```
    class(muli_trapezium_node_class), pointer :: down => null()
```

Methoden

```
   contains
    !Überschriebene measurable_class Methoden
    procedure ::write_to_marker=>muli_trapezium_tree_write_to_marker
    procedure ::read_from_marker=>muli_trapezium_tree_read_from_marker
    procedure ::print_to_unit=>muli_trapezium_tree_print_to_unit
    procedure,nopass ::get_type=>muli_trapezium_tree_get_type
    procedure,nopass ::verify_type=>muli_trapezium_tree_verify_type
    !Überschriebene muli_trapezium_node_class Methoden
    procedure,public ::nullify=>muli_trapezium_tree_nullify
    procedure,public ::finalize=>muli_trapezium_tree_finalize
    procedure,public ::decide_by_value=>muli_trapezium_tree_decide_by_value
    procedure,public ::decide_by_position=>muli_trapezium_tree_decide_by_position
    procedure,public ::decide_decreasing=>muli_trapezium_tree_decide_decreasing
    !Originäre muli_trapezium_tree_type Methoden
    procedure,public ::get_left_list=>muli_trapezium_tree_get_left_list
    procedure,public ::get_right_list=>muli_trapezium_tree_get_right_list
    procedure,public ::find_by_value=>muli_trapezium_tree_find_by_value
    procedure,public ::find_by_position=>muli_trapezium_tree_find_by_position
    procedure,public ::find_decreasing=>muli_trapezium_tree_find_decreasing
    procedure,public ::approx_by_integral=>muli_trapezium_tree_approx_by_integral
    procedure,public ::approx_by_propability=>muli_trapezium_tree_approx_by_propability
    procedure,public ::to_tree=>muli_trapezium_tree_to_tree
    generic,public::find=>find_by_value,find_by_position
    procedure::append=>muli_trapezium_tree_append
    procedure::gnuplot=>muli_trapezium_tree_gnuplot
  end type muli_trapezium_tree_type
```

8.3.4 muli_trapezium_list_type

muli_trapezium_node_class in der Ausprägung "Liste".

```
type,extends(muli_trapezium_node_class) :: muli_trapezium_list_type
```

Methoden

```
 contains
   !Überschriebene measurable_class Methoden
   procedure ::write_to_marker=>muli_trapezium_list_write_to_marker
   procedure ::read_from_marker=>muli_trapezium_list_read_from_marker
   procedure ::read_target_from_marker=>muli_trapezium_list_read_target_from_marker
   procedure ::print_to_unit=>muli_trapezium_list_print_to_unit
   procedure,nopass ::get_type=>muli_trapezium_list_get_type
   procedure,nopass ::verify_type=>muli_trapezium_list_verify_type
   !Originäre muli_trapezium_list_type Methoden
   procedure,public ::finalize=>muli_trapezium_list_finalize
   procedure,public ::insert_right_a=>muli_trapezium_list_insert_right_a
   generic,public ::insert_right=>insert_right_a!,insert_right_b
   procedure,public ::insert_left_a=>muli_trapezium_list_insert_left_a
   generic,public ::insert_left=>insert_left_a!,insert_left_b
   procedure::append=>muli_trapezium_list_append
   procedure,public ::to_tree=>muli_trapezium_list_to_tree
   procedure,public ::gnuplot=>muli_trapezium_list_gnuplot
   procedure,public ::integrate=>muli_trapezium_list_integrate
   procedure,public ::check=>muli_trapezium_list_check
   procedure,public ::apply=>muli_trapezium_list_apply
end type muli_trapezium_list_type
```

8.4 Schnittstellen

```
abstract interface
   subroutine muli_trapezium_append_interface(this,right)
     import muli_trapezium_node_class
     class(muli_trapezium_node_class),intent(inout),target :: this,right
   end subroutine muli_trapezium_append_interface
   subroutine muli_trapezium_final_interface(this)
     import muli_trapezium_node_class
     class(muli_trapezium_node_class),intent(inout) :: this
   end subroutine muli_trapezium_final_interface
   end interface
```

8.5 Implementierung der Prozeduren

8.5.1 Methoden für qcd_2_2_type

Überschriebene serializable_class Methoden

muli_trapezium_write_to_marker ↑

```
subroutine muli_trapezium_write_to_marker (this,marker,status)
  class(muli_trapezium_type), intent(in) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  ! local variables
  integer::dim
  call marker%mark_begin("muli_trapezium_type")
  call marker%mark("dim",this%dim)
  call marker%mark("r_position",this%r_position)
  call marker%mark("d_position",this%d_position)
  if(allocated(this%values))then
     call marker%mark("values",this%values)
  else
     call marker%mark_null("values")
  end if
  call marker%mark_end("muli_trapezium_type")
end subroutine muli_trapezium_write_to_marker
```

muli_trapezium_read_from_marker ↑

```
subroutine muli_trapezium_read_from_marker (this,marker,status)
  class(muli_trapezium_type), intent(out) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  ! local variables
  integer::dim
  call marker%pick_begin("muli_trapezium_type",status=status)
  call marker%pick("dim",this%dim,status)
  call marker%pick("r_position",this%r_position,status)
  call marker%pick("d_position",this%d_position,status)
  if(allocated(this%values))deallocate(this%values)
  call marker%verify_nothing("values",status)
  if(status==serialize_ok)then
     allocate(this%values(0:this%dim-1,7))
     call marker%pick("values",this%values,status)
  end if
  call marker%pick_end("muli_trapezium_type",status)
end subroutine muli_trapezium_read_from_marker
```

muli_trapezium_print_to_unit ↑

```
subroutine muli_trapezium_print_to_unit(this,unit,parents,components,peers)
  class(muli_trapezium_type),intent(in)::this
  integer,intent(in)::unit
```

```
    integer(kind=dik),intent(in)::parents,components,peers
    write(unit,'("Components of muli_trapezium_type:")')
    write(unit,fmt=*)"Dimension:        ",this%dim
    write(unit,fmt=*)"Right position:   ",this%r_position
    write(unit,fmt=*)"Position step:    ",this%d_position
    if(allocated(this%values))then
       if(components>0)then
          write(unit,fmt=*)"Right values:     ",muli_trapezium_get_r_value_array(this)
          write(unit,fmt=*)"Value step:       ",this%get_d_value()
          write(unit,fmt=*)"Right integrals:  ",this%get_r_integral()
          write(unit,fmt=*)"Integral step:    ",this%get_d_integral()
          write(unit,fmt=*)"Right propabilities:",this%get_r_propability()
          write(unit,fmt=*)"Propability step: ",this%get_d_propability()
          write(unit,fmt=*)"Errors:           ",this%get_error()
       else
          write(unit,fmt=*)"Values are allocated."
       end if
    else
       write(unit,fmt=*)"Values are not allocated."
    end if
  end subroutine muli_trapezium_print_to_unit
```

muli_trapezium_get_type ↑

```
  pure subroutine muli_trapezium_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="muli_trapezium_type")
  end subroutine muli_trapezium_get_type
```

muli_trapezium_verify_type ↑

```
  elemental logical function muli_trapezium_verify_type(type) result(match)
    character(*),intent(in)::type
    match=type=="muli_trapezium_type"
  end function muli_trapezium_verify_type
```

Überschriebene measurable_type Methoden

muli_trapezium_measure ↑

```
  elemental function muli_trapezium_measure(this)
    class(muli_trapezium_type),intent(in)::this
    real(kind=double)::muli_trapezium_measure
    muli_trapezium_measure=this%measure_comp
  end function muli_trapezium_measure
```

Originäre muli_trapezium_type Methoden

muli_trapezium_initialize ↑

```
subroutine muli_trapezium_initialize(this,dim,r_position,d_position)
  class(muli_trapezium_type),intent(inout)::this
  integer,intent(in)::dim
  real(kind=double),intent(in)::r_position,d_position
  integer::dim1,dim2
  this%dim=dim
  this%r_position=r_position
  this%d_position=d_position
  if(allocated(this%values))deallocate(this%values)
  allocate(this%values(0:dim-1,value_dimension))
  do dim2=1,value_dimension-1
     do dim1=0,dim-1
        this%values(dim1,dim2)=0D0
     end do
  end do
  do dim1=0,dim-1
     this%values(dim1,value_dimension)=huge(1D0)
  end do
  this%measure_comp=huge(1D0)
end subroutine muli_trapezium_initialize
```

!!! components !!!

muli_trapezium_get_dimension ↑

```
elemental function muli_trapezium_get_dimension(this) result(dim)
  class(muli_trapezium_type),intent(in)::this
  integer::dim
  dim=this%dim
end function muli_trapezium_get_dimension
```

muli_trapezium_get_l_position ↑

```
pure function muli_trapezium_get_l_position(this) result(pos)
  class(muli_trapezium_type),intent(in)::this
  real(kind=double)::pos
  pos=this%r_position-this%d_position
end function muli_trapezium_get_l_position
```

muli_trapezium_get_r_position ↑

```
pure function muli_trapezium_get_r_position(this) result(pos)
  class(muli_trapezium_type),intent(in)::this
  real(kind=double)::pos
  pos=this%r_position
end function muli_trapezium_get_r_position
```

muli_trapezium_get_d_position ↑

```
pure function muli_trapezium_get_d_position(this) result(pos)
  class(muli_trapezium_type),intent(in)::this
  real(kind=double)::pos
  pos=this%d_position
end function muli_trapezium_get_d_position
```

muli_trapezium_get_error_sum ↑

```
pure function muli_trapezium_get_error_sum(this) result(error)
  class(muli_trapezium_type),intent(in)::this
  real(kind=double)::error
  error=sum(this%values(0:this%dim-1,error_index))
end function muli_trapezium_get_error_sum
```

muli_trapezium_get_integral_sum ↑

```
pure function muli_trapezium_get_integral_sum(this) result(error)
  class(muli_trapezium_type),intent(in)::this
  real(kind=double)::error
  error=sum(this%values(0:this%dim-1,d_integral_index))
end function muli_trapezium_get_integral_sum
```

muli_trapezium_get_l_value_element ↑

```
pure function muli_trapezium_get_l_value_element(this,set) result(element)
  class(muli_trapezium_type),intent(in)::this
  integer,intent(in)::set
  real(kind=double)::element
  element=this%values(set,r_value_index)-this%values(set,d_value_index)
end function muli_trapezium_get_l_value_element
```

muli_trapezium_get_l_value_array ↑

```
pure function muli_trapezium_get_l_value_array(this) result(subarray)
  class(muli_trapezium_type),intent(in)::this
  real(kind=double),dimension(this%dim)::subarray
  subarray=this%values(0:this%dim-1,r_value_index)-this%values(0:this%dim-1,d_value_index)
end function muli_trapezium_get_l_value_array
```

muli_trapezium_get_r_value_element ↑

```
pure function muli_trapezium_get_r_value_element(this,set) result(element)
  class(muli_trapezium_type),intent(in)::this
  integer,intent(in)::set
  real(kind=double)::element
  element=this%values(set,r_value_index)
end function muli_trapezium_get_r_value_element
```

muli_trapezium_get_r_value_array ↑

```
pure function muli_trapezium_get_r_value_array(this) result(subarray)
  class(muli_trapezium_type),intent(in)::this
  real(kind=double),dimension(this%dim)::subarray
  subarray=this%values(0:this%dim-1,r_value_index)
end function muli_trapezium_get_r_value_array
```

muli_trapezium_get_d_value_element ↑

```
pure function muli_trapezium_get_d_value_element(this,set) result(element)
  class(muli_trapezium_type),intent(in)::this
  integer,intent(in)::set
  real(kind=double)::element
  element=this%values(set,d_value_index)
end function muli_trapezium_get_d_value_element
```

muli_trapezium_get_d_value_array ↑

```
pure function muli_trapezium_get_d_value_array(this) result(subarray)
  class(muli_trapezium_type),intent(in)::this
  real(kind=double),dimension(this%dim)::subarray
  subarray=this%values(0:this%dim-1,d_value_index)
end function muli_trapezium_get_d_value_array
```

muli_trapezium_get_l_integral_element ↑

```
pure function muli_trapezium_get_l_integral_element(this,set) result(element)
  class(muli_trapezium_type),intent(in)::this
  integer,intent(in)::set
  real(kind=double)::element
  element=this%values(set,r_integral_index)-this%values(set,d_integral_index)
end function muli_trapezium_get_l_integral_element
```

muli_trapezium_get_l_integral_array ↑

```
pure function muli_trapezium_get_l_integral_array(this) result(subarray)
  class(muli_trapezium_type),intent(in)::this
  real(kind=double),dimension(this%dim)::subarray
  subarray=this%values(0:this%dim-1,r_integral_index)-this%values(0:this%dim-1,d_integral_index)
end function muli_trapezium_get_l_integral_array
```

muli_trapezium_get_r_integral_element ↑

```
pure function muli_trapezium_get_r_integral_element(this,set) result(element)
  class(muli_trapezium_type),intent(in)::this
  integer,intent(in)::set
  real(kind=double)::element
  element=this%values(set,r_integral_index)
end function muli_trapezium_get_r_integral_element
```

muli_trapezium_get_r_integral_array ↑

```
pure function muli_trapezium_get_r_integral_array(this) result(subarray)
  class(muli_trapezium_type),intent(in)::this
  real(kind=double),dimension(this%dim)::subarray
  subarray=this%values(0:this%dim-1,r_integral_index)
end function muli_trapezium_get_r_integral_array
```

muli_trapezium_get_d_integral_element ↑

```
pure function muli_trapezium_get_d_integral_element(this,set) result(element)
  class(muli_trapezium_type),intent(in)::this
  integer,intent(in)::set
  real(kind=double)::element
  element=this%values(set,d_integral_index)
end function muli_trapezium_get_d_integral_element
```

muli_trapezium_get_d_integral_array ↑

```
pure function muli_trapezium_get_d_integral_array(this) result(subarray)
  class(muli_trapezium_type),intent(in)::this
  real(kind=double),dimension(this%dim)::subarray
  subarray=this%values(0:this%dim-1,d_integral_index)
end function muli_trapezium_get_d_integral_array
```

muli_trapezium_get_l_propability_element ↑

```
pure function muli_trapezium_get_l_propability_element(this,set) result(element)
  class(muli_trapezium_type),intent(in)::this
  integer,intent(in)::set
  real(kind=double)::element
  element=this%values(set,r_propability_index)-this%values(set,d_propability_index)
end function muli_trapezium_get_l_propability_element
```

muli_trapezium_get_l_propability_array ↑

```
pure function muli_trapezium_get_l_propability_array(this) result(subarray)
  class(muli_trapezium_type),intent(in)::this
  real(kind=double),dimension(this%dim)::subarray
  subarray=&
    this%values(0:this%dim-1,r_propability_index)&
   -this%values(0:this%dim-1,d_propability_index)
end function muli_trapezium_get_l_propability_array
```

muli_trapezium_get_r_propability_element ↑

```
pure function muli_trapezium_get_r_propability_element(this,set) result(element)
  class(muli_trapezium_type),intent(in)::this
  integer,intent(in)::set
  real(kind=double)::element
  element=this%values(set,r_propability_index)
end function muli_trapezium_get_r_propability_element
```

muli_trapezium_get_r_propability_array ↑

```
pure function muli_trapezium_get_r_propability_array(this) result(subarray)
  class(muli_trapezium_type),intent(in)::this
  real(kind=double),dimension(this%dim)::subarray
  subarray=this%values(0:this%dim-1,r_propability_index)
end function muli_trapezium_get_r_propability_array
```

muli_trapezium_get_d_propability_array ↑

```
  pure function muli_trapezium_get_d_propability_array(this) result(subarray)
    class(muli_trapezium_type),intent(in)::this
    real(kind=double),dimension(this%dim)::subarray
    subarray=this%values(0:this%dim-1,d_propability_index)
  end function muli_trapezium_get_d_propability_array
```

muli_trapezium_get_d_propability_element ↑

```
  pure function muli_trapezium_get_d_propability_element(this,set) result(element)
    class(muli_trapezium_type),intent(in)::this
    integer,intent(in)::set
    real(kind=double)::element
    element=this%values(set,d_propability_index)
  end function muli_trapezium_get_d_propability_element
```

muli_trapezium_get_error ↑

```
  pure function muli_trapezium_get_error(this) result(error)
    class(muli_trapezium_type),intent(in)::this
    real(kind=double),dimension(this%dim)::error
    error=this%values(0:this%dim-1,error_index)
  end function muli_trapezium_get_error
```

! interpolation

muli_trapezium_get_value_at_position ↑

```
  subroutine muli_trapezium_get_value_at_position(this,pos,subarray)
    class(muli_trapezium_type),intent(in)::this
    real(kind=double),intent(in)::pos
    real(kind=double),dimension(this%dim),intent(out)::subarray
    subarray=this%get_r_value_array()-this%get_d_value()*this%d_position/(this%r_position-pos)
  end subroutine muli_trapezium_get_value_at_position
```

! write access

muli_trapezium_set_r_value ↑

```
  subroutine muli_trapezium_set_r_value(this,subarray)
    class(muli_trapezium_type),intent(inout)::this
    real(kind=double),intent(in),dimension(0:this%dim-1)::subarray
    this%values(0:this%dim-1,r_value_index)=subarray
  end subroutine muli_trapezium_set_r_value
```

muli_trapezium_set_d_value ↑

```
  subroutine muli_trapezium_set_d_value(this,subarray)
    class(muli_trapezium_type),intent(inout)::this
    real(kind=double),intent(in),dimension(0:this%dim-1)::subarray
    this%values(0:this%dim-1,d_value_index)=subarray
  end subroutine muli_trapezium_set_d_value
```

muli_trapezium_set_r_integral ↑

```
  subroutine muli_trapezium_set_r_integral(this,subarray)
    class(muli_trapezium_type),intent(inout)::this
    real(kind=double),intent(in),dimension(0:this%dim-1)::subarray
    this%values(0:this%dim-1,r_integral_index)=subarray
  end subroutine muli_trapezium_set_r_integral
```

muli_trapezium_set_d_integral ↑

```
  subroutine muli_trapezium_set_d_integral(this,subarray)
    class(muli_trapezium_type),intent(inout)::this
    real(kind=double),intent(in),dimension(0:this%dim-1)::subarray
    this%values(0:this%dim-1,d_integral_index)=subarray
  end subroutine muli_trapezium_set_d_integral
```

muli_trapezium_set_r_propability ↑

```
  subroutine muli_trapezium_set_r_propability(this,subarray)
    class(muli_trapezium_type),intent(inout)::this
    real(kind=double),intent(in),dimension(0:this%dim-1)::subarray
    this%values(0:this%dim-1,r_propability_index)=subarray
  end subroutine muli_trapezium_set_r_propability
```

muli_trapezium_set_d_propability ↑

```
  subroutine muli_trapezium_set_d_propability(this,subarray)
    class(muli_trapezium_type),intent(inout)::this
    real(kind=double),intent(in),dimension(0:this%dim-1)::subarray
    this%values(0:this%dim-1,d_propability_index)=subarray
  end subroutine muli_trapezium_set_d_propability
```

muli_trapezium_set_error ↑

```
  subroutine muli_trapezium_set_error(this,subarray)
    class(muli_trapezium_type),intent(inout)::this
    real(kind=double),intent(in),dimension(0:this%dim-1)::subarray
    this%values(0:this%dim-1,error_index)=subarray
    this%measure_comp=sum(subarray)
  end subroutine muli_trapezium_set_error
```

! tests

muli_trapezium_is_left_of ↑

```
  pure function muli_trapezium_is_left_of(this,that) result(is_left)
    logical::is_left
    class(muli_trapezium_type),intent(in)::this,that
    is_left=this%r_position<=that%r_position!-that%d_position
  end function muli_trapezium_is_left_of
```

muli_trapezium_includes ↑

```
  elemental logical function muli_trapezium_includes&
    (this,dim,position,value,integral,propability) result(includes)
    class(muli_trapezium_type),intent(in)::this
    integer,intent(in)::dim
    real(kind=double),intent(in),optional::position,value,integral,propability
    includes=.true.
    if(present(position))then
       if(this%get_l_position()>position.or.position>=this%get_r_position())&
         includes=.false.
    end if
    if(present(value))then
       if(this%get_l_value(dim)>value.or.value>=this%get_r_value(dim))&
         includes=.false.
    end if
    if(present(integral))then
       if(this%get_l_integral(dim)>integral.or.integral>=this%get_r_integral(dim))&
         includes=.false.
    end if
    if(present(propability))then
       if(this%get_l_propability(dim)>propability.or.propability>=this%get_r_propability(dim))&
         includes=.false.
    end if
  end function muli_trapezium_includes
```

muli_trapezium_update ↑

```
  subroutine muli_trapezium_update(this)
    class(muli_trapezium_type),intent(inout) :: this
    real(kind=double),dimension(:),allocatable :: int
    allocate(int(0:this%dim-1),source=this%get_d_integral())
    call this%set_d_integral(-this%d_position*(this%get_r_value_array()-this%get_d_value()/2D0))
    call this%set_error(abs(this%get_d_integral()-int))
!    print('(11(D20.10))'),this%get_d_integral()
  end subroutine muli_trapezium_update
```

muli_trapezium_split ↑

```
  subroutine muli_trapezium_split(this,c_value,c_position,new_node)
    class(muli_trapezium_type),intent(inout) :: this
    real(kind=double),intent(in) :: c_position
    real(kind=double),intent(in),dimension(this%dim) :: c_value
    class(muli_trapezium_type),intent(out),pointer :: new_node
    real(kind=double) :: ndpr,ndpl
    real(kind=double),dimension(:),allocatable::ov,edv
    ndpr=this%r_position-c_position
    ndpl=this%d_position-ndpr
    allocate(ov(0:this%dim-1),&
      source=this%get_r_value_array()-ndpr*this%get_d_value()/this%d_position)
    allocate(edv(0:this%dim-1),source=c_value-ov)
    allocate(new_node)
    call new_node%initialize(dim=this%dim,&
         &r_position=c_position,&
```

```
        &d_position=ndpl)
    call new_node%set_r_value(c_value)
    call new_node%set_d_value(this%get_d_value()+c_value-this%get_r_value_array())
    call new_node%set_d_integral(ndpl*(this%get_d_value()-this%get_r_value_array()-c_value)/2
    call new_node%set_error(abs((edv*ndpl)/2D0))
    !new_node%measure_comp=sum(abs((edv*ndpl)/2D0))
    this%d_position=ndpr
    call this%set_d_value(this%get_r_value_array()-c_value)
    call this%set_d_integral(-(ndpr*(this%get_r_value_array()+c_value)/2D0))
    call this%set_error(abs(edv*ndpr/2D0))
    !this%measure_comp=sum(abs(edv*ndpr/2D0))
!     print ('("muli_trapezium_split: new errors:")')
!     print ('(E14.7)'),this%get_error()
!     print ('(E14.7)'),new_node%get_error()
!     print('(11(D20.10))'),new_node%get_d_integral()
!     print('(11(D20.10))'),this%get_d_integral()
  end subroutine muli_trapezium_split
```

muli_trapezium_approx_value ↑

```
  pure function muli_trapezium_approx_value(this,x) result(val)
    ! returns the values at x
    class(muli_trapezium_type),intent(in) :: this
    real(kind=double),dimension(this%dim) :: val
    real(kind=double), intent(in) :: x
    val = this%get_r_value_array()&
          +(x-this%r_position)*this%get_d_value()/this%d_position
  end function muli_trapezium_approx_value
```

muli_trapezium_approx_value_n ↑

```
  elemental function muli_trapezium_approx_value_n(this,x,n) result(val)
    ! returns the value at x
    class(muli_trapezium_type),intent(in) :: this
    real(kind=double)::val
    real(kind=double), intent(in) :: x
    integer,intent(in)::n
    val = this%get_r_value_element(n)&
          +(x-this%r_position)*this%get_d_value_element(n)/this%d_position
  end function muli_trapezium_approx_value_n
```

muli_trapezium_approx_integral ↑

```
  pure function muli_trapezium_approx_integral(this,x)
    ! returns the integral from x to r_position
    class(muli_trapezium_type),intent(in) :: this
    real(kind=double),dimension(this%dim) :: muli_trapezium_approx_integral
    real(kind=double), intent(in) :: x
    muli_trapezium_approx_integral = &
         &this%get_r_integral()+&
         &((this%r_position-x)*&
         &(-this%get_d_value()*(this%r_position-x)&
```

```
        +2*this%d_position*this%get_r_value_array()))/&
      &(2*this%d_position)
  end function muli_trapezium_approx_integral
```

muli_trapezium_approx_integral_n ↑

```
  elemental function muli_trapezium_approx_integral_n(this,x,n) result(val)
    ! returns the integral from x to r_position
    class(muli_trapezium_type),intent(in) :: this
    real(kind=double)::val
    real(kind=double), intent(in) :: x
    integer,intent(in)::n
    val = &
         &this%get_r_integral_element(n)+&
         &((this%r_position-x)*&
         &(-this%get_d_value_element(n)*(this%r_position-x)&
           +2*this%d_position*this%get_r_value_element(n)))/&
         &(2*this%d_position)
  end function muli_trapezium_approx_integral_n
```

muli_trapezium_approx_propability ↑

```
   pure function muli_trapezium_approx_propability(this,x) result(prop)
    ! returns the vlaues at x
    class(muli_trapezium_type),intent(in) :: this
    real(kind=double),dimension(this%dim) :: prop
    real(kind=double), intent(in) :: x
    prop=exp(-this%approx_integral(x))
  end function muli_trapezium_approx_propability
```

muli_trapezium_approx_propability_n ↑

```
  elemental function muli_trapezium_approx_propability_n(this,x,n) result(val)
    ! returns the integral from x to r_position
    class(muli_trapezium_type),intent(in) :: this
    real(kind=double)::val
    real(kind=double), intent(in) :: x
    integer,intent(in)::n
    val = exp(-this%approx_integral_n(x,n))
  end function muli_trapezium_approx_propability_n
```

muli_trapezium_approx_position_by_integral ↑

```
  elemental function muli_trapezium_approx_position_by_integral(this,dim,int) result(val)
    class(muli_trapezium_type),intent(in) :: this
    real(kind=double)::val
    integer,intent(in)::dim
    real(kind=double),intent(in)::int
    real(kind=double)::dpdv
    dpdv=(this%d_position/this%values(dim,d_value_index))
    val=this%r_position-dpdv*&
         (this%values(dim,r_value_index)-&
           sqrt(((this%values(dim,r_integral_index)-int)*2D0/dpdv)&
```

```
                +this%values(dim,r_value_index)**2))
end function muli_trapezium_approx_position_by_integral
```

muli_trapezium_to_node ↑

```
subroutine muli_trapezium_to_node(this,value,list,tree)
  class(muli_trapezium_type),intent(in) :: this
  real(kind=double),intent(in) :: value
  class(muli_trapezium_list_type),optional,pointer,intent(out) :: list
  class(muli_trapezium_tree_type),optional,pointer,intent(out) :: tree
  if(present(list))then
     allocate(list)
     list%dim=this%dim
     list%r_position=this%r_position
     list%d_position=this%d_position
     allocate(list%values(0:this%dim-1,value_dimension),source=this%values)
  end if
  if(present(tree))then
     allocate(tree)
     tree%dim=this%dim
     tree%r_position=this%r_position
     tree%d_position=this%d_position
     allocate(tree%values(0:this%dim-1,value_dimension),source=this%values)
  end if
end subroutine muli_trapezium_to_node
```

muli_trapezium_sum_up ↑

```
subroutine muli_trapezium_sum_up(this)
  class(muli_trapezium_type),intent(inout) :: this
  integer::i
  if(allocated(this%values))then
     do i=1,7
        this%values(0,i)=sum(this%values(1:this%dim-1,i))
     end do
  end if
end subroutine muli_trapezium_sum_up
```

8.5.2 Methoden für muli_trapezium_node_class

muli_trapezium_node_deserialize_from_marker ↑

```
subroutine muli_trapezium_node_deserialize_from_marker(this,name,marker)
  class(muli_trapezium_node_class), intent(out) :: this
  character(*),intent(in)::name
  class(marker_type),intent(inout)::marker
  integer(kind=dik)::status
  class(serializable_class),pointer::ser
  allocate(muli_trapezium_tree_type::ser)
  call marker%push_reference(ser)
  allocate(muli_trapezium_list_type::ser)
```

```
    call marker%push_reference(ser)
    call serializable_deserialize_from_marker(this,name,marker)
    call marker%pop_reference(ser)
    deallocate(ser)
    call marker%pop_reference(ser)
    deallocate(ser)
  end subroutine muli_trapezium_node_deserialize_from_marker
```

muli_trapezium_node_nullify ↑

```
  subroutine muli_trapezium_node_nullify(this)
    class(muli_trapezium_node_class),intent(out) :: this
    nullify(this%left)
    nullify(this%right)
  end subroutine muli_trapezium_node_nullify
```

muli_trapezium_node_get_left ↑

```
  subroutine muli_trapezium_node_get_left(this,left)
    class(muli_trapezium_node_class),intent(in) :: this
    class(muli_trapezium_node_class),pointer,intent(out) :: left
    left=>this%left
  end subroutine muli_trapezium_node_get_left
```

muli_trapezium_node_get_right ↑

```
  subroutine muli_trapezium_node_get_right(this,right)
    class(muli_trapezium_node_class),intent(in) :: this
    class(muli_trapezium_node_class),pointer,intent(out) :: right
    right=>this%right
  end subroutine muli_trapezium_node_get_right
```

muli_trapezium_node_get_leftmost ↑

```
  subroutine muli_trapezium_node_get_leftmost(this,node)
    class(muli_trapezium_node_class),intent(in) :: this
    class(muli_trapezium_node_class),pointer,intent(out) :: node
    if (associated(this%left)) then
       node=>this%left
       do while (associated(node%left))
          node=>node%left
       end do
    else
       nullify(node)
    end if
  end subroutine muli_trapezium_node_get_leftmost
```

muli_trapezium_node_get_rightmost ↑

```
  subroutine muli_trapezium_node_get_rightmost(this,right)
    class(muli_trapezium_node_class),intent(in) :: this
    class(muli_trapezium_node_class),pointer,intent(out) :: right
    if (associated(this%right)) then
```

```
      right=>this%right
      do while (associated(right%right))
         right=>right%right
      end do
   else
      nullify(right)
   end if
 end subroutine muli_trapezium_node_get_rightmost
```

muli_trapezium_node_decide_by_value ↑

```
 subroutine muli_trapezium_node_decide_by_value(this,value,dim,record,node)
   class(muli_trapezium_node_class),intent(in) :: this
   real(kind=double),intent(in)::value
   integer,intent(in)::record,dim
   class(muli_trapezium_node_class),pointer,intent(out) :: node
   if(this%values(dim,record)>value)then
      node=>this%left
   else
      node=>this%right
   end if
 end subroutine muli_trapezium_node_decide_by_value
```

muli_trapezium_node_decide_by_position ↑

```
 subroutine muli_trapezium_node_decide_by_position(this,position,node)
   class(muli_trapezium_node_class),intent(in) :: this
   real(kind=double),intent(in)::position
   class(muli_trapezium_node_class),pointer,intent(out) :: node
   if(this%r_position>position)then
      node=>this%left
   else
      node=>this%right
   end if
 end subroutine muli_trapezium_node_decide_by_position
```

muli_trapezium_node_decide_decreasing ↑

```
 subroutine muli_trapezium_node_decide_decreasing(this,value,dim,record,node)
   class(muli_trapezium_node_class),intent(in) :: this
   real(kind=double),intent(in)::value
   integer,intent(in)::record,dim
   class(muli_trapezium_node_class),pointer,intent(out) :: node
   if(this%values(dim,record)<=value)then
      node=>this%left
   else
      node=>this%right
   end if
 end subroutine muli_trapezium_node_decide_decreasing
```

muli_trapezium_node_untangle ↑

```
subroutine muli_trapezium_node_untangle(this)
  class(muli_trapezium_node_class),intent(inout),target :: this
  if(associated(this%left))then
     if(associated(this%left%right,this))then
        nullify(this%left%right)
        nullify(this%left)
     end if
  end if
end subroutine muli_trapezium_node_untangle
```

muli_trapezium_node_apply ↑

```
recursive subroutine muli_trapezium_node_apply(this,proc)
  class(muli_trapezium_node_class),intent(inout) :: this
  interface
     subroutine proc(this)
       import muli_trapezium_node_class
       class(muli_trapezium_node_class),intent(inout) :: this
     end subroutine proc
  end interface
  if(associated(this%right))call proc(this%right)
  if(associated(this%left))call proc(this%left)
  call proc(this)
end subroutine muli_trapezium_node_apply
```

8.5.3 Methoden für muli_trapezium_list_type

muli_trapezium_list_write_to_marker ↑

```
recursive subroutine muli_trapezium_list_write_to_marker (this,marker,status)
  class(muli_trapezium_list_type), intent(in) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  ! local variables
  class(serializable_class),pointer::ser
  call marker%mark_begin("muli_trapezium_list_type")
  call muli_trapezium_write_to_marker(this,marker,status)
  ser=>this%right
  call marker%mark_pointer("right",ser)
  call marker%mark_end("muli_trapezium_list_type")
end subroutine muli_trapezium_list_write_to_marker
```

muli_trapezium_list_read_from_marker ↑

```
recursive subroutine muli_trapezium_list_read_from_marker (this,marker,status)
  class(muli_trapezium_list_type), intent(out) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  print *,"muli_trapezium_list_read_from_marker:"
  print *,"You cannot deserialize a list with this subroutine."
```

```
    print *,"Use muli_trapezium_list_read_target_from_marker instead."
  end subroutine muli_trapezium_list_read_from_marker
```

muli_trapezium_list_read_target_from_marker ↑

```
  recursive subroutine muli_trapezium_list_read_target_from_marker (this,marker,status)
    class(muli_trapezium_list_type),target,intent(out) :: this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    ! local variables
    class(serializable_class),pointer::ser
    call marker%pick_begin("muli_trapezium_list_type",status=status)
    call muli_trapezium_read_from_marker(this,marker,status)
    call marker%pick_pointer("right",ser)
    if(associated(ser))then
       select type(ser)
       class is (muli_trapezium_list_type)
          this%right=>ser
          ser%left=>this
       class default
          nullify(this%right)
          print *,"muli_trapezium_list_read_target_from_marker:"
          print *,"Unexpected type for right component."
       end select
    else
       nullify(this%right)
    end if
    call marker%pick_end("muli_trapezium_list_type",status)
  end subroutine muli_trapezium_list_read_target_from_marker
```

muli_trapezium_list_print_to_unit ↑

```
  recursive subroutine muli_trapezium_list_print_to_unit&
    (this,unit,parents,components,peers)
    class(muli_trapezium_list_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    class(serializable_class),pointer::ser
    if(parents>0)call muli_trapezium_print_to_unit(this,unit,parents-1,components,peers)
    ser=>this%left
    call serialize_print_peer_pointer(ser,unit,-one,-one,-one,"LEFT")
    ser=>this%right
    call serialize_print_peer_pointer(ser,unit,parents,components,peers,"RIGHT")
  end subroutine muli_trapezium_list_print_to_unit
```

muli_trapezium_list_get_type ↑

```
  pure subroutine muli_trapezium_list_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="muli_trapezium_list_type")
  end subroutine muli_trapezium_list_get_type
```

muli_trapezium_list_verify_type ↑

```
elemental logical function muli_trapezium_list_verify_type(type) result(match)
  character(*),intent(in)::type
  match=type=="muli_trapezium_list_type"
end function muli_trapezium_list_verify_type
```

muli_trapezium_list_finalize ↑

```
recursive subroutine muli_trapezium_list_finalize(this)
  class(muli_trapezium_list_type),intent(inout)::this
  if (associated(this%right)) then
     call this%right%finalize()
     deallocate(this%right)
  end if
  this%dim=0
end subroutine muli_trapezium_list_finalize
```

muli_trapezium_list_insert_left_a ↑

```
subroutine muli_trapezium_list_insert_left_a(this,value,content,new_node)
  class(muli_trapezium_list_type),intent(inout),target :: this
  real(kind=double),intent(in) :: value
  class(muli_trapezium_type),intent(in) :: content
  class(muli_trapezium_list_type),pointer,intent(out) :: new_node
  call content%to_node(value,list=new_node)
  new_node%right=>this
  if(associated(this%left))then
     new_node%left=>this%left
     this%left%right=>new_node
  else
     nullify(new_node%left)
  end if
  this%left=>new_node
end subroutine muli_trapezium_list_insert_left_a
```

muli_trapezium_list_insert_right_a ↑

```
subroutine muli_trapezium_list_insert_right_a(this,value,content,new_node)
  class(muli_trapezium_list_type),intent(inout),target :: this
  real(kind=double),intent(in) :: value
  class(muli_trapezium_type),intent(in) :: content
  class(muli_trapezium_list_type),pointer,intent(out) :: new_node
  class(muli_trapezium_list_type),pointer :: tmp_list
  call content%to_node(value,list=tmp_list)
  if(associated(this%right))then
     this%right%left=>tmp_list
     tmp_list%right=>this%right
  else
     nullify(tmp_list%right)
  end if
  this%right=>tmp_list
  tmp_list%left=>this
  new_node=>tmp_list
end subroutine muli_trapezium_list_insert_right_a
```

muli_trapezium_list_append ↑

```
subroutine muli_trapezium_list_append(this,right)
  class(muli_trapezium_list_type),intent(inout),target :: this
  class(muli_trapezium_node_class),intent(inout),target :: right
  this%right=>right
  right%left=>this
end subroutine muli_trapezium_list_append
```

muli_trapezium_list_to_tree ↑

```
subroutine muli_trapezium_list_to_tree(this,out_tree)
  class(muli_trapezium_list_type),target,intent(in) :: this
  class(muli_trapezium_tree_type),intent(out) :: out_tree
  type(muli_trapezium_tree_type),target :: do_list
  class(muli_trapezium_node_class),pointer :: this_entry,do_list_entry,node
  class(muli_trapezium_tree_type),pointer :: tree1,tree2
  integer :: ite,log,n_deep,n_leaves
  n_leaves=0
  this_entry => this
  count: do while(associated(this_entry))
     n_leaves=n_leaves+1
     this_entry=>this_entry%right
  end do count
  call ilog2(n_leaves,log,n_deep)
  this_entry => this
  do_list_entry => do_list
  deep: do ite=0,n_deep-1
     allocate(tree1)
     tree1%down=>this_entry%right
     allocate(tree2)
     tree2%down=>this_entry
     tree2%left=>this_entry
     tree2%right=>this_entry%right
     tree1%left=>tree2
     this_entry => this_entry%right%right
     do_list_entry%right=>tree1
     do_list_entry=>tree1
  end do deep
  rest: do while(associated(this_entry))
     allocate(tree1)
     tree1%down=>this_entry
     tree1%left=>this_entry
     do_list_entry%right => tree1
     do_list_entry => tree1
     this_entry => this_entry%right
     ite=ite+1
  end do rest
  tree: do while(ite>2)
     do_list_entry => do_list%right
     node=>do_list
```

```
      level: do while(associated(do_list_entry))
         node%right=>do_list_entry%right
         node=>do_list_entry%right
         do_list_entry%right=>node%left
         node%left=>do_list_entry
         do_list_entry=>node%right
         ite=ite-1
      end do level
    end do tree
    node=>do_list%right
    select type(node)
    type is (muli_trapezium_tree_type)
       call node%to_tree(out_tree)
    class default
       print *,"muli_trapezium_list_to_tree"
       print *,"unexpeted type for do_list%right"
    end select
    out_tree%right=>out_tree%right%left
    if(allocated(out_tree%values))then
       deallocate(out_tree%values)
    end if
    deallocate(do_list%right%right)
    deallocate(do_list%right)
  end subroutine muli_trapezium_list_to_tree
```

muli_trapezium_list_gnuplot ↑

```
  subroutine muli_trapezium_list_gnuplot(this,dir)
    class(muli_trapezium_list_type),intent(in),target :: this
    character(len=*),intent(in)::dir
    character(len=*),parameter::val_file="/value.plot"
    character(len=*),parameter::int_file="/integral.plot"
    character(len=*),parameter::err_file="/integral_error.plot"
    character(len=*),parameter::pro_file="/propability.plot"
    character(len=*),parameter::den_file="/density.plot"
    character(len=*),parameter::fmt='(E20.10)'
    class(muli_trapezium_node_class),pointer::list
    integer::val_unit,err_unit,int_unit,pro_unit,den_unit
    list=>this
    call generate_unit(val_unit,100,1000)
    open(val_unit,file=dir//val_file)
    call generate_unit(int_unit,100,1000)
    open(int_unit,file=dir//int_file)
    call generate_unit(err_unit,100,1000)
    open(err_unit,file=dir//err_file)
    call generate_unit(pro_unit,100,1000)
    open(pro_unit,file=dir//pro_file)
    call generate_unit(den_unit,100,1000)
    open(den_unit,file=dir//den_file)
    do while (associated(list))
!       print *,list%r_position,list%get_r_value()
```

```
      write(val_unit,fmt,advance='NO')list%r_position
      call write_array(val_unit,list%get_r_value_array(),fmt)
      write(int_unit,fmt,advance='NO')list%r_position
      call write_array(int_unit,list%get_r_integral(),fmt)
      write(err_unit,fmt,advance='NO')list%r_position
      call write_array(err_unit,list%get_error(),fmt)
      write(pro_unit,fmt,advance='NO')list%r_position
      call write_array(pro_unit,list%get_r_propability(),fmt)
      write(den_unit,fmt,advance='NO')list%r_position
      call write_array(den_unit,list%get_r_propability()*list%get_r_value_array(),fmt)
      list=>list%right
   end do
   close(val_unit)
   close(int_unit)
   close(err_unit)
   close(pro_unit)
   close(den_unit)
   contains
     subroutine write_array(unit,array,form)
       integer,intent(in)::unit
       real(kind=double),dimension(:),intent(in)::array
       character(len=*),intent(in)::form
       integer::n
       do n=1,size(array)
          write(unit,form,ADVANCE='NO')array(n)
          flush(unit)
       end do
       write(unit,'("")')
     end subroutine write_array
 end subroutine muli_trapezium_list_gnuplot
```

muli_trapezium_list_integrate ↑

```
 subroutine muli_trapezium_list_integrate(this,integral_sum,error_sum)
   class(muli_trapezium_list_type),intent(in),target :: this
   real(kind=double),intent(out)::error_sum,integral_sum
   real(kind=double),dimension(:),allocatable::integral
   class(muli_trapezium_node_class),pointer :: node
   allocate(integral(0:this%dim-1))
   call this%get_rightmost(node)
   integral=0D0
   integral_sum=0D0
   error_sum=0D0
   integrate: do while(associated(node))
      node%values(1,r_value_index)=sum(node%values(1:this%dim-1,r_value_index))
      node%values(1,d_value_index)=sum(node%values(1:this%dim-1,d_value_index))
      node%values(1,error_index)=sum(node%values(1:this%dim-1,error_index))
      error_sum=error_sum+node%values(1,error_index)
      call node%set_d_integral(&
        node%get_d_position()*(node%get_d_value()/2D0-node%get_r_value_array()))
      call node%set_r_propability(exp(-integral))
```

```
      call node%set_r_integral(integral)
      integral=integral-node%get_d_integral()
      call node%set_d_propability(node%get_r_propability()-exp(-integral))
      call node%get_left(node)
    end do integrate
    integral_sum=integral(1)
  end subroutine muli_trapezium_list_integrate
```

muli_trapezium_list_check ↑

```
  recursive subroutine muli_trapezium_list_check(this)
    class(muli_trapezium_list_type),intent(in),target :: this
    class(muli_trapezium_node_class),pointer::tn,next
    real(kind=double),parameter::eps=1d-10
    logical::test
    if(associated(this%right))then
      next=>this%right
      test=(this%r_position.le.this%right%get_l_position()+eps)
      print *,"position check:  ",test
      if(.not.test)then
        call this%print_parents()
        call next%print_parents()
      end if
      select type (next)
      class is (muli_trapezium_list_type)
        tn=>this
        print *,"structure check: ",associated(tn,next%left)
        print *,"class check:    T"
        call next%check()
      class default
        print *,"class check:    F"
      end select
    else
      print *,"end of list at ",this%r_position
    end if
  end subroutine muli_trapezium_list_check
```

muli_trapezium_list_apply ↑

```
  recursive subroutine muli_trapezium_list_apply(this,proc)
    class(muli_trapezium_list_type),intent(inout) :: this
    interface
      subroutine proc(this)
        import muli_trapezium_node_class
        class(muli_trapezium_node_class),intent(inout) :: this
      end subroutine proc
    end interface
    if(associated(this%right))call this%right%apply(proc)
    call proc(this)
  end subroutine muli_trapezium_list_apply
```

8.5.4 Methoden für muli_trapezium_tree_type

muli_trapezium_tree_write_to_marker ↑

```
subroutine muli_trapezium_tree_write_to_marker (this,marker,status)
  class(muli_trapezium_tree_type), intent(in) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  ! local variables
  class(muli_trapezium_list_type),pointer::list
  class(serializable_class),pointer::ser
  call marker%mark_begin("muli_trapezium_tree_type")
  call this%get_left_list(list)
  ser=>list
  call marker%mark_pointer("list",ser)
  call marker%mark_end("muli_trapezium_tree_type")
end subroutine muli_trapezium_tree_write_to_marker
```

muli_trapezium_tree_read_from_marker ↑

```
subroutine muli_trapezium_tree_read_from_marker (this,marker,status)
  class(muli_trapezium_tree_type), intent(out) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  ! local variables
  class(serializable_class),pointer::ser
  call marker%pick_begin("muli_trapezium_tree_type",status=status)
  call marker%pick_pointer("list",ser)
  if(associated(ser))then
     select type(ser)
     class is (muli_trapezium_list_type)
        call ser%to_tree(this)
     class default
        nullify(this%left)
        nullify(this%right)
        nullify(this%down)
     end select
  else
     nullify(this%left)
     nullify(this%right)
     nullify(this%down)
  end if
  call marker%pick_end("muli_trapezium_tree_type",status)
end subroutine muli_trapezium_tree_read_from_marker
```

muli_trapezium_tree_print_to_unit ↑

```
recursive subroutine muli_trapezium_tree_print_to_unit(this,unit,parents,components,peers)
  class(muli_trapezium_tree_type),intent(in)::this
  integer,intent(in)::unit
  integer(kind=dik),intent(in)::parents,components,peers
  class(serializable_class),pointer::ser
```

```
    if(parents>0)call muli_trapezium_print_to_unit(this,unit,parents-1,components,peers)
    ser=>this%down
    call serialize_print_peer_pointer(ser,unit,one,zero,one,"DOWN")
    if(associated(this%left))then
       select type(sertmp=>this%left)
       class is(muli_trapezium_list_type)
          ser=>sertmp
          call serialize_print_peer_pointer(ser,unit,parents,components,zero,"LEFT")
       class default
          call serialize_print_peer_pointer(ser,unit,parents,components,peers,"LEFT")
       end select
    else
       write(unit,fmt=*)"Left is not associated."
    end if
    if(associated(this%right))then
       select type(sertmp=>this%right)
       class is(muli_trapezium_list_type)
          ser=>sertmp
          call serialize_print_peer_pointer(ser,unit,parents,components,zero,"RIGHT")
       class default
          call serialize_print_peer_pointer(ser,unit,parents,components,peers,"RIGHT")
       end select
    else
       write(unit,fmt=*)"Right is not associated."
    end if
  end subroutine muli_trapezium_tree_print_to_unit
```

muli_trapezium_tree_get_type ↑

```
  pure subroutine muli_trapezium_tree_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="muli_trapezium_tree_type")
  end subroutine muli_trapezium_tree_get_type
```

muli_trapezium_tree_verify_type ↑

```
  elemental logical function muli_trapezium_tree_verify_type(type) result(match)
    character(*),intent(in)::type
    match=type=="muli_trapezium_tree_type"
  end function muli_trapezium_tree_verify_type
```

Überschriebene muli_trapezium_node_class Methoden

muli_trapezium_tree_nullify ↑

```
  subroutine muli_trapezium_tree_nullify(this)
    class(muli_trapezium_tree_type),intent(out) :: this
    call muli_trapezium_node_nullify(this)
    nullify(this%down)
  end subroutine muli_trapezium_tree_nullify
```

muli_trapezium_tree_get_left_list ↑

```
subroutine muli_trapezium_tree_get_left_list(this,list)
  class(muli_trapezium_tree_type),intent(in) :: this
  class(muli_trapezium_list_type),pointer,intent(out) :: list
  class(muli_trapezium_node_class),pointer::node
  call this%get_leftmost(node)
  if(associated(node))then
     select type(node)
     class is (muli_trapezium_list_type)
        list=>node
     class default
        nullify(list)
     end select
  else
     nullify(list)
  end if
end subroutine muli_trapezium_tree_get_left_list
```

muli_trapezium_tree_get_right_list ↑

```
subroutine muli_trapezium_tree_get_right_list(this,list)
  class(muli_trapezium_tree_type),intent(in) :: this
  class(muli_trapezium_list_type),pointer,intent(out) :: list
  class(muli_trapezium_node_class),pointer::node
  call this%get_rightmost(node)
  if(associated(node))then
     select type(node)
     class is (muli_trapezium_list_type)
        list=>node
     class default
        nullify(list)
     end select
  else
     nullify(list)
  end if
end subroutine muli_trapezium_tree_get_right_list
```

muli_trapezium_tree_finalize ↑

```
recursive subroutine muli_trapezium_tree_finalize(this)
  class(muli_trapezium_tree_type),intent(inout) :: this
  if (associated(this%right)) then
     call this%right%untangle()
     call this%right%finalize()
     deallocate(this%right)
  end if
  if (associated(this%left)) then
     call this%left%untangle()
     call this%left%finalize()
     deallocate(this%left)
  end if
  this%dim=0
end subroutine muli_trapezium_tree_finalize
```

muli_trapezium_tree_decide_by_value ↑

```
subroutine muli_trapezium_tree_decide_by_value(this,value,dim,record,node)
  class(muli_trapezium_tree_type),intent(in) :: this
  real(kind=double),intent(in)::value
  integer,intent(in)::record,dim
  class(muli_trapezium_node_class),pointer,intent(out) :: node
  if(this%down%values(dim,record)>value)then
     node=>this%left
  else
     node=>this%right
  end if
end subroutine muli_trapezium_tree_decide_by_value
```

muli_trapezium_tree_decide_by_position ↑

```
subroutine muli_trapezium_tree_decide_by_position(this,position,node)
  class(muli_trapezium_tree_type),intent(in) :: this
  real(kind=double),intent(in)::position
  class(muli_trapezium_node_class),pointer,intent(out) :: node
  if(this%down%r_position>position)then
     node=>this%left
  else
     node=>this%right
  end if
end subroutine muli_trapezium_tree_decide_by_position
```

muli_trapezium_tree_decide_decreasing ↑

```
subroutine muli_trapezium_tree_decide_decreasing(this,value,dim,record,node)
  class(muli_trapezium_tree_type),intent(in) :: this
  real(kind=double),intent(in)::value
  integer,intent(in)::record,dim
  class(muli_trapezium_node_class),pointer,intent(out) :: node
  if(this%down%values(dim,record)<=value)then
     node=>this%left
  else
     node=>this%right
  end if
end subroutine muli_trapezium_tree_decide_decreasing
```

muli_trapezium_tree_find_by_value ↑

```
subroutine muli_trapezium_tree_find_by_value(this,value,dim,record,node)
  class(muli_trapezium_tree_type),intent(in),target :: this
  real(kind=double),intent(in)::value
  integer,intent(in)::record,dim
  class(muli_trapezium_node_class),pointer,intent(out) :: node
  node=>this
  do while(.not.allocated(node%values))
     call node%decide(value,dim,record,node)
  end do
end subroutine muli_trapezium_tree_find_by_value
```

muli_trapezium_tree_find_by_position ↑

```
subroutine muli_trapezium_tree_find_by_position(this,position,node)
  class(muli_trapezium_tree_type),intent(in),target :: this
  real(kind=double),intent(in)::position
  class(muli_trapezium_node_class),pointer,intent(out) :: node
  node=>this
  do while(.not.allocated(node%values))
     call node%decide(position,node)
  end do
end subroutine muli_trapezium_tree_find_by_position
```

muli_trapezium_tree_find_decreasing ↑

```
subroutine muli_trapezium_tree_find_decreasing(this,value,dim,node)
  class(muli_trapezium_tree_type),intent(in),target :: this
  real(kind=double),intent(in)::value
  integer,intent(in)::dim
  class(muli_trapezium_node_class),pointer,intent(out) :: node
  node=>this
  do while(.not.allocated(node%values))
     call node%decide_decreasing(value,dim,r_integral_index,node)
  end do
end subroutine muli_trapezium_tree_find_decreasing
```

muli_trapezium_tree_approx_by_integral ↑

```
subroutine muli_trapezium_tree_approx_by_integral&
  (this,int,dim,in_range,position,value,integral,content)
  class(muli_trapezium_tree_type),intent(in),target :: this
  real(kind=double),intent(in) :: int
  integer,intent(in)::dim
  logical,intent(out) :: in_range
  class(muli_trapezium_node_class),pointer,intent(out),optional :: content
  real(kind=double),intent(out),optional :: position,value,integral
  integer::i
  real(kind=double) :: DINT!,l_prop,r_prop,d_prop
  real(kind=double)::RP,DP,RV,DV,RI!FC = gfortran
  class(muli_trapezium_node_class),pointer :: node
  node=>this
  do while(.not.allocated(node%values))
     call node%decide_decreasing(INT,dim,r_integral_index,node)
  end do
  if(   int<=node%values(dim,r_integral_index)-node%values(dim,d_integral_index)&
       &.and.&
       &int>=node%values(dim,r_integral_index))then
     in_range=.true.
       RP=node%r_position!FC = gfortran
       DP=node%d_position!FC = gfortran
       RV=node%values(dim,r_value_index)!FC = gfortran
       DV=node%values(dim,d_value_index)!FC = gfortran
       RI=node%values(dim,r_integral_index)!FC = gfortran
```

```
      if (present(position)) then
         DINT=(ri-int)*2D0*dv/dp
         position=rp-(dp/dv)*(rv-sqrt(dint+rv**2))
      end if
      if (present(value)) then
         value=Sqrt(dp*(-2*dv*int + 2*dv*ri + dp*rv**2))/dp
      end if
      if (present(integral)) then
         integral=int
      end if
      if (present(content)) then
         content=>node
      end if
   else
      in_range=.false.
   end if
 end subroutine muli_trapezium_tree_approx_by_integral
```

muli_trapezium_tree_approx_by_propability ↑

```
 subroutine muli_trapezium_tree_approx_by_propability&
   (this,prop,dim,in_range,position,value,integral,content)
   class(muli_trapezium_tree_type),intent(in),target :: this
   real(kind=double),intent(in) :: prop
   integer,intent(in)::dim
   logical,intent(out) :: in_range
   class(muli_trapezium_node_class),pointer,intent(out),optional :: content
   real(kind=double),intent(out),optional :: position,value,integral
   integer::i
   real(kind=double) :: INT,DINT,l_prop,r_prop,d_prop
   class(muli_trapezium_node_class),pointer :: node
   if(0D0<prop.and.prop<1D0)then
      node=>this
      INT=-log(prop)
      call muli_trapezium_tree_approx_by_integral&
        (this,int,dim,in_range,position,value,integral,content)
   else
      in_range=.false.
   end if
 end subroutine muli_trapezium_tree_approx_by_propability
```

muli_trapezium_tree_to_tree ↑

```
 subroutine muli_trapezium_tree_to_tree(this,out_tree)
   class(muli_trapezium_tree_type),intent(in) :: this
   class(muli_trapezium_tree_type),intent(out) :: out_tree
   out_tree%left=>this%left
   out_tree%right=>this%right
   out_tree%down=>this%down
 end subroutine muli_trapezium_tree_to_tree
```

muli_trapezium_tree_append ↑

```
subroutine muli_trapezium_tree_append(this,right)
  class(muli_trapezium_tree_type),intent(inout),target :: this
  class(muli_trapezium_node_class),intent(inout),target :: right
  print ('("muli_trapezium_tree_append: Not yet implemented.")')
end subroutine muli_trapezium_tree_append
```

muli_trapezium_tree_gnuplot ↑

```
subroutine muli_trapezium_tree_gnuplot(this,dir)
  class(muli_trapezium_tree_type),intent(in) :: this
  character(len=*),intent(in)::dir
  class(muli_trapezium_list_type),pointer::list
  call this%get_left_list(list)
  call list%gnuplot(dir)
end subroutine muli_trapezium_tree_gnuplot
```

9 Modul muli_fibonacci_tree

9.1 Abhängigkeiten

```
use muli_basic
```

9.2 Parameter

```
character(*),parameter,private :: no_par = "edge=\noparent"
character(*),parameter,private :: no_ret = "edge=\noreturn"
character(*),parameter,private :: no_kid = "edge=\nochild"
character(*),parameter,private :: le_kid = "edge=\childofleave"
```

9.3 Derived Types

9.3.1 fibonacci_node_type

```
  type,extends(measurable_class) :: fibonacci_node_type
!     private
     class(fibonacci_node_type), pointer :: up => null()
     class(measurable_class), pointer :: down => null()
     class(fibonacci_node_type), pointer :: left => null()
     class(fibonacci_node_type), pointer :: right => null()
     integer :: depth = 0
   contains
     ! overridden serializable_class procedures
     procedure::write_to_marker=>fibonacci_node_write_to_marker
     procedure::read_from_marker=>fibonacci_node_read_from_marker
     procedure::read_target_from_marker=>fibonacci_node_read_target_from_marker
     procedure::print_to_unit=>fibonacci_node_print_to_unit
     procedure,nopass::get_type=>fibonacci_node_get_type
     procedure::deserialize_from_marker=>fibonacci_node_deserialize_from_marker
     ! overridden measurable_class procedures
     procedure::measure=>fibonacci_node_measure
     ! init/final
     procedure,public ::deallocate_tree=>fibonacci_node_deallocate_tree
     procedure,public ::deallocate_all=>fibonacci_node_deallocate_all
!     interface
     procedure,public ::get_depth=>fibonacci_node_get_depth
     procedure,public ::count_leaves=>fibonacci_node_count_leaves
```

```
! public tests
    procedure,public,nopass ::is_leave=>fibonacci_node_is_leave
    procedure,public,nopass ::is_root=>fibonacci_node_is_root
    procedure,public,nopass ::is_inner=>fibonacci_node_is_inner
! print methods
    procedure,public ::write_association=>fibonacci_node_write_association
    procedure,public ::write_contents=>fibonacci_node_write_contents
    procedure,public ::write_values=>fibonacci_node_write_values
    procedure,public ::write_leaves=>fibonacci_node_write_leaves
    !procedure,public ::write=>fibonacci_node_write_contents
! write methods
    procedure,public ::write_pstricks=>fibonacci_node_write_pstricks
! elaborated functions
    procedure,public ::copy_node=>fibonacci_node_copy_node
    procedure,public ::find_root=>fibonacci_node_find_root
    procedure,public ::find_leftmost=>fibonacci_node_find_leftmost
    procedure,public ::find_rightmost=>fibonacci_node_find_rightmost
    procedure,public ::find=>fibonacci_node_find
    procedure,public ::find_left_leave=>fibonacci_node_find_left_leave
    procedure,public ::find_right_leave=>fibonacci_node_find_right_leave
    procedure,public ::apply_to_leaves=>fibonacci_node_apply_to_leaves
    procedure,public ::apply_to_leaves_rl=>fibonacci_node_apply_to_leaves_rl
! private procedures: these are unsafe!
    procedure ::set_depth=>fibonacci_node_set_depth
    procedure ::append_left=>fibonacci_node_append_left
    procedure ::append_right=>fibonacci_node_append_right
    procedure ::replace=>fibonacci_node_replace
    procedure ::swap=>fibonacci_node_swap_nodes
    procedure ::flip=>fibonacci_node_flip_children
    procedure ::rip=>fibonacci_node_rip
    procedure ::remove_and_keep_parent=>fibonacci_node_remove_and_keep_parent
    procedure ::remove_and_keep_twin=>fibonacci_node_remove_and_keep_twin
    procedure ::rotate_left=>fibonacci_node_rotate_left
    procedure ::rotate_right=>fibonacci_node_rotate_right
    procedure ::rotate=>fibonacci_node_rotate
    procedure ::balance_node=>fibonacci_node_balance_node
    procedure ::update_depth_save=>fibonacci_node_update_depth_save
    procedure ::update_depth_unsave=>fibonacci_node_update_depth_unsave
    procedure ::repair=>fibonacci_node_repair
! tests: these are save when type is fibonacci_node_type and else unsafe.
    procedure ::is_left_short=>fibonacci_node_is_left_short
    procedure ::is_right_short=>fibonacci_node_is_right_short
    procedure ::is_unbalanced=>fibonacci_node_is_unbalanced
    procedure ::is_left_too_short=>fibonacci_node_is_left_too_short
    procedure ::is_right_too_short=>fibonacci_node_is_right_too_short
    procedure ::is_too_unbalanced=>fibonacci_node_is_too_unbalanced
    procedure ::is_left_child=>fibonacci_node_is_left_child
    procedure ::is_right_child=>fibonacci_node_is_right_child
  end type fibonacci_node_type
```

9.3.2 fibonacci_leave_type

```
  type,extends(fibonacci_node_type) :: fibonacci_leave_type
!     class(measurable_class),pointer :: content
  contains
     ! overridden serializable_class procedures
     procedure::print_to_unit=>fibonacci_leave_print_to_unit
     procedure,nopass::get_type=>fibonacci_leave_get_type
     procedure,public ::deallocate_all=>fibonacci_leave_deallocate_all
     ! new procedures
     procedure,public ::pick=>fibonacci_leave_pick
     procedure,public ::get_left=>fibonacci_leave_get_left
     procedure,public ::get_right=>fibonacci_leave_get_right
     procedure,public ::write_pstricks=>fibonacci_leave_write_pstricks
     procedure,public ::copy_content=>fibonacci_leave_copy_content
     procedure,public ::set_content=>fibonacci_leave_set_content
     procedure,public ::get_content=>fibonacci_leave_get_content
     procedure,public,nopass ::is_inner=>fibonacci_leave_is_inner
     procedure,public,nopass ::is_leave=>fibonacci_leave_is_leave
     procedure ::insert_leave_by_node=>fibonacci_leave_insert_leave_by_node
     procedure ::is_left_short=>fibonacci_leave_is_left_short
     procedure ::is_right_short=>fibonacci_leave_is_right_short
     procedure ::is_unbalanced=>fibonacci_leave_is_unbalanced
     procedure ::is_left_too_short=>fibonacci_leave_is_left_too_short
     procedure ::is_right_too_short=>fibonacci_leave_is_right_too_short
     procedure ::is_too_unbalanced=>fibonacci_leave_is_too_unbalanced
  end type fibonacci_leave_type
```

9.3.3 fibonacci_root_type

```
  type,extends(fibonacci_node_type) :: fibonacci_root_type
     logical::is_valid_c=.false.
     class(fibonacci_leave_type),pointer ::leftmost=>null()
     class(fibonacci_leave_type),pointer ::rightmost=>null()
  contains
     ! overridden serializable_class procedures
     procedure::write_to_marker=>fibonacci_root_write_to_marker
     procedure::read_target_from_marker=>fibonacci_root_read_target_from_marker
     procedure::print_to_unit=>fibonacci_root_print_to_unit
     procedure,nopass::get_type=>fibonacci_root_get_type
     ! new procedures
     procedure::get_leftmost=>fibonacci_root_get_leftmost
     procedure::get_rightmost=>fibonacci_root_get_rightmost
! public tests
     procedure,public,nopass ::is_root=>fibonacci_root_is_root
     procedure,public,nopass ::is_inner=>fibonacci_root_is_inner
     procedure,public ::is_valid=>fibonacci_root_is_valid
     procedure,public ::count_leaves=>fibonacci_root_count_leaves
     procedure,public ::write_pstricks=>fibonacci_root_write_pstricks
     procedure,public ::copy_root=>fibonacci_root_copy_root
```

```
    procedure,public ::push_by_content=>fibonacci_root_push_by_content
    procedure,public ::push_by_leave=>fibonacci_root_push_by_leave
    procedure,public ::pop_left=>fibonacci_root_pop_left
    procedure,public ::pop_right=>fibonacci_root_pop_right
    procedure,public ::merge=>fibonacci_root_merge
    procedure,public ::set_leftmost=>fibonacci_root_set_leftmost
    procedure,public ::set_rightmost=>fibonacci_root_set_rightmost
    procedure,public ::init_by_leave=>fibonacci_root_init_by_leave
    procedure,public ::init_by_content=>fibonacci_root_init_by_content
    procedure,public ::reset=>fibonacci_root_reset
    ! init/final
    procedure,public ::deallocate_tree=>fibonacci_root_deallocate_tree
    procedure,public ::deallocate_all=>fibonacci_root_deallocate_all
    procedure ::is_left_child=>fibonacci_root_is_left_child
    procedure ::is_right_child=>fibonacci_root_is_right_child
  end type fibonacci_root_type
```

9.3.4 fibonacci_stub_type

```
   type,extends(fibonacci_root_type) :: fibonacci_stub_type
   contains
    ! overridden serializable_class procedures
    procedure,nopass::get_type=>fibonacci_stub_get_type
    ! overridden fibonacci_root_type procedures
    procedure,public ::push_by_content=>fibonacci_stub_push_by_content
    procedure,public ::push_by_leave=>fibonacci_stub_push_by_leave
    procedure,public ::pop_left=>fibonacci_stub_pop_left
    procedure,public ::pop_right=>fibonacci_stub_pop_right
  end type fibonacci_stub_type
```

9.3.5 fibonacci_leave_list_type

```
  type fibonacci_leave_list_type
    class(fibonacci_leave_type),pointer ::leave=>null()
    class(fibonacci_leave_list_type),pointer :: next => null()
  end type fibonacci_leave_list_type
```

9.4 Implementierung der Prozeduren

9.4.1 Methoden für fibonacci_node_type

Überschriebene serializable_class Methoden

fibonacci_node_write_to_marker ↑

```
 recursive  subroutine fibonacci_node_write_to_marker(this,marker,status)
    class(fibonacci_node_type), intent(in) :: this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
! local variables
    class(serializable_class),pointer::ser
    call marker%mark_begin("fibonacci_node_type")
    ser=>this%left
    call marker%mark_pointer("left",ser)
    ser=>this%right
    call marker%mark_pointer("right",ser)
    ser=>this%xxxx
    call marker%mark_pointer("down",ser)
    call marker%mark_end("fibonacci_node_type")
  end subroutine fibonacci_node_write_to_marker
```

fibonacci_node_read_from_marker ↑

```
  recursive subroutine fibonacci_node_read_from_marker (this,marker,status)
    class(fibonacci_node_type), intent(out) :: this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    print *,"fibonacci_node_read_from_marker: You cannot deserialize a list with this subroutine."
    print *,"Use fibonacci_node_read_target_from_marker instead."
  end subroutine fibonacci_node_read_from_marker
```

fibonacci_node_read_target_from_marker ↑

```
recursive subroutine fibonacci_node_read_target_from_marker(this,marker,status)
    class(fibonacci_node_type),target,intent(out) :: this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
! local variables
    class(serializable_class),pointer::ser
    call marker%pick_begin("fibonacci_node_type",status=status)
    call marker%pick_pointer("left",ser)
    if(status==0)then
       select type(ser)
       class is (fibonacci_node_type)
          this%left=>ser
          this%left%up=>this
       end select
    end if
    call marker%pick_pointer("right",ser)
    if(status==0)then
       select type(ser)
       class is (fibonacci_node_type)
          this%right=>ser
          this%right%up=>this
       end select
    end if
    call marker%pick_pointer("down",ser)
```

```
    if(status==0)then
       select type(ser)
       class is (measurable_class)
          this%xxxx=>ser
       end select
    end if
    call marker%pick_end("fibonacci_node_type",status)
  end subroutine fibonacci_node_read_target_from_marker
```

fibonacci_node_get_type ↑

```
  pure subroutine fibonacci_node_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="fibonacci_node_type")
  end subroutine fibonacci_node_get_type
```

fibonacci_node_deserialize_from_marker ↑

```
  subroutine fibonacci_node_deserialize_from_marker(this,name,marker)
    class(fibonacci_node_type),intent(out)::this
    character(*),intent(in)::name
    class(marker_type),intent(inout)::marker
    class(serializable_class),pointer::ser
    allocate(fibonacci_leave_type::ser)
    call marker%push_reference(ser)
    allocate(fibonacci_node_type::ser)
    call marker%push_reference(ser)
    call serializable_deserialize_from_marker(this,name,marker)
    call marker%pop_reference(ser)
    deallocate(ser)
    call marker%pop_reference(ser)
    deallocate(ser)
  end subroutine fibonacci_node_deserialize_from_marker
```

fibonacci_node_print_to_unit ↑

```
  recursive subroutine fibonacci_node_print_to_unit(this,unit,parents,components,peers)
    class(fibonacci_node_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    class(serializable_class),pointer::ser
    write(unit,'("Components of fibonacci_node_type:")')
    write(unit,'("Depth:   ",I22)')this%depth
    write(unit,'("Value:   ",E23.16)')this%measure()
    ser=>this%up
    call serialize_print_comp_pointer(ser,unit,parents,-one,-one,"Up:     ")
    ser=>this%left
    call serialize_print_peer_pointer(ser,unit,parents,components,peers,"Left:   ")
    ser=>this%right
    call serialize_print_peer_pointer(ser,unit,parents,components,peers,"Right:  ")
  end subroutine fibonacci_node_print_to_unit
```

fibonacci_node_measure ↑

```
  elemental function fibonacci_node_measure(this)
    class(fibonacci_node_type),intent(in)::this
    real(kind=double)::fibonacci_node_measure
    fibonacci_node_measure=this%down%measure()
  end function fibonacci_node_measure
```

! init/final

fibonacci_node_deallocate_tree ↑

```
  recursive subroutine fibonacci_node_deallocate_tree(this)
    class(fibonacci_node_type),intent(inout) :: this
    if (associated(this%left)) then
       call this%left%deallocate_tree()
       deallocate(this%left)
    end if
    if (associated(this%right)) then
       call this%right%deallocate_tree()
       deallocate(this%right)
    end if
    call this%set_depth(0)
  end subroutine fibonacci_node_deallocate_tree
```

fibonacci_node_deallocate_all ↑

```
  recursive subroutine fibonacci_node_deallocate_all(this)
    class(fibonacci_node_type),intent(inout) :: this
    if (associated(this%left)) then
       call this%left%deallocate_all()
       deallocate(this%left)
    end if
    if (associated(this%right)) then
       call this%right%deallocate_all()
       deallocate(this%right)
    end if
    call this%set_depth(0)
  end subroutine fibonacci_node_deallocate_all
```

fibonacci_node_set_depth ↑

```
  subroutine fibonacci_node_set_depth(this,depth)
    class(fibonacci_node_type),intent(inout) :: this
    integer,intent(in) :: depth
    this%depth=depth
  end subroutine fibonacci_node_set_depth
```

fibonacci_node_get_depth ↑

```
  elemental function fibonacci_node_get_depth(this)
    class(fibonacci_node_type),intent(in) :: this
    integer :: fibonacci_node_get_depth
    fibonacci_node_get_depth = this%depth
  end function fibonacci_node_get_depth
```

fibonacci_node_is_leave ↑

```
elemental function fibonacci_node_is_leave()
  logical :: fibonacci_node_is_leave
  fibonacci_node_is_leave = .false.
end function fibonacci_node_is_leave
```

fibonacci_node_is_root ↑

```
elemental function fibonacci_node_is_root()
  logical :: fibonacci_node_is_root
  fibonacci_node_is_root = .false.
end function fibonacci_node_is_root
```

fibonacci_node_is_inner ↑

```
elemental function fibonacci_node_is_inner()
  logical :: fibonacci_node_is_inner
  fibonacci_node_is_inner = .true.
end function fibonacci_node_is_inner
```

fibonacci_node_write_leaves ↑

```
subroutine fibonacci_node_write_leaves(this,unit)
  class(fibonacci_node_type),intent(in),target :: this
  integer,intent(in),optional :: unit
  call this%apply_to_leaves(fibonacci_leave_write,unit)
end subroutine fibonacci_node_write_leaves
```

fibonacci_node_write_contents ↑

```
subroutine fibonacci_node_write_contents(this,unit)
  class(fibonacci_node_type),intent(in),target :: this
  integer,intent(in),optional :: unit
  call this%apply_to_leaves(fibonacci_leave_write_content,unit)
end subroutine fibonacci_node_write_contents
```

fibonacci_node_write_values ↑

```
subroutine fibonacci_node_write_values(this,unit)
  class(fibonacci_node_type),intent(in),target :: this
  integer,intent(in),optional :: unit
  call this%apply_to_leaves(fibonacci_leave_write_value,unit)
end subroutine fibonacci_node_write_values
```

fibonacci_node_write_association ↑

```
subroutine fibonacci_node_write_association(this,that)
  class(fibonacci_node_type),intent(in),target :: this
  class(fibonacci_node_type),intent(in),target :: that
  if (associated(that%left,this)) then
     write(*,'("this is left child of that")')
  end if
  if (associated(that%right,this)) then
```

```
      write(*,'("this is right child of that")')
   end if
   if (associated(that%up,this)) then
      write(*,'("this is parent of that")')
   end if
   if (associated(this%left,that)) then
      write(*,'("that is left child of this")')
   end if
   if (associated(this%right,that)) then
      write(*,'("that is right child of this")')
   end if
   if (associated(this%up,that)) then
      write(*,'("that is parent of this")')
   end if
 end subroutine fibonacci_node_write_association
```

fibonacci_node_write_pstricks ↑

```
 recursive subroutine fibonacci_node_write_pstricks(this,unitnr)
   class(fibonacci_node_type),intent(in),target :: this
   integer,intent(in) :: unitnr
   if (associated(this%up)) then
      if (associated(this%up%left,this).neqv.(associated(this%up%right,this))) then
         write(unitnr,'("\beginpsTree\Toval\node",i3,"",f9.3,"")')&
           int(this%depth),this%measure()
      else
         write(unitnr,'("\beginpsTree\Toval[",a,"]\node",i3,"",f9.3,"")')&
           no_ret,int(this%depth),this%measure()
      end if
   else
      write(unitnr,'("\beginpsTree\Toval[",a,"]\node",i3,"",f9.3,"")')&
        no_par,int(this%depth),this%measure()
   end if
   if (associated(this%left)) then
      call this%left%write_pstricks(unitnr)
   else
      write(unitnr,'("\Tr[edge=brokenline]")')
   end if
   if (associated(this%right)) then
      call this%right%write_pstricks(unitnr)
   else
      write(unitnr,'("\Tr[edge=brokenline]")')
   end if
   write(unitnr,'("\endpsTree")')
 end subroutine fibonacci_node_write_pstricks
```

fibonacci_node_copy_node ↑

```
 subroutine fibonacci_node_copy_node(this,primitive)
   class(fibonacci_node_type),intent(out) :: this
   class(fibonacci_node_type),intent(in) :: primitive
   this%up => primitive%up
```

```
    this%left => primitive%left
    this%right => primitive%right
    this%depth = primitive%depth
    this%down=> primitive%down
  end subroutine fibonacci_node_copy_node
```

fibonacci_node_find_root ↑

```
  subroutine fibonacci_node_find_root(this,root)
    class(fibonacci_node_type),intent(in),target  :: this
    class(fibonacci_root_type),pointer,intent(out) :: root
    class(fibonacci_node_type),pointer :: node
    node=>this
    do while(associated(node%up))
       node=>node%up
    end do
    select type (node)
    class is (fibonacci_root_type)
       root=>node
    class default
       nullify(root)
       print *,"fibonacci_node_find_root: root is not type compatible to&
       & fibonacci_root_type. Retured NULL()."
    end select
  end subroutine fibonacci_node_find_root
```

fibonacci_node_find_leftmost ↑

```
  subroutine fibonacci_node_find_leftmost(this,leave)
    class(fibonacci_node_type),intent(in), target  :: this
    class(fibonacci_leave_type),pointer,intent(out) :: leave
    class(fibonacci_node_type), pointer :: node
    node=>this
    do while(associated(node%left))
       node=>node%left
    end do
    select type (node)
    class is (fibonacci_leave_type)
       leave => node
    class default
       leave => null()
    end select
  end subroutine fibonacci_node_find_leftmost
```

fibonacci_node_find_rightmost ↑

```
  subroutine fibonacci_node_find_rightmost(this,leave)
    class(fibonacci_node_type),intent(in), target  :: this
    class(fibonacci_leave_type),pointer,intent(out) :: leave
    class(fibonacci_node_type), pointer :: node
    node=>this
    do while(associated(node%right))
```

```
      node=>node%right
   end do
   select type (node)
   class is (fibonacci_leave_type)
      leave => node
   class default
      leave => null()
   end select
 end subroutine fibonacci_node_find_rightmost
```

fibonacci_node_find ↑

```
 subroutine fibonacci_node_find(this,value,leave)
   class(fibonacci_node_type),intent(in),target  :: this
   real(kind=double),intent(in) :: value
   class(fibonacci_leave_type),pointer,intent(out) :: leave
   class(fibonacci_node_type), pointer :: node
   node=>this
   do
      if (node>=value) then
         if (associated(node%left)) then
            node=>node%left
         else
            print *,"fibonacci_node_find: broken tree!"
            leave => null()
            return
         end if
      else
         if (associated(node%right)) then
            node=>node%right
         else
            print *,"fibonacci_node_find: broken tree!"
            leave => null()
            return
         end if
      end if
      select type (node)
      class is (fibonacci_leave_type)
         leave => node
         exit
      end select
   end do
 end subroutine fibonacci_node_find
```

fibonacci_node_find_left_leave ↑

```
 subroutine fibonacci_node_find_left_leave(this,leave)
   class(fibonacci_node_type),intent(in),target  :: this
   class(fibonacci_node_type),pointer  :: node
   class(fibonacci_leave_type),pointer,intent(out)  :: leave
   nullify(leave)
   node=>this
```

```
    do while (associated(node%up))
       if (associated(node%up%right,node)) then
          node=>node%up%left
          do while (associated(node%right))
             node=>node%right
          end do
          select type (node)
          class is (fibonacci_leave_type)
          leave=>node
          end select
          exit
       end if
       node=>node%up
    end do
  end subroutine fibonacci_node_find_left_leave
```

fibonacci_node_find_right_leave ↑

```
  subroutine fibonacci_node_find_right_leave(this,leave)
    class(fibonacci_node_type),intent(in),target  :: this
    class(fibonacci_node_type),pointer  :: node
    class(fibonacci_leave_type),pointer,intent(out)  :: leave
    nullify(leave)
    node=>this
    do while (associated(node%up))
       if (associated(node%up%left,node)) then
          node=>node%up%right
          do while (associated(node%left))
             node=>node%left
          end do
          select type (node)
          class is (fibonacci_leave_type)
          leave=>node
          end select
          exit
       end if
       node=>node%up
    end do
  end subroutine fibonacci_node_find_right_leave
```

fibonacci_node_replace ↑

```
  subroutine fibonacci_node_replace(this,old_node)
    class(fibonacci_node_type),intent(inout),target :: this
    class(fibonacci_node_type),target :: old_node
    if (associated(old_node%up)) then
       if (old_node%is_left_child()) then
          old_node%up%left => this
       else
          if (old_node%is_right_child()) then
             old_node%up%right => this
          end if
```

```
      end if
      this%up => old_node%up
   else
      nullify(this%up)
   end if
 end subroutine fibonacci_node_replace
```

fibonacci_node_swap_nodes ↑

```
 subroutine fibonacci_node_swap_nodes(left,right)
   class(fibonacci_node_type),target,intent(inout) :: left,right
   class(fibonacci_node_type),pointer :: left_left,right_right
   class(measurable_class),pointer::down
   ! swap branches
   left_left   =>left%left
   right_right=>right%right
   left%left   =>right%right
   right%right=>left_left
   ! repair up components
   right_right%up=>left
   left_left%up   =>right
   ! repair down components
         down =>  left%down
    left%down => right%down
   right%down =>      down
 end subroutine fibonacci_node_swap_nodes
```

fibonacci_node_swap_nodes ↑

```
!  subroutine fibonacci_node_swap_nodes(this,that)
!    class(fibonacci_node_type),target :: this
!    class(fibonacci_node_type),pointer,intent(in) :: that
!    class(fibonacci_node_type),pointer :: par_i,par_a
!    par_i => this%up
!    par_a => that%up
!    if (associated(par_i%left,this)) then
!       par_i%left => that
!    else
!       par_i%right => that
!    end if
!    if (associated(par_a%left,that)) then
!       par_a%left => this
!    else
!       par_a%right => this
!    end if
!    this%up => par_a
!    that%up => par_i
!  end subroutine fibonacci_node_swap_nodes
```

fibonacci_node_flip_children ↑

```
subroutine fibonacci_node_flip_children(this)
  class(fibonacci_node_type),intent(inout) :: this
  class(fibonacci_node_type),pointer :: child
  child => this%left
  this%left=>this%right
  this%right => child
end subroutine fibonacci_node_flip_children
```

fibonacci_node_rip ↑

```
subroutine fibonacci_node_rip(this)
  class(fibonacci_node_type),intent(inout),target :: this
  if (this%is_left_child()) then
     nullify(this%up%left)
  end if
  if (this%is_right_child()) then
     nullify(this%up%right)
  end if
  nullify(this%up)
end subroutine fibonacci_node_rip
```

fibonacci_node_remove_and_keep_twin ↑

```
subroutine fibonacci_node_remove_and_keep_twin(this,twin)
  class(fibonacci_node_type),intent(inout),target  :: this
  class(fibonacci_node_type),intent(out),pointer :: twin
  class(fibonacci_node_type),pointer :: pa
  if (.not. (this%is_root())) then
     pa=>this%up
     if (.not. pa%is_root()) then
        if (this%is_left_child()) then
           twin => pa%right
        else
           twin => pa%left
        end if
        if (pa%is_left_child()) then
           pa%up%left => twin
        else
           pa%up%right => twin
        end if
     end if
     twin%up => pa%up
     if(associated(this%right))then
        this%right%left=>this%left
     end if
     if(associated(this%left))then
        this%left%right=>this%right
     end if
     nullify(this%left)
     nullify(this%right)
     nullify(this%up)
     deallocate(pa)
```

```
    end if
  end subroutine fibonacci_node_remove_and_keep_twin
```

fibonacci_node_remove_and_keep_parent ↑

```
  subroutine fibonacci_node_remove_and_keep_parent(this,pa)
    class(fibonacci_node_type),intent(inout),target  :: this
    class(fibonacci_node_type),intent(out),pointer :: pa
    class(fibonacci_node_type),pointer :: twin
    if (.not. (this%is_root())) then
       pa=>this%up
       if (this%is_left_child()) then
          twin => pa%right
       else
          twin => pa%left
       end if
       twin%up=>pa%up
       if (associated(twin%left)) then
          twin%left%up => pa
       end if
       if (associated(twin%right)) then
          twin%right%up => pa
       end if
       call pa%copy_node(twin)
       select type(pa)
       class is (fibonacci_root_type)
          call pa%set_leftmost()
          call pa%set_rightmost()
       end select
       if(associated(this%right))then
          this%right%left=>this%left
       end if
       if(associated(this%left))then
          this%left%right=>this%right
       end if
       nullify(this%left)
       nullify(this%right)
       nullify(this%up)
       deallocate(twin)
    else
       pa=>this
    end if
  end subroutine fibonacci_node_remove_and_keep_parent
```

fibonacci_leave_pick ↑

```
  subroutine fibonacci_leave_pick(this)
    class(fibonacci_leave_type),target,intent(inout) :: this
    class(fibonacci_node_type),pointer :: other
    class(fibonacci_root_type),pointer :: root
!     call this%up%print_parents()
    call this%find_root(root)
```

```
    if(associated(this%up,root))then
       if(this%up%depth<2)then
          print *,"fibonacci_leave_pick: Cannot pick leave. &
          &Tree must have at least three leaves."
          return
       else
          call this%remove_and_keep_parent(other)
          call other%repair()
       end if
    else
       call this%remove_and_keep_twin(other)
       call other%up%repair()
    end if
    if(associated(root%leftmost,this))call root%set_leftmost()
    if(associated(root%rightmost,this))call root%set_rightmost()
  end subroutine fibonacci_leave_pick
```

fibonacci_node_append_left ↑

```
  subroutine fibonacci_node_append_left(this,new_branch)
    class(fibonacci_node_type),target :: this
    class(fibonacci_node_type),target :: new_branch
    this%left => new_branch
    new_branch%up => this
  end subroutine fibonacci_node_append_left
```

fibonacci_node_append_right ↑

```
  subroutine fibonacci_node_append_right(this,new_branch)
    class(fibonacci_node_type),intent(inout),target :: this
    class(fibonacci_node_type),target :: new_branch
    this%right => new_branch
    new_branch%up => this
  end subroutine fibonacci_node_append_right
```

fibonacci_node_rotate_left ↑

```
  subroutine fibonacci_node_rotate_left(this)
    class(fibonacci_node_type),intent(inout),target  :: this
    call this%swap(this%right)
    call this%right%flip()
    call this%right%update_depth_unsave()
    call this%flip()
!    value = this%value
!    this%value = this%left%value
!    this%left%value = value
  end subroutine fibonacci_node_rotate_left
```

fibonacci_node_rotate_right ↑

```
  subroutine fibonacci_node_rotate_right(this)
    class(fibonacci_node_type),intent(inout),target  :: this
    call this%left%swap(this)
```

```
    call this%left%flip()
    call this%left%update_depth_unsave()
    call this%flip()
!     value = this%value
!     this%value = this%right%value
!     this%right%value = value
  end subroutine fibonacci_node_rotate_right
```

fibonacci_node_rotate ↑

```
  subroutine fibonacci_node_rotate(this)
    class(fibonacci_node_type),intent(inout),target  :: this
    if (this%is_left_short()) then
       call this%rotate_left()
    else
       if (this%is_right_short()) then
          call this%rotate_right()
       end if
    end if
  end subroutine fibonacci_node_rotate
```

fibonacci_node_balance_node ↑

```
  subroutine fibonacci_node_balance_node(this,changed)
    class(fibonacci_node_type),intent(inout),target  :: this
    logical,intent(out) :: changed
    changed=.false.
    if (this%is_left_too_short()) then
       if (this%right%is_right_short()) then
          call this%right%rotate_right
       end if
       call this%rotate_left()
       changed=.true.
    else
       if (this%is_right_too_short()) then
          if (this%left%is_left_short()) then
             call this%left%rotate_left
          end if
          call this%rotate_right()
          changed=.true.
       end if
    end if
  end subroutine fibonacci_node_balance_node
```

fibonacci_node_update_depth_unsave ↑

```
  subroutine fibonacci_node_update_depth_unsave(this)
    class(fibonacci_node_type),intent(inout)  :: this
    this%depth=max(this%left%depth+1,this%right%depth+1)
  end subroutine fibonacci_node_update_depth_unsave
```

fibonacci_node_update_depth_save ↑

```
subroutine fibonacci_node_update_depth_save(this,updated)
  class(fibonacci_node_type),intent(inout)  :: this
  logical,intent(out) :: updated
  integer :: left,right,new_depth
  if (associated(this%left)) then
     left=this%left%depth+1
  else
     left=-1
  end if
  if (associated(this%right)) then
     right=this%right%depth+1
  else
     right=-1
  end if
  new_depth=max(left,right)
  if (this%depth == new_depth) then
     updated = .false.
  else
     this%depth=new_depth
     updated = .true.
  end if
end subroutine fibonacci_node_update_depth_save
```

fibonacci_node_repair ↑

```
subroutine fibonacci_node_repair(this)
  class(fibonacci_node_type),intent(inout),target :: this
  class(fibonacci_node_type),pointer:: node
  logical :: new_depth,new_balance
  new_depth = .true.
  node=>this
  do while((new_depth .or. new_balance) .and. (associated(node)))
     call node%balance_node(new_balance)
     call node%update_depth_save(new_depth)
     node=>node%up
  end do
end subroutine fibonacci_node_repair
```

fibonacci_node_is_left_short ↑

```
elemental logical function fibonacci_node_is_left_short(this)
  class(fibonacci_node_type),intent(in) :: this
  fibonacci_node_is_left_short = (this%left%depth<this%right%depth)
end function fibonacci_node_is_left_short
```

fibonacci_node_is_right_short ↑

```
elemental logical function fibonacci_node_is_right_short(this)
  class(fibonacci_node_type),intent(in) :: this
  fibonacci_node_is_right_short = (this%right%depth<this%left%depth)
end function fibonacci_node_is_right_short
```

fibonacci_node_is_unbalanced ↑

```
elemental logical function fibonacci_node_is_unbalanced(this)
  class(fibonacci_node_type),intent(in) :: this
  fibonacci_node_is_unbalanced = (this%is_left_short() .or. this%is_right_short())
end function fibonacci_node_is_unbalanced
```

fibonacci_node_is_left_too_short ↑

```
elemental logical function fibonacci_node_is_left_too_short(this)
  class(fibonacci_node_type),intent(in) :: this
  fibonacci_node_is_left_too_short = (this%left%depth+1<this%right%depth)
end function fibonacci_node_is_left_too_short
```

fibonacci_node_is_right_too_short ↑

```
elemental logical function fibonacci_node_is_right_too_short(this)
  class(fibonacci_node_type),intent(in) :: this
  fibonacci_node_is_right_too_short = (this%right%depth+1<this%left%depth)
end function fibonacci_node_is_right_too_short
```

fibonacci_node_is_too_unbalanced ↑

```
elemental logical function fibonacci_node_is_too_unbalanced(this)
  class(fibonacci_node_type),intent(in) :: this
  fibonacci_node_is_too_unbalanced = (this%is_left_too_short() .or. this%is_right_too_short())
end function fibonacci_node_is_too_unbalanced
```

fibonacci_node_is_left_child ↑

```
elemental logical function fibonacci_node_is_left_child(this)
  class(fibonacci_node_type),intent(in),target :: this
  fibonacci_node_is_left_child = associated(this%up%left,this)
end function fibonacci_node_is_left_child
```

fibonacci_node_is_right_child ↑

```
elemental logical function fibonacci_node_is_right_child(this)
  class(fibonacci_node_type),intent(in),target :: this
  fibonacci_node_is_right_child = associated(this%up%right,this)
end function fibonacci_node_is_right_child
```

fibonacci_node_apply_to_leaves ↑

```
recursive subroutine fibonacci_node_apply_to_leaves(node,func,unit)
  class(fibonacci_node_type),intent(in),target :: node
  interface
     subroutine func(this,unit)
       import fibonacci_leave_type
       class(fibonacci_leave_type),intent(in),target :: this
       integer,intent(in),optional :: unit
     end subroutine func
  end interface
  integer,intent(in),optional :: unit
  select type (node)
  class is (fibonacci_leave_type)
```

```
      call func(node,unit)
   class default
      call node%left%apply_to_leaves(func,unit)
      call node%right%apply_to_leaves(func,unit)
   end select
 end subroutine fibonacci_node_apply_to_leaves
```

fibonacci_node_apply_to_leaves_rl ↑

```
 recursive subroutine fibonacci_node_apply_to_leaves_RL(node,func,unit)
   class(fibonacci_node_type),intent(in),target :: node
   interface
      subroutine func(this,unit)
        import fibonacci_leave_type
        class(fibonacci_leave_type),intent(in),target :: this
        integer,intent(in),optional :: unit
      end subroutine func
   end interface
   integer,intent(in),optional :: unit
   select type (node)
   class is (fibonacci_leave_type)
      call func(node,unit)
   class default
      call node%right%apply_to_leaves_rl(func,unit)
      call node%left%apply_to_leaves_rl(func,unit)
   end select
 end subroutine fibonacci_node_apply_to_leaves_RL
```

fibonacci_node_count_leaves ↑

```
 recursive subroutine fibonacci_node_count_leaves(this,n)
   class(fibonacci_node_type),intent(in) :: this
   integer,intent(out) :: n
   integer::n1,n2
   if(associated(this%left).and.associated(this%right)) then
      call fibonacci_node_count_leaves(this%left,n1)
      call fibonacci_node_count_leaves(this%right,n2)
      n=n1+n2
   else
      n=1
   end if
 end subroutine fibonacci_node_count_leaves
```

9.4.2 Methoden für fibonacci_root_type

fibonacci_root_write_to_marker ↑

```
 SUBROUTINE fibonacci_root_write_to_marker(this,marker,status)
   CLASS(fibonacci_root_type), INTENT(IN) :: this
   class(marker_type),intent(inout)::marker
   integer(kind=dik),intent(out)::status
```

```
!    call marker%mark_begin("FIBONACCI_ROOT_TYPE")
    call fibonacci_node_write_to_marker(this,marker,status)
!    marker%mark_end("FIBONACCI_ROOT_TYPE")
  end SUBROUTINE fibonacci_root_write_to_marker
```

fibonacci_root_read_target_from_marker ↑

```
  SUBROUTINE fibonacci_root_read_target_from_marker(this,marker,status)
    CLASS(fibonacci_root_type),target,INTENT(out) :: this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
!    call marker%pick_begin("FIBONACCI_ROOT_TYPE",status)
    call fibonacci_node_read_from_marker(this,marker,status)
    call this%find_leftmost(this%leftmost)
    call this%find_rightmost(this%rightmost)
!    call marker%pick_end("FIBONACCI_ROOT_TYPE",status)
  end SUBROUTINE fibonacci_root_read_target_from_marker
```

fibonacci_root_print_to_unit ↑

```
  subroutine fibonacci_root_print_to_unit(this,unit,parents,components,peers)
    class(fibonacci_root_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    class(serializable_class),pointer::ser
    if(parents>0)call fibonacci_node_print_to_unit(this,unit,parents-1,components,peers)
    write(unit,'("Components of fibonacci_root_type:")')
    ser=>this%leftmost
    call serialize_print_peer_pointer(ser,unit,parents,components,min(peers,one),"Leftmost: ")
    ser=>this%rightmost
    call serialize_print_peer_pointer(ser,unit,parents,components,min(peers,one),"Rightmost:")
  end subroutine fibonacci_root_print_to_unit
```

fibonacci_root_get_type ↑

```
  pure subroutine fibonacci_root_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="fibonacci_root_type")
  end subroutine fibonacci_root_get_type
```

fibonacci_root_get_leftmost ↑

```
  subroutine fibonacci_root_get_leftmost(this,leftmost)
    class(fibonacci_root_type),intent(in)::this
    class(fibonacci_leave_type),pointer::leftmost
    leftmost=>this%leftmost
  end subroutine fibonacci_root_get_leftmost
```

fibonacci_root_get_rightmost ↑

```
  subroutine fibonacci_root_get_rightmost(this,rightmost)
    class(fibonacci_root_type),intent(in)::this
    class(fibonacci_leave_type),pointer::rightmost
```

```
    rightmost=>this%rightmost
  end subroutine fibonacci_root_get_rightmost
```

fibonacci_root_is_inner ↑

```
  elemental function fibonacci_root_is_inner()
    logical::fibonacci_root_is_inner
    fibonacci_root_is_inner=.false.
  end function fibonacci_root_is_inner
```

fibonacci_root_is_root ↑

```
  elemental function fibonacci_root_is_root()
    logical::fibonacci_root_is_root
    fibonacci_root_is_root=.true.
  end function fibonacci_root_is_root
```

fibonacci_root_is_valid ↑

```
  elemental function fibonacci_root_is_valid(this)
    class(fibonacci_root_type),intent(in) :: this
    logical :: fibonacci_root_is_valid
    fibonacci_root_is_valid=this%is_valid_c
  end function fibonacci_root_is_valid
```

fibonacci_root_count_leaves ↑

```
  subroutine fibonacci_root_count_leaves(this,n)
    class(fibonacci_root_type),intent(in) :: this
    integer,intent(out) :: n
    n=0
    call fibonacci_node_count_leaves(this,n)
  end subroutine fibonacci_root_count_leaves
```

fibonacci_root_write_pstricks ↑

```
  subroutine fibonacci_root_write_pstricks(this,unitnr)
    class(fibonacci_root_type),intent(in),target :: this
    integer,intent(in) :: unitnr
    logical :: is_opened
    character :: is_sequential,is_formatted,is_writeable
    print *,"pstricks"
    inquire(unitnr,opened=is_opened,&
         &sequential=is_sequential,formatted=is_formatted,write=is_writeable)
    if (is_opened) then
       if (is_sequential=="Y" .and. is_formatted=="Y" .and. is_writeable=="Y") then
          write(unitnr,'("\beginpsTree\Toval[linecolor=blue]\node",i3,"",f9.3,"")')&
            this%depth,this%measure()
          if (associated(this%leftmost)) then
             call this%leftmost%write_pstricks(unitnr)
          else
             write(unitnr,'("\Tr[",a,"]")') no_kid
          end if
```

```
        if (associated(this%left)) then
           call this%left%write_pstricks(unitnr)
        else
           write(unitnr,'("\Tr[",a,"]")') no_kid
        end if
        if (associated(this%right)) then
           call this%right%write_pstricks(unitnr)
        else
           write(unitnr,'("\Tr[",a,"]")') no_kid
        end if
        if (associated(this%rightmost)) then
           call this%rightmost%write_pstricks(unitnr)
        else
           write(unitnr,'("\Tr[",a,"]")') no_kid
        end if
        write(unitnr,'("\endpsTree")')
        write(unitnr,'("\\")')
     else
        print '("fibonacci_node_write_pstricks: Unit ",I2," is not opened properly.")',unitnr
        print '("No output is written to unit.")'
     end if
  else
     print '("fibonacci_node_write_pstricks: Unit ",I2," is not opened.")',unitnr
     print '("No output is written to unit.")'
  end if
end subroutine fibonacci_root_write_pstricks
```

fibonacci_root_copy_root ↑

```
subroutine fibonacci_root_copy_root(this,primitive)
  class(fibonacci_root_type),intent(out) :: this
  class(fibonacci_root_type),intent(in) :: primitive
  call fibonacci_node_copy_node(this,primitive)
  this%leftmost => primitive%leftmost
  this%rightmost => primitive%rightmost
end subroutine fibonacci_root_copy_root
```

fibonacci_root_push_by_content ↑

```
  subroutine fibonacci_root_push_by_content(this,content)
    class(fibonacci_root_type),target,intent(inout)  :: this
    class(measurable_class),target,intent(in)::content
    class(fibonacci_leave_type),pointer :: node
!     print *,"fibonacci_root_push_by_content: ",content%measure()
    allocate(node)
    node%down=>content
    call this%push_by_leave(node)
  end subroutine fibonacci_root_push_by_content
```

fibonacci_root_push_by_leave ↑

```
! this is a workaround for BUG 44696. This subroutine is a merge of
! fibonacci_tree_push_by_node
! fibonacci_node_find
! fibonacci_leave_insert_leave_by_node
subroutine fibonacci_root_push_by_leave(this,new_leave)
  class(fibonacci_root_type),target,intent(inout)   :: this
  class(fibonacci_leave_type),pointer,intent(inout) :: new_leave
  class(fibonacci_leave_type),pointer :: old_leave
  class(fibonacci_node_type), pointer :: node,new_node,leave_c
  if (new_leave<=this%leftmost) then
     old_leave=>this%leftmost
     this%leftmost=>new_leave
     node=>old_leave%up
     call fibonacci_node_spawn&
       (new_node,new_leave,old_leave,old_leave%left,old_leave%right)
     call node%append_left(new_node)
  else
     if (new_leave>this%rightmost) then
        old_leave=>this%rightmost
        this%rightmost=>new_leave
        node=>old_leave%up
        call fibonacci_node_spawn&
          (new_node,old_leave,new_leave,old_leave%left,old_leave%right)
        call node%append_right(new_node)
     else
        node=>this
        do
           if (new_leave<=node) then
              leave_c=>node%left
              select type (leave_c)
              class is (fibonacci_leave_type)
                 if(new_leave<=leave_c)then
                    call fibonacci_node_spawn&
                      (new_node,new_leave,leave_c,leave_c%left,leave_c%right)
                 else
                    call fibonacci_node_spawn&
                      (new_node,leave_c,new_leave,leave_c%left,leave_c%right)
                 end if
                 call node%append_left(new_node)
                 exit
              class default
                 node=>node%left
              end select
           else
              leave_c=>node%right
              select type (leave_c)
              class is (fibonacci_leave_type)
                 if(new_leave<=leave_c)then
                    call fibonacci_node_spawn&
                      (new_node,new_leave,leave_c,leave_c%left,leave_c%right)
                 else
```

```
                  call fibonacci_node_spawn&
                    (new_node,leave_c,new_leave,leave_c%left,leave_c%right)
               end if
               call node%append_right(new_node)
               exit
            class default
               node=>node%right
            end select
         end if
      end do
   end if
 end if
 call node%repair()
end subroutine fibonacci_root_push_by_leave
```

fibonacci_root_pop_left ↑

```
subroutine fibonacci_root_pop_left(this,leave)
  class(fibonacci_root_type),intent(inout),target  :: this
  class(fibonacci_leave_type),pointer,intent(out) :: leave
  class(fibonacci_node_type),pointer  :: parent,grand
  !write(11,fmt=*)"fibonacci root pop left                \"!PSTRICKS
  !flush(11)!PSTRICKS
  leave => this%leftmost
  if (this%left%depth>=1) then
     parent => leave%up
     grand=>parent%up
     grand%left => parent%right
     parent%right%up=>grand
     deallocate(parent)
     parent=>grand%left
     if (.not.parent%is_leave())then
        parent=>parent%left
     end if
     select type (parent)
     class is (fibonacci_leave_type)
        this%leftmost => parent
     class default
        print *,"fibonacci_root_pop_left: ERROR: leftmost is no leave."
        call parent%print_all()
        STOP
     end select
     !call this%write_pstricks(11)!PSTRICKS
     !flush(11)!PSTRICKS
     !write(11,fmt=*)"fibonacci node repair              \"!PSTRICKS
     !flush(11)!PSTRICKS
     call grand%repair()
  else
     if (this%left%depth==0.and.this%right%depth==1) then
        parent => this%right
        parent%right%up=>this
```

```
        parent%left%up=>this
        this%left=>parent%left
        this%right=>parent%right
        this%depth=1
        deallocate(parent)
        parent=>this%left
        select type (parent)
        class is (fibonacci_leave_type)
        this%leftmost => parent
        end select
        this%down=>this%leftmost%down
     end if
  end if
  nullify(leave%right%left)
  nullify(leave%up)
  nullify(leave%right)
  nullify(this%leftmost%left)
  !call this%write_pstricks(11)!PSTRICKS
  !flush(11)!PSTRICKS
end subroutine fibonacci_root_pop_left
```

fibonacci_root_pop_right ↑

```
subroutine fibonacci_root_pop_right(this,leave)
  class(fibonacci_root_type),intent(inout),target  :: this
  class(fibonacci_leave_type),pointer,intent(out) :: leave
  class(fibonacci_node_type),pointer  :: parent,grand
  leave => this%rightmost
  if (this%right%depth>=1) then
     parent => leave%up
     grand=>parent%up
     grand%right => parent%left
     parent%left%up=>grand
     deallocate(parent)
     parent=>grand%right
     if (.not.parent%is_leave())then
        parent=>parent%right
     end if
     select type (parent)
     class is (fibonacci_leave_type)
        this%rightmost => parent
     class default
        print *,"fibonacci_root_pop_left: ERROR: leftmost is no leave."
        call parent%print_all()
        STOP
     end select
     call grand%repair()
  else
     if (this%right%depth==0.and.this%left%depth==1) then
        parent => this%left
        parent%left%up=>this
```

```
        parent%right%up=>this
        this%right=>parent%right
        this%left=>parent%left
        this%depth=1
        deallocate(parent)
        parent=>this%right
        select type (parent)
        class is (fibonacci_leave_type)
        this%rightmost => parent
        end select
        this%down=>this%rightmost%down
      end if
    end if
  end subroutine fibonacci_root_pop_right
```

fibonacci_root_merge ↑

```
  subroutine fibonacci_root_merge(this_tree,that_tree,merge_tree)
    ! I neither used nor revised this procedure for a long time, so it might be broken.
    class(fibonacci_root_type),intent(in) :: this_tree
    class(fibonacci_root_type),intent(in) :: that_tree
    class(fibonacci_root_type),pointer,intent(out) :: merge_tree
    class(fibonacci_leave_type),pointer :: this_leave,that_leave,old_leave
    type(fibonacci_leave_list_type),target :: leave_list
    class(fibonacci_leave_list_type),pointer :: last_leave
    integer :: n_leaves
    if (associated(this_tree%leftmost).and.associated(that_tree%leftmost)) then
      n_leaves=1
      this_leave=>this_tree%leftmost
      that_leave=>that_tree%leftmost
      if (this_leave < that_leave) then
        allocate(leave_list%leave,source=this_leave)
        call this_leave%find_right_leave(this_leave)
      else
        allocate(leave_list%leave,source=that_leave)
        call that_leave%find_right_leave(that_leave)
      end if
      last_leave=>leave_list
      do while (associated(this_leave).and.associated(that_leave))
        if (this_leave < that_leave) then
          old_leave=>this_leave
          call this_leave%find_right_leave(this_leave)
        else
          old_leave=>that_leave
          call that_leave%find_right_leave(that_leave)
        end if
        allocate(last_leave%next)
        last_leave=>last_leave%next
        allocate(last_leave%leave,source=old_leave)
        n_leaves=n_leaves+1
      end do
```

```
      if (associated(this_leave)) then
         old_leave=>this_leave
      else
         old_leave=>that_leave
      end if
      do while (associated(old_leave))
         allocate(last_leave%next)
         last_leave=>last_leave%next
         allocate(last_leave%leave,source=old_leave)
         n_leaves=n_leaves+1
         call old_leave%find_right_leave(old_leave)
      end do
      allocate(merge_tree)
      call fibonacci_root_list_to_tree(merge_tree,n_leaves,leave_list)
   else
      n_leaves=0
   end if
   if(associated(leave_list%next)) then
      last_leave=>leave_list%next
      do while (associated(last_leave%next))
         leave_list%next=>last_leave%next
         deallocate(last_leave)
         last_leave=>leave_list%next
      end do
      deallocate(last_leave)
   end if
 end subroutine fibonacci_root_merge
```

fibonacci_root_set_leftmost ↑

```
 subroutine fibonacci_root_set_leftmost(this)
   class(fibonacci_root_type) :: this
   call this%find_leftmost(this%leftmost)
 end subroutine fibonacci_root_set_leftmost
```

fibonacci_root_set_rightmost ↑

```
 subroutine fibonacci_root_set_rightmost(this)
   class(fibonacci_root_type) :: this
   call this%find_rightmost(this%rightmost)
 end subroutine fibonacci_root_set_rightmost
```

fibonacci_root_init_by_leave ↑

```
 subroutine fibonacci_root_init_by_leave(this,left_leave,right_leave)
   class(fibonacci_root_type),target,intent(out) :: this
   class(fibonacci_leave_type),target,intent(in) :: left_leave,right_leave
   if (left_leave <= right_leave) then
      this%left  =>  left_leave
      this%right => right_leave
      this%leftmost => left_leave
      this%rightmost => right_leave
```

```
    else
       this%left  => right_leave
       this%right  => left_leave
       this%leftmost => right_leave
       this%rightmost => left_leave
    end if
    this%left%up => this
    this%right%up => this
    this%down=>this%leftmost%down
    this%depth = 1
    this%leftmost%right=>this%rightmost
    this%rightmost%left=>this%leftmost
    this%is_valid_c=.true.
  end subroutine fibonacci_root_init_by_leave
```

fibonacci_root_init_by_content ↑

```
  subroutine fibonacci_root_init_by_content(this,left_content,right_content)
    class(fibonacci_root_type),target,intent(out) :: this
    class(measurable_class),intent(in),target :: left_content,right_content
    call fibonacci_root_reset(this)
    print *,"fibonacci_root_init_by_content: ",left_content%measure(),right_content%measure()
    if (left_content<right_content) then
       call this%leftmost%set_content(left_content)
       call this%rightmost%set_content(right_content)
    else
       call this%leftmost%set_content(right_content)
       call this%rightmost%set_content(left_content)
    end if
    this%down=>this%leftmost%down
    this%is_valid_c=.true.
  end subroutine fibonacci_root_init_by_content
```

fibonacci_root_reset ↑

```
  subroutine fibonacci_root_reset(this)
    class(fibonacci_root_type),target,intent(inout) :: this
    call fibonacci_root_deallocate_tree(this)
    allocate (this%leftmost)
    allocate (this%rightmost)
    this%depth=1
    this%leftmost%depth=0
    this%rightmost%depth=0
    this%left=>this%leftmost
    this%right=>this%rightmost
    this%left%up=>this
    this%right%up=>this
    this%leftmost%right=>this%rightmost
    this%rightmost%left=>this%leftmost
  end subroutine fibonacci_root_reset
```

fibonacci_root_deallocate_tree ↑

```
recursive subroutine fibonacci_root_deallocate_tree(this)
  class(fibonacci_root_type),intent(inout) :: this
  call fibonacci_node_deallocate_tree(this)
  nullify(this%leftmost)
  nullify(this%rightmost)
end subroutine fibonacci_root_deallocate_tree
```

fibonacci_root_deallocate_all ↑

```
recursive subroutine fibonacci_root_deallocate_all(this)
  class(fibonacci_root_type),intent(inout) :: this
  call fibonacci_node_deallocate_all(this)
  nullify(this%leftmost)
  nullify(this%rightmost)
end subroutine fibonacci_root_deallocate_all
```

fibonacci_root_is_left_child ↑

```
elemental logical function fibonacci_root_is_left_child(this)
  class(fibonacci_root_type),target,intent(in) :: this
  fibonacci_root_is_left_child = .false.
end function fibonacci_root_is_left_child
```

fibonacci_root_is_right_child ↑

```
elemental logical function fibonacci_root_is_right_child(this)
  class(fibonacci_root_type),target,intent(in) :: this
  fibonacci_root_is_right_child = .false.
end function fibonacci_root_is_right_child
```

9.4.3 Methoden für fibonacci_stub_type

fibonacci_stub_get_type ↑

```
pure subroutine fibonacci_stub_get_type(type)
  character(:),allocatable,intent(out)::type
  allocate(type,source="fibonacci_stub_type")
end subroutine fibonacci_stub_get_type
```

fibonacci_stub_push_by_content ↑

```
subroutine fibonacci_stub_push_by_content(this,content)
  class(fibonacci_stub_type),target,intent(inout)  :: this
  class(measurable_class),target,intent(in)::content
  class(fibonacci_leave_type),pointer::leave
  allocate(leave)
  call leave%set_content(content)
  call this%push_by_leave(leave)
end subroutine fibonacci_stub_push_by_content
```

fibonacci_stub_push_by_leave ↑

```
subroutine fibonacci_stub_push_by_leave(this,new_leave)
  class(fibonacci_stub_type),target,intent(inout)   :: this
  class(fibonacci_leave_type),pointer,intent(inout) :: new_leave
  class(fibonacci_leave_type),pointer::old_leave
  if(this%depth<1)then
     if(associated(this%leftmost))then
        old_leave=>this%leftmost
        call this%init_by_leave(old_leave,new_leave)
     else
        this%leftmost=>new_leave
     end if
  else
     call fibonacci_root_push_by_leave(this,new_leave)
  end if
end subroutine fibonacci_stub_push_by_leave
```

fibonacci_stub_pop_left ↑

```
subroutine fibonacci_stub_pop_left(this,leave)
  class(fibonacci_stub_type),intent(inout),target  :: this
  class(fibonacci_leave_type),pointer,intent(out) :: leave
  if(this%depth<2)then
     if(this%depth==1)then
        leave=>this%leftmost
        this%leftmost=>this%rightmost
        nullify(this%rightmost)
        nullify(this%right)
        nullify(this%left)
        this%depth=0
        this%is_valid_c=.false.
     else
        if(associated(this%leftmost))then
           leave=>this%leftmost
           nullify(this%leftmost)
        end if
     end if
  else
     call fibonacci_root_pop_left(this,leave)
  end if
end subroutine fibonacci_stub_pop_left
```

fibonacci_stub_pop_right ↑

```
subroutine fibonacci_stub_pop_right(this,leave)
  class(fibonacci_stub_type),intent(inout),target  :: this
  class(fibonacci_leave_type),pointer,intent(out) :: leave
  if(this%depth<2)then
     if(this%depth==1)then
        this%is_valid_c=.false.
        if(associated(this%rightmost))then
           leave=>this%rightmost
           nullify(this%rightmost)
```

```
            nullify(this%right)
         else
            if(associated(this%leftmost))then
               leave=>this%leftmost
               nullify(this%leftmost)
               nullify(this%left)
            else
               nullify(leave)
            end if
         end if
      end if
   else
      call fibonacci_root_pop_right(this,leave)
   end if
  end subroutine fibonacci_stub_pop_right
```

9.4.4 Methoden für fibonacci_leave_type

fibonacci_leave_get_type ↑

```
  pure subroutine fibonacci_leave_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="fibonacci_leave_type")
  end subroutine fibonacci_leave_get_type
```

fibonacci_leave_print_to_unit ↑

```
  subroutine fibonacci_leave_print_to_unit(this,unit,parents,components,peers)
    class(fibonacci_leave_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    class(serializable_class),pointer::ser
    if(parents>0)call fibonacci_node_print_to_unit(this,unit,parents-one,components,-one)
    write(unit,'("Components of fibonacci_leave_type:")')
    ser=>this%down
    call serialize_print_comp_pointer(ser,unit,parents,components,peers,"Content:")
  end subroutine fibonacci_leave_print_to_unit
```

fibonacci_leave_get_left ↑

```
  subroutine fibonacci_leave_get_left(this,leave)
    class(fibonacci_leave_type),intent(in) :: this
    class(fibonacci_leave_type),intent(out),pointer :: leave
    class(fibonacci_node_type),pointer::node
    node=>this%left
    select type(node)
    class is (fibonacci_leave_type)
       leave=>node
    end select
  end subroutine fibonacci_leave_get_left
```

fibonacci_leave_get_right ↑

```
  subroutine fibonacci_leave_get_right(this,leave)
    class(fibonacci_leave_type),intent(in) :: this
    class(fibonacci_leave_type),intent(out),pointer :: leave
    class(fibonacci_node_type),pointer::node
!    print *,"fibonacci_leave_get_right"
!    call this%down%print_little
    if(associated(this%right))then
       node=>this%right
!       call node%down%print_little
       select type(node)
       class is (fibonacci_leave_type)
          leave=>node
       end select
    else
!       print *,"no right leave"
       nullify(leave)
    end if
  end subroutine fibonacci_leave_get_right
```

fibonacci_leave_deallocate_all ↑

```
  subroutine fibonacci_leave_deallocate_all(this)
    class(fibonacci_leave_type),intent(inout) :: this
    if (associated(this%down)) then
       deallocate(this%down)
    end if
  end subroutine fibonacci_leave_deallocate_all
```

fibonacci_leave_write_pstricks ↑

```
  subroutine fibonacci_leave_write_pstricks(this,unitnr)
    class(fibonacci_leave_type),intent(in),target :: this
    integer,intent(in) :: unitnr
    write(unitnr,'("\beginpsTree\Toval[linecolor=green]\node",i3,"",f9.3,"")')&
      this%depth,this%measure()
    if (associated(this%left)) then
       write(unitnr,'("\Tr[",a,"]")') le_kid
    end if
    if (associated(this%right)) then
       write(unitnr,'("\Tr[",a,"]")') le_kid
    end if
    write(unitnr,'("\endpsTree")')
  end subroutine fibonacci_leave_write_pstricks
```

fibonacci_leave_insert_leave_by_node ↑

```
  subroutine fibonacci_leave_insert_leave_by_node(this,new_leave)
    class(fibonacci_leave_type),target,intent(inout) :: this,new_leave
    class(fibonacci_node_type),pointer :: parent,new_node
    parent=>this%up
    !print *,associated(this%left),associated(this%right)
```

```
    if(this<new_leave)then
       call fibonacci_node_spawn(new_node,this,new_leave,this%left,this%right)
       !print *,"Repair! ",this%measure(),new_leave%measure()
    else
       call fibonacci_node_spawn(new_node,new_leave,this,this%left,this%right)
    end if
    if(associated(parent%left,this))then
       call parent%append_left(new_node)
    else
       call parent%append_right(new_node)
    end if
    call parent%repair()
  end subroutine fibonacci_leave_insert_leave_by_node
```

fibonacci_leave_copy_content ↑

```
  subroutine fibonacci_leave_copy_content(this,content)
    class(fibonacci_leave_type) :: this
    class(measurable_class),intent(in) :: content
    allocate(this%down,source=content)
  end subroutine fibonacci_leave_copy_content
```

fibonacci_leave_set_content ↑

```
  subroutine fibonacci_leave_set_content(this,content)
    class(fibonacci_leave_type) :: this
    class(measurable_class),target,intent(in) :: content
    this%down => content
  end subroutine fibonacci_leave_set_content
```

fibonacci_leave_get_content ↑

```
  subroutine fibonacci_leave_get_content(this,content)
    class(fibonacci_leave_type),intent(in) :: this
    class(measurable_class),pointer :: content
    content => this%down
  end subroutine fibonacci_leave_get_content
```

fibonacci_leave_is_inner ↑

```
  elemental logical function fibonacci_leave_is_inner()
    fibonacci_leave_is_inner = .false.
  end function fibonacci_leave_is_inner
```

fibonacci_leave_is_leave ↑

```
  elemental logical function fibonacci_leave_is_leave()
    fibonacci_leave_is_leave = .true.
  end function fibonacci_leave_is_leave
```

fibonacci_leave_is_left_short ↑

```
elemental logical function fibonacci_leave_is_left_short(this)
  class(fibonacci_leave_type),intent(in) :: this
  fibonacci_leave_is_left_short = .false.
end function fibonacci_leave_is_left_short
```

fibonacci_leave_is_right_short ↑

```
elemental logical function fibonacci_leave_is_right_short(this)
  class(fibonacci_leave_type),intent(in) :: this
  fibonacci_leave_is_right_short = .false.
end function fibonacci_leave_is_right_short
```

fibonacci_leave_is_unbalanced ↑

```
elemental logical function fibonacci_leave_is_unbalanced(this)
  class(fibonacci_leave_type),intent(in) :: this
  fibonacci_leave_is_unbalanced = .false.
end function fibonacci_leave_is_unbalanced
```

fibonacci_leave_is_left_too_short ↑

```
elemental logical function fibonacci_leave_is_left_too_short(this)
  class(fibonacci_leave_type),intent(in) :: this
  fibonacci_leave_is_left_too_short = .false.
end function fibonacci_leave_is_left_too_short
```

fibonacci_leave_is_right_too_short ↑

```
elemental logical function fibonacci_leave_is_right_too_short(this)
  class(fibonacci_leave_type),intent(in) :: this
  fibonacci_leave_is_right_too_short = .false.
end function fibonacci_leave_is_right_too_short
```

fibonacci_leave_is_too_unbalanced ↑

```
elemental logical function fibonacci_leave_is_too_unbalanced(this)
  class(fibonacci_leave_type),intent(in) :: this
  fibonacci_leave_is_too_unbalanced = .false.
end function fibonacci_leave_is_too_unbalanced
```

9.4.5 Sonstige Prozeduren

fibonacci_leave_write_content

```
subroutine fibonacci_leave_write_content(this,unit)
  class(fibonacci_leave_type),intent(in),target :: this
  integer,optional,intent(in)::unit
  call this%down%print_all(unit)
end subroutine fibonacci_leave_write_content
```

fibonacci_leave_write

```
  subroutine fibonacci_leave_write(this,unit)
    class(fibonacci_leave_type),intent(in),target :: this
    integer,optional,intent(in)::unit
    call this%print_all(unit)
  end subroutine fibonacci_leave_write
```

fibonacci_leave_write_value

```
  subroutine fibonacci_leave_write_value(this,unit)
    class(fibonacci_leave_type),intent(in),target :: this
    integer,intent(in),optional::unit
    if(present(unit))then
       write(unit,fmt=*)this%measure()
    else
       print *,this%measure()
    end if
!     call this%print_little(unit)
  end subroutine fibonacci_leave_write_value
```

fibonacci_node_spawn

```
  subroutine fibonacci_node_spawn&
    (new_node,left_leave,right_leave,left_left_leave,right_right_leave)
    class(fibonacci_node_type),pointer,intent(out) :: new_node
    class(fibonacci_leave_type),target,intent(inout) :: left_leave,right_leave
    class(fibonacci_node_type),pointer,intent(inout) :: left_left_leave,right_right_leave
    allocate(new_node)
    new_node%depth=1
    if(associated(left_left_leave))then
       left_left_leave%right=>left_leave
       left_leave%left=>left_left_leave
    else
       nullify(left_leave%left)
    end if
    if(associated(right_right_leave))then
       right_right_leave%left=>right_leave
       right_leave%right=>right_right_leave
    else
       nullify(right_leave%right)
    end if
    new_node%left=>left_leave
    new_node%right=>right_leave
    new_node%down=>left_leave%down
    new_node%depth=1
    left_leave%up=>new_node
    right_leave%up=>new_node
    left_leave%right=>right_leave
    right_leave%left=>left_leave
  end subroutine fibonacci_node_spawn
```

fibonacci_root_list_to_tree

```
  subroutine fibonacci_root_list_to_tree(this,n_leaves,leave_list_target)
    class(fibonacci_root_type),target :: this
    integer,intent(in) :: n_leaves
    type(fibonacci_leave_list_type),target,intent(in) :: leave_list_target
!     class(fibonacci_root_type),pointer,intent(out) :: tree
    integer:: depth,n_deep,n_merge
    class(fibonacci_node_type),pointer :: node
    class(fibonacci_leave_list_type),pointer :: leave_list
    class(fibonacci_leave_type),pointer::content
    real(kind=double) :: up_value
    leave_list=>leave_list_target
    call ilog2(n_leaves,depth,n_deep)
    n_deep=n_deep*2
    n_merge=0
    this%depth=depth
    node=>this
    outer: do
       do while(depth>1)
          depth=depth-1
          allocate(node%left)
          node%left%up=>node
          node=>node%left
          node%depth=depth
       end do
       node%left=>leave_list%leave
       node%down=>leave_list%leave%down
       leave_list=>leave_list%next
       node%right=>leave_list%leave
       content => leave_list%leave
       leave_list=>leave_list%next
       n_merge=n_merge+2
       inner: do
          if (associated(node%up)) then
             if (node%is_left_child()) then
                if (n_merge==n_deep.and.depth==1) then
                   node=>node%up
                   node%right=>leave_list%leave
                   node%right%up=>node
                   node%down=>content%down
                   content=>leave_list%leave
                   leave_list=>leave_list%next
                   n_merge=n_merge+1
                   cycle
                end if
                exit inner
             else
                node=>node%up
                depth=depth+1
             end if
          else
             exit outer
```

```
        end if
      end do inner
      node=>node%up
      node%down=>content%down
      allocate(node%right)
      node%right%up => node
      node=>node%right
      if (n_deep==n_merge) then
        depth=depth-1
      end if
      node%depth=depth
    end do outer
    call this%set_leftmost
    call this%set_rightmost
  end subroutine fibonacci_root_list_to_tree
```

10 Modul muli_interactions

10.1 Abhängigkeiten

```
use muli_momentum
```

10.2 Parameter

```
implicit none
!process parameters
integer,parameter::hadron_A_kind=2212  ! Proton
integer,parameter::hadron_B_kind=-2212 ! Anti Proton
integer,dimension(4),parameter::parton_kind_of_int_kind=[1,1,2,2]
real(kind=double), parameter :: b_sigma_tot_all = 100 !mb PDG
real(kind=double), parameter :: b_sigma_tot_nd = 0.5*b_sigma_tot_all !phys.rev.d v49 n5 1994
real(kind=double), parameter :: gev_cme_tot = 14000 !total center of mass energie
real(kind=double), parameter :: gev2_cme_tot = gev_cme_tot**2 !s
real(kind=double), parameter :: gev_pt_max = gev_cme_tot/2D0
real(kind=double), parameter :: gev2_pt_max = gev2_cme_tot/4D0
!model parameters
real(kind=double), parameter :: gev_pt_min = 8D-1
real(kind=double), parameter :: gev2_pt_min = gev_pt_min**2
real(kind=double), parameter :: pts_min = gev_pt_min/gev_pt_max
real(kind=double), parameter :: pts2_min = gev2_pt_min/gev2_pt_max
real(kind=double), parameter :: gev_p_t_0 = 2.0
real(kind=double), parameter :: gev2_p_t_0 = gev_p_t_0**2
real(kind=double), parameter :: norm_p_t_0 = gev_p_t_0/gev_pt_max
real(kind=double), parameter :: norm2_p_t_0 = gev2_p_t_0/gev2_pt_max
!mathematical constants
real(kind=double),private,parameter :: pi = 3.14159265
real(kind=double),parameter         :: euler = exp(1D0)
!physical constants
real(kind=double), parameter :: gev2_mbarn = 0.389379304D0
real(kind=double), parameter :: const_pref=pi*gev2_mbarn/(gev2_cme_tot*b_sigma_tot_nd)
  !parton kind parameters
integer,parameter,public::lha_flavor_at=-6
integer,parameter,public::lha_flavor_ab=-5
integer,parameter,public::lha_flavor_ac=-4
integer,parameter,public::lha_flavor_as=-3
integer,parameter,public::lha_flavor_au=-2
integer,parameter,public::lha_flavor_ad=-1
integer,parameter,public::lha_flavor_g=0
```

```
integer,parameter,public::lha_flavor_d=1
integer,parameter,public::lha_flavor_u=2
integer,parameter,public::lha_flavor_s=3
integer,parameter,public::lha_flavor_c=4
integer,parameter,public::lha_flavor_b=5
integer,parameter,public::lha_flavor_t=6
integer,parameter,public::pdg_flavor_at=-6
integer,parameter,public::pdg_flavor_ab=-5
integer,parameter,public::pdg_flavor_ac=-4
integer,parameter,public::pdg_flavor_as=-3
integer,parameter,public::pdg_flavor_au=-2
integer,parameter,public::pdg_flavor_ad=-1
integer,parameter,public::pdg_flavor_g=21
integer,parameter,public::pdg_flavor_d=1
integer,parameter,public::pdg_flavor_u=2
integer,parameter,public::pdg_flavor_s=3
integer,parameter,public::pdg_flavor_c=4
integer,parameter,public::pdg_flavor_b=5
integer,parameter,public::pdg_flavor_t=6
integer,parameter,public::parton_kind_sea=1
integer,parameter,public::parton_kind_valence=2
integer,parameter,public::parton_kind_sea_and_valence=3
integer,parameter,public::parton_kind_twin=4
integer,parameter,public::parton_kind_sea_and_twin=5
integer,parameter,public::parton_kind_valence_and_twin=6
integer,parameter,public::parton_kind_all=7
integer,parameter,public::pdf_int_kind_undef=0
integer,parameter,public::pdf_int_kind_gluon=1
integer,parameter,public::pdf_int_kind_sea=2
integer,parameter,public::pdf_int_kind_val_down=3
integer,parameter,public::pdf_int_kind_val_up=4
integer,parameter,public::pdf_int_kind_twin=5
character(len=2),dimension(-6:6),parameter :: integer_parton_names = &
     &["-6","-5","-4","-3","-2","-1","00","+1","+2","+3","+4","+5","+6"]
character,dimension(-6:6),parameter :: traditional_parton_names = &
     &["T","B","C","S","U","D","g","d","u","s","c","b","t"]
!ps polynom coefficients
! evolution variable is pt2s/(x1*x2)
real(kind=double),dimension(1:4,1:5),parameter :: phase_space_coefficients_in&
  = reshape(source=[&
     & 6144D0, -4608D0,  +384D0,    0D0,&
     & 6144D0, -5120D0,  +384D0,    0D0,&
     & 6144D0, -2048D0,  +128D0, -576D0,&
     &13824D0, -9600D0, +1056D0,    0D0,&
     &31104D0,-19872D0, +2160D0, +486D0],&
     &shape=[4,5])

! evolution variable is pt2s/(x1*x2)
real(kind=double),dimension(1:4,1:8),parameter :: phase_space_coefficients_inout&
  = reshape(source=[&
     &3072,  -2304,  +192,    0, &
```

```
      &6144,  -5120,  +384,    0, &
      &0,         0,   192,  -96, &
      &3072,  -2048,  +192,  -96, &
      &0,      2048, -2176, +576, &
      &0,       288,  -306,  +81, &
      &6912,  -4800,  +528,    0, &
      &31104,-23328, +5832, -486],&
      &shape=[4,8])

 integer,dimension(1:4,0:8),parameter :: inout_signatures = reshape(source=[&
       1, 1, 1, 1,&!1a
      -1, 1,-1, 1,&!1b
       1, 1, 1, 1,&!2
       1,-1, 1,-1,&!3
       1,-1, 1,-1,&!4
       1,-1, 0, 0,&!5
       0, 0, 1,-1,&!6
       1, 0, 1, 0,&!7
       0, 0, 0, 0],&
       shape=[4,9])

 integer,dimension(6,-234:234),parameter::valid_processes=reshape([&
-6,  -6,  -6,  -6,   2,   2,&!-234
-6,  -5,  -6,  -5,   1,   1,&!-233
-6,  -5,  -5,  -6,   1,   1,&!-232
-6,  -4,  -6,  -4,   1,   1,&!-231
-6,  -4,  -4,  -6,   1,   1,&!-230
-6,  -3,  -6,  -3,   1,   1,&!-229
-6,  -3,  -3,  -6,   1,   1,&!-228
-6,  -2,  -6,  -2,   1,   1,&!-227
-6,  -2,  -2,  -6,   1,   1,&!-226
-6,  -1,  -6,  -1,   1,   1,&!-225
-6,  -1,  -1,  -6,   1,   1,&!-224
-6,   0,  -6,   0,   4,   7,&!-223
-6,   0,   0,  -6,   4,   7,&!-222
-6,   1,  -6,   1,   1,   1,&!-221
-6,   1,   1,  -6,   1,   1,&!-220
-6,   2,  -6,   2,   1,   1,&!-219
-6,   2,   2,  -6,   1,   1,&!-218
-6,   3,  -6,   3,   1,   1,&!-217
-6,   3,   3,  -6,   1,   1,&!-216
-6,   4,  -6,   4,   1,   1,&!-215
-6,   4,   4,  -6,   1,   1,&!-214
-6,   5,  -6,   5,   1,   1,&!-213
-6,   5,   5,  -6,   1,   1,&!-212
-6,   6,  -6,   6,   3,   4,&!-211
-6,   6,  -5,   5,   3,   3,&!-210
-6,   6,  -4,   4,   3,   3,&!-209
-6,   6,  -3,   3,   3,   3,&!-208
-6,   6,  -2,   2,   3,   3,&!-207
-6,   6,  -1,   1,   3,   3,&!-206
```

```
 -6,   6,   0,   0,   3,   5,&!-205
 -6,   6,   1,  -1,   3,   3,&!-204
 -6,   6,   2,  -2,   3,   3,&!-203
 -6,   6,   3,  -3,   3,   3,&!-202
 -6,   6,   4,  -4,   3,   3,&!-201
 -6,   6,   5,  -5,   3,   3,&!-200
 -6,   6,   6,  -6,   3,   4,&!-199
 -5,  -6,  -6,  -5,   1,   1,&!-198
 -5,  -6,  -5,  -6,   1,   1,&!-197
 -5,  -5,  -5,  -5,   2,   2,&!-196
 -5,  -4,  -5,  -4,   1,   1,&!-195
 -5,  -4,  -4,  -5,   1,   1,&!-194
 -5,  -3,  -5,  -3,   1,   1,&!-193
 -5,  -3,  -3,  -5,   1,   1,&!-192
 -5,  -2,  -5,  -2,   1,   1,&!-191
 -5,  -2,  -2,  -5,   1,   1,&!-190
 -5,  -1,  -5,  -1,   1,   1,&!-189
 -5,  -1,  -1,  -5,   1,   1,&!-188
 -5,   0,  -5,   0,   4,   7,&!-187
 -5,   0,   0,  -5,   4,   7,&!-186
 -5,   1,  -5,   1,   1,   1,&!-185
 -5,   1,   1,  -5,   1,   1,&!-184
 -5,   2,  -5,   2,   1,   1,&!-183
 -5,   2,   2,  -5,   1,   1,&!-182
 -5,   3,  -5,   3,   1,   1,&!-181
 -5,   3,   3,  -5,   1,   1,&!-180
 -5,   4,  -5,   4,   1,   1,&!-179
 -5,   4,   4,  -5,   1,   1,&!-178
 -5,   5,  -6,   6,   3,   3,&!-177
 -5,   5,  -5,   5,   3,   4,&!-176
 -5,   5,  -4,   4,   3,   3,&!-175
 -5,   5,  -3,   3,   3,   3,&!-174
 -5,   5,  -2,   2,   3,   3,&!-173
 -5,   5,  -1,   1,   3,   3,&!-172
 -5,   5,   0,   0,   3,   5,&!-171
 -5,   5,   1,  -1,   3,   3,&!-170
 -5,   5,   2,  -2,   3,   3,&!-169
 -5,   5,   3,  -3,   3,   3,&!-168
 -5,   5,   4,  -4,   3,   3,&!-167
 -5,   5,   5,  -5,   3,   4,&!-166
 -5,   5,   6,  -6,   3,   3,&!-165
 -5,   6,  -5,   6,   1,   1,&!-164
 -5,   6,   6,  -5,   1,   1,&!-163
 -4,  -6,  -6,  -4,   1,   1,&!-162
 -4,  -6,  -4,  -6,   1,   1,&!-161
 -4,  -5,  -5,  -4,   1,   1,&!-160
 -4,  -5,  -4,  -5,   1,   1,&!-159
 -4,  -4,  -4,  -4,   2,   2,&!-158
 -4,  -3,  -4,  -3,   1,   1,&!-157
 -4,  -3,  -3,  -4,   1,   1,&!-156
 -4,  -2,  -4,  -2,   1,   1,&!-155
```

```
-4,  -2,  -2,  -4,   1,   1,&!-154
-4,  -1,  -4,  -1,   1,   1,&!-153
-4,  -1,  -1,  -4,   1,   1,&!-152
-4,   0,  -4,   0,   4,   7,&!-151
-4,   0,   0,  -4,   4,   7,&!-150
-4,   1,  -4,   1,   1,   1,&!-149
-4,   1,   1,  -4,   1,   1,&!-148
-4,   2,  -4,   2,   1,   1,&!-147
-4,   2,   2,  -4,   1,   1,&!-146
-4,   3,  -4,   3,   1,   1,&!-145
-4,   3,   3,  -4,   1,   1,&!-144
-4,   4,  -6,   6,   3,   3,&!-143
-4,   4,  -5,   5,   3,   3,&!-142
-4,   4,  -4,   4,   3,   4,&!-141
-4,   4,  -3,   3,   3,   3,&!-140
-4,   4,  -2,   2,   3,   3,&!-139
-4,   4,  -1,   1,   3,   3,&!-138
-4,   4,   0,   0,   3,   5,&!-137
-4,   4,   1,  -1,   3,   3,&!-136
-4,   4,   2,  -2,   3,   3,&!-135
-4,   4,   3,  -3,   3,   3,&!-134
-4,   4,   4,  -4,   3,   4,&!-133
-4,   4,   5,  -5,   3,   3,&!-132
-4,   4,   6,  -6,   3,   3,&!-131
-4,   5,  -4,   5,   1,   1,&!-130
-4,   5,   5,  -4,   1,   1,&!-129
-4,   6,  -4,   6,   1,   1,&!-128
-4,   6,   6,  -4,   1,   1,&!-127
-3,  -6,  -6,  -3,   1,   1,&!-126
-3,  -6,  -3,  -6,   1,   1,&!-125
-3,  -5,  -5,  -3,   1,   1,&!-124
-3,  -5,  -3,  -5,   1,   1,&!-123
-3,  -4,  -4,  -3,   1,   1,&!-122
-3,  -4,  -3,  -4,   1,   1,&!-121
-3,  -3,  -3,  -3,   2,   2,&!-120
-3,  -2,  -3,  -2,   1,   1,&!-119
-3,  -2,  -2,  -3,   1,   1,&!-118
-3,  -1,  -3,  -1,   1,   1,&!-117
-3,  -1,  -1,  -3,   1,   1,&!-116
-3,   0,  -3,   0,   4,   7,&!-115
-3,   0,   0,  -3,   4,   7,&!-114
-3,   1,  -3,   1,   1,   1,&!-113
-3,   1,   1,  -3,   1,   1,&!-112
-3,   2,  -3,   2,   1,   1,&!-111
-3,   2,   2,  -3,   1,   1,&!-110
-3,   3,  -6,   6,   3,   3,&!-109
-3,   3,  -5,   5,   3,   3,&!-108
-3,   3,  -4,   4,   3,   3,&!-107
-3,   3,  -3,   3,   3,   4,&!-106
-3,   3,  -2,   2,   3,   3,&!-105
-3,   3,  -1,   1,   3,   3,&!-104
```

```
-3,   3,   0,   0,   3,   5,&!-103
-3,   3,   1,  -1,   3,   3,&!-102
-3,   3,   2,  -2,   3,   3,&!-101
-3,   3,   3,  -3,   3,   4,&!-100
-3,   3,   4,  -4,   3,   3,&! -99
-3,   3,   5,  -5,   3,   3,&! -98
-3,   3,   6,  -6,   3,   3,&! -97
-3,   4,  -3,   4,   1,   1,&! -96
-3,   4,   4,  -3,   1,   1,&! -95
-3,   5,  -3,   5,   1,   1,&! -94
-3,   5,   5,  -3,   1,   1,&! -93
-3,   6,  -3,   6,   1,   1,&! -92
-3,   6,   6,  -3,   1,   1,&! -91
-2,  -6,  -6,  -2,   1,   1,&! -90
-2,  -6,  -2,  -6,   1,   1,&! -89
-2,  -5,  -5,  -2,   1,   1,&! -88
-2,  -5,  -2,  -5,   1,   1,&! -87
-2,  -4,  -4,  -2,   1,   1,&! -86
-2,  -4,  -2,  -4,   1,   1,&! -85
-2,  -3,  -3,  -2,   1,   1,&! -84
-2,  -3,  -2,  -3,   1,   1,&! -83
-2,  -2,  -2,  -2,   2,   2,&! -82
-2,  -1,  -2,  -1,   1,   1,&! -81
-2,  -1,  -1,  -2,   1,   1,&! -80
-2,   0,  -2,   0,   4,   7,&! -79
-2,   0,   0,  -2,   4,   7,&! -78
-2,   1,  -2,   1,   1,   1,&! -77
-2,   1,   1,  -2,   1,   1,&! -76
-2,   2,  -6,   6,   3,   3,&! -75
-2,   2,  -5,   5,   3,   3,&! -74
-2,   2,  -4,   4,   3,   3,&! -73
-2,   2,  -3,   3,   3,   3,&! -72
-2,   2,  -2,   2,   3,   4,&! -71
-2,   2,  -1,   1,   3,   3,&! -70
-2,   2,   0,   0,   3,   5,&! -69
-2,   2,   1,  -1,   3,   3,&! -68
-2,   2,   2,  -2,   3,   4,&! -67
-2,   2,   3,  -3,   3,   3,&! -66
-2,   2,   4,  -4,   3,   3,&! -65
-2,   2,   5,  -5,   3,   3,&! -64
-2,   2,   6,  -6,   3,   3,&! -63
-2,   3,  -2,   3,   1,   1,&! -62
-2,   3,   3,  -2,   1,   1,&! -61
-2,   4,  -2,   4,   1,   1,&! -60
-2,   4,   4,  -2,   1,   1,&! -59
-2,   5,  -2,   5,   1,   1,&! -58
-2,   5,   5,  -2,   1,   1,&! -57
-2,   6,  -2,   6,   1,   1,&! -56
-2,   6,   6,  -2,   1,   1,&! -55
-1,  -6,  -6,  -1,   1,   1,&! -54
-1,  -6,  -1,  -6,   1,   1,&! -53
```

```
-1,  -5,  -5,  -1,   1,   1,&! -52
-1,  -5,  -1,  -5,   1,   1,&! -51
-1,  -4,  -4,  -1,   1,   1,&! -50
-1,  -4,  -1,  -4,   1,   1,&! -49
-1,  -3,  -3,  -1,   1,   1,&! -48
-1,  -3,  -1,  -3,   1,   1,&! -47
-1,  -2,  -2,  -1,   1,   1,&! -46
-1,  -2,  -1,  -2,   1,   1,&! -45
-1,  -1,  -1,  -1,   2,   2,&! -44
-1,   0,  -1,   0,   4,   7,&! -43
-1,   0,   0,  -1,   4,   7,&! -42
-1,   1,  -6,   6,   3,   3,&! -41
-1,   1,  -5,   5,   3,   3,&! -40
-1,   1,  -4,   4,   3,   3,&! -39
-1,   1,  -3,   3,   3,   3,&! -38
-1,   1,  -2,   2,   3,   3,&! -37
-1,   1,  -1,   1,   3,   4,&! -36
-1,   1,   0,   0,   3,   5,&! -35
-1,   1,   1,  -1,   3,   4,&! -34
-1,   1,   2,  -2,   3,   3,&! -33
-1,   1,   3,  -3,   3,   3,&! -32
-1,   1,   4,  -4,   3,   3,&! -31
-1,   1,   5,  -5,   3,   3,&! -30
-1,   1,   6,  -6,   3,   3,&! -29
-1,   2,  -1,   2,   1,   1,&! -28
-1,   2,   2,  -1,   1,   1,&! -27
-1,   3,  -1,   3,   1,   1,&! -26
-1,   3,   3,  -1,   1,   1,&! -25
-1,   4,  -1,   4,   1,   1,&! -24
-1,   4,   4,  -1,   1,   1,&! -23
-1,   5,  -1,   5,   1,   1,&! -22
-1,   5,   5,  -1,   1,   1,&! -21
-1,   6,  -1,   6,   1,   1,&! -20
-1,   6,   6,  -1,   1,   1,&! -19
 0,  -6,  -6,   0,   4,   7,&! -18
 0,  -6,   0,  -6,   4,   7,&! -17
 0,  -5,  -5,   0,   4,   7,&! -16
 0,  -5,   0,  -5,   4,   7,&! -15
 0,  -4,  -4,   0,   4,   7,&! -14
 0,  -4,   0,  -4,   4,   7,&! -13
 0,  -3,  -3,   0,   4,   7,&! -12
 0,  -3,   0,  -3,   4,   7,&! -11
 0,  -2,  -2,   0,   4,   7,&! -10
 0,  -2,   0,  -2,   4,   7,&!  -9
 0,  -1,  -1,   0,   4,   7,&!  -8
 0,  -1,   0,  -1,   4,   7,&!  -7
 0,   0,  -6,   6,   5,   6,&!  -6
 0,   0,  -5,   5,   5,   6,&!  -5
 0,   0,  -4,   4,   5,   6,&!  -4
 0,   0,  -3,   3,   5,   6,&!  -3
 0,   0,  -2,   2,   5,   6,&!  -2
```

```
  0,   0,  -1,   1,   5,   6,&!  -1
  0,   0,   0,   0,   5,   8,&!   0
  0,   0,   1,  -1,   5,   6,&!   1
  0,   0,   2,  -2,   5,   6,&!   2
  0,   0,   3,  -3,   5,   6,&!   3
  0,   0,   4,  -4,   5,   6,&!   4
  0,   0,   5,  -5,   5,   6,&!   5
  0,   0,   6,  -6,   5,   6,&!   6
  0,   1,   0,   1,   4,   7,&!   7
  0,   1,   1,   0,   4,   7,&!   8
  0,   2,   0,   2,   4,   7,&!   9
  0,   2,   2,   0,   4,   7,&!  10
  0,   3,   0,   3,   4,   7,&!  11
  0,   3,   3,   0,   4,   7,&!  12
  0,   4,   0,   4,   4,   7,&!  13
  0,   4,   4,   0,   4,   7,&!  14
  0,   5,   0,   5,   4,   7,&!  15
  0,   5,   5,   0,   4,   7,&!  16
  0,   6,   0,   6,   4,   7,&!  17
  0,   6,   6,   0,   4,   7,&!  18
  1,  -6,  -6,   1,   1,   1,&!  19
  1,  -6,   1,  -6,   1,   1,&!  20
  1,  -5,  -5,   1,   1,   1,&!  21
  1,  -5,   1,  -5,   1,   1,&!  22
  1,  -4,  -4,   1,   1,   1,&!  23
  1,  -4,   1,  -4,   1,   1,&!  24
  1,  -3,  -3,   1,   1,   1,&!  25
  1,  -3,   1,  -3,   1,   1,&!  26
  1,  -2,  -2,   1,   1,   1,&!  27
  1,  -2,   1,  -2,   1,   1,&!  28
  1,  -1,  -6,   6,   3,   3,&!  29
  1,  -1,  -5,   5,   3,   3,&!  30
  1,  -1,  -4,   4,   3,   3,&!  31
  1,  -1,  -3,   3,   3,   3,&!  32
  1,  -1,  -2,   2,   3,   3,&!  33
  1,  -1,  -1,   1,   3,   4,&!  34
  1,  -1,   0,   0,   3,   5,&!  35
  1,  -1,   1,  -1,   3,   4,&!  36
  1,  -1,   2,  -2,   3,   3,&!  37
  1,  -1,   3,  -3,   3,   3,&!  38
  1,  -1,   4,  -4,   3,   3,&!  39
  1,  -1,   5,  -5,   3,   3,&!  40
  1,  -1,   6,  -6,   3,   3,&!  41
  1,   0,   0,   1,   4,   7,&!  42
  1,   0,   1,   0,   4,   7,&!  43
  1,   1,   1,   1,   2,   2,&!  44
  1,   2,   1,   2,   1,   1,&!  45
  1,   2,   2,   1,   1,   1,&!  46
  1,   3,   1,   3,   1,   1,&!  47
  1,   3,   3,   1,   1,   1,&!  48
  1,   4,   1,   4,   1,   1,&!  49
```

```
1,   4,   4,   1,   1,   1,&!  50
1,   5,   1,   5,   1,   1,&!  51
1,   5,   5,   1,   1,   1,&!  52
1,   6,   1,   6,   1,   1,&!  53
1,   6,   6,   1,   1,   1,&!  54
2,  -6,  -6,   2,   1,   1,&!  55
2,  -6,   2,  -6,   1,   1,&!  56
2,  -5,  -5,   2,   1,   1,&!  57
2,  -5,   2,  -5,   1,   1,&!  58
2,  -4,  -4,   2,   1,   1,&!  59
2,  -4,   2,  -4,   1,   1,&!  60
2,  -3,  -3,   2,   1,   1,&!  61
2,  -3,   2,  -3,   1,   1,&!  62
2,  -2,  -6,   6,   3,   3,&!  63
2,  -2,  -5,   5,   3,   3,&!  64
2,  -2,  -4,   4,   3,   3,&!  65
2,  -2,  -3,   3,   3,   3,&!  66
2,  -2,  -2,   2,   3,   4,&!  67
2,  -2,  -1,   1,   3,   3,&!  68
2,  -2,   0,   0,   3,   5,&!  69
2,  -2,   1,  -1,   3,   3,&!  70
2,  -2,   2,  -2,   3,   4,&!  71
2,  -2,   3,  -3,   3,   3,&!  72
2,  -2,   4,  -4,   3,   3,&!  73
2,  -2,   5,  -5,   3,   3,&!  74
2,  -2,   6,  -6,   3,   3,&!  75
2,  -1,  -1,   2,   1,   1,&!  76
2,  -1,   2,  -1,   1,   1,&!  77
2,   0,   0,   2,   4,   7,&!  78
2,   0,   2,   0,   4,   7,&!  79
2,   1,   1,   2,   1,   1,&!  80
2,   1,   2,   1,   1,   1,&!  81
2,   2,   2,   2,   2,   2,&!  82
2,   3,   2,   3,   1,   1,&!  83
2,   3,   3,   2,   1,   1,&!  84
2,   4,   2,   4,   1,   1,&!  85
2,   4,   4,   2,   1,   1,&!  86
2,   5,   2,   5,   1,   1,&!  87
2,   5,   5,   2,   1,   1,&!  88
2,   6,   2,   6,   1,   1,&!  89
2,   6,   6,   2,   1,   1,&!  90
3,  -6,  -6,   3,   1,   1,&!  91
3,  -6,   3,  -6,   1,   1,&!  92
3,  -5,  -5,   3,   1,   1,&!  93
3,  -5,   3,  -5,   1,   1,&!  94
3,  -4,  -4,   3,   1,   1,&!  95
3,  -4,   3,  -4,   1,   1,&!  96
3,  -3,  -6,   6,   3,   3,&!  97
3,  -3,  -5,   5,   3,   3,&!  98
3,  -3,  -4,   4,   3,   3,&!  99
3,  -3,  -3,   3,   3,   4,&! 100
```

```
   3,   -3,   -2,    2,    3,    3,&! 101
   3,   -3,   -1,    1,    3,    3,&! 102
   3,   -3,    0,    0,    3,    5,&! 103
   3,   -3,    1,   -1,    3,    3,&! 104
   3,   -3,    2,   -2,    3,    3,&! 105
   3,   -3,    3,   -3,    3,    4,&! 106
   3,   -3,    4,   -4,    3,    3,&! 107
   3,   -3,    5,   -5,    3,    3,&! 108
   3,   -3,    6,   -6,    3,    3,&! 109
   3,   -2,   -2,    3,    1,    1,&! 110
   3,   -2,    3,   -2,    1,    1,&! 111
   3,   -1,   -1,    3,    1,    1,&! 112
   3,   -1,    3,   -1,    1,    1,&! 113
   3,    0,    0,    3,    4,    7,&! 114
   3,    0,    3,    0,    4,    7,&! 115
   3,    1,    1,    3,    1,    1,&! 116
   3,    1,    3,    1,    1,    1,&! 117
   3,    2,    2,    3,    1,    1,&! 118
   3,    2,    3,    2,    1,    1,&! 119
   3,    3,    3,    3,    2,    2,&! 120
   3,    4,    3,    4,    1,    1,&! 121
   3,    4,    4,    3,    1,    1,&! 122
   3,    5,    3,    5,    1,    1,&! 123
   3,    5,    5,    3,    1,    1,&! 124
   3,    6,    3,    6,    1,    1,&! 125
   3,    6,    6,    3,    1,    1,&! 126
   4,   -6,   -6,    4,    1,    1,&! 127
   4,   -6,    4,   -6,    1,    1,&! 128
   4,   -5,   -5,    4,    1,    1,&! 129
   4,   -5,    4,   -5,    1,    1,&! 130
   4,   -4,   -6,    6,    3,    3,&! 131
   4,   -4,   -5,    5,    3,    3,&! 132
   4,   -4,   -4,    4,    3,    4,&! 133
   4,   -4,   -3,    3,    3,    3,&! 134
   4,   -4,   -2,    2,    3,    3,&! 135
   4,   -4,   -1,    1,    3,    3,&! 136
   4,   -4,    0,    0,    3,    5,&! 137
   4,   -4,    1,   -1,    3,    3,&! 138
   4,   -4,    2,   -2,    3,    3,&! 139
   4,   -4,    3,   -3,    3,    3,&! 140
   4,   -4,    4,   -4,    3,    4,&! 141
   4,   -4,    5,   -5,    3,    3,&! 142
   4,   -4,    6,   -6,    3,    3,&! 143
   4,   -3,   -3,    4,    1,    1,&! 144
   4,   -3,    4,   -3,    1,    1,&! 145
   4,   -2,   -2,    4,    1,    1,&! 146
   4,   -2,    4,   -2,    1,    1,&! 147
   4,   -1,   -1,    4,    1,    1,&! 148
   4,   -1,    4,   -1,    1,    1,&! 149
   4,    0,    0,    4,    4,    7,&! 150
   4,    0,    4,    0,    4,    7,&! 151
```

```
4,   1,   1,   4,   1,   1,&! 152
4,   1,   4,   1,   1,   1,&! 153
4,   2,   2,   4,   1,   1,&! 154
4,   2,   4,   2,   1,   1,&! 155
4,   3,   3,   4,   1,   1,&! 156
4,   3,   4,   3,   1,   1,&! 157
4,   4,   4,   4,   2,   2,&! 158
4,   5,   4,   5,   1,   1,&! 159
4,   5,   5,   4,   1,   1,&! 160
4,   6,   4,   6,   1,   1,&! 161
4,   6,   6,   4,   1,   1,&! 162
5,  -6,  -6,   5,   1,   1,&! 163
5,  -6,   5,  -6,   1,   1,&! 164
5,  -5,  -6,   6,   3,   3,&! 165
5,  -5,  -5,   5,   3,   4,&! 166
5,  -5,  -4,   4,   3,   3,&! 167
5,  -5,  -3,   3,   3,   3,&! 168
5,  -5,  -2,   2,   3,   3,&! 169
5,  -5,  -1,   1,   3,   3,&! 170
5,  -5,   0,   0,   3,   5,&! 171
5,  -5,   1,  -1,   3,   3,&! 172
5,  -5,   2,  -2,   3,   3,&! 173
5,  -5,   3,  -3,   3,   3,&! 174
5,  -5,   4,  -4,   3,   3,&! 175
5,  -5,   5,  -5,   3,   4,&! 176
5,  -5,   6,  -6,   3,   3,&! 177
5,  -4,  -4,   5,   1,   1,&! 178
5,  -4,   5,  -4,   1,   1,&! 179
5,  -3,  -3,   5,   1,   1,&! 180
5,  -3,   5,  -3,   1,   1,&! 181
5,  -2,  -2,   5,   1,   1,&! 182
5,  -2,   5,  -2,   1,   1,&! 183
5,  -1,  -1,   5,   1,   1,&! 184
5,  -1,   5,  -1,   1,   1,&! 185
5,   0,   0,   5,   4,   7,&! 186
5,   0,   5,   0,   4,   7,&! 187
5,   1,   1,   5,   1,   1,&! 188
5,   1,   5,   1,   1,   1,&! 189
5,   2,   2,   5,   1,   1,&! 190
5,   2,   5,   2,   1,   1,&! 191
5,   3,   3,   5,   1,   1,&! 192
5,   3,   5,   3,   1,   1,&! 193
5,   4,   4,   5,   1,   1,&! 194
5,   4,   5,   4,   1,   1,&! 195
5,   5,   5,   5,   2,   2,&! 196
5,   6,   5,   6,   1,   1,&! 197
5,   6,   6,   5,   1,   1,&! 198
6,  -6,  -6,   6,   3,   4,&! 199
6,  -6,  -5,   5,   3,   3,&! 200
6,  -6,  -4,   4,   3,   3,&! 201
6,  -6,  -3,   3,   3,   3,&! 202
```

```
  6,  -6,  -2,   2,   3,   3,&! 203
  6,  -6,  -1,   1,   3,   3,&! 204
  6,  -6,   0,   0,   3,   5,&! 205
  6,  -6,   1,  -1,   3,   3,&! 206
  6,  -6,   2,  -2,   3,   3,&! 207
  6,  -6,   3,  -3,   3,   3,&! 208
  6,  -6,   4,  -4,   3,   3,&! 209
  6,  -6,   5,  -5,   3,   3,&! 210
  6,  -6,   6,  -6,   3,   4,&! 211
  6,  -5,  -5,   6,   1,   1,&! 212
  6,  -5,   6,  -5,   1,   1,&! 213
  6,  -4,  -4,   6,   1,   1,&! 214
  6,  -4,   6,  -4,   1,   1,&! 215
  6,  -3,  -3,   6,   1,   1,&! 216
  6,  -3,   6,  -3,   1,   1,&! 217
  6,  -2,  -2,   6,   1,   1,&! 218
  6,  -2,   6,  -2,   1,   1,&! 219
  6,  -1,  -1,   6,   1,   1,&! 220
  6,  -1,   6,  -1,   1,   1,&! 221
  6,   0,   0,   6,   4,   7,&! 222
  6,   0,   6,   0,   4,   7,&! 223
  6,   1,   1,   6,   1,   1,&! 224
  6,   1,   6,   1,   1,   1,&! 225
  6,   2,   2,   6,   1,   1,&! 226
  6,   2,   6,   2,   1,   1,&! 227
  6,   3,   3,   6,   1,   1,&! 228
  6,   3,   6,   3,   1,   1,&! 229
  6,   4,   4,   6,   1,   1,&! 230
  6,   4,   6,   4,   1,   1,&! 231
  6,   5,   5,   6,   1,   1,&! 232
  6,   5,   6,   5,   1,   1,&! 233
  6,   6,   6,   6,   2,   2&! 234
],[6,469])


  integer,dimension(2,0:16),parameter::double_pdf_kinds=reshape([&
       &0,0,&
       &1,1,&
       &1,2,&
       &1,3,&
       &1,4,&
       &2,1,&
       &2,2,&
       &2,3,&
       &2,4,&
       &3,1,&
       &3,2,&
       &3,3,&
       &3,4,&
       &4,1,&
       &4,2,&
```

```
     &4,3,&
     &4,4&
     &],[2,17])
integer,parameter,dimension(371)::int_all=[&
     &  -6,   -5,   -4,   -3,   -2,   -1,    0,    1,    2,    3,    4,    5,    6,  -14,  -13,  -
     &  -9,   -8,   -7,    7,    8,    9,   10,   11,   12,   13,   14,    7,    8,    9,   10,  -
     &-114,  -79,  -78,  -43,  -42,   42,   43,   78,   79,  114,  115,  150,  151, -158, -157,  -
     &-153, -152, -149, -148, -147, -146, -145, -144, -143, -142, -141, -140, -139, -138, -137,  -
     &-133, -132, -131, -122, -121, -120, -119, -118, -117, -116, -113, -112, -111, -110, -109,  -
     &-105, -104, -103, -102, -101, -100,  -99,  -98,  -97,  -96,  -95,  -86,  -85,  -84,  -83,
     & -77,  -76,  -75,  -74,  -73,  -72,  -71,  -70,  -69,  -68,  -67,  -66,  -65,  -64,  -63,
     & -59,  -50,  -49,  -48,  -47,  -46,  -45,  -44,  -41,  -40,  -39,  -38,  -37,  -36,  -35,
     & -31,  -30,  -29,  -28,  -27,  -26,  -25,  -24,  -23,   23,   24,   25,   26,   27,   28,
     &  32,   33,   34,   35,   36,   37,   38,   39,   40,   41,   44,   45,   46,   47,   48,
     &  60,   61,   62,   63,   64,   65,   66,   67,   68,   69,   70,   71,   72,   73,   74,
     &  80,   81,   82,   83,   84,   85,   86,   95,   96,   97,   98,   99,  100,  101,  102,
     & 106,  107,  108,  109,  110,  111,  112,  113,  116,  117,  118,  119,  120,  121,  122,
     & 134,  135,  136,  137,  138,  139,  140,  141,  142,  143,  144,  145,  146,  147,  148,
     & 154,  155,  156,  157,  158, -149, -148, -113, -112,  -77,  -76,  -41,  -40,  -39,  -38,
     & -34,  -33,  -32,  -31,  -30,  -29,   44,   80,   81,  116,  117,  152,  153, -147, -146,  -
     & -74,  -73,  -72,  -71,  -70,  -69,  -68,  -67,  -66,  -65,  -64,  -63,  -28,  -27,   45,
     & 119,  154,  155,   42,   43,   23,   24,   25,   26,   27,   28,   29,   30,   31,   32,
     &  36,   37,   38,   39,   40,   41,   44,   45,   46,   47,   48,   49,   50,   44,   45,
     &  59,   60,   61,   62,   63,   64,   65,   66,   67,   68,   69,   70,   71,   72,   73,
     &  77,   80,   81,   82,   83,   84,   85,   86,   80,   81,   82]

integer,parameter,dimension(16)::int_sizes_all=[13,16,2,2,16,208,26,26,2,26,1,2,2,26,2,1]

integer,parameter,dimension(3,0:8)::muli_flow_stats=&
     reshape([&
      1, 2,4,&
      3, 4,4,&
      5, 6,8,&
      7, 8,4,&
      9,10,8,&
     11,16,16,&
     17,22,16,&
     23,28,16,&
     29,52,96],&
     [3,9])

integer,parameter,dimension(0:4,52)::muli_flows=&
     reshape([&
     3,0,0,1,2,&!1a
     1,0,0,2,1,&
     1,2,0,0,3,&!1b
     3,3,0,0,2,&
     4,0,0,1,2,&!2
     4,0,0,2,1,&
     3,2,0,0,3,&!3
     1,3,0,0,2,&
```

```
4,2,0,0,3,&!4
4,3,0,0,2,&
4,0,1,3,4,&!5
4,0,1,4,3,&
2,0,3,1,4,&
2,0,4,1,3,&
2,0,3,4,1,&
2,0,4,3,1,&
4,1,2,4,0,&!6
2,1,4,2,0,&
4,2,1,4,0,&
2,4,1,2,0,&
2,2,4,1,0,&
2,4,2,1,0,&
2,0,1,2,4,&!7
2,0,1,4,2,&
4,0,2,1,4,&
4,0,4,1,2,&
2,0,2,4,1,&
2,0,4,2,1,&
9,1,2,3,4,&!8
5,1,2,4,3,&
5,1,3,2,4,&
3,1,4,2,3,&
3,1,3,4,2,&
5,1,4,3,2,&
5,2,1,3,4,&
5,2,1,4,3,&
3,3,1,2,4,&
3,4,1,2,3,&
3,3,1,4,2,&
3,4,1,3,2,&
3,2,3,1,4,&
3,2,4,1,3,&
5,3,2,1,4,&
3,4,2,1,3,&
5,3,4,1,2,&
3,4,3,1,2,&
3,2,3,4,1,&
3,2,4,3,1,&
3,3,2,4,1,&
5,4,2,3,1,&
3,3,4,2,1,&
5,4,3,2,1],&
[5,52])
```

10.3 Interfaces

```
abstract interface
   function trafo_in(in)
     use kinds!NODEP!
     real(kind=double),dimension(3)::trafo_in
     real(kind=double),dimension(3),intent(in)::in
   end function trafo_in
end interface
abstract interface
   pure function coord_scalar_in(hyp)
     use kinds!NODEP!
     real(kind=double)::coord_scalar_in
     real(kind=double),dimension(3),intent(in)::hyp
   end function coord_scalar_in
end interface
abstract interface
   subroutine coord_hcd_in(hyp,cart,denom)
     use kinds!NODEP!
     real(kind=double),dimension(3),intent(in)::hyp
     real(kind=double),dimension(3),intent(out)::cart
     real(kind=double),intent(out)::denom
   end subroutine coord_hcd_in
end interface
interface
   pure function alphaspdf(Q)
     use kinds!NODEP!
     real(kind=double)::alphaspdf
     real(kind=double),intent(in)::Q
   end function alphaspdf
end interface
interface
   pure subroutine evolvepdf(x,q,f)
     use kinds!NODEP!
     real(kind=double),intent(in)::x,q
     real(kind=double),intent(out),dimension(-6:6)::f
   end subroutine evolvepdf
end interface
real(kind=double)::pts2_scale
```

10.4 Implementierung der Prozeduren

muli_get_state_transformations(inout_kind,lha_flavors) result

```
pure function muli_get_state_transformations(inout_kind,lha_flavors) result(transformations)
  integer,intent(in)::inout_kind
  integer,dimension(4),intent(in)::lha_flavors
  integer,dimension(4)::signature
  logical,dimension(3)::transformations
  where(lha_flavors>0)
     signature=1
```

```
    elsewhere(lha_flavors<0)
       signature=-1
    elsewhere
       signature=0
    end where
    !print *,"inout_kind=",inout_kind
    !print *,"lha_flavors=",lha_flavors
    !print *,"signature",signature
    if(&
         (sum(inout_signatures(1:2,inout_kind))==sum(signature(1:2))).and.&
         (sum(inout_signatures(3:4,inout_kind))==sum(signature(3:4)))&
         )then
       transformations(1)=.false.
    else
       transformations(1)=.true.
       signature=-signature
    end if
    if(all(inout_signatures(1:2,inout_kind)==signature(1:2)))then
       transformations(2)=.false.
    else
       transformations(2)=.true.
    end if
    if(all(inout_signatures(3:4,inout_kind)==signature(3:4)))then
       transformations(3)=.false.
    else
       transformations(3)=.true.
    end if
    !print *,"signature",signature
    !print *,"transformations=",transformations
  end function muli_get_state_transformations
```

id

```
  pure function id(a)
    real(kind=double),dimension(:),intent(in)::a
    real(kind=double),dimension(size(a))::id
    id=a
  end function id
```

h_to_c_ort

```
  pure function h_to_c_ort(hyp)
    real(kind=double),dimension(3)::h_to_c_ort
    real(kind=double),dimension(3),intent(in)::hyp
    h_to_c_ort=&
         &[sqrt(sqrt(((hyp(1)*(1D0-hyp(3)))+hyp(3))**2+(hyp(2)-(5D-1))**2)&
                     -(hyp(2)-(5D-1)))&
         &,sqrt(sqrt(((hyp(1)*(1D0-hyp(3)))+hyp(3))**2+(hyp(2)-(5D-1))**2)&
                     +(hyp(2)-(5D-1)))&
         &,hyp(3)]
  end function h_to_c_ort
```

c_to_h_ort

```
pure function c_to_h_ort(cart)
  real(kind=double),dimension(3)::c_to_h_ort
  real(kind=double),dimension(3),intent(in)::cart
  c_to_h_ort=[(cart(3)-(cart(1)*cart(2)))/(cart(3)-1D0),&
              (1D0 - cart(1)**2 + cart(2)**2)/2D0,cart(3)]
end function c_to_h_ort
```

h_to_c_noparam

```
pure function h_to_c_noparam(hyp)
  real(kind=double),dimension(2)::h_to_c_noparam
  real(kind=double),dimension(2),intent(in)::hyp
  h_to_c_noparam=&
       &[sqrt(sqrt(hyp(1)**8+(((hyp(2)-(5D-1))**3)*4)**2)-((hyp(2)-(5D-1))**3)*4)&
       &,sqrt(sqrt(hyp(1)**8+(((hyp(2)-(5D-1))**3)*4)**2)+((hyp(2)-(5D-1))**3)*4)]
end function h_to_c_noparam
```

c_to_h_noparam

```
pure function c_to_h_noparam(cart)
  real(kind=double),dimension(2)::c_to_h_noparam
  real(kind=double),dimension(2),intent(in)::cart
  c_to_h_noparam=&
       &[sqrt(sqrt(cart(1)*cart(2)))&
       &,(1D0+sign(abs((cart(2)**2) - (cart(1)**2))**(1/3D0),cart(2)-cart(1)))/2D0]
end function c_to_h_noparam
```

h_to_c_param

```
pure function h_to_c_param(hyp)
  real(kind=double),dimension(3)::h_to_c_param
  real(kind=double),dimension(3),intent(in)::hyp
  h_to_c_param=&
       &[sqrt(sqrt((((hyp(1)**4)*(1D0-hyp(3)))+hyp(3))**2+(((hyp(2)-(5D-1))**3)*4)**2)&
                   -((hyp(2)-(5D-1))**3)*4)&
       &,sqrt(sqrt((((hyp(1)**4)*(1D0-hyp(3)))+hyp(3))**2+(((hyp(2)-(5D-1))**3)*4)**2)&
                   +((hyp(2)-(5D-1))**3)*4)&
       &,hyp(3)]
end function h_to_c_param
```

c_to_h_param

```
pure function c_to_h_param(cart)
  real(kind=double),dimension(3)::c_to_h_param
  real(kind=double),dimension(3),intent(in)::cart
  c_to_h_param=&
       &[(((cart(1)*cart(2)) - cart(3))/(1D0 - cart(3)))**(1/4D0)&
       &,(1D0+sign(abs((cart(2)**2) - (cart(1)**2))**(1/3D0),cart(2)-cart(1)))/2D0&
       &,cart(3)]
end function c_to_h_param
```

h_to_c_smooth

```
pure function h_to_c_smooth(hyp)
  real(kind=double),dimension(3)::h_to_c_smooth
  real(kind=double),dimension(3),intent(in)::hyp
  real(kind=double)::h2
  h2=(((hyp(2)-5D-1)**3)*4D0+hyp(2)-5D-1)/2D0
  h_to_c_smooth=&
       &[sqrt(sqrt((((hyp(1)**4)*(1D0-hyp(3)))+hyp(3))**2+h2**2)-h2)&
       &,sqrt(sqrt((((hyp(1)**4)*(1D0-hyp(3)))+hyp(3))**2+h2**2)+h2)&
       &,hyp(3)]
end function h_to_c_smooth
```

c_to_h_smooth

```
pure function c_to_h_smooth(cart)
  real(kind=double),dimension(3)::c_to_h_smooth
  real(kind=double),dimension(3),intent(in)::cart
  c_to_h_smooth=&
       [((product(cart(1:2))-cart(3))/(1D0-cart(3)))**(1/4D0),&
       (3D0-3D0**(2D0/3)/&
       (-9D0*cart(1)**2 + 9D0*cart(2)**2 + sqrt(3D0+81D0*(cart(1)**2-cart(2)**2)**2))&
         **(1D0/3)&
       + 3**(1D0/3)*(-9D0*cart(1)**2 + 9D0*cart(2)**2 + sqrt(3D0 + 81D0*(cart(1)**2&
       - cart(2)**2)**2))**(1D0/3))/6D0,cart(3)]
end function c_to_h_smooth
```

h_to_c_ort_def

```
pure function h_to_c_ort_def(hyp)
  real(kind=double),dimension(3)::h_to_c_ort_def
  real(kind=double),dimension(3),intent(in)::hyp
  h_to_c_ort_def=h_to_c_ort([hyp(1),hyp(2),pts2_scale])
end function h_to_c_ort_def
```

c_to_h_ort_def

```
pure function c_to_h_ort_def(cart)
  real(kind=double),dimension(3)::c_to_h_ort_def
  real(kind=double),dimension(3),intent(in)::cart
  c_to_h_ort_def=c_to_h_ort([cart(1),cart(2),pts2_scale])
end function c_to_h_ort_def
```

h_to_c_param_def

```
pure function h_to_c_param_def(hyp)
  real(kind=double),dimension(3)::h_to_c_param_def
  real(kind=double),dimension(3),intent(in)::hyp
  h_to_c_param_def=h_to_c_param([hyp(1),hyp(2),pts2_scale])
end function h_to_c_param_def
```

c_to_h_param_def

```
pure function c_to_h_param_def(cart)
  real(kind=double),dimension(3)::c_to_h_param_def
  real(kind=double),dimension(3),intent(in)::cart
  if(product(cart(1:2))>=pts2_scale)then
     c_to_h_param_def=c_to_h_param([cart(1),cart(2),pts2_scale])
  else
     c_to_h_param_def=[-1D0,-1D0,-1D0]
  end if
end function c_to_h_param_def
```

h_to_c_smooth_def

```
pure function h_to_c_smooth_def(hyp)
  real(kind=double),dimension(3)::h_to_c_smooth_def
  real(kind=double),dimension(3),intent(in)::hyp
  h_to_c_smooth_def=h_to_c_smooth([hyp(1),hyp(2),pts2_scale])
end function h_to_c_smooth_def
```

c_to_h_smooth_def

```
pure function c_to_h_smooth_def(cart)
  real(kind=double),dimension(3)::c_to_h_smooth_def
  real(kind=double),dimension(3),intent(in)::cart
  if(product(cart(1:2))>=pts2_scale)then
     c_to_h_smooth_def=c_to_h_smooth([cart(1),cart(2),pts2_scale])
  else
     c_to_h_smooth_def=[-1D0,-1D0,-1D0]
  end if
end function c_to_h_smooth_def
```

voxel_h_to_c_ort

```
pure function voxel_h_to_c_ort(hyp)
  real(kind=double)::voxel_h_to_c_ort
  real(kind=double),dimension(3),intent(in)::hyp
  real(kind=double)::T,TH1
  T=1D0-hyp(3)
  TH1=T*(1D0-hyp(1))
  voxel_h_to_c_ort=Sqrt(T**2/(5D0-4D0*(1D0-hyp(2))*hyp(2)-4D0*(2D0-TH1)*TH1))
end function voxel_h_to_c_ort
```

voxel_c_to_h_ort

```
pure function voxel_c_to_h_ort(cart)
  real(kind=double)::voxel_c_to_h_ort
  real(kind=double),dimension(3),intent(in)::cart
  real(kind=double)::P
  P=product(cart(1:2))
  if(P>cart(3))then
     voxel_c_to_h_ort=(cart(1)**2 + cart(2)**2)/(1D0-cart(3))
  else
     voxel_c_to_h_ort=0D0
```

```
    end if
  end function voxel_c_to_h_ort
```

voxel_h_to_c_noparam

```
  pure function voxel_h_to_c_noparam(hyp)
    real(kind=double)::voxel_h_to_c_noparam
    real(kind=double),dimension(3),intent(in)::hyp
    voxel_h_to_c_noparam=&
      12D0*Sqrt((hyp(1)**6*(1D0-2D0*hyp(2))**4)/(4*hyp(1)**8+(1D0-2D0*hyp(2))**6))
  end function voxel_h_to_c_noparam
```

voxel_c_to_h_noparam

```
  pure function voxel_c_to_h_noparam(cart)
    real(kind=double)::voxel_c_to_h_noparam
    real(kind=double),dimension(3),intent(in)::cart
    real(kind=double)::P
    voxel_c_to_h_noparam=(cart(1)**2+cart(2)**2)/(12D0*(cart(1)*cart(2))**(3D0/4D0)&
         *(cart(2)**2+cart(1)**2)**(2D0/3D0))
  end function voxel_c_to_h_noparam
```

voxel_h_to_c_param

```
  pure function voxel_h_to_c_param(hyp)
    real(kind=double)::voxel_h_to_c_param
    real(kind=double),dimension(3),intent(in)::hyp
    voxel_h_to_c_param=12*Sqrt((hyp(1)**6*(1D0-2D0*hyp(2))**4*(hyp(3)-1D0)**2)&
         /((1D0-2D0*hyp(2))**6+4D0*(hyp(3)-(hyp(1)**4*(hyp(3)-1D0)))**2))
  end function voxel_h_to_c_param
```

voxel_c_to_h_param

```
  pure function voxel_c_to_h_param(cart)
    real(kind=double)::voxel_c_to_h_param
    real(kind=double),dimension(3),intent(in)::cart
    real(kind=double)::P,T,CP,CM
    P=product(cart(1:2))
    if(P>cart(3))then
       P=P-cart(3)
       CP=cart(1)**2+cart(2)**2
       CM=abs(cart(2)**2-cart(1)**2)
       T=1-cart(3)
       voxel_c_to_h_param=(Cp*sqrt(sqrt(P/T)))/(12*Cm**(2D0/3D0)*P)
    else
       voxel_c_to_h_param=0D0
    end if
  end function voxel_c_to_h_param
```

voxel_h_to_c_smooth

```
  pure function voxel_h_to_c_smooth(hyp)
    real(kind=double)::voxel_h_to_c_smooth
    real(kind=double),dimension(3),intent(in)::hyp
    real(kind=double)::T
    T=1D0-hyp(3)
    voxel_h_to_c_smooth=&
         &8D0*(hyp(1)**3*(1D0+3D0*(hyp(2)-1D0)*hyp(2))*T)&
         &/sqrt((1D0-2D0*hyp(2)*(2D0+hyp(2)*(2D0*hyp(2)-3D0)))**2+4D0*(1D0+(hyp(1)**4-1D0)*T)**2)
  end function voxel_h_to_c_smooth
```

voxel_c_to_h_smooth

```
  pure function voxel_c_to_h_smooth(cart)
    real(kind=double)::voxel_c_to_h_smooth
    real(kind=double),dimension(3),intent(in)::cart
    real(kind=double)::P,S,T,CM,CP
    P=product(cart(1:2))
    if(P>cart(3))then
       P=P-cart(3)
       CP=cart(1)**2+cart(2)**2
       CM=cart(2)**2-cart(1)**2
       T=1-cart(3)
       S=sqrt(3D0+81D0*cm**2)
       voxel_c_to_h_smooth=(3D0**(1D0/3D0)*Cp*(3D0**(1D0/3D0)+(9D0*Cm+S)**(2D0/3D0))&
            *sqrt(sqrt(P/T)))/(4D0*P*S*(9D0*Cm+S)**(1D0/3D0))
    else
       voxel_c_to_h_smooth=0D0
    end if
end function voxel_c_to_h_smooth

!
```

voxel_h_to_c_ort_def

```
  pure function voxel_h_to_c_ort_def(hyp)
    real(kind=double)::voxel_h_to_c_ort_def
    real(kind=double),dimension(3),intent(in)::hyp
    voxel_h_to_c_ort_def=voxel_h_to_c_ort(hyp)
  end function voxel_h_to_c_ort_def
```

voxel_c_to_h_ort_def

```
  pure function voxel_c_to_h_ort_def(cart)
    real(kind=double)::voxel_c_to_h_ort_def
    real(kind=double),dimension(3),intent(in)::cart
    voxel_c_to_h_ort_def=voxel_c_to_h_ort(cart)
  end function voxel_c_to_h_ort_def
```

voxel_h_to_c_param_def

```
  pure function voxel_h_to_c_param_def(hyp)
    real(kind=double)::voxel_h_to_c_param_def
    real(kind=double),dimension(3),intent(in)::hyp
```

```
    voxel_h_to_c_param_def=voxel_h_to_c_param(hyp)
  end function voxel_h_to_c_param_def
```

voxel_c_to_h_param_def

```
  pure function voxel_c_to_h_param_def(cart)
    real(kind=double)::voxel_c_to_h_param_def
    real(kind=double),dimension(3),intent(in)::cart
    voxel_c_to_h_param_def=voxel_c_to_h_param(cart)
  end function voxel_c_to_h_param_def
```

voxel_h_to_c_smooth_def

```
  pure function voxel_h_to_c_smooth_def(hyp)
    real(kind=double)::voxel_h_to_c_smooth_def
    real(kind=double),dimension(3),intent(in)::hyp
    voxel_h_to_c_smooth_def=voxel_h_to_c_smooth(hyp)
  end function voxel_h_to_c_smooth_def
```

voxel_c_to_h_smooth_def

```
  pure function voxel_c_to_h_smooth_def(cart)
    real(kind=double)::voxel_c_to_h_smooth_def
    real(kind=double),dimension(3),intent(in)::cart
    voxel_c_to_h_smooth_def=voxel_c_to_h_smooth(cart)
  end function voxel_c_to_h_smooth_def
```

denom_cart

```
  pure function denom_cart(cart)
    real(kind=double)::denom_cart
    real(kind=double),dimension(3),intent(in)::cart
    denom_cart=1D0/(864D0*Sqrt(cart(3)**3*(1D0-cart(3)/product(cart(1:2)))))
  end function denom_cart
```

denom_ort

```
  pure function denom_ort(hyp)
    real(kind=double)::denom_ort
    real(kind=double),dimension(3),intent(in)::hyp
    real(kind=double)::Y,P
    Y=(1D0-2D0*hyp(2))**2
    P=1D0-hyp(3)
    if(hyp(1)>0D0.and.hyp(3)>0D0)then
       denom_ort=sqrt((P + (-1 + Hyp(1))*P**2)&
                     /(746496*hyp(1)*hyp(3)**3*(4*(1 + (-1 + hyp(1))*P)**2 + Y)))

    else
       denom_ort=0D0
    end if
  end function denom_ort
```

denom_param

```
pure function denom_param(hyp)
  real(kind=double)::denom_param
  real(kind=double),dimension(3),intent(in)::hyp
  real(kind=double)::X,Y,P
  X=hyp(1)**4
  Y=1D0-2D0*hyp(2)
  P=1D0-hyp(3)
  if(hyp(3)>0D0)then
     denom_param=sqrt((P*(1+P*(X-1))*Sqrt(X)*Y**4)/(5184*(4*(1+P*(X-1))**2+Y**6)*hyp(3)**3))
  else
     denom_param=0D0
  end if
end function denom_param
```

denom_param_reg

```
pure function denom_param_reg(hyp)
  real(kind=double)::denom_param_reg
  real(kind=double),dimension(3),intent(in)::hyp
  real(kind=double)::X,Y,P
  X=hyp(1)**4
  Y=1D0-2D0*hyp(2)
  P=1D0-hyp(3)
  if(hyp(3)>0D0)then
     denom_param_reg=sqrt((P*(1+P*(X-1))*Sqrt(X)*Y**4)&
                         /(5184*(4*(1+P*(X-1))**2+Y**6)*(hyp(3)+norm2_p_t_0)**3))
  else
     denom_param_reg=0D0
  end if
end function denom_param_reg
```

denom_smooth

```
pure function denom_smooth(hyp)
  real(kind=double)::denom_smooth
  real(kind=double),dimension(3),intent(in)::hyp
  real(kind=double)::X,Y,P
  X=hyp(1)**2
  Y=(1D0-2D0*hyp(2))**2
  P=1D0-hyp(3)
  if(hyp(3)>0D0)then
     denom_smooth=sqrt((P*X*(1 + P*(-1 + X**2))*(1 + 3*Y)**2)/(46656*hyp(3)**3&
          *(16*(1 + P*(-1 + X**2))**2 + Y + 2*Y**2 + Y**3)))
  else
     denom_smooth=0D0
  end if
end function denom_smooth
```

denom_smooth_reg

```
pure function denom_smooth_reg(hyp)
  real(kind=double)::denom_smooth_reg
```

```
    real(kind=double),dimension(3),intent(in)::hyp
    real(kind=double)::X,Y,P
    X=hyp(1)**2
    Y=(1D0-2D0*hyp(2))**2
    P=1D0-hyp(3)
    if(hyp(3)>0D0)then
       denom_smooth_reg=&
         sqrt((P*X*(1 + P*(-1 + X**2))*(1 + 3*Y)**2)&
              /(46656*(hyp(3)+norm2_p_t_0)**3&
                *(16*(1 + P*(-1 + X**2))**2 + Y + 2*Y**2 + Y**3)))
    else
       denom_smooth_reg=0D0
    end if
  end function denom_smooth_reg
```

denom_cart_save

```
   pure function denom_cart_save(cart)
    real(kind=double)::denom_cart_save
    real(kind=double),dimension(3),intent(in)::cart
    if(product(cart(1:2))>cart(3))then
       denom_cart_save=denom_cart(cart)
    else
       denom_cart_save=0D0
    end if
  end function denom_cart_save
```

denom_ort_save

```
  pure function denom_ort_save(hyp)
    real(kind=double)::denom_ort_save
    real(kind=double),dimension(3),intent(in)::hyp
    real(kind=double)::Y,Z,W
    real(kind=double),dimension(3)::cart
    cart=h_to_c_ort(hyp)
    if(cart(1)>1D0.or.cart(2)>1D0)then
       denom_ort_save=0D0
    else
       denom_ort_save=denom_ort(hyp)
    end if
  end function denom_ort_save
```

denom_param_save

```
  pure function denom_param_save(hyp)
    real(kind=double)::denom_param_save
    real(kind=double),dimension(3),intent(in)::hyp
    real(kind=double)::Y,Z,W
    real(kind=double),dimension(3)::cart
    cart=h_to_c_param(hyp)
    if(cart(1)>1D0.or.cart(2)>1D0)then
       denom_param_save=0D0
```

```
    else
       denom_param_save=denom_param(hyp)
    end if
  end function denom_param_save
```

! pure **denom_smooth_save**

```
  function denom_smooth_save(hyp)
    real(kind=double)::denom_smooth_save
    real(kind=double),dimension(3),intent(in)::hyp
    real(kind=double)::Y,Z,W
    real(kind=double),dimension(3)::cart
    cart=h_to_c_smooth(hyp)
    if(cart(1)>1D0.or.cart(2)>1D0)then
       denom_smooth_save=0D0
    else
       denom_smooth_save=denom_smooth(hyp)
    end if
  end function denom_smooth_save
```

denom_cart_cuba_int

```
  subroutine denom_cart_cuba_int(d_cart,cart,d_denom,denom,pt2s)
    real(kind=double),dimension(3),intent(in)::cart
    real(kind=double),dimension(1),intent(out)::denom
    real(kind=double),intent(in) :: pt2s
    integer,intent(in)::d_cart,d_denom
    denom(1)=denom_cart_save([cart(1),cart(2),pt2s])
  end subroutine denom_cart_cuba_int
```

denom_ort_cuba_int

```
  subroutine denom_ort_cuba_int(d_hyp,hyp,d_denom,denom,pt2s)
    real(kind=double),dimension(3),intent(in)::hyp
    real(kind=double),dimension(1),intent(out)::denom
    real(kind=double),intent(in) :: pt2s
    integer,intent(in)::d_hyp,d_denom
    denom(1)=denom_ort_save([hyp(1),hyp(2),pt2s])
  end subroutine denom_ort_cuba_int
```

denom_param_cuba_int

```
  subroutine denom_param_cuba_int(d_hyp,hyp,d_denom,denom,pt2s)
    real(kind=double),dimension(3),intent(in)::hyp
    real(kind=double),dimension(1),intent(out)::denom
    real(kind=double),intent(in) :: pt2s
    integer,intent(in)::d_hyp,d_denom
    denom(1)=denom_param_save([hyp(1),hyp(2),pt2s])
  end subroutine denom_param_cuba_int
```

denom_smooth_cuba_int

```
subroutine denom_smooth_cuba_int(d_hyp,hyp,d_denom,denom,pt2s)
  real(kind=double),dimension(3),intent(in)::hyp
  real(kind=double),dimension(1),intent(out)::denom
  real(kind=double),intent(in) :: pt2s
  integer,intent(in)::d_hyp,d_denom
  denom(1)=denom_smooth_save([hyp(1),hyp(2),pt2s])
end subroutine denom_smooth_cuba_int
```

coordinates_hcd_cart

```
subroutine coordinates_hcd_cart(hyp,cart,denom)
  real(kind=double),dimension(3),intent(in)::hyp
  real(kind=double),dimension(3),intent(out)::cart
  real(kind=double),intent(out)::denom
  cart=hyp
  denom=denom_cart_save(cart)
end subroutine coordinates_hcd_cart
```

coordinates_hcd_ort

```
subroutine coordinates_hcd_ort(hyp,cart,denom)
  real(kind=double),dimension(3),intent(in)::hyp
  real(kind=double),dimension(3),intent(out)::cart
  real(kind=double),intent(out)::denom
  cart=h_to_c_ort(hyp)
  denom=denom_ort(hyp)
end subroutine coordinates_hcd_ort
```

coordinates_hcd_param

```
subroutine coordinates_hcd_param(hyp,cart,denom)
  real(kind=double),dimension(3),intent(in)::hyp
  real(kind=double),dimension(3),intent(out)::cart
  real(kind=double),intent(out)::denom
  cart=h_to_c_param(hyp)
  denom=denom_param(hyp)
end subroutine coordinates_hcd_param
```

coordinates_hcd_param_reg

```
subroutine coordinates_hcd_param_reg(hyp,cart,denom)
  real(kind=double),dimension(3),intent(in)::hyp
  real(kind=double),dimension(3),intent(out)::cart
  real(kind=double),intent(out)::denom
  cart=h_to_c_param(hyp)
  denom=denom_param_reg(hyp)
end subroutine coordinates_hcd_param_reg
```

coordinates_hcd_smooth

```
subroutine coordinates_hcd_smooth(hyp,cart,denom)
  real(kind=double),dimension(3),intent(in)::hyp
  real(kind=double),dimension(3),intent(out)::cart
```

```
    real(kind=double),intent(out)::denom
    cart=h_to_c_smooth(hyp)
    denom=denom_smooth(hyp)
  end subroutine coordinates_hcd_smooth
```

coordinates_hcd_smooth_reg

```
  subroutine coordinates_hcd_smooth_reg(hyp,cart,denom)
    real(kind=double),dimension(3),intent(in)::hyp
    real(kind=double),dimension(3),intent(out)::cart
    real(kind=double),intent(out)::denom
    cart=h_to_c_smooth(hyp)
    denom=denom_smooth_reg(hyp)
  end subroutine coordinates_hcd_smooth_reg
```

pdf_in_in_kind

```
  pure function pdf_in_in_kind(process_id,double_pdf_id,c1,c2,gev_pt)
    real(kind=double)::pdf_in_in_kind
    real(kind=double),intent(in)::c1,c2,gev_pt
    integer,intent(in)::process_id,double_pdf_id
    real(kind=double)::pdf1,pdf2
    call single_pdf(valid_processes(1,process_id),&
                    double_pdf_kinds(1,double_pdf_id),&
                    c1,&
                    gev_pt,&
                    pdf1)
    call single_pdf(valid_processes(2,process_id),&
                    double_pdf_kinds(2,double_pdf_id),&
                    c2,&
                    gev_pt,&
                    pdf2)
    pdf_in_in_kind=pdf1*pdf2
  contains
    pure subroutine single_pdf(flavor,pdf_kind,c,gev_pt,pdf)
      integer,intent(in)::flavor,pdf_kind
      real(kind=double),intent(in)::c,gev_pt
      real(kind=double),intent(out)::pdf
      real(kind=double),dimension(-6:6)::lha_pdf
      call evolvePDF(c,gev_pt,lha_pdf)
      select case(pdf_kind)
      case(1)
         pdf=lha_pdf(0)
      case(2)
         if(flavor==1.or.flavor==2)then
            pdf=lha_pdf(-flavor)
         else
            pdf=lha_pdf(flavor)
         end if
      case(3)
         pdf=lha_pdf(1)-lha_pdf(-1)
      case(4)
```

```
      pdf=lha_pdf(2)-lha_pdf(-2)
    end select
  end subroutine single_pdf
end function pdf_in_in_kind
```

ps_io_pol

```
elemental function ps_io_pol(process_io_id,pt2shat)
  real(kind=double)::ps_io_pol
  integer,intent(in)::process_io_id
  real(kind=double),intent(in)::pt2shat
  ps_io_pol=dot_product(&
       [1D0,pt2shat,pt2shat**2,pt2shat**3]&
       ,phase_space_coefficients_inout(1:4,valid_processes(6,process_io_id)))
end function ps_io_pol
```

interactions_dddsigma

```
  pure subroutine interactions_dddsigma(process_id,double_pdf_id,hyp,cart,dddsigma)
    real(kind=double),intent(out)::dddsigma
    integer,intent(in)::process_id,double_pdf_id
    real(kind=double),dimension(3),intent(in)::hyp
    real(kind=double),dimension(3),intent(out)::cart
    real(kind=double)::a,pt2shat,gev_pt
    cart=h_to_c_param(hyp)
    a=product(cart(1:2))
    if(cart(1)<=1D0.and.cart(2)<=1D0)then
       pt2shat=hyp(3)/a
       gev_pt=sqrt(hyp(3))*gev_pt_max
!       print *,process_id,pt2shat
       dddsigma=&
            &const_pref&
            &*alphasPDF(gev_pt)**2&
            &*ps_io_pol(process_id,pt2shat)&
            &*pdf_in_in_kind(process_id,double_pdf_id,cart(1),cart(2),gev_pt)&
            &*denom_param(hyp)&
            &/a
    else
       dddsigma=0D0
    end if
  end subroutine interactions_dddsigma
```

interactions_dddsigma_reg

```
pure subroutine interactions_dddsigma_reg(process_id,double_pdf_id,hyp,cart,dddsigma)
  real(kind=double),intent(out)::dddsigma
  integer,intent(in)::process_id,double_pdf_id
  real(kind=double),dimension(3),intent(in)::hyp
  real(kind=double),dimension(3),intent(out)::cart
  real(kind=double)::a,pt2shat,gev_pt,gev2_pt
  cart=h_to_c_param(hyp)
  a=product(cart(1:2))
```

```
    if(cart(1)<=1D0.and.cart(2)<=1D0)then
       pt2shat=hyp(3)/a
       gev_pt=sqrt(hyp(3))*gev_pt_max
       gev2_pt=hyp(3)*gev2_pt_max
!        print *,process_id,pt2shat
       dddsigma=&
            &const_pref&
            &*alphasPDF(sqrt(gev2_pt+gev2_p_t_0))**2&
            &*ps_io_pol(process_id,pt2shat)&
            &*pdf_in_in_kind(process_id,double_pdf_id,cart(1),cart(2),gev_pt)&
            &*denom_param_reg(hyp)&
            &/a
    else
       dddsigma=0D0
    end if
  end subroutine interactions_dddsigma_reg
```

interactions_dddsigma_print

```
  subroutine interactions_dddsigma_print(process_id,double_pdf_id,hyp,cart,dddsigma)
    real(kind=double),intent(out)::dddsigma
    integer,intent(in)::process_id,double_pdf_id
    real(kind=double),dimension(3),intent(in)::hyp
    real(kind=double),dimension(3),intent(out)::cart
    real(kind=double)::a,pt2shat,gev_pt
    cart=h_to_c_param(hyp)
    a=product(cart(1:2))
    if(cart(1)<=1D0.and.cart(2)<=1D0)then
       pt2shat=hyp(3)/a
       gev_pt=sqrt(hyp(3))*gev_pt_max
!        print *,process_id,pt2shat
       dddsigma=&
            &const_pref&
!             &*alphasPDF(gev_pt)**2&
            &*ps_io_pol(process_id,pt2shat)&
            &*pdf_in_in_kind(process_id,double_pdf_id,cart(1),cart(2),gev_pt)&
            &*denom_param(hyp)&
            &/a
    else
       dddsigma=0D0
    end if
    write(11,fmt=*)dddsigma,pt2shat,&
         pdf_in_in_kind(process_id,double_pdf_id,cart(1),cart(2),&
         gev_pt),ps_io_pol(process_id,pt2shat),const_pref,denom_param(hyp),a
    flush(11)
  end subroutine interactions_dddsigma_print
```

interactions_dddsigma_cart

```
  pure subroutine interactions_dddsigma_cart(process_id,double_pdf_id,cart,dddsigma)
    real(kind=double),intent(out)::dddsigma
    integer,intent(in)::process_id,double_pdf_id
```

```
    real(kind=double),dimension(3),intent(in)::cart
    real(kind=double)::a,pt2shat,gev_pt
    a=product(cart(1:2))
    if(cart(1)<=1D0.and.cart(2)<=1D0)then
       pt2shat=cart(3)/a
       gev_pt=sqrt(cart(3))*gev_pt_max
!       print *,process_id,pt2shat
       dddsigma=&
            &const_pref&
            &*alphasPDF(gev_pt)**2&
            &*ps_io_pol(process_id,pt2shat)&
            &*pdf_in_in_kind(process_id,double_pdf_id,cart(1),cart(2),gev_pt)&
            &*denom_cart(cart)&
            &/a
    else
       dddsigma=0D0
    end if
  end subroutine interactions_dddsigma_cart
```

cuba_gg_me_smooth

```
  subroutine cuba_gg_me_smooth(d_hyp,hyp,d_me,me,pt2s)
    integer,intent(in)::d_hyp,d_me
    real(kind=double),dimension(d_hyp),intent(in)::hyp
    real(kind=double),dimension(1),intent(out)::me
    real(kind=double),dimension(3)::cart
    real(kind=double),intent(in)::pt2s
    real(kind=double)::p,p2
    if(d_hyp==3)then
       p=hyp(3)
       p2=hyp(3)**2
    else
       if(d_hyp==2)then
          p=sqrt(pt2s)
          p2=pt2s
       end if
    end if
    cart=h_to_c_smooth([hyp(1),hyp(2),p2])
    if(p>pts_min.and.product(cart(1:2))>p2)then
       me(1)=&
            &const_pref&
            &*alphasPDF(p*gev_pt_max)**2&
            &*ps_io_pol(109,p2)&
            &*pdf_in_in_kind(109,11,cart(1),cart(2),p2)&
            &*denom_smooth([hyp(1),hyp(2),p2])&
            &/product(cart(1:2))
    else
       me(1)=0D0
    end if
  end subroutine cuba_gg_me_smooth
```

cuba_gg_me_param

```
subroutine cuba_gg_me_param(d_hyp,hyp,d_me,me,pt2s)
  integer,intent(in)::d_hyp,d_me
  real(kind=double),dimension(d_hyp),intent(in)::hyp
  real(kind=double),dimension(1),intent(out)::me
  real(kind=double),dimension(3)::cart
  real(kind=double),intent(in)::pt2s
  real(kind=double)::p,p2
  if(d_hyp==3)then
     p=hyp(3)
     p2=hyp(3)**2
  else
     if(d_hyp==2)then
        p=sqrt(pt2s)
        p2=pt2s
     end if
  end if
  cart=h_to_c_param([hyp(1),hyp(2),p2])
  if(p>pts_min.and.product(cart(1:2))>p2)then
     me(1)=&
          &const_pref&
          &*alphasPDF(p*gev_pt_max)**2&
          &*ps_io_pol(109,p2)&
          &*pdf_in_in_kind(109,11,cart(1),cart(2),p2)&
          &*denom_param([hyp(1),hyp(2),p2])&
          &/product(cart(1:2))
  else
     me(1)=0D0
  end if
end subroutine cuba_gg_me_param
```

cuba_gg_me_ort

```
subroutine cuba_gg_me_ort(d_hyp,hyp,d_me,me,pt2s)
  integer,intent(in)::d_hyp,d_me
  real(kind=double),dimension(d_hyp),intent(in)::hyp
  real(kind=double),dimension(1),intent(out)::me
  real(kind=double),dimension(3)::cart
  real(kind=double),intent(in)::pt2s
  real(kind=double)::p,p2
  if(d_hyp==3)then
     p=hyp(3)
     p2=hyp(3)**2
  else
     if(d_hyp==2)then
        p=sqrt(pt2s)
        p2=pt2s
     end if
  end if
  cart=h_to_c_ort([hyp(1),cart(2),p2])
  if(p>pts_min.and.product(cart(1:2))>p2)then
     me(1)=&
```

```
            &const_pref&
            &*alphasPDF(p*gev_pt_max)**2&
            &*ps_io_pol(109,p2)&
            &*pdf_in_in_kind(109,11,cart(1),cart(2),p2)&
            &*denom_ort([hyp(1),hyp(2),p2])&
            &/product(cart(1:2))
    else
       me(1)=0D0
    end if
  end subroutine cuba_gg_me_ort
```

cuba_gg_me_cart

```
  subroutine cuba_gg_me_cart(d_cart,cart,d_me,me,pt2s)
    integer,intent(in)::d_cart,d_me
    real(kind=double),dimension(d_cart),intent(in)::cart
    real(kind=double),dimension(1),intent(out)::me
    real(kind=double),intent(in)::pt2s
    real(kind=double)::a,p,p2
    if(d_cart==3)then
       p=cart(3)
       p2=cart(3)**2
    else
       if(d_cart==2)then
          p=sqrt(pt2s)
          p2=pt2s
       end if
    end if
    a=product(cart(1:2))
    if(p>pts_min.and.a>p2)then
       me(1)=&
            &const_pref&
            &*alphasPDF(p*gev_pt_max)**2&
            &*ps_io_pol(109,p2)&
            &*pdf_in_in_kind(109,11,cart(1),cart(2),p2)&
            &*denom_cart([cart(1),cart(2),p2])&
            &/a
    else
       me(1)=0D0
    end if
  end subroutine cuba_gg_me_cart
```

interactions_proton_proton_integrand_generic_17_reg

```
  subroutine interactions_proton_proton_integrand_generic_17_reg(hyp_2,trafo,f,pt)
    real(kind=double),dimension(2),intent(in)::hyp_2
    procedure(coord_hcd_in)::trafo
    real(kind=double),dimension(17),intent(out)::f
    class(transversal_momentum_type), intent(in) :: pt
    real(kind=double),dimension(3)::cart,hyp_3
    real(kind=double),dimension(5)::psin
    real(kind=double),dimension(-6:6)::c,d
```

```
real(kind=double)::gev_pt,gev2_pt,pts,pt2s,pt2shat,a,&
     pdf_seaquark_seaquark,pdf_seaquark_gluon,pdf_gluon_gluon,&
     pdf_up_seaquark,pdf_up_gluon,pdf_down_seaquark,pdf_down_gluon,&
     v1u,v1d,v2u,v2d,denom

pts=pt%get_unit_scale()
pt2s=pt%get_unit2_scale()
gev_pt=pt%get_gev_scale()
gev2_pt=pt%get_gev2_scale()

hyp_3(1:2)=hyp_2
hyp_3(3)=pt2s
call trafo(hyp_3,cart,denom)
a=product(cart(1:2))
if(cart(1)<=1D0.and.cart(2)<=1D0.and.a>pt2s)then
   pt2shat=pt2s/a

   ! phase space polynom
   psin=matmul([1D0,pt2shat,pt2shat**2,pt2shat**3],phase_space_coefficients_in)
   ! pdf
   call evolvepdf(cart(1),gev_pt,c)
   call evolvepdf(cart(2),gev_pt,d)
   !c=[1,1,1,1,1,1,1,1,1,1,1,1,1]*1D0
   !d=c
   v1d=c(1)-c(-1)
   v1u=c(2)-c(-2)
   v2d=d(1)-d(-1)
   v2u=d(2)-d(-2)
   c(1)=c(-1)
   c(2)=c(-2)
   d(1)=d(-1)
   d(2)=d(-2)
   f(1)=0D0
   !gluon_gluon
   f( 2)=(&
        !type5
        c(0)*d(0)&
        )*psin(5)
   !gluon_seaquark
   f( 3)=(&
        !type4
        c(0)*d(-4)+c(0)*d(-3)+c(0)*d(-2)+c(0)*d(-1)+c(0)*d(1)+c(0)*d(2)+c(0)*d(3)+c(0)*d(4)&
        )*psin(4)
   !gluon_down
   f( 4)=(&
        !type4
        c( 0)*v2d&
        )*psin(4)
   !gluon_up
   f( 5)=(&
        !type4
```

```
      c(0)*v2u&
      )*psin(4)
 !seaquark_gluon
 f( 6)=(&
      !type4
      c(-4)*d(0)+c(-3)*d(0)+c(-2)*d(0)+c(-1)*d(0)&
      +c(1)*d(0)+c( 2)*d(0)+c( 3)*d(0)+c( 4)*d(0)&
      )*psin(4)
 !seaquark_seaquark
 f( 7)=&
      !type1
      (c(-4)*d(-3)+c(-4)*d(-2)+c(-4)*d(-1)+c(-4)*d( 1)+c(-4)*d( 2)+c(-4)*d( 3)+&
       c(-3)*d(-4)+c(-3)*d(-2)+c(-3)*d(-1)+c(-3)*d( 1)+c(-3)*d( 2)+c(-3)*d( 4)+&
       c(-2)*d(-4)+c(-2)*d(-3)+c(-2)*d(-1)+c(-2)*d( 1)+c(-2)*d( 3)+c(-2)*d( 4)+&
       c(-1)*d(-4)+c(-1)*d(-3)+c(-1)*d(-2)+c(-1)*d( 2)+c(-1)*d( 3)+c(-1)*d( 4)+&
       c( 1)*d(-4)+c( 1)*d(-3)+c( 1)*d(-2)+c( 1)*d( 2)+c( 1)*d( 3)+c( 1)*d( 4)+&
       c( 2)*d(-4)+c( 2)*d(-3)+c( 2)*d(-1)+c( 2)*d( 1)+c( 2)*d( 3)+c( 2)*d( 4)+&
       c( 3)*d(-4)+c( 3)*d(-2)+c( 3)*d(-1)+c( 3)*d( 1)+c( 3)*d( 2)+c( 3)*d( 4)+&
       c( 4)*d(-3)+c( 4)*d(-2)+c( 4)*d(-1)+c( 4)*d( 1)+c( 4)*d( 2)+c( 4)*d( 3))&
       *psin(1)&
      !type2
       +( c(-4)*d(-4)+c(-3)*d(-3)+c(-2)*d(-2)+c(-1)*d(-1)&
         +c( 4)*d( 4)+c( 3)*d( 3)+c( 2)*d( 2)+c( 1)*d( 1))&
       *psin(2)&
       !type3
       +( c(-4)*d( 4)+c(-3)*d( 3)+c(-2)*d( 2)+c(-1)*d( 1)&
         +c( 4)*d(-4)+c( 3)*d(-3)+c( 2)*d(-2)+c( 1)*d(-1))&
       *psin(3)
 !seaquark_down
 f( 8)=&
      !type1
      (c(-4)*v2d+c(-3)*v2d+c(-2)*v2d+c( 2)*v2d+c( 3)*v2d+c( 4)*v2d)&
      *psin(1)&
      !type2
      +c( 1)*v2d&
      *psin(2)&
      !type3
      +c(-1)*v2d&
      *psin(3)
 !seaquark_up
 f( 9)=&
      !type1
      (c(-4)*v2u+c(-3)*v2u+c(-1)*v2u+c( 1)*v2u+c( 3)*v2u+c( 4)*v2u)&
      *psin(1)&
      !type2
      +c(2)*v2u&
      *psin(2)&
      !type3
      +c(-2)*v2u&
      *psin(3)
 !down_gluon
```

```
      f(10)=(&
           !type4
           v1d*d( 0)&
           )*psin(4)
      !down_seaquark
      f(11)=&
           !type1
           (v1d*d(-4)+v1d*d(-3)+v1d*d(-2)+v1d*d( 2)+v1d*d( 3)+v1d*d( 4))&
           *psin(1)&
           !type2
           +v1d*d( 1)&
           *psin(2)&
           !type3
           +v1d*d(-1)&
           *psin(3)
      !down_down
      f(12)=v1d*v2d*psin(2)
      !down_up
      f(13)=v1d*v2u*psin(1)
      !up_gluon
      f(14)=(&
           !type4
           &v1u*d(0)&
           &)*psin(4)
      !up_seaquark
      f(15)=&
           !type1
           (v1u*d(-4)+v1u*d(-3)+v1u*d(-1)+v1u*d( 1)+v1u*d( 3)+v1u*d( 4))&
           *psin(1)&
           !type2
           +v1u*d(2)&
           *psin(2)&
           !type3
           +v1u*d(-2)&
           *psin(3)
      !up_down
      f(16)=v1u*v2d*psin(1)
      !up_up
      f(17)=v1u*v2u*psin(2)
      f=f&
      *const_pref&
           *alphasPDF(sqrt(gev2_pt+gev2_p_t_0))**2&
           *denom&
           /a
      !       print *,const_pref,alphasPDF(gev_pt)**2,denom_smooth(hyp),a
   else
      f=[0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0,0D0]
   end if
!    print *,pt2shat,c(0)*d(0),psin(5),const_pref,alphasPDF(gev_pt)**2,denom,a
  end subroutine interactions_proton_proton_integrand_generic_17_reg
```

interactions_proton_proton_integrand_param_17_reg

```
subroutine interactions_proton_proton_integrand_param_17_reg(d_hyp,hyp_2,d_f,f,pt)
  integer,intent(in)::d_hyp,d_f
  real(kind=double),dimension(2),intent(in)::hyp_2
  real(kind=double),dimension(17),intent(out)::f
  class(transversal_momentum_type), intent(in) :: pt
  call interactions_proton_proton_integrand_generic_17_reg&
    (hyp_2,coordinates_hcd_param_reg,f,pt)
  !    write (53,*)hyp_2,momentum_get_pts_scale(),f
end subroutine interactions_proton_proton_integrand_param_17_reg
```

interactions_proton_proton_integrand_smooth_17_reg

```
subroutine interactions_proton_proton_integrand_smooth_17_reg(d_hyp,hyp_2,d_f,f,pt)
  integer,intent(in)::d_hyp,d_f
  real(kind=double),dimension(2),intent(in)::hyp_2
  real(kind=double),dimension(17),intent(out)::f
  class(transversal_momentum_type), intent(in) :: pt
  call interactions_proton_proton_integrand_generic_17_reg&
    (hyp_2,coordinates_hcd_smooth_reg,f,pt)
  !    write (53,*)hyp_2,momentum_get_pts_scale(),f
end subroutine interactions_proton_proton_integrand_smooth_17_reg
```

11 Modul muli_mcint

11.1 Abhängigkeiten

```
use muli_basic
use tao_random_numbers !NODEP!
use muli_interactions
```

11.2 Parameter

```
integer,private,parameter::max_n=2**30
real(kind=double),private,parameter::max_d=1D0*max_n
real(kind=double),private,parameter,dimension(2,2)::unit_square=reshape([0D0,0D0,1D0,1D0],[2,2])
```

11.3 Derived Types

11.3.1 sample_region_type

```
type,extends(serializable_class)::sample_region_type
   integer::n_hits=0
   integer::n_alloc=0
   real(kind=double),dimension(2,2)::corners=unit_square
   real(kind=double),dimension(:,:),allocatable::hyp_hits
 contains
   !Überschriebene serializable_class Methoden
   procedure ::write_to_marker=>sample_region_write_to_marker
   procedure ::read_from_marker=>sample_region_read_from_marker
   procedure ::print_to_unit=>sample_region_print_to_unit
   procedure,nopass ::get_type=>sample_region_get_type
   !Originäre sample_region_type Methoden
   procedure ::initialize=>sample_region_initialize
   procedure ::generate_hit=>sample_region_generate_hit
   procedure ::confirm_hit=>sample_region_confirm_hit
   procedure ::split=>sample_region_split
   procedure ::write_hits=>sample_region_write_hits
   procedure ::is_full=>sample_region_is_full
   procedure ::move_components=>sample_region_move_components
   procedure ::mean=>sample_region_mean
   procedure ::area=>sample_region_area
   procedure ::density=>sample_region_density
```

```
    procedure ::contains=>sample_region_contains
    procedure ::to_generator=>sample_region_to_generator
 end type sample_region_type
```

11.3.2 sample_2d_type

```
 type,extends(serializable_class)::sample_2d_type
    integer::n_regions=0
    integer::n_alloc=0
    integer::n_hits=0
    real(kind=double),dimension(2)::range=[0,1]
    type(sample_region_type),dimension(:),allocatable::regions
  contains
    !Überschriebene serializable_class Methoden
    procedure ::write_to_marker=>sample_2d_write_to_marker
    procedure ::read_from_marker=>sample_2d_read_from_marker
    procedure ::print_to_unit=>sample_2d_print_to_unit
    procedure,nopass ::get_type=>sample_2d_get_type
    !Originäre sample_2d_type Methoden
    procedure ::initialize=>sample_2d_initialize
    procedure ::contains=>sample_2d_contains
    procedure ::generate_hit=>sample_2d_generate_hit
    procedure ::confirm_hit=>sample_2d_confirm_hit
    procedure ::split=>sample_2d_split
    procedure ::push=>sample_2d_push
    procedure ::write_hits=>sample_2d_write_hits
    procedure ::is_full=>sample_2d_is_full
    procedure ::move_components=>sample_2d_move_components
    procedure ::thickness=>sample_2d_thickness
    procedure ::analyse=>sample_2d_analyse
    procedure ::to_generator=>sample_2d_to_generator
    procedure ::mean=>sample_2d_mean
 end type sample_2d_type
```

11.3.3 sample_3d_type

```
 type,extends(serializable_class)::sample_3d_type
    integer::n_slices=0
    integer::n_alloc=0
    type(sample_2d_type),dimension(:),allocatable::slices
  contains
    !Überschriebene serializable_class Methoden
    procedure ::write_to_marker=>sample_3d_write_to_marker
    procedure ::read_from_marker=>sample_3d_read_from_marker
    procedure ::print_to_unit=>sample_3d_print_to_unit
    procedure,nopass ::get_type=>sample_3d_get_type
    ! overridden measurable_class procedures
    procedure ::measure=>sample_3d_measure
    !Originäre sample_3d_type Methoden
```

```
    procedure ::to_generator=>sample_3d_to_generator
    procedure :: sample_3d_initialize
    procedure :: sample_3d_generate_hit
    procedure :: sample_3d_confirm_hit
    procedure ::enlarge=>sample_3d_enlarge
    generic::initialize=>sample_3d_initialize
    generic::generate_hit=>sample_3d_generate_hit
    generic::confirm_hit=>sample_3d_confirm_hit
 end type sample_3d_type
```

11.3.4 sample_int_kind_type

```
 type,extends(sample_3d_type)::sample_int_kind_type
    integer::n_proc=0
    integer(kind=i64)::n_tries=0
    integer::n_hits=0
    integer::n_over=0
    integer,dimension(:),allocatable::hits,weights,processes
    real(kind=double)::overall_boost=1D-1
  contains
    !Überschriebene serializable_class Methoden
    procedure ::write_to_marker=>sample_int_kind_write_to_marker
    procedure ::read_from_marker=>sample_int_kind_read_from_marker
    procedure ::print_to_unit=>sample_int_kind_print_to_unit
    procedure,nopass ::get_type=>sample_int_kind_get_type
    ! overridden sample_3d_type procedures
    procedure ::to_generator=>sample_int_kind_to_generator
    !Originäre sample_int_kind_type Methoden
    procedure ::process_id=>sample_int_kind_process_id
    procedure :: sample_int_kind_initialize
    procedure :: sample_int_kind_generate_hit
    procedure ::mcgenerate_hit=>sample_int_kind_mcgenerate_hit
    procedure :: sample_int_kind_confirm_hit
    procedure ::analyse=>sample_int_kind_analyse
    generic::initialize=>sample_int_kind_initialize
    generic::generate_hit=>sample_int_kind_generate_hit
    generic::confirm_hit=>sample_int_kind_confirm_hit
 end type sample_int_kind_type
```

11.3.5 sample_inclusive_type

```
 type,extends(serializable_class)::sample_inclusive_type
    integer::n_alloc=0
    integer(kind=i64)::n_tries_sum=zero
    integer(kind=i64)::n_over_sum=zero
    integer(kind=i64)::n_hits_sum=zero
    type(sample_int_kind_type),dimension(:),allocatable::int_kinds
  contains
    !Überschriebene serializable_class Methoden
```

```
    procedure ::write_to_marker=>sample_inclusive_write_to_marker
    procedure ::read_from_marker=>sample_inclusive_read_from_marker
    procedure ::print_to_unit=>sample_inclusive_print_to_unit
    procedure,nopass ::get_type=>sample_inclusive_get_type
    !Originäre sample_inclusive_type Methoden
    procedure ::process_id=>sample_inclusive_process_id
    procedure ::initialize=>sample_inclusive_initialize
    procedure ::finalize=>sample_inclusive_finalize
    procedure ::generate_hit=>sample_inclusive_generate_hit
    procedure ::mcgenerate_hit=>sample_inclusive_mcgenerate_hit
    procedure ::confirm_hit=>sample_inclusive_confirm_hit
    procedure ::sum_up=>sample_inclusive_sum_up
    procedure ::analyse=>sample_inclusive_analyse
    procedure ::to_generator=>sample_inclusive_to_generator
    procedure ::allocate=>sample_inclusive_allocate
  end type sample_inclusive_type
```

11.4 Implementierung der Prozeduren

11.4.1 Methoden für sample_region_type

sample_region_write_to_marker ↑

```
  subroutine sample_region_write_to_marker(this,marker,status)
    class(sample_region_type),intent(in) :: this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    integer::n
    call marker%mark_begin("sample_region_type")
    call marker%mark("n_hits",this%n_hits)
    call marker%mark("n_alloc",this%n_alloc)
    call marker%mark("lower_corner",this%corners(1:2,1))
    call marker%mark("upper_corner",this%corners(1:2,2))
    if(allocated(this%hyp_hits))then
       call marker%mark("hyp_hits",this%hyp_hits(1:3,:this%n_hits))
    else
       call marker%mark_nothing("hyp_hits")
    end if
    call marker%mark_end("sample_region_type")
  end subroutine sample_region_write_to_marker
```

sample_region_read_from_marker ↑

```
  subroutine sample_region_read_from_marker(this,marker,status)
    class(sample_region_type),intent(out) :: this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    integer::n
    call marker%pick_begin("sample_region_type",status=status)
    call marker%pick("n_hits",this%n_hits,status)
```

```
    call marker%pick("n_alloc",this%n_alloc,status)
    call marker%pick("lower_corner",this%corners(1:2,1),status)
    call marker%pick("upper_corner",this%corners(1:2,2),status)
    if(allocated(this%hyp_hits))deallocate(this%hyp_hits)
    call marker%verify_nothing("hyp_hits",status)
    if(.not.status==serialize_nothing)then
       allocate(this%hyp_hits(3,this%n_alloc))
       call marker%pick("hyp_hits",this%hyp_hits(1:3,:this%n_hits),status)
    end if
    call marker%pick_end("sample_region_type",status)
  end subroutine sample_region_read_from_marker
```

sample_region_print_to_unit ↑

```
  subroutine sample_region_print_to_unit(this,unit,parents,components,peers)
    class(sample_region_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    write(unit,fmt=*)"components of sample_region_type"
    write(unit,'("n_hits:           ",i10)')this%n_hits
    write(unit,'("n_alloc:          ",i10)')this%n_alloc
    write(unit,'("corners:          ",4(e20.10))')this%corners
    if(allocated(this%hyp_hits).and.this%n_hits>0)then
       if(components>0)then
          write(unit,'("hits:")')
          print *,shape(this%hyp_hits)
          write(unit,fmt='(3(e20.10))')this%hyp_hits(1:3,this%n_hits)
       else
          write(unit,fmt=*)"skipping hits."
       end if
    else
       write(unit,fmt=*)"hits are not allocated."
    end if
  end subroutine sample_region_print_to_unit
```

sample_region_get_type ↑

```
  pure subroutine sample_region_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="sample_region_type")
  end subroutine sample_region_get_type
```

sample_region_initialize ↑

```
  subroutine sample_region_initialize(this,n_alloc)
    class(sample_region_type),intent(out)::this
    integer,intent(in)::n_alloc
    if(allocated(this%hyp_hits))deallocate(this%hyp_hits)
    allocate(this%hyp_hits(3,n_alloc))
    this%n_alloc=n_alloc
  end subroutine sample_region_initialize
```

sample_region_generate_hit ↑

```
  pure subroutine sample_region_generate_hit(this,rnd,area,hit)
    class(sample_region_type),intent(in)::this
    integer,intent(in),dimension(2)::rnd
    real(kind=double),dimension(2),intent(out)::hit
    real(kind=double),intent(out)::area
    call muli_mcint_generate_hit(rnd,this%corners,hit)
    area=this%area()
  end subroutine sample_region_generate_hit
```

sample_region_confirm_hit ↑

```
  subroutine sample_region_confirm_hit(this,hit)
    class(sample_region_type),intent(inout)::this
    real(kind=double),dimension(3),intent(in)::hit
!    print *,"sample_region_confirm_hit: ",this%n_hits,this%n_alloc,hit
    this%n_hits=this%n_hits+1
    if(this%n_hits<=this%n_alloc)then
       this%hyp_hits(1:3,this%n_hits)=hit
    else
       print *,"sample_region_confirm_hit: Region is already full."
    end if
  end subroutine sample_region_confirm_hit
```

sample_region_split ↑

```
  subroutine sample_region_split(this,pos,dimX,n_alloc,lower,upper)
    class(sample_region_type),intent(in)::this
    type(sample_region_type),intent(out)::lower,upper
    real(kind=double),dimension(3)::hit
    real(kind=double),intent(in)::pos
    integer,intent(in)::dimX,n_alloc
    integer::n_hit
    call lower%initialize(n_alloc)
    call upper%initialize(n_alloc)
    do n_hit=1,this%n_hits
       hit=this%hyp_hits(1:3,n_hit)
       if(hit(dimX)<pos)then
          call lower%confirm_hit(hit)
       else
          call upper%confirm_hit(hit)
       end if
    end do
    lower%corners=this%corners
    upper%corners=this%corners
    if(dimX<3)then
       lower%corners(dimX,2)=pos
       upper%corners(dimX,1)=pos
    end if
  end subroutine sample_region_split
```

sample_region_write_hits ↑

```
subroutine sample_region_write_hits(this,unit)
  class(sample_region_type),intent(in)::this
  integer,intent(in)::unit
  integer::n
  do n=1,this%n_hits
     write(unit,fmt=*)this%hyp_hits(1:3,n)
  end do
end subroutine sample_region_write_hits
```

sample_region_is_full ↑

```
elemental logical function sample_region_is_full(this)
  class(sample_region_type),intent(in)::this
  sample_region_is_full=this%n_alloc==this%n_hits
end function sample_region_is_full
```

sample_region_move_components ↑

```
subroutine sample_region_move_components(this,that)
  class(sample_region_type),intent(inout)::this
  class(sample_region_type),intent(out)::that
  that%n_alloc=this%n_alloc
  that%n_hits=this%n_hits
  that%corners=this%corners
  call move_alloc(this%hyp_hits,that%hyp_hits)
  this%n_alloc=0
  this%n_hits=0
end subroutine sample_region_move_components
```

sample_region_mean ↑

```
elemental function sample_region_mean(this,dim)
  real(kind=double)::sample_region_mean
  class(sample_region_type),intent(in)::this
  integer,intent(in)::dim
  sample_region_mean=sum(this%hyp_hits(dim,1:this%n_hits))/this%n_hits
end function sample_region_mean
```

sample_region_area ↑

```
elemental function sample_region_area(this)
  real(kind=double)::sample_region_area
  class(sample_region_type),intent(in)::this
  sample_region_area=product(this%corners(1:2,2)-this%corners(1:2,1))
end function sample_region_area
```

sample_region_density ↑

```
elemental function sample_region_density(this)
  real(kind=double)::sample_region_density
  class(sample_region_type),intent(in)::this
  sample_region_density=this%n_hits/this%area()
end function sample_region_density
```

sample_region_contains ↑

```
pure logical function sample_region_contains(this,hit)
  class(sample_region_type),intent(in)::this
  real(kind=double),intent(in),dimension(3)::hit
  sample_region_contains=(this%corners(1,1)<=hit(1)&
       .and.hit(1)<=this%corners(1,2)&
       .and.this%corners(2,1)<=hit(2)&
       .and.hit(2)<=this%corners(2,2))
end function sample_region_contains
```

sample_region_to_generator ↑

```
subroutine sample_region_to_generator(this)
  class(sample_region_type),intent(inout)::this
  if(allocated(this%hyp_hits))deallocate(this%hyp_hits)
  this%n_alloc=0
end subroutine sample_region_to_generator
```

11.4.2 Methoden für sample_2d_type

sample_2d_write_to_marker ↑

```
subroutine sample_2d_write_to_marker(this,marker,status)
  class(sample_2d_type),intent(in) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  integer::n
  call marker%mark_begin("sample_2d_type")
  call marker%mark("n_regions",this%n_regions)
  call marker%mark("n_alloc",this%n_alloc)
  call marker%mark("n_hits",this%n_hits)
  call marker%mark("range",this%range)
  if(this%n_regions>0)then
     call marker%mark_instance_begin(this%regions(1),name="sample_2d_type",shape=shape(this
     do n=1,this%n_regions
        call sample_region_write_to_marker(this%regions(n),marker,status)
     end do
     call marker%mark_instance_end()
  end if
  call marker%mark_end("sample_2d_type")
end subroutine sample_2d_write_to_marker
```

sample_2d_read_from_marker ↑

```
subroutine sample_2d_read_from_marker(this,marker,status)
  class(sample_2d_type),intent(out) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  integer::n
  call marker%pick_begin("sample_2d_type",status=status)
```

```
    call marker%pick("n_regions",this%n_regions,status)
    call marker%pick("n_alloc",this%n_alloc,status)
    call marker%pick("n_hits",this%n_hits,status)
    call marker%pick("range",this%range,status)
    if(this%n_regions>0)then
       call marker%pick_begin("regions",status=status)
       allocate(this%regions(this%n_regions))
       do n=1,this%n_regions
          call sample_region_read_from_marker(this%regions(n),marker,status)
       end do
       call marker%pick_end("regions",status)
    end if
    call marker%pick_end("sample_2d_type",status)
  end subroutine sample_2d_read_from_marker
```

sample_2d_print_to_unit ↑

```
  subroutine sample_2d_print_to_unit(this,unit,parents,components,peers)
    class(sample_2d_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    integer::n
    write(unit,fmt=*)"components of sample_2d_type"
    write(unit,'("n_regions:        ",i10)')this%n_regions
    write(unit,'("n_alloc:          ",i10)')this%n_alloc
    write(unit,'("range:            ",2(e20.10))')this%range
    if(allocated(this%regions))then
       if(components>0)then
          write(unit,'("regions:")')
          do n=1,this%n_regions
             call this%regions(n)%print_to_unit(unit,parents,components-1,peers)
          end do
       else
          write(unit,fmt=*)"skipping regions."
       end if
    else
       write(unit,fmt=*)"regions are not allocated."
    end if
  end subroutine sample_2d_print_to_unit
```

sample_2d_get_type ↑

```
  pure subroutine sample_2d_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="sample_2d_type")
  end subroutine sample_2d_get_type
```

sample_2d_initialize ↑

```
  subroutine sample_2d_initialize(this,n_alloc)
    class(sample_2d_type),intent(out)::this
    integer,intent(in)::n_alloc
```

```
    integer::n
    if(allocated(this%regions))deallocate(this%regions)
    allocate(this%regions(n_alloc))
    this%n_alloc=n_alloc
    this%n_regions=1
    call this%regions(1)%initialize(n_alloc)
!    do n=1,n_alloc
!       call this%regions(n)%initialize(n_alloc)
!    end do
  end subroutine sample_2d_initialize
```

sample_2d_contains ↑

```
  pure logical function sample_2d_contains(this,pts2)
    class(sample_2d_type),intent(in)::this
    real(kind=double),intent(in)::pts2
    sample_2d_contains=this%range(1)<=pts2.and.pts2<=this%range(2)
  end function sample_2d_contains
```

sample_2d_generate_hit ↑

```
  pure subroutine sample_2d_generate_hit(this,rnd,boost,hit,region)
    class(sample_2d_type),intent(in)::this
    integer,dimension(3),intent(in)::rnd
    integer,intent(out)::region
    integer::n,sum
    real(kind=double),dimension(2),intent(out)::hit
    real(kind=double),intent(out)::boost
    if(0<this%n_hits.and.this%n_hits<10)then
       sum=modulo(rnd(1),this%n_hits)+1!this should be improved
       region=0
       do while(sum>0)
          region=region+1
          sum=sum-this%regions(region)%n_hits
       end do
       call this%regions(region)%generate_hit(rnd(2:3),boost,hit)
       boost=boost*this%n_hits/this%regions(region)%n_hits
    else
       if(this%n_regions>1)then
          region=modulo(rnd(1),this%n_regions)+1!this should be improved
          call this%regions(region)%generate_hit(rnd(2:3),boost,hit)
          boost=boost*this%n_regions
       else
          region=1
          call this%regions(1)%generate_hit(rnd(2:3),boost,hit)
       end if
    end if
  end subroutine sample_2d_generate_hit
```

sample_2d_confirm_hit ↑

```
subroutine sample_2d_confirm_hit(this,hit,region,full)
  class(sample_2d_type),intent(inout)::this
  integer,intent(in)::region
  real(kind=double),dimension(3),intent(in)::hit
  type(sample_region_type),allocatable::old_region
  real(kind=double),dimension(2)::mean,var,diff,cm,cv,c
  integer::n,n_alloc,dim
  logical,intent(out)::full
  this%n_hits=this%n_hits+1
  if(region<=this%n_alloc)then
     full=.false.
     call this%regions(region)%confirm_hit(hit)
     n_alloc=this%regions(region)%n_alloc
     if(this%regions(region)%is_full())then
        if(this%is_full())then
           full=.true.
        else
           this%n_regions=this%n_regions+1
           allocate(old_region)
           call this%regions(region)%move_components(old_region)
           mean=sum(old_region%hyp_hits(1:2,:),dim=2)/n_alloc
           var=0D0
           do n=1,n_alloc
              var=var+abs(mean-old_region%hyp_hits(1:2,n))
           end do
           var=var/n_alloc
           diff=old_region%corners(1:2,2)-old_region%corners(1:2,1)
           cm=abs([0.5D0,0.5D0]-(old_region%corners(1:2,2)-mean)/diff)
           cv=abs(2*([0.25D0,0.25D0]-var/diff))
           c=max(cm,cv)
           if(c(1)<c(2))then
              dim=2
           else
              dim=1
           end if
           call old_region%split(&
             mean(dim),&
             dim,&
             this%n_alloc,&
             this%regions(region),&
             this%regions(this%n_regions))
        end if
     end if
  else
     print *,"sample_2d_confirm_hit: Region ",region," not allocated."
  end if
end subroutine sample_2d_confirm_hit
```

sample_2d_is_full ↑

```
  elemental logical function sample_2d_is_full(this)
    class(sample_2d_type),intent(in)::this
    sample_2d_is_full=this%n_alloc==this%n_regions
  end function sample_2d_is_full
```

sample_2d_split ↑

```
  recursive subroutine sample_2d_split(this,n_alloc,pos,lower,upper)
    class(sample_2d_type),intent(in)::this
    integer,intent(in)::n_alloc
    real(kind=double),intent(in)::pos
    type(sample_2d_type),intent(out)::lower,upper
    integer::n_r,n_h
    real(kind=double),dimension(3)::hit
    !print *,"sample_2d_split: ",pos,this%range
    call lower%initialize(4*n_alloc)
    call upper%initialize(4*n_alloc)
    do n_r=this%n_regions,1,-1
       do n_h=1,this%regions(n_r)%n_hits
          hit=this%regions(n_r)%hyp_hits(1:3,n_h)
          if(hit(3)>pos)then
             call upper%push(hit)
          else
             call lower%push(hit)
          end if
       end do
    end do
    lower%range=[this%range(1),pos]
    upper%range=[pos,this%range(2)]
  end subroutine sample_2d_split
```

sample_2d_push ↑

```
  subroutine sample_2d_push(this,hit)
    class(sample_2d_type),intent(inout)::this
    real(kind=double),dimension(3),intent(in)::hit
    integer::region
    logical::full
    do region=1,this%n_regions
       if(this%regions(region)%contains(hit))then
          call this%confirm_hit(hit,region,full)
!           call this%regions(region)%confirm_hit(hit)
          if(full)print *,"sample_2d_push: region is full now"
          exit
       end if
    end do
    if(region>this%n_regions)print *,"sample_2d_push: no region contains ",hit
  end subroutine sample_2d_push
```

sample_2d_write_hits ↑

```
subroutine sample_2d_write_hits(this,unit)
  class(sample_2d_type),intent(in)::this
  integer,intent(in)::unit
  integer::n
  do n=1,this%n_regions
     call this%regions(n)%write_hits(unit)
  end do
end subroutine sample_2d_write_hits
```

sample_2d_move_components ↑

```
subroutine sample_2d_move_components(this,that)
  class(sample_2d_type),intent(inout)::this
  class(sample_2d_type),intent(out)::that
  that%n_alloc=this%n_alloc
  that%n_regions=this%n_regions
  that%n_hits=this%n_hits
  that%range=this%range
  call move_alloc(this%regions,that%regions)
  this%n_alloc=0
  this%n_regions=0
  this%n_hits=0
  this%range=[0D0,0D0]
end subroutine sample_2d_move_components
```

sample_2d_thickness ↑

```
elemental function sample_2d_thickness(this)
  class(sample_2d_type),intent(in)::this
  real(kind=double)::sample_2d_thickness
  sample_2d_thickness=this%range(2)-this%range(1)
end function sample_2d_thickness
```

sample_2d_analyse ↑

```
subroutine sample_2d_analyse(this,dir,file)
  class(sample_2d_type),intent(in)::this
  character(*),intent(in)::dir,file
  integer::u
  real(kind=double),dimension(1:2,0:100,0:100)::grid
  integer,dimension(0:100,0:100)::i_grid
  integer::r,x,y
  integer,dimension(2,2)::i
  call generate_unit(u)
  print *,"sample_2d_analyse: ",dir//"/"//file
  open(u,file=dir//"/"//file)
  do x=0,100
     do y=0,100
        grid(1:2,x,y)=[-1D0,-1D0]
     end do
  end do
  do r=1,this%n_regions
```

```
      i=int(this%regions(r)%corners*1D2)
      do x=i(1,1),i(1,2)
         do y=i(2,1),i(2,2)
            i_grid(x,y)=this%regions(r)%n_hits
            grid(1,x,y)=1D0/this%regions(r)%area()
            grid(2,x,y)=this%regions(r)%density()
         end do
      end do
   end do
   do x=0,100
      do y=0,100
         write(u,fmt=*)x,y,i_grid(x,y),grid(1:2,x,y)
      end do
      write(u,fmt=*)""
   end do
   close(u)
 end subroutine sample_2d_analyse
```

sample_2d_to_generator ↑

```
 subroutine sample_2d_to_generator(this)
   class(sample_2d_type),intent(inout)::this
   integer::region
   do region=1,this%n_regions
      call this%regions(region)%to_generator()
   end do
 end subroutine sample_2d_to_generator
```

sample_2d_mean ↑

```
 elemental function sample_2d_mean(this,dim) result(mean)
   class(sample_2d_type),intent(in)::this
   integer,intent(in)::dim
   real(kind=double)::mean
   integer::region,hit
   mean=0D0
   do region=1,this%n_regions
      do hit=1,this%regions(region)%n_hits
         mean=mean+this%regions(region)%hyp_hits(dim,hit)
      end do
   end do
   mean=mean/this%n_hits
 end function sample_2d_mean
```

11.4.3 Methoden für sample_3d_type

sample_3d_write_to_marker ↑

```
 subroutine sample_3d_write_to_marker(this,marker,status)
   class(sample_3d_type),intent(in) :: this
   class(marker_type),intent(inout)::marker
```

```
    integer(kind=dik),intent(out)::status
    integer::n
    call marker%mark_begin("sample_3d_type")
    call marker%mark("n_slices",this%n_slices)
    call marker%mark("n_alloc",this%n_alloc)
    if(this%n_slices>0)then
       call marker%mark_instance_begin&
         (this%slices(1),"slices",shape=shape(this%slices))
       do n=1,this%n_slices
          call sample_2d_write_to_marker(this%slices(n),marker,status)
       end do
       call marker%mark_instance_end()
    end if
    call marker%mark_end("sample_3d_type")
  end subroutine sample_3d_write_to_marker
```

sample_3d_read_from_marker ↑

```
  subroutine sample_3d_read_from_marker(this,marker,status)
    class(sample_3d_type),intent(out) :: this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    integer::n
    call marker%pick_begin("sample_3d_type",status=status)
    call marker%pick("n_slices",this%n_slices,status)
    call marker%pick("n_alloc",this%n_alloc,status)
    if(this%n_slices>0)then
       call marker%pick_instance_begin("slices",status=status)
       allocate(this%slices(this%n_slices))
       do n=1,this%n_slices
          call sample_2d_read_from_marker(this%slices(n),marker,status)
       end do
       call marker%pick_instance_end(status)
    end if
    call marker%pick_end("sample_3d_type",status)
  end subroutine sample_3d_read_from_marker
```

sample_3d_print_to_unit ↑

```
  subroutine sample_3d_print_to_unit(this,unit,parents,components,peers)
    class(sample_3d_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    integer::n
    write(unit,fmt=*)"components of sample_3d_type"
    write(unit,'("n_slices:          ",i10)')this%n_slices
    write(unit,'("n_alloc:           ",i10)')this%n_alloc
    if(allocated(this%slices))then
       if(components>0)then
          do n=1,this%n_slices
             call this%slices(n)%print_to_unit(unit,parents,components-1,peers)
          end do
```

```
      else
         write(unit,fmt=*)"skipping slices."
      end if
   else
      write(unit,fmt=*)"slices are not allocated."
   end if
 end subroutine sample_3d_print_to_unit
```

sample_3d_get_type ↑

```
 pure subroutine sample_3d_get_type(type)
   character(:),allocatable,intent(out)::type
   allocate(type,source="sample_3d_type")
 end subroutine sample_3d_get_type
```

sample_3d_measure ↑

```
 elemental function sample_3d_measure(this)
   real(kind=double)::sample_3d_measure
   class(sample_3d_type),intent(in)::this
   sample_3d_measure=1D0
 end function sample_3d_measure
```

sample_3d_to_generator ↑

```
 subroutine sample_3d_to_generator(this)
   class(sample_3d_type),intent(inout)::this
   integer::slice
   do slice=1,this%n_slices
      call this%slices(slice)%to_generator()
   end do
 end subroutine sample_3d_to_generator
```

sample_3d_initialize ↑

```
 subroutine sample_3d_initialize(this,n_alloc)
   class(sample_3d_type),intent(out)::this
   integer,intent(in)::n_alloc
   if(allocated(this%slices))deallocate(this%slices)
   if(n_alloc>0)then
      allocate(this%slices(n_alloc))
      this%n_alloc=n_alloc
      this%n_slices=1
      call this%slices(1)%initialize(n_alloc)
   else
      this%n_alloc=0
   end if
 end subroutine sample_3d_initialize
```

sample_3d_generate_hit ↑

```
pure subroutine sample_3d_generate_hit(this,rnd,pts2,boost,hit,region,slice)
  class(sample_3d_type),intent(in)::this
  integer,intent(in),dimension(3)::rnd
  real(kind=double),intent(in)::pts2
  integer,intent(out)::slice,region
  real(kind=double),dimension(3),intent(out)::hit
  real(kind=double),intent(out)::boost
  if(this%n_slices==0)then
     call muli_mcint_generate_hit(rnd,unit_square,hit(1:2))
     boost=1D0
     slice=1
     region=1
  else
     do slice=1,this%n_slices
        if(this%slices(slice)%contains(pts2))exit
     end do
     call this%slices(slice)%generate_hit(rnd,boost,hit(1:2),region)
  end if
  hit(3)=pts2
end subroutine sample_3d_generate_hit
```

sample_3d_confirm_hit ↑

```
subroutine sample_3d_confirm_hit(this,hit,region,slice)
  class(sample_3d_type),intent(inout)::this
  integer,intent(in)::slice,region
  real(kind=double),intent(in),dimension(3)::hit
  type(sample_2d_type),allocatable::old_slice
  integer::n
  logical::full
  if(this%n_alloc<slice)then
     print *,"sample_3d_confirm_hit: Slice ",slice," not allocated."
  else
  !if(.not.allocated(this%slices))call this%initialize(2)
     call this%slices(slice)%confirm_hit(hit,region,full)
     if(full)then
        if(this%n_alloc==this%n_slices)call this%enlarge()
        this%n_slices=this%n_slices+1
        allocate(old_slice)
        call this%slices(slice)%move_components(old_slice)
        call sample_2d_split(&
             old_slice,&
             this%n_alloc,&
             old_slice%mean(3),&
             this%slices(slice),&
             this%slices(this%n_slices))
     end if
  end if
end subroutine sample_3d_confirm_hit
```

sample_3d_enlarge ↑

```
subroutine sample_3d_enlarge(this)
  class(sample_3d_type),intent(inout)::this
  type(sample_2d_type),allocatable,dimension(:)::old_slices
  integer::n
  print *,"sample_3d_enlarge"
  call move_alloc(this%slices,old_slices)
  this%n_alloc=this%n_alloc*2
  allocate(this%slices(this%n_alloc))
  do n=1,size(old_slices)
     call old_slices(n)%move_components(this%slices(n))
  end do
end subroutine sample_3d_enlarge
```

11.4.4 Methoden für sample_int_kind_type

sample_int_kind_write_to_marker ↑

```
subroutine sample_int_kind_write_to_marker(this,marker,status)
  class(sample_int_kind_type),intent(in) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  call marker%mark_begin("sample_int_kind_type")
  call sample_3d_write_to_marker(this,marker,status)
  call marker%mark("n_hits",this%n_hits)
  call marker%mark("n_proc",this%n_proc)
  call marker%mark("boost",this%overall_boost)
  if(this%n_hits>0)then
     call marker%mark("hits",this%hits)
  end if
  if(this%n_proc>0)then
     call marker%mark("processes",this%processes)
     call marker%mark("weights",this%weights)
  end if
  call marker%mark_end("sample_int_kind_type")
end subroutine sample_int_kind_write_to_marker
```

sample_int_kind_read_from_marker ↑

```
subroutine sample_int_kind_read_from_marker(this,marker,status)
  class(sample_int_kind_type),intent(out) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  call marker%pick_begin("sample_int_kind_type",status=status)
  call sample_3d_read_from_marker(this,marker,status)
  call marker%pick("n_hits",this%n_hits,status)
  call marker%pick("n_proc",this%n_proc,status)
  call marker%pick("boost",this%overall_boost,status)
  if(this%n_hits>0)then
     allocate(this%hits(this%n_hits))
     call marker%pick("hits",this%hits,status)
  end if
```

```
    if(this%n_proc>0)then
       allocate(this%processes(this%n_proc))
       call marker%pick("processes",this%processes,status)
       allocate(this%weights(this%n_proc))
       call marker%pick("weights",this%weights,status)
    end if
    call marker%pick_end("sample_int_kind_type",status)
  end subroutine sample_int_kind_read_from_marker
```

sample_int_kind_print_to_unit ↑

```
  subroutine sample_int_kind_print_to_unit(this,unit,parents,components,peers)
    class(sample_int_kind_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    integer::n
    if(parents>0)call sample_3d_print_to_unit(this,unit,parents,components,peers)
    write(unit,fmt=*)"components of sample_int_kind_type"
    write(unit,'("n_hits:          ",i10)')this%n_hits
    write(unit,'("n_proc:          ",i10)')this%n_proc
    write(unit,'("overall_boost:   ",e14.7)')this%overall_boost
    write(unit,'("hits:")')
    write(unit,'(10(i0," "))')this%hits(1:this%n_hits)
    write(unit,'("weights:")')
    write(unit,'(10(i0," "))')this%weights
    write(unit,'("processes:")')
    write(unit,'(2(i0," "))')this%processes
  end subroutine sample_int_kind_print_to_unit
```

sample_int_kind_get_type ↑

```
  pure subroutine sample_int_kind_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="sample_int_kind_type")
  end subroutine sample_int_kind_get_type
```

sample_int_kind_to_generator ↑

```
  subroutine sample_int_kind_to_generator(this)
    class(sample_int_kind_type),intent(inout)::this
    integer::int_kind
    if(allocated(this%hits))deallocate(this%hits)
    call sample_3d_to_generator(this)
  end subroutine sample_int_kind_to_generator
```

sample_int_kind_process_id ↑

```
  elemental integer function sample_int_kind_process_id(this,subprocess)
    class(sample_int_kind_type),intent(in)::this
    integer,intent(in)::subprocess
    sample_int_kind_process_id=this%processes(subprocess)
  end function sample_int_kind_process_id
```

sample_int_kind_initialize ↑

```
  subroutine sample_int_kind_initialize(this,n_alloc,processes,overall_boost)
    class(sample_int_kind_type),intent(out)::this
    integer,intent(in)::n_alloc
    integer,intent(in),dimension(:)::processes
    real(kind=double),optional,intent(in)::overall_boost
    integer::s,n
    s=size(processes)
    call sample_3d_initialize(this,n_alloc)
    if(allocated(this%hits))deallocate(this%hits)
    allocate(this%hits(n_alloc))
    if(allocated(this%weights))deallocate(this%weights)
    allocate(this%weights(s))
    if(allocated(this%processes))deallocate(this%processes)
    allocate(this%processes(s),source=processes)
    do n=1,s
       this%weights(n)=0
    end do
    this%n_alloc=n_alloc
    this%n_hits=0
    this%n_proc=s
    if(present(overall_boost))this%overall_boost=overall_boost
    this%overall_boost=this%overall_boost*this%n_proc
!    print *,this%weights
  end subroutine sample_int_kind_initialize
```

sample_int_kind_generate_hit ↑

```
  pure subroutine sample_int_kind_generate_hit&
    (this,rnd,pts2,boost,hit,region,slice,subprocess)
    class(sample_int_kind_type),intent(in)::this
    integer,dimension(4),intent(in)::rnd
    real(kind=double),intent(in)::pts2
    real(kind=double),dimension(3),intent(out)::hit
    integer,intent(out)::region,slice,subprocess
    real(kind=double),intent(out)::boost
    integer::n_n
!    print *,rnd,pts2,boost,hit,region,slice,subprocess
    call sample_3d_generate_hit(this,rnd(2:4),pts2,boost,hit,region,slice)
    n_n=modulo(rnd(1),this%n_hits+size(this%weights))+1
    if(n_n>this%n_hits)then
       subprocess=n_n-this%n_hits
    else
       subprocess=this%hits(n_n)
    end if
    boost=boost*this%overall_boost*(this%n_proc+this%n_hits)&
          /(this%n_proc*(this%weights(subprocess)+1))
  end subroutine sample_int_kind_generate_hit
```

sample_int_kind_mcgenerate_hit ↑

```
subroutine sample_int_kind_mcgenerate_hit&
  (this,pts2,mean,integrand_kind,tao_rnd,process_id,cart_hit)
  class(sample_int_kind_type),intent(inout)::this
  integer,intent(in)::integrand_kind
  real(kind=double),intent(in)::pts2,mean
  type(tao_random_state),intent(inout)::tao_rnd
  real(kind=double),dimension(3),intent(out)::cart_hit
  integer,intent(out)::process_id
  real(kind=double)::boost
  integer::region,slice,subprocess
  integer,dimension(4)::i_rnd
  real(kind=double)::dddsigma,d_rnd
  real(kind=double),dimension(3)::hyp_hit
  MC:do
     this%n_tries=this%n_tries+1
     call tao_random_number(tao_rnd,i_rnd)
     call tao_random_number(tao_rnd,d_rnd)
     !print *,pts2,mean,integrand_kind,process_id,cart_hit
     call this%generate_hit(i_rnd,pts2,boost,hyp_hit,region,slice,subprocess)
     process_id=this%process_id(subprocess)
     call interactions_dddsigma_reg(process_id,integrand_kind,hyp_hit,cart_hit,dddsigma)
     dddsigma=dddsigma*boost
     if(d_rnd*mean<dddsigma)then
        exit MC
     end if
  end do MC
  if(mean<dddsigma)then
     call this%confirm_hit(hyp_hit,region,slice,subprocess,.true.)
  else
     call this%confirm_hit(hyp_hit,region,slice,subprocess,.false.)
  end if
end subroutine sample_int_kind_mcgenerate_hit
```

sample_int_kind_confirm_hit ↑

```
subroutine sample_int_kind_confirm_hit(this,hit,region,slice,subprocess,over)
  class(sample_int_kind_type),intent(inout)::this
  real(kind=double),dimension(3),intent(in)::hit
  integer,intent(in)::region,slice,subprocess
  integer,dimension(:),allocatable::tmp_hits
  logical,optional,intent(in)::over
  this%n_hits=this%n_hits+1
  if(present(over))then
     if(over)then
        this%n_over=this%n_over+1
        this%overall_boost=this%overall_boost/1.1D0
     else
        this%overall_boost=this%overall_boost*1.0001D0
     end if
  end if
  if(0<size(this%hits))then
```

```
      if(this%n_hits>size(this%hits))then
         call move_alloc(this%hits,tmp_hits)
         allocate(this%hits(2*size(tmp_hits)))
         this%hits(1:size(tmp_hits))=tmp_hits
      end if
      this%hits(this%n_hits)=subprocess
   end if
   this%weights(subprocess)=this%weights(subprocess)+1
   call sample_3d_confirm_hit(this,hit,region,slice)
 end subroutine sample_int_kind_confirm_hit
```

sample_int_kind_analyse ↑

```
 subroutine sample_int_kind_analyse(this,dir,prefix)
   class(sample_int_kind_type),intent(in)::this
   character(*),intent(in)::dir,prefix
   integer::slices_unit,subprocs_unit
   integer::n,slice
   character(3)::slice_name
   integer,dimension(:),allocatable::int_a
   real(kind=double),dimension(:),allocatable::real_a
   call generate_unit(slices_unit)
   print *,"sample_int_kind_analyse: ",dir//"/"//prefix//"slice_distribution.plot"
   open(slices_unit,file=dir//"/"//prefix//"slice_distribution.plot")
   call generate_unit(subprocs_unit)
   print *,"sample_int_kind_analyse: ",dir//"/"//prefix//"subproc_distribution.plot"
   open(subprocs_unit,file=dir//"/"//prefix//"subproc_distribution.plot")
   allocate(real_a(this%n_slices))
   allocate(int_a(this%n_slices))
   do n=1,this%n_slices
      real_a(n)=this%slices(n)%range(1)
   end do
   call misc_sort(real_a,int_a)
   do n=1,size(this%weights)
      if(this%n_hits>0)then
         write(subprocs_unit,fmt=*)&
         real(this%weights(n)),real(this%weights(n)+1)/this%n_hits
      else
         write(subprocs_unit,fmt=*)0,0
      end if
   end do
   do n=1,this%n_slices
      slice=int_a(n)
      call integer_with_leading_zeros(n,3,slice_name)
      call sample_2d_analyse(this%slices(slice),dir,prefix//slice_name//".plot")
      print *,this%n_hits,this%slices(slice)%range(2)-this%slices(slice)%range(1)
      if (this%n_hits>0)then
         write(slices_unit,fmt=*)&
              &this%slices(slice)%range(1),&
              &this%slices(slice)%range(2),&
              &this%slices(slice)%n_hits,&
```

```
                &real(this%slices(slice)%n_hits)/&
                (this%n_hits*(this%slices(slice)%range(2)-this%slices(slice)%range(1)))
        else
           write(slices_unit,fmt=*)&
                &this%slices(slice)%range(1),&
                &this%slices(slice)%range(2),&
                &this%slices(slice)%n_hits,&
                &0D0
        end if
     end do
     write(slices_unit,fmt=*)1D0,0D0,0D0,0D0
     close(slices_unit)
     close(subprocs_unit)
  end subroutine sample_int_kind_analyse
```

11.4.5 Methoden für sample_inclusive_type

sample_inclusive_write_to_marker ↑

```
  subroutine sample_inclusive_write_to_marker(this,marker,status)
    class(sample_inclusive_type),intent(in) :: this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    integer::n
    call marker%mark_begin("sample_inclusive_type")
    call marker%mark("n_alloc",this%n_alloc)
    if(allocated(this%int_kinds))then
       call marker%mark_begin(tag="int_kinds",shape=shape(this%int_kinds))
       do n=1,size(this%int_kinds)
          call this%int_kinds(n)%write_to_marker(marker,status)
       end do
       call marker%mark_instance_end()
    else
       call marker%mark_empty(tag="int_kinds",shape=[0])
    end if
    call marker%mark_end("sample_inclusive_type")
  end subroutine sample_inclusive_write_to_marker
```

sample_inclusive_read_from_marker ↑

```
  subroutine sample_inclusive_read_from_marker(this,marker,status)
    class(sample_inclusive_type),intent(out) :: this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    integer::n
    integer,dimension(:),allocatable::s
    call marker%pick_begin("sample_inclusive_type",status=status)
    call marker%pick("n_alloc",this%n_alloc,status)
    call marker%pick_begin("int_kinds",shape=s,status=status)
    if(s(1)>0)then
       do n=1,size(this%int_kinds)
```

```
        call this%int_kinds(n)%read_from_marker(marker,status)
      end do
      call marker%pick_end("int_kinds",status)
   end if
   call marker%pick_end("sample_inclusive_type",status)
 end subroutine sample_inclusive_read_from_marker
```

sample_inclusive_print_to_unit ↑

```
 subroutine sample_inclusive_print_to_unit(this,unit,parents,components,peers)
   class(sample_inclusive_type),intent(in)::this
   integer,intent(in)::unit
   integer(kind=dik),intent(in)::parents,components,peers
   integer::n
   write(unit,fmt=*)"components of sample_inclusive_type"
   write(unit,'("n_alloc:          ",i10)')this%n_alloc
   if(allocated(this%int_kinds))then
      if(components>0)then
         write(unit,'("int_kinds:")')
         do n=1,this%n_alloc
            call this%int_kinds(n)%print_to_unit(unit,parents,components-1,peers)
         end do
      else
         write(unit,fmt=*)"skipping int_kinds."
      end if
   else
      write(unit,fmt=*)"int_kinds are not allocated."
   end if
 end subroutine sample_inclusive_print_to_unit
```

sample_inclusive_get_type ↑

```
 pure subroutine sample_inclusive_get_type(type)
   character(:),allocatable,intent(out)::type
   allocate(type,source="sample_inclusive_type")
 end subroutine sample_inclusive_get_type
```

sample_inclusive_process_id ↑

```
 elemental integer function sample_inclusive_process_id(this,subprocess,int_kind)
   class(sample_inclusive_type),intent(in)::this
   integer,intent(in)::subprocess,int_kind
   sample_inclusive_process_id=this%int_kinds(int_kind)%processes(subprocess)
 end function sample_inclusive_process_id
```

sample_inclusive_initialize ↑

```
 subroutine sample_inclusive_initialize(this,n_alloc,sizes,processes,overall_boost)
   class(sample_inclusive_type),intent(out)::this
   integer,intent(in)::n_alloc
   integer,dimension(:),intent(in)::sizes,processes
   real(kind=double),optional,intent(in)::overall_boost
   integer::n,sum
```

```
    this%n_tries_sum=zero
    this%n_over_sum=0
    this%n_alloc=size(sizes)
    if(allocated(this%int_kinds))deallocate(this%int_kinds)
    allocate(this%int_kinds(this%n_alloc))
    sum=0
    do n=1,this%n_alloc
       call this%int_kinds(n)%initialize(n_alloc,processes(sum+1:sum+sizes(n)),overall_boost)
       sum=sum+sizes(n)
    end do
  end subroutine sample_inclusive_initialize
```

sample_inclusive_finalize ↑

```
  subroutine sample_inclusive_finalize(this)
    class(sample_inclusive_type),intent(inout)::this
    deallocate(this%int_kinds)
    this%n_alloc=0
  end subroutine sample_inclusive_finalize
```

sample_inclusive_generate_hit ↑

```
  pure subroutine sample_inclusive_generate_hit&
    (this,rnd,pts2,int_kind,hit,region,boost,slice,process)
    class(sample_inclusive_type),intent(in)::this
    integer,dimension(4),intent(in)::rnd
    real(kind=double),intent(in)::pts2
    integer,intent(in)::int_kind
    real(kind=double),dimension(3),intent(out)::hit
    integer,intent(out)::region,slice,process
    real(kind=double),intent(out)::boost
    call this%int_kinds(int_kind)%generate_hit(rnd,pts2,boost,hit,region,slice,process)
  end subroutine sample_inclusive_generate_hit
```

sample_inclusive_mcgenerate_hit ↑

```
  subroutine sample_inclusive_mcgenerate_hit&
    (this,pts2,mean,integrand_kind,tao_rnd,process_id,cart_hit)
    class(sample_inclusive_type),intent(inout)::this
    real(kind=double),intent(in)::pts2,mean
    integer,intent(in)::integrand_kind
    type(tao_random_state),intent(inout)::tao_rnd
    real(kind=double),dimension(3),intent(out)::cart_hit
    integer,intent(out)::process_id
    call sample_int_kind_mcgenerate_hit(&
         this%int_kinds(integrand_kind),pts2,mean,integrand_kind,tao_rnd,process_id,cart_hit)
  end subroutine sample_inclusive_mcgenerate_hit
```

sample_inclusive_confirm_hit ↑

```
  subroutine sample_inclusive_confirm_hit(this,hit,int_kind,region,slice,process,over)
    class(sample_inclusive_type),intent(inout)::this
    real(kind=double),dimension(3),intent(in)::hit
```

```
    integer,intent(in)::int_kind,region,slice,process
    logical,optional,intent(in)::over
    call this%int_kinds(int_kind)%confirm_hit(hit,region,slice,process,over)
  end subroutine sample_inclusive_confirm_hit
```

sample_inclusive_sum_up ↑

```
  subroutine sample_inclusive_sum_up(this)
    class(sample_inclusive_type),intent(inout)::this
    integer::n
    this%n_tries_sum=zero
    this%n_hits_sum=zero
    this%n_over_sum=zero
    do n=1,this%n_alloc
       this%n_tries_sum=this%n_tries_sum+this%int_kinds(n)%n_tries
       this%n_hits_sum=this%n_hits_sum+this%int_kinds(n)%n_hits
       this%n_over_sum=this%n_over_sum+this%int_kinds(n)%n_over
    end do
  end subroutine sample_inclusive_sum_up
```

sample_inclusive_analyse ↑

```
  subroutine sample_inclusive_analyse(this,dir,subdirs)
    class(sample_inclusive_type),intent(in)::this
    character(*),intent(in)::dir
    logical,intent(in)::subdirs
    integer::inclusive_unit
    integer::n,n_hits
    character(2)::sample_name
    call generate_unit(inclusive_unit)
    open(inclusive_unit,file=dir//"/int_kinds.plot")
    n_hits=0
    do n=1,size(this%int_kinds)
       n_hits=n_hits+this%int_kinds(n)%n_hits
    end do
    do n=1,size(this%int_kinds)
       write(inclusive_unit,fmt=*)n,real(this%int_kinds(n)%n_hits)/n_hits
       call integer_with_leading_zeros(n,2,sample_name)
       if(subdirs)then
          call sample_int_kind_analyse(&
               this%int_kinds(n),&
               dir//"/"//sample_name,&
               "")
       else
          call sample_int_kind_analyse(&
               this%int_kinds(n),&
               dir,&
               sample_name//"_")
       end if
    end do
    close(inclusive_unit)
  end subroutine sample_inclusive_analyse
```

sample_inclusive_to_generator ↑

```
subroutine sample_inclusive_to_generator(this)
  class(sample_inclusive_type),intent(inout)::this
  integer::int_kind
  do int_kind=1,size(this%int_kinds)
     call this%int_kinds(int_kind)%to_generator()
  end do
end subroutine sample_inclusive_to_generator
```

sample_inclusive_allocate ↑

```
subroutine sample_inclusive_allocate(this,n_alloc)
  class(sample_inclusive_type),intent(out)::this
  integer,intent(in)::n_alloc
  allocate(this%int_kinds(n_alloc))
  this%n_alloc=n_alloc
end subroutine sample_inclusive_allocate
```

11.4.6 Sonstige Prozeduren

muli_mcint_generate_hit

```
pure subroutine muli_mcint_generate_hit(rnd,corners,hit)
  real(kind=double),dimension(2),intent(out)::hit
  integer,intent(in),dimension(2)::rnd
  real(kind=double),dimension(2,2),intent(in)::corners
  !print *,hit
  !print *,corners
  !print *,(corners(1:2,2)-corners(1:2,1))
  hit=(rnd/max_d)*(corners(1:2,2)-corners(1:2,1))+corners(1:2,1)
end subroutine muli_mcint_generate_hit
```

plot_pstvue3d

```
subroutine plot_pstvue3d(unit,corners,density)
  integer,intent(in)::unit
  real(kind=double),dimension(2,2),intent(in)::corners
  real(kind=double),intent(in)::density
  real(kind=double),dimension(2)::width,mean
  real(kind=double),dimension(3,3)::plot
  width=(corners(:,2)-corners(:,1))/2D0
  mean=(corners(:,1)+corners(:,2))/2D0
  plot(1,1)=width(1)
  plot(2,1)=width(2)
  plot(3,1)=density/2D0
  plot(1,2)=mean(1)
  plot(2,2)=mean(2)
  plot(3,2)=density/2D0
  call log_color_code(density,plot(1:3,3))
  if(density>1D0)then
```

```
      write(unit,fmt='("           mybigcube",F14.7,"",F14.7,"",F14.7,"&
          &",F14.7,"",F14.7,"",F14.7,"",F14.7,"",F14.7,"",F14.7,"")')plot
      return
   end if
   write(unit,fmt='("               mycube",F14.7,"",F14.7,"",F14.7,"&
       &",F14.7,"",F14.7,"",F14.7,"",F14.7,"",F14.7,"",F14.7,"")')plot
 end subroutine plot_pstvue3d
```

log_color_code

```
 subroutine log_color_code(number,rgb)
   real(kind=double),intent(in)::number
   real(kind=double),dimension(3),intent(out)::rgb
   if(number<exp(-5D0))then
      rgb=[0D0,0D0,exp(5D0)*number]
   else
      if(number<exp(-4D0))then
         rgb=[0D0,(number-exp(-5D0))/(exp(-4D0)-exp(-5D0)),1D0]
      else
         if(number<exp(-3D0))then
            rgb=[0D0,1D0,1D0-((number-exp(-4D0))/(exp(-3D0)-exp(-4D0)))]
         else
            if(number<exp(-2D0))then
               rgb=[(number-exp(-3D0))/(exp(-2D0)-exp(-3D0)),1D0,0D0]
            else
               if(number<exp(-1D0))then
                  rgb=[1D0,1D0-(number-exp(-2D0))/(exp(-1D0)-exp(-2D0)),0D0]
               else
                  if(number<1D0)then
                     rgb=[1D0,0D0,(number-exp(-3D0))/(1D0-exp(-3D0))]
                  else
                     rgb=[exp(1D0),1D0,1D0]*exp(-number)
                     return
                  end if
               end if
            end if
         end if
      end if
   end if
 end subroutine log_color_code
```

misc_sort

```
 recursive subroutine misc_sort(in,out)
   real(kind=double),dimension(:),intent(in)::in
   integer,dimension(:),intent(out)::out
   integer,dimension(:),allocatable::tmp
   integer::n,k,l,cut
   if(size(in)==1)then
      out=[1]
   else
      if(size(in)==2)then
```

```
        if(in(1)<=in(2))then
           out=[1,2]
        else
           out=[2,1]
        end if
     else
        cut=size(in)/2
        k=1
        l=cut+1
        allocate(tmp(size(in)))
        call misc_sort(in(1:cut),tmp(1:cut))
        call misc_sort(in(cut+1:),tmp(cut+1:))
        do n=cut+1,size(in)
           tmp(n)=tmp(n)+cut
        end do
        do n=1,size(in)
           if(k>cut)then
              out(n)=tmp(l)
              l=l+1
           else
              if(l>size(tmp))then
                 out(n)=tmp(k)
                 k=k+1
              else
                 if(in(tmp(k))<in(tmp(l)))then
                    out(n)=tmp(k)
                    k=k+1
                 else
                    out(n)=tmp(l)
                    l=l+1
                 end if
              end if
           end if
        end do
     end if
  end if
end subroutine misc_sort
```

12 Modul muli_cuba

12.1 Abhängigkeiten

```
use muli_momentum
```

12.2 Parameter

```
integer, parameter :: max_maxeval = huge(1)
```

12.3 Derived Types

12.3.1 cuba_class

```
type,extends(serializable_class), abstract :: cuba_class
   ! private
   real(kind=drk) :: start_time=0D0
   real(kind=drk) :: stop_time=0D0
   real(kind=drk) :: run_time=0D0
   ! common input
   integer :: dim_x = 2
   integer :: dim_f = 1
   type(transversal_momentum_type) :: userdata
   real(kind=drk) :: eps_rel = 1D-3
   real(kind=drk) :: eps_abs = 0D0
   integer :: flags = 0
   integer :: seed = 1
   integer :: min_eval = 0
   integer :: max_eval = max_maxeval
   ! common output
   integer :: neval = 0
   integer,public :: fail = -1
   integer :: nregions = 0
   real(kind=drk), dimension(:), allocatable :: integral
   real(kind=drk), dimension(:), allocatable :: error
   real(kind=drk), dimension(:), allocatable :: prob

   procedure(integrand_interface),nopass,pointer::integrand
 contains
   !Überschriebene serializable_class Methoden
```

```
    procedure::write_to_marker=>cuba_write_to_marker
    procedure::read_from_marker=>cuba_read_from_marker
    procedure::print_to_unit=>cuba_print_to_unit
    !Originäre cuba_class Methoden
    procedure ::get_integral_array=>cuba_get_integral_array
    procedure ::get_integral_1=>cuba_get_integral_1
    generic   ::get_integral=>get_integral_array,get_integral_1
    procedure ::copy_common=>cuba_copy_common
    procedure ::set_common=>cuba_set_common
    procedure ::set_dim_f=>cuba_set_dim_f
    procedure ::set_dim_x=>cuba_set_dim_x
    procedure ::reset_timer=>cuba_reset_timer
    procedure ::integrate_with_timer=>cuba_integrate_with_timer
    procedure ::integrate_associated=>cuba_integrate_associated
    procedure(integrate_interface), deferred :: integrate_nd
    procedure(integrate_userdata_interface), deferred :: integrate_userdata
    procedure(cuba_copy_interface), deferred :: copy

    procedure ::dealloc_dim_f=>cuba_dealloc_dim_f
    procedure ::alloc_dim_f=>cuba_alloc_dim_f
    procedure ::dealloc=>cuba_dealloc
    procedure ::alloc=>cuba_alloc
    generic ::integrate=>integrate_nd,integrate_userdata
 end type cuba_class
```

12.3.2 cuba_cuhre_type

```
 type,extends(cuba_class) :: cuba_cuhre_type
    private
    integer :: key = 13
  contains
    !Überschriebene serializable_class Methoden
    procedure::write_to_marker=>cuba_cuhre_write_to_marker
    procedure::read_from_marker=>cuba_cuhre_read_from_marker
    procedure::print_to_unit=>cuba_cuhre_print_to_unit
    procedure,nopass::get_type=>cuba_cuhre_get_type
    !Überschriebene cuba_class Methoden
    procedure ::integrate_nd=>integrate_cuhre
    procedure ::integrate_userdata=>integrate_cuhre_userdata
    procedure ::copy=>cuba_cuhre_copy
    procedure ::set_deferred=>cuba_cuhre_set_deferred
 end type cuba_cuhre_type
```

12.3.3 cuba_suave_type

```
 type,extends(cuba_class) :: cuba_suave_type
    private
    integer :: nnew = 10000 !1000
    integer :: flatness = 5 !50
```

```
  contains
    !Überschriebene serializable_class Methoden
    procedure::write_to_marker=>cuba_suave_write_to_marker
    procedure::read_from_marker=>cuba_suave_read_from_marker
    procedure::print_to_unit=>cuba_suave_print_to_unit
    procedure,nopass::get_type=>cuba_suave_get_type
    !Überschriebene cuba_class Methoden
    procedure ::integrate_nd=>integrate_suave
    procedure ::integrate_userdata=>integrate_suave_userdata
    procedure ::copy=>cuba_suave_copy
end type cuba_suave_type
```

12.3.4 cuba_divonne_type

```
type,extends(cuba_class) :: cuba_divonne_type
    private
    integer :: key1 = 13
    integer :: key2 = 13
    integer :: key3 = 13
    integer :: maxpass = 2
    real(kind=drk) :: border = 0D0
    real(kind=drk) :: maxchisq = 10D0
    real(kind=drk) :: mindeviation = .25D0
    integer :: ngiven = 0
    integer :: ldxgiven = 0
    real(kind=drk),dimension(:,:),allocatable :: xgiven
    integer :: nextra = 0
  contains
    !Überschriebene serializable_class Methoden
    procedure::write_to_marker=>cuba_divonne_write_to_marker
    procedure::read_from_marker=>cuba_divonne_read_from_marker
    procedure::print_to_unit=>cuba_divonne_print_to_unit
    procedure,nopass::get_type=>cuba_divonne_get_type
    !Überschriebene cuba_class Methoden
    procedure ::integrate_nd=>integrate_divonne
    procedure ::integrate_userdata=>integrate_divonne_userdata
    procedure ::copy=>cuba_divonne_copy
    procedure ::set_deferred=>cuba_divonne_set_deferred
end type cuba_divonne_type
```

12.3.5 cuba_vegas_type

```
type,extends(cuba_class) :: cuba_vegas_type
    private
    integer :: nstart = 500
    integer :: nincrease = 1000
    integer :: nbatch = 1000
    integer :: gridno = 0
    character(len=8),pointer :: statefile => null()
```

```
  contains
    !Überschriebene serializable_class Methoden
    procedure::write_to_marker=>cuba_vegas_write_to_marker
    procedure::read_from_marker=>cuba_vegas_read_from_marker
    procedure::print_to_unit=>cuba_vegas_print_to_unit
    procedure,nopass::get_type=>cuba_vegas_get_type
    !Überschriebene cuba_class Methoden
    procedure ::integrate_nd=>integrate_vegas
    procedure ::integrate_userdata=>integrate_vegas_userdata
    procedure ::copy=>cuba_vegas_copy
    procedure ::set_deferred=>cuba_vegas_set_deferred
end type cuba_vegas_type
```

12.4 Interfaces

```
interface
   subroutine integrand_interface(dim_x, x, dim_f, f,userdata)
     use kinds !NODEP!
     use muli_momentum
     integer, intent(in) :: dim_x, dim_f
     real(kind=drk), dimension(dim_x), intent(in) :: x
     real(kind=drk), dimension(dim_f), intent(out) :: f
     class(transversal_momentum_type), intent(in) :: userdata
   end subroutine integrand_interface
end interface
interface
   subroutine cuba_copy_interface(this,source)
     import :: cuba_class
     class(cuba_class),intent(out)::this
     class(cuba_class),intent(in)::source
   end subroutine cuba_copy_interface
   subroutine ca_plain(this)
     import :: cuba_class
     class(cuba_class) :: this
   end subroutine ca_plain
   subroutine integrate_interface(this, integrand)
     import :: cuba_class
     class(cuba_class),intent(inout) :: this
     interface
        subroutine integrand(dim_x, x, dim_f, f,userdata)
          use kinds !NODEP!
          use muli_momentum
          integer, intent(in) :: dim_x, dim_f
          real(kind=drk), dimension(dim_x), intent(in) :: x
          real(kind=drk), dimension(dim_f), intent(out) :: f
          class(transversal_momentum_type), intent(in) :: userdata
        end subroutine integrand
     end interface
   end subroutine integrate_interface
```

```
end interface
interface
   subroutine integrate_userdata_interface(this, integrand,userdata)
     use muli_momentum
     import :: cuba_class
     class(cuba_class),intent(inout) :: this
     interface
        subroutine integrand(dim_x, x, dim_f, f,userdata)
          use kinds !NODEP!
          use muli_momentum
          integer, intent(in) :: dim_x, dim_f
          real(kind=drk), dimension(dim_x), intent(in) :: x
          real(kind=drk), dimension(dim_f), intent(out) :: f
          class(transversal_momentum_type), intent(in) :: userdata
        end subroutine integrand
     end interface
     class(transversal_momentum_type),intent(in)::userdata
   end subroutine integrate_userdata_interface
end interface
```

12.5 Implementierung der Prozeduren

12.5.1 Methoden für cuba_class

Überschriebene serializable_class Methoden

cuba_write_to_marker ↑

```
subroutine cuba_write_to_marker(this,marker,status)
  class(cuba_class),intent(in)::this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  call marker%mark_begin("cuba_class")
  call marker%mark("dim_x",this%dim_x)
  call marker%mark("dim_f",this%dim_f)
  call marker%mark("eps_rel",this%eps_rel)
  call marker%mark("eps_abs",this%eps_abs)
  call marker%mark("flags",this%flags)
  call marker%mark("min_eval",this%min_eval)
  call marker%mark("max_eval",this%max_eval)
  call marker%mark("neval",this%neval)
  call marker%mark("fail",this%fail)
  call marker%mark("nregions",this%nregions)
  if(allocated(this%integral))then
     call marker%mark("integral",this%integral)
  else
     call marker%mark_null("integral")
  end if
  if(allocated(this%error))then
```

```
      call marker%mark("error",this%error)
   else
      call marker%mark_null("error")

   end if
   if(allocated(this%prob))then
      call marker%mark("prob",this%prob)
   else
      call marker%mark_null("prob")
   end if
   call marker%mark_null("cuba_class")
 end subroutine cuba_write_to_marker
```

cuba_read_from_marker ↑

```
 subroutine cuba_read_from_marker(this,marker,status)
   class(cuba_class),intent(out) :: this
   class(marker_type), intent(inout) :: marker
   integer(kind=dik),intent(out)::status
   call marker%pick_begin("CUBA_CLASS",status=status)
   call marker%pick("dim_x",this%dim_x,status)
   call marker%pick("dim_f",this%dim_f,status)
   call marker%pick("eps_rel",this%eps_rel,status)
   call marker%pick("eps_abs",this%eps_abs,status)
   call marker%pick("flags",this%flags,status)
   call marker%pick("min_eval",this%min_eval,status)
   call marker%pick("max_eval",this%max_eval,status)
   call marker%pick("neval",this%neval,status)
   call marker%pick("fail",this%fail,status)
   call marker%pick("nregions",this%nregions,status)
   call marker%verify_nothing("integral",status)
   if(allocated(this%integral))deallocate(this%integral)
   if(status==serialize_ok)then
      allocate(this%integral(this%dim_f))
      call marker%pick("integral",this%integral,status)
   end if
   call marker%verify_nothing("error",status)
   if(allocated(this%error))deallocate(this%error)
   if(status==serialize_ok)then
      allocate(this%error(this%dim_f))
      call marker%pick("error",this%error,status)
   end if
   call marker%verify_nothing("prob",status)
   if(allocated(this%prob))deallocate(this%prob)
   if(status==serialize_ok)then
      allocate(this%prob(this%dim_f))
      call marker%pick("prob",this%prob,status)
   end if
   call marker%pick_end("cuba_class",status)
 END SUBROUTINE cuba_read_from_marker
```

cuba_print_to_unit ↑

```
subroutine cuba_print_to_unit(this,unit,parents,components,peers)
  class(cuba_class),intent(in) :: this
  INTEGER, INTENT(IN) :: unit
  integer(kind=dik),intent(in)::parents,components,peers
  character(11)::n
  write(n,'("(",I2,"(E12.4))")')this%dim_f
  write(unit,'("Components of cuba_class:")')
  write(unit,'("Parameters:")')
  write(unit,'("dim_f:     ",I10)')   this%dim_f
  write(unit,'("dim_x:     ",I10)')   this%dim_x
  call this%userdata%print_to_unit(unit,parents,components-1,peers)
  write(unit,'("eps_rel:   ",E10.4)') this%eps_rel
  write(unit,'("eps_abs:   ",E10.4)') this%eps_abs
  write(unit,'("flags:     ",I10)')   this%flags
  write(unit,'("seed:      ",I10)')   this%seed
  write(unit,'("min_eval:  ",I10)')   this%min_eval
  write(unit,'("max_eval:  ",I10)')   this%max_eval
  write(unit,'("Results:")')
  write(unit,'("neval:     ",I10)')   this%neval
  write(unit,'("fail:      ",I10)')   this%fail
  write(unit,'("integral:  ")',advance="no")
  write(unit,fmt=n)this%integral
  write(unit,'("error:     ")',advance="no")
  write(unit,fmt=n)this%error
  write(unit,'("prob:      ")',advance="no")
  write(unit,fmt=n)this%prob
  write(unit,'("time:      ",E10.4)') this%stop_time-this%start_time
  !    write(unit,'("time:      ",E10.4)') this%run_time
end subroutine cuba_print_to_unit
```

Originäre cuba_class Methoden

cuba_integrate_associated ↑

```
subroutine cuba_integrate_associated(this)
  class(cuba_class),intent(inout)::this
  call cuba_integrate_with_timer(this,this%integrand)
end subroutine cuba_integrate_associated
```

cuba_integrate_with_timer ↑

```
subroutine cuba_integrate_with_timer(this,integrand)
  class(cuba_class),intent(inout)::this
  procedure(integrand_interface)::integrand
  call cpu_time(this%start_time)
  call this%integrate(integrand)
  call cpu_time(this%stop_time)
  this%run_time=this%run_time+this%stop_time-this%start_time
end subroutine cuba_integrate_with_timer
```

cuba_reset_timer ↑

```
  subroutine cuba_reset_timer(this)
    class(cuba_class),intent(inout)::this
    this%start_time=0D0
    this%stop_time=0D0
    this%run_time=0D0
  end subroutine cuba_reset_timer
```

cuba_get_integral_array ↑

```
  subroutine cuba_get_integral_array(this,integral)
    class(cuba_class) :: this
    real(kind=drk),intent(out),dimension(:) :: integral
    integral=this%integral
  end subroutine cuba_get_integral_array
```

cuba_get_integral_1 ↑

```
  subroutine cuba_get_integral_1(this,integral)
    class(cuba_class) :: this
    real(kind=drk),intent(out) :: integral
    integral=this%integral(1)
  end subroutine cuba_get_integral_1
```

cuba_dealloc_dim_f ↑

```
  subroutine cuba_dealloc_dim_f(this)
    class(cuba_class) :: this
    !      print '("cuba_dealloc_dim_f...")'
    if (allocated(this%integral)) then
       deallocate(this%integral)
    end if
    if (allocated(this%error)) then
       deallocate(this%error)
    end if
    if (allocated(this%prob)) then
       deallocate(this%prob)
    end if
    !      print '("done")'
  end subroutine cuba_dealloc_dim_f
```

cuba_dealloc ↑

```
  subroutine cuba_dealloc(this)
    class(cuba_class) :: this
    call this%dealloc_dim_f
  end subroutine cuba_dealloc
```

cuba_alloc_dim_f ↑

```
  subroutine cuba_alloc_dim_f(this)
    class(cuba_class) :: this
    call this%dealloc_dim_f()
    allocate(this%integral(this%dim_f))
```

```
    allocate(this%error(this%dim_f))
    allocate(this%prob(this%dim_f))
  end subroutine cuba_alloc_dim_f
```

cuba_alloc ↑

```
  subroutine cuba_alloc(this)
    class(cuba_class) :: this
    call this%alloc_dim_f
  end subroutine cuba_alloc
```

cuba_set_common ↑

```
  subroutine cuba_set_common&
    (this,dim_x,dim_f,eps_rel,eps_abs,flags,seed,min_eval,max_eval,integrand,userdata)
    class(cuba_class),intent(inout) :: this
    integer,intent(in),optional :: dim_x,dim_f,flags,min_eval,max_eval,seed
    real(kind=drk),intent(in),optional :: eps_rel,eps_abs
    type(transversal_momentum_type),intent(in),optional :: userdata
    procedure(integrand_interface),optional::integrand
    if(present(dim_x))then
       call this%set_dim_x(dim_x)
    end if
    if(present(dim_f))then
       call this%set_dim_f(dim_f)
    end if
    if(present(flags))then
       this%flags=flags
    end if
    if(present(seed))then
       this%seed=seed
    end if
    if(present(min_eval))then
       this%min_eval=min_eval
    end if
    if(present(max_eval))then
       if(max_eval<max_maxeval)then
          this%max_eval=max_eval
       else
          print '("cuba_set_common: Value of max_eval is too large.")'
          this%max_eval=max_maxeval
       end if
    end if
    if(present(eps_rel))then
       this%eps_rel=eps_rel
    end if
    if(present(eps_abs))then
       this%eps_abs=eps_abs
    end if
    if(present(integrand))this%integrand=>integrand
    if(present(userdata))this%userdata=userdata
  end subroutine cuba_set_common
```

cuba_set_dim_f ↑

```
subroutine cuba_set_dim_f(this,new_dim_f)
  class(cuba_class) :: this
  integer,intent(in) :: new_dim_f
  if (new_dim_f>0) then
    this%dim_f = new_dim_f
    call this%alloc_dim_f
  else
     write (*,'("cuba_set_dim_f: New value for dim_f is negative. dim_f is not set.")')
  end if
end subroutine cuba_set_dim_f
```

cuba_set_dim_x ↑

```
subroutine cuba_set_dim_x(this,new_dim_x)
  class(cuba_class) :: this
  integer,intent(in) :: new_dim_x
  if (new_dim_x>0) then
     this%dim_x = new_dim_x
  else
     write (*,'("cuba_set_dim_x: New value for dim_x is negative. dim_x is not set.")')
  end if
end subroutine cuba_set_dim_x
```

cuba_copy_common ↑

```
subroutine cuba_copy_common(this,source)
  class(cuba_class),intent(out) :: this
  class(cuba_class),intent(in) :: source
  this%dim_x = source%dim_x
  this%dim_f = source%dim_f
  this%eps_rel = source%eps_rel
  this%eps_abs = source%eps_abs
  this%flags = source%flags
  this%min_eval = source%min_eval
  this%max_eval = source%max_eval
  call this%alloc()
end subroutine cuba_copy_common
```

12.5.2 Methoden für cuba_vegas_type

Überschriebene serializable_class Methoden

cuba_vegas_write_to_marker ↑

```
subroutine cuba_vegas_write_to_marker(this,marker,status)
  class(cuba_vegas_type),intent(in) :: this
  class(marker_type), intent(inout) :: marker
  integer(kind=dik),intent(out)::status
  call marker%mark_begin("cuba_vegas_type")
```

```
    call cuba_write_to_marker(this,marker,status)
    call marker%mark("nstart",this%nstart)
    call marker%mark("nincrease",this%nincrease)
    call marker%mark_null("cuba_vegas_type")
  end subroutine cuba_vegas_write_to_marker
```

cuba_vegas_read_from_marker ↑

```
  subroutine cuba_vegas_read_from_marker(this,marker,status)
    class(cuba_vegas_type),intent(out) :: this
    class(marker_type), intent(inout) :: marker
    integer(kind=dik),intent(out)::status
    call marker%pick_begin("cuba_vegas_type",status=status)
    call cuba_read_from_marker(this,marker,status)
    call marker%pick("nstart",this%nstart,status)
    call marker%pick("nincrease",this%nincrease,status)
    call marker%pick_end("cuba_vegas_type",status)
  end subroutine cuba_vegas_read_from_marker
```

cuba_vegas_print_to_unit ↑

```
  subroutine cuba_vegas_print_to_unit(this,unit,parents,components,peers)
    class(cuba_vegas_type),intent(in) :: this
    INTEGER, INTENT(IN) :: unit
    integer(kind=dik),intent(in)::parents,components,peers
    if(parents>0)call cuba_print_to_unit(this,unit,parents-1,components,peers)
    write(unit,'("Components of cuba_vegas_type:")')
    write(unit,'("nstart:    ",I10)')   this%nstart
    write(unit,'("nincrease: ",I10)')   this%nincrease
    write(unit,'("nbatch:    ",I10)')   this%nbatch
    write(unit,'("gridno:    ",I10)')   this%gridno
    if(associated(this%statefile))then
       write(unit,'("statefile:",a)')   this%statefile
    else
       write(unit,'("statefile:",a)')   "not associated"
    end if
  end subroutine cuba_vegas_print_to_unit
```

cuba_vegas_get_type ↑

```
  pure subroutine cuba_vegas_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="cuba_vegas_type")
  end subroutine cuba_vegas_get_type
```

Überschriebene cuba_class Methoden

cuba_vegas_set_deferred ↑

```
subroutine cuba_vegas_set_deferred(this,n_start,n_increase,nbatch,gridno,statefile)
  class(cuba_vegas_type),intent(inout) :: this
  integer,intent(in),optional :: n_start,n_increase,nbatch,gridno
  character(len=*),intent(in),target,optional::statefile
  if(present(n_start))this%nstart=n_start
  if(present(n_increase))this%nincrease=n_increase
  if(present(nbatch))this%nbatch=nbatch
  if(present(gridno))this%gridno=gridno
  if(present(statefile))this%statefile=>statefile
end subroutine cuba_vegas_set_deferred
```

cuba_vegas_copy ↑

```
subroutine cuba_vegas_copy(this,source)
  class(cuba_vegas_type),intent(out) :: this
  class(cuba_class),intent(in) :: source
  select type(source)
  class is (cuba_vegas_type)
     call this%copy_common(source)
     this%nstart=source%nstart
     this%nincrease=source%nincrease
  class default
     print *,"cuba_vegas_copy: type of source is not type compatible with &
     &cuba_vegas_type."
  end select
end subroutine cuba_vegas_copy
```

integrate_vegas ↑

```
subroutine integrate_vegas(this,integrand)
  class(cuba_vegas_type),intent(inout) :: this
  procedure(integrand_interface)::integrand
  !      print '("vegas")'
  call vegas(&
       this%dim_x, &
       this%dim_f, &
       integrand, &
       this%userdata, &
       this%eps_rel, &
       this%eps_abs, &
       this%flags, &
       this%seed, &
       this%min_eval, &
       this%max_eval, &
       this%nstart, &
       this%nincrease, &
       this%nbatch, &
       this%gridno, &
       this%statefile, &
       this%neval, &
       this%fail, &
       this%integral, &
```

```
        this%error, &
        this%prob)
  end subroutine integrate_vegas
```

integrate_vegas_userdata ↑

```
  subroutine integrate_vegas_userdata(this,integrand,userdata)
    class(cuba_vegas_type),intent(inout) :: this
    procedure(integrand_interface)::integrand
    class(transversal_momentum_type),intent(in)::userdata
    !      print '("vegas")'
    call vegas(&
         this%dim_x, &
         this%dim_f, &
         integrand, &
         userdata, &
         this%eps_rel, &
         this%eps_abs, &
         this%flags, &
         this%seed, &
         this%min_eval, &
         this%max_eval, &
         this%nstart, &
         this%nincrease, &
         this%nbatch, &
         this%gridno, &
         this%statefile, &
         this%neval, &
         this%fail, &
         this%integral, &
         this%error, &
         this%prob)
  end subroutine integrate_vegas_userdata
```

12.5.3 Methoden für cuba_suave_type

Überschriebene serializable_class Methoden

cuba_suave_write_to_marker ↑

```
  subroutine cuba_suave_write_to_marker(this,marker,status)
    class(cuba_suave_type),intent(in) :: this
    class(marker_type), intent(inout) :: marker
    integer(kind=dik),intent(out)::status
    call marker%mark_begin("cuba_suave_type")
    call cuba_write_to_marker(this,marker,status)
    call marker%mark("nnew",this%nnew)
    call marker%mark("flatness",this%flatness)
    call marker%mark_null("cuba_suave_type")
  end subroutine cuba_suave_write_to_marker
```

cuba_suave_read_from_marker ↑

```
subroutine cuba_suave_read_from_marker(this,marker,status)
  class(cuba_suave_type),intent(out) :: this
  class(marker_type), intent(inout) :: marker
  integer(kind=dik),intent(out)::status
  call marker%pick_begin("cuba_suave_type",status=status)
  call cuba_read_from_marker(this,marker,status)
  call marker%pick("nnew",this%nnew,status)
  call marker%pick("flatnes",this%flatness,status)
  call marker%pick_end("cuba_suave_type",status)
end subroutine cuba_suave_read_from_marker
```

cuba_suave_print_to_unit ↑

```
subroutine cuba_suave_print_to_unit(this,unit,parents,components,peers)
  class(cuba_suave_type),intent(in) :: this
  INTEGER, INTENT(IN) :: unit
  integer(kind=dik),intent(in)::parents,components,peers
  if(parents>0)call cuba_print_to_unit(this,unit,parents-1,components,peers)
  write(unit,'("Components of cuba_suave_type:")')
  write(unit,'("nnew:       ",I10)')   this%nnew
  write(unit,'("flatness:   ",I10)')   this%flatness
end subroutine cuba_suave_print_to_unit
```

cuba_suave_get_type ↑

```
pure subroutine cuba_suave_get_type(type)
  character(:),allocatable,intent(out)::type
  allocate(type,source="cuba_suave_type")
end subroutine cuba_suave_get_type
```

Überschriebene cuba_class Methoden

integrate_suave ↑

```
subroutine integrate_suave(this,integrand)
  class(cuba_suave_type),intent(inout) :: this
  procedure(integrand_interface)::integrand
  !      print '("suave")'
  call suave(&
       this%dim_x, &
       this%dim_f, &
       integrand, &
       this%userdata, &
       this%eps_rel, &
       this%eps_abs, &
       this%flags, &
       this%seed, &
       this%min_eval, &
       this%max_eval, &
```

```
        this%nnew, &
        this%flatness, &
        this%nregions, &
        this%neval, &
        this%fail, &
        this%integral, &
        this%error, &
        this%prob)
  end subroutine integrate_suave
```

integrate_suave_userdata ↑

```
   subroutine integrate_suave_userdata(this,integrand,userdata)
    class(cuba_suave_type),intent(inout) :: this
    procedure(integrand_interface)::integrand
    class(transversal_momentum_type),intent(in)::userdata
    !      print '("suave")'
    call suave(&
        this%dim_x, &
        this%dim_f, &
        integrand, &
        userdata, &
        this%eps_rel, &
        this%eps_abs, &
        this%flags, &
        this%seed, &
        this%min_eval, &
        this%max_eval, &
        this%nnew, &
        this%flatness, &
        this%nregions, &
        this%neval, &
        this%fail, &
        this%integral, &
        this%error, &
        this%prob)
  end subroutine integrate_suave_userdata
```

cuba_suave_copy ↑

```
  subroutine cuba_suave_copy(this,source)
    class(cuba_suave_type),intent(out) :: this
    class(cuba_class),intent(in) :: source
    select type(source)
    class is (cuba_suave_type)
       call this%copy_common(source)
       this%nnew = source%nnew
       this%flatness = source%flatness
    class default
       print *,"cuba_suave_copy: type of source is not type compatible with cuba_suave_type."
    end select
  end subroutine cuba_suave_copy
```

12.5.4 Methoden für cuba_divonne_type

Überschriebene serializable_class Methoden

cuba_divonne_write_to_marker ↑

```
subroutine cuba_divonne_write_to_marker(this,marker,status)
  class(cuba_divonne_type),intent(in) :: this
  class(marker_type), intent(inout) :: marker
  integer(kind=dik),intent(out)::status
  call marker%mark_begin("cuba_divonne_type")
  call cuba_write_to_marker(this,marker,status)
  call marker%mark("key1",this%key1)
  call marker%mark("key2",this%key2)
  call marker%mark("key3",this%key3)
  call marker%mark("maxpass",this%maxpass)
  call marker%mark("border",this%border)
  call marker%mark("maxchisq",this%maxchisq)
  call marker%mark("mindeviation",this%mindeviation)
  call marker%mark("ngiven",this%ngiven)
  call marker%mark("ldxgiven",this%ldxgiven)
  call marker%mark("nextra",this%nextra)
  call marker%mark("xgiven",this%xgiven)
  call marker%mark_null("cuba_divonne_type")
end subroutine cuba_divonne_write_to_marker
```

cuba_divonne_read_from_marker ↑

```
subroutine cuba_divonne_read_from_marker(this,marker,status)
  class(cuba_divonne_type),intent(out) :: this
  class(marker_type), intent(inout) :: marker
  integer(kind=dik),intent(out)::status
  call marker%pick_begin("cuba_divonne_type",status=status)
  call cuba_read_from_marker(this,marker,status)
  call marker%pick("key1",this%key1,status)
  call marker%pick("key2",this%key2,status)
  call marker%pick("key3",this%key3,status)
  call marker%pick("maxpass",this%maxpass,status)
  call marker%pick("border",this%border,status)
  call marker%pick("maxchisq",this%maxchisq,status)
  call marker%pick("mindeviation",this%mindeviation,status)
  call marker%pick("ngiven",this%ngiven,status)
  call marker%pick("ldxgiven",this%ldxgiven,status)
  call marker%pick("nextra",this%nextra,status)
  if(allocated(this%xgiven))deallocate(this%xgiven)
  allocate(this%xgiven(this%ldxgiven,this%ngiven))
  call marker%pick("xgiven",this%xgiven,status)
  call marker%pick_end("cuba_divonne_type",status)
end subroutine cuba_divonne_read_from_marker
```

cuba_divonne_print_to_unit ↑

```
subroutine cuba_divonne_print_to_unit(this,unit,parents,components,peers)
  class(cuba_divonne_type),intent(in) :: this
  INTEGER, INTENT(IN) :: unit
  integer(kind=dik),intent(in)::parents,components,peers
  if(parents>0)call cuba_print_to_unit(this,unit,parents-1,components,peers)
  write(unit,'("Components of cuba_divonne_type:")')
  write(unit,'("key1:       ",I10)')   this%key1
  write(unit,'("key2:       ",I10)')   this%key2
  write(unit,'("key3:       ",I10)')   this%key3
  write(unit,'("maxpass:    ",I10)')   this%maxpass
  write(unit,'("ngiven:     ",I10)')   this%ngiven
  write(unit,'("ldxgiven:   ",I10)')   this%ldxgiven
  write(unit,'("nextra:     ",I10)')   this%nextra
  write(unit,'("border:     ",E10.4)') this%border
  write(unit,'("maxchisq:   ",E10.4)') this%maxchisq
  write(unit,'("mindeviation:",E10.4)') this%mindeviation
  write(unit,'("xgiven:     ",2(E10.4))') this%xgiven
end subroutine cuba_divonne_print_to_unit
```

cuba_divonne_get_type ↑

```
pure subroutine cuba_divonne_get_type(type)
  character(:),allocatable,intent(out)::type
  allocate(type,source="cuba_divonne_type")
end subroutine cuba_divonne_get_type
```

Überschriebene cuba_class Methoden

integrate_divonne ↑

```
subroutine integrate_divonne(this,integrand)
  class(cuba_divonne_type),intent(inout) :: this
  procedure(integrand_interface)::integrand
  !      call this%reset_output()
  !    print '("divonne")'
  call divonne(&
       & this%dim_x, &
       & this%dim_f, &
       & integrand, &
       & this%userdata,&
       & this%eps_rel, &
       & this%eps_abs, &
       & this%flags, &
       & this%seed, &
       & this%min_eval, &
       & this%max_eval, &
       & this%key1, &
       & this%key2, &
       & this%key3, &
       & this%maxpass, &
       & this%border, &
```

```
         & this%maxchisq, &
         & this%mindeviation, &
         & this%ngiven, &
         & this%ldxgiven, &
         & this%xgiven, &
         & this%nextra, &
                                !         & this%peakfinder, &
         & 0,&
         & this%nregions, &
         & this%neval, &
         & this%fail, &
         & this%integral, &
         & this%error, &
         & this%prob)
  end subroutine integrate_divonne
```

integrate_divonne_userdata ↑

```
subroutine integrate_divonne_userdata(this,integrand,userdata)
    class(cuba_divonne_type),intent(inout) :: this
    procedure(integrand_interface)::integrand
    class(transversal_momentum_type),intent(in)::userdata
    !      call this%reset_output()
    !    print '("divonne")'
    call divonne(&
         & this%dim_x, &
         & this%dim_f, &
         & integrand, &
         & userdata,&
         & this%eps_rel, &
         & this%eps_abs, &
         & this%flags, &
         & this%seed, &
         & this%min_eval, &
         & this%max_eval, &
         & this%key1, &
         & this%key2, &
         & this%key3, &
         & this%maxpass, &
         & this%border, &
         & this%maxchisq, &
         & this%mindeviation, &
         & this%ngiven, &
         & this%ldxgiven, &
         & this%xgiven, &
         & this%nextra, &
                                !         & this%peakfinder, &
         & 0,&
         & this%nregions, &
         & this%neval, &
         & this%fail, &
```

```
        & this%integral, &
        & this%error, &
        & this%prob)
  end subroutine integrate_divonne_userdata
```

cuba_divonne_copy ↑

```
  subroutine cuba_divonne_copy(this,source)
    class(cuba_divonne_type),intent(out) :: this
    class(cuba_class),intent(in) :: source
    select type(source)
    class is (cuba_divonne_type)
       call this%copy_common(source)
       call this%set_deferred(&
       &source%key1,&
       &source%key2,&
       &source%key3,&
       &source%maxpass,&
       &source%border,&
       &source%maxchisq,&
       &source%mindeviation,&
       &source%xgiven&
       &)
    class default
       print *,"cuba_divonne_copy: type of source is not type compatible with cuba_divonne_type."
    end select
  end subroutine cuba_divonne_copy
```

cuba_divonne_set_deferred ↑

```
  subroutine cuba_divonne_set_deferred&
    (this,key1,key2,key3,maxpass,border,maxchisq,mindeviation,xgiven,xgiven_flat)
    class(cuba_divonne_type) :: this
    integer,optional,intent(in)::key1,key2,key3,maxpass
    real(kind=drk),optional,intent(in)::border,maxchisq,mindeviation
    real(kind=drk),dimension(:,:),optional,intent(in)::xgiven
    real(kind=drk),dimension(:),optional,intent(in)::xgiven_flat
    integer,dimension(2)::s
    if(present(key1))this%key1=key1
    if(present(key2))this%key2=key2
    if(present(key3))this%key3=key3
    if(present(maxpass))this%maxpass=maxpass
    if(present(border))this%border=border
    if(present(maxchisq))this%maxchisq=maxchisq
    if(present(mindeviation))this%mindeviation=mindeviation
    if(present(xgiven))then
       if(allocated(this%xgiven))deallocate(this%xgiven)
       s=shape(xgiven)
       if(s(1)==this%dim_x)then
          allocate(this%xgiven(s(1),s(2)),source=xgiven)
          this%ldxgiven=s(1)
          this%ngiven=s(2)
```

```
      else
         print *,"cuba_divonne_set_deferred: shape of xgiven is not [dim_x,:]."
         this%ngiven=0
      end if
   end if
   if(present(xgiven_flat))then
      if(allocated(this%xgiven))deallocate(this%xgiven)
      if(mod(size(xgiven_flat),this%dim_x)==0)then
         this%ngiven=size(xgiven_flat)/this%dim_x
         this%ldxgiven=this%dim_x
         allocate(this%xgiven(this%ldxgiven,this%ngiven),&
                  source=reshape(xgiven_flat,[this%ldxgiven,this%ngiven]))
      else
         print *,"cuba_divonne_set_deferred: size of xgiven_flat is no multiple of dim_x."
         this%ngiven=0
      end if
   end if
 end subroutine cuba_divonne_set_deferred
```

12.5.5 Methoden für cuba_vegas_type

Überschriebene serializable_class Methoden

cuba_cuhre_write_to_marker ↑

```
 subroutine cuba_cuhre_write_to_marker(this,marker,status)
   class(cuba_cuhre_type),intent(in) :: this
   class(marker_type), intent(inout) :: marker
   integer(kind=dik),intent(out)::status
   call marker%mark_begin("cuba_cuhre_type")
   call cuba_write_to_marker(this,marker,status)
   call marker%mark("key",this%key)
   call marker%pick_end("cuba_cuhre_type",status)
 end subroutine cuba_cuhre_write_to_marker
```

cuba_cuhre_read_from_marker ↑

```
 subroutine cuba_cuhre_read_from_marker(this,marker,status)
   class(cuba_cuhre_type),intent(out) :: this
   class(marker_type), intent(inout) :: marker
   integer(kind=dik),intent(out)::status
   call marker%pick_begin("cuba_cuhre_type",status=status)
   call cuba_read_from_marker(this,marker,status)
   call marker%pick("key",this%key,status)
   call marker%pick_end("cuba_cuhre_type",status)
 end subroutine cuba_cuhre_read_from_marker
```

cuba_cuhre_print_to_unit ↑

```
  subroutine cuba_cuhre_print_to_unit(this,unit,parents,components,peers)
    class(cuba_cuhre_type),intent(in) :: this
    integer, intent(in) :: unit
    integer(kind=dik),intent(in)::parents,components,peers
    if(parents>0)call cuba_print_to_unit(this,unit,parents-1,components,peers)
    write(unit,'("Components of cuba_cuhre_type:")')
    write(unit,'("key:       ",I10)')   this%key
  end subroutine cuba_cuhre_print_to_unit
```

cuba_cuhre_get_type ↑

```
  pure subroutine cuba_cuhre_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="cuba_cuhre_type")
  end subroutine cuba_cuhre_get_type
```

Überschriebene cuba_class Methoden

integrate_cuhre ↑

```
  subroutine integrate_cuhre(this,integrand)
    class(cuba_cuhre_type),intent(inout) :: this
    procedure(integrand_interface)::integrand
    !c      print '("cuhre")'
    call cuhre(&
         this%dim_x, &
         this%dim_f, &
         integrand, &
         this%userdata, &
         this%eps_rel, &
         this%eps_abs, &
         this%flags, &
!         this%seed, &
         this%min_eval, &
         this%max_eval, &
         this%key, &
         this%nregions, &
         this%neval, &
         this%fail, &
         this%integral, &
         this%error, &
         this%prob)
  end subroutine integrate_cuhre
```

integrate_cuhre_userdata ↑

```
  subroutine integrate_cuhre_userdata(this,integrand,userdata)
    class(cuba_cuhre_type),intent(inout) :: this
    procedure(integrand_interface)::integrand
    class(transversal_momentum_type),intent(in)::userdata
    !c      print '("cuhre")'
```

```
    call cuhre(&
         this%dim_x, &
         this%dim_f, &
         integrand, &
         userdata, &
         this%eps_rel, &
         this%eps_abs, &
         this%flags, &
!          this%seed, &
         this%min_eval, &
         this%max_eval, &
         this%key, &
         this%nregions, &
         this%neval, &
         this%fail, &
         this%integral, &
         this%error, &
         this%prob)
  end subroutine integrate_cuhre_userdata
```

cuba_cuhre_copy ↑

```
  subroutine cuba_cuhre_copy(this,source)
    class(cuba_cuhre_type),intent(out) :: this
    class(cuba_class),intent(in) :: source
    select type(source)
    class is (cuba_cuhre_type)
       call this%copy_common(source)
       this%key=source%key
    class default
       print *,"cuba_cuhre_copy: type of source is not type compatible with &
       &cuba_cuhre_type."
    end select
  end subroutine cuba_cuhre_copy
```

cuba_cuhre_set_deferred ↑

```
  subroutine cuba_cuhre_set_deferred(this,key)
    class(cuba_cuhre_type),intent(inout) :: this
    integer, intent(in) :: key
    this%key = key
  end subroutine cuba_cuhre_set_deferred
```

13 Modul muli_basic

13.1 Abhängigkeiten

```
use,intrinsic::iso_fortran_env
use kinds !NODEP!
use iso_varying_string, string_t=>varying_string !NODEP!
```

13.2 Parameter

```
! bitmodel parameters
integer,public,parameter::drk=double
integer,public,parameter::dik=i64
integer(kind=dik),public,parameter::one=int(1,kind=dik)
integer(kind=dik),public,parameter::zero=int(0,kind=dik)
! serialization parameters
integer(kind=dik),public,parameter::serialize_page_size=1024
integer(kind=dik),public,parameter::serialize_ok=0000
integer(kind=dik),public,parameter::serialize_syntax_error=1001
integer(kind=dik),public,parameter::serialize_wrong_tag=1002
integer(kind=dik),public,parameter::serialize_wrong_id=1003
integer(kind=dik),public,parameter::serialize_wrong_type=1004
integer(kind=dik),public,parameter::serialize_wrong_name=1005
integer(kind=dik),public,parameter::serialize_no_target=1006
integer(kind=dik),public,parameter::serialize_no_pointer=1007
integer(kind=dik),public,parameter::serialize_wrong_action=1008
integer(kind=dik),public,parameter::serialize_unexpected_content=1009
integer(kind=dik),public,parameter::serialize_null=1010
integer(kind=dik),public,parameter::serialize_nothing=1011
logical,public,parameter::serialize_default_indent=.true.
logical,public,parameter::serialize_default_line_break=.true.
logical,public,parameter::serialize_default_asynchronous=.false.
! private components
integer(kind=dik),private::last_id=0
character(len=*),private,parameter::serialize_integer_characters="-0123456789"
```

13.3 Derived Types

13.3.1 serializable_class

```
type,public,abstract::serializable_class
 contains
   procedure(ser_write_if),deferred::write_to_marker
   procedure(ser_read_if),deferred::read_from_marker
   procedure(ser_unit),deferred::print_to_unit
   procedure(ser_type),nopass,deferred::get_type
   procedure,nopass::verify_type=>serializable_verify_type
   procedure::read_target_from_marker=>serializable_read_target_from_marker
   procedure::write_type=>serializable_write_type
   procedure::print=>serializable_print
   procedure::print_error=>serializable_print_error
   procedure::print_all=>serializable_print_all
   procedure::print_little=>serializable_print_little
   procedure::print_parents=>serializable_print_parents
   procedure::print_components=>serializable_print_components
   procedure::print_peers=>serializable_print_peers
   procedure::serialize_to_file=>serializable_serialize_to_file
   procedure::serialize_to_unit=>serializable_serialize_to_unit
   procedure::serialize_to_marker=>serializable_serialize_to_marker
   procedure::deserialize_from_file=>serializable_deserialize_from_file
   procedure::deserialize_from_unit=>serializable_deserialize_from_unit
   procedure::deserialize_from_marker=>serializable_deserialize_from_marker
   generic::serialize=>serialize_to_file,serialize_to_unit,serialize_to_marker
   generic::deserialize=>deserialize_from_file,deserialize_from_unit,deserialize_from_marke
end type serializable_class
```

13.3.2 measurable_class

```
type,public,abstract,extends(serializable_class)::measurable_class
 contains
   procedure(measure_int),public,deferred::measure
end type measurable_class
```

13.3.3 identified_type

```
type,public,extends(serializable_class)::identified_type
   private
   integer(kind=dik)::id
   type(string_t)::name
 contains
   ! overridden serializable_class procedures
   procedure,public::write_to_marker=>identified_write_to_marker
   procedure,public::read_from_marker=>identified_read_from_marker
   procedure,public::print_to_unit=>identified_print_to_unit
   procedure,public,nopass::get_type=>identified_get_type
   procedure,nopass::verify_type=>identified_verify_type
   ! new procedures
   procedure,public::identified_initialize
   procedure,public::get_id=>identified_get_id
```

```
    procedure,public::get_name=>identified_get_name
    generic,public::initialize=>identified_initialize
 end type identified_type
```

13.3.4 unique_type

```
 type,public,extends(identified_type)::unique_type
    private
    integer(kind=dik)::unique_id
  contains
    ! overridden serializable_class procedures
    procedure,public,nopass::get_type=>unique_get_type
    procedure,nopass::verify_type=>unique_verify_type
    procedure,public::write_to_marker=>unique_write_to_marker
    procedure,public::read_from_marker=>unique_read_from_marker
    procedure,public::print_to_unit=>unique_print_to_unit
    ! overridden identified_type procedures
    procedure,public::identified_initialize=>unique_initialize
    ! new procedures
    procedure,public::get_unique_id=>unique_get_unique_id
 end type unique_type
```

13.3.5 serializable_ref_type

```
 type,private::serializable_ref_type
    private
    integer(kind=dik)::id
    class(serializable_class),pointer::ref=>null()
    class(serializable_ref_type),pointer::next=>null()
  contains
    procedure,public::finalize=>serializable_ref_finalize
 end type serializable_ref_type
```

13.3.6 position_stack_type

```
 type::position_stack_type
    private
    integer(kind=dik),dimension(2)::position
    class(position_stack_type),pointer::next=>null()
  contains
    procedure,public::push_head=>position_stack_push_head
    procedure,public::push_given=>position_stack_push_given
    procedure,public::position_stack_pop
    procedure,public::position_stack_drop
    procedure,public::nth_position=>position_stack_nth_position
    procedure,public::first=>position_stack_first
    procedure,public::last=>position_stack_last
    procedure,public::range=>position_stack_range
```

```
    generic,public::push=>push_head
    generic,public::push=>push_given
    generic,public::pop=>position_stack_pop
    generic,public::push=>position_stack_drop
  end type position_stack_type
```

13.3.7 page_ring_type

```
  type,public::page_ring_type
    private
    logical::asynchronous=serialize_default_asynchronous
    logical::eof_reached=.false.
    integer::unit=-1
    integer(kind=dik)::ring_size=2
    integer(kind=dik)::action=0
    integer(kind=dik)::eof_int=-1
    integer(kind=dik)::out_unit=output_unit
    integer(kind=dik)::err_unit=error_unit
    integer(kind=dik),dimension(2)::active_pages=[0,-1]
    integer(kind=dik),dimension(2)::eof_pos=[-1,-1]
    type(string_t)::eof_string
    type(position_stack_type)::position_stack
    character(serialize_page_size),dimension(:),allocatable::ring
   contains
    ! read access only procedures:
    procedure,public::open_for_read_access=>page_ring_open_for_read_access
    procedure,public::read_page=>page_ring_read_page
    ! write access only procedures:
    procedure,public::open_for_write_access=>page_ring_open_for_write_access
    procedure,public::flush=>page_ring_flush
    procedure,public::break=>page_ring_break
    ! comparing
    procedure,public::str_equal=>page_ring_str_equal
    ! searching:
    procedure,public::find_pure=>page_ring_find_pure
    generic,  public::find=>page_ring_find,page_ring_find_default
    ! positioning:
    procedure,public::set_position=>page_ring_set_position
    procedure,public::turn_page=>page_ring_turn_page
    procedure,public::proceed=>page_ring_proceed
    generic,  public::push_position=>push_actual_position,push_given_position
    generic,  public::pop_position=>pop_actual_position,pop_given_position
    generic,  public::get_position=>page_ring_get_position1,page_ring_get_position2
    ! printing:
    procedure,public::print_to_unit=>page_ring_print_to_unit
    procedure,public::print_ring=>page_ring_print_ring
    procedure,public::print_position=>page_ring_print_position
    ! writing:
    procedure,public::put=>page_ring_put
    generic,  public::push=>push_string,push_integer,push_integer_dik,push_double,push_integ
```

```
! reading:
procedure,public::get_character=>page_ring_get_character
procedure,public::allocate_substring=>page_ring_allocate_substring
procedure,public::pop_character=>page_ring_pop_character
procedure,public::pop_by_keys=>page_ring_pop_by_keys
generic,  public::substring=>page_ring_substring1,page_ring_substring2
generic,  public::substring_by_keys=>page_ring_character_by_keys,page_ring_positions_by_keys
generic,  public::pop=>pop_string,pop_integer,pop_integer_dik,pop_double,pop_logical,pop_integ
! misc:
procedure,public::close=>page_ring_close
procedure,public::ring_index=>page_ring_ring_index
! private:
procedure,private::activate_next_page=>page_ring_activate_next_page
procedure,private::enlarge=>page_ring_enlarge
! specific names for generic procedures:
procedure,private::page_ring_substring1
procedure,private::page_ring_substring2
procedure,private::page_ring_character_by_keys
procedure,private::page_ring_positions_by_keys
procedure,private::push_string=>page_ring_push_string
procedure,private::push_integer=>page_ring_push_integer
procedure,private::push_integer_dik=>page_ring_push_integer_dik
procedure,private::push_integer_array=>page_ring_push_integer_array
procedure,private::push_integer_array_dik=>page_ring_push_integer_array_dik
procedure,private::push_double=>page_ring_push_double
procedure,private::push_double_array=>page_ring_push_double_array
procedure,private::pop_string=>page_ring_pop_string
procedure,private::pop_integer=>page_ring_pop_integer
procedure,private::pop_integer_dik=>page_ring_pop_integer_dik
procedure,private::pop_logical=>page_ring_pop_logical
procedure,private::pop_integer_array=>page_ring_pop_integer_array
procedure,private::pop_integer_array_dik=>page_ring_pop_integer_array_dik
procedure,private::pop_double=>page_ring_pop_double
procedure,private::pop_double_array=>page_ring_pop_double_array
procedure,private::page_ring_find
procedure,private::page_ring_find_default

procedure,private::actual_index=>page_ring_actual_index
procedure,private::actual_page=>page_ring_actual_page
procedure,private::actual_offset=>page_ring_actual_offset
procedure,private::actual_position=>page_ring_actual_position
procedure,private::first_index=>page_ring_first_index
procedure,private::first_page=>page_ring_first_page
procedure,private::last_index=>page_ring_last_index
procedure,private::last_page=>page_ring_last_page
procedure,private::push_actual_position=>page_ring_ring_push_actual_position
procedure,private::push_given_position=>page_ring_ring_push_given_position
procedure,private::pop_actual_position=>page_ring_ring_pop_actual_position
procedure,private::pop_given_position=>page_ring_ring_pop_given_position
procedure,private::page_ring_get_position1
procedure,private::page_ring_get_position2
```

```
end type page_ring_type
```

13.3.8 marker_type

```
type,public,extends(page_ring_type)::marker_type
   private
   integer(kind=dik)::indentation=0
   integer(kind=dik)::n_instances=0
   logical::do_break=.true.
   logical::do_indent=.false.
   class(serializable_ref_type),pointer::heap=>null()
   class(serializable_ref_type),pointer::references=>null()
 contains
   procedure::mark_begin=>marker_mark_begin
   procedure::mark_instance_begin=>marker_mark_instance_begin
   procedure::mark_end=>marker_mark_end
   procedure::mark_instance_end=>marker_mark_instance_end
   procedure::mark_logical=>marker_mark_logical
   procedure::mark_integer=>marker_mark_integer
   procedure::mark_integer_array=>marker_mark_integer_array
   procedure::mark_integer_matrix=>marker_mark_integer_matrix
   procedure::mark_integer_dik=>marker_mark_integer_dik
   procedure::mark_integer_array_dik=>marker_mark_integer_array_dik
   procedure::mark_integer_matrix_dik=>marker_mark_integer_matrix_dik
   procedure::mark_double=>marker_mark_double
   procedure::mark_double_array=>marker_mark_double_array
   procedure::mark_double_matrix=>marker_mark_double_matrix
   procedure::mark_string=>marker_mark_string
   procedure::mark_instance=>marker_mark_instance
   procedure::mark_target=>marker_mark_target
   procedure::mark_allocatable=>marker_mark_allocatable
   procedure::mark_pointer=>marker_mark_pointer
   procedure::mark_null=>marker_mark_null
   procedure::mark_nothing=>marker_mark_nothing
   procedure::mark_empty=>marker_mark_empty
   procedure::pick_begin=>marker_pick_begin
   procedure::query_instance_begin=>marker_query_instance_begin
   procedure::pick_instance_begin=>marker_pick_instance_begin
   procedure::pick_end=>marker_pick_end
   procedure::pick_instance_end=>marker_pick_instance_end
   procedure::pick_instance=>marker_pick_instance
   procedure::pick_target=>marker_pick_target
   procedure::pick_allocatable=>marker_pick_allocatable
   procedure::pick_pointer=>marker_pick_pointer
   procedure::pick_logical=>marker_pick_logical
   procedure::pick_integer=>marker_pick_integer
   procedure::pick_integer_array=>marker_pick_integer_array
   procedure::pick_integer_matrix=>marker_pick_integer_matrix
   procedure::pick_integer_dik=>marker_pick_integer_dik
   procedure::pick_integer_array_dik=>marker_pick_integer_array_dik
```

```
    procedure::pick_integer_matrix_dik=>marker_pick_integer_matrix_dik
    procedure::pick_double=>marker_pick_double
    procedure::pick_double_array=>marker_pick_double_array
    procedure::pick_double_matrix=>marker_pick_double_matrix
    procedure::pick_string=>marker_pick_string
    generic,public::mark=>mark_logical,&
         mark_integer,mark_integer_array,mark_integer_matrix,&
         mark_integer_dik,mark_integer_array_dik,mark_integer_matrix_dik,&
         mark_double,mark_double_array,mark_double_matrix,mark_string
    generic,public::pick=>pick_logical,&
         pick_integer,pick_integer_array,pick_integer_matrix,&
         pick_integer_dik,pick_integer_array_dik,pick_integer_matrix_dik,&
         pick_double,pick_double_array,pick_double_matrix,pick_string
    procedure::verify_nothing=>marker_verify_nothing
    procedure::indent=>marker_indent
    procedure::push_heap=>marker_push_heap
    procedure::pop_heap=>marker_pop_heap
    procedure::search_heap_by_id=>marker_search_heap_by_id
    procedure::search_heap_by_ref=>marker_search_heap_by_ref
    procedure::push_reference=>marker_push_reference
    procedure::pop_reference=>marker_pop_reference
    procedure::reset_references=>marker_reset_references
    procedure::search_reference=>marker_search_reference
    procedure::reset_heap=>marker_reset_heap
    procedure::finalize=>marker_finalize
    generic::search_heap=>search_heap_by_id
    generic::search_heap=>search_heap_by_ref
end type marker_type
```

13.4 Interfaces

```
abstract interface
   elemental function measure_int(this)
     import measurable_class
     import drk
     class(measurable_class),intent(in)::this
     real(kind=drk)::measure_int
   end function measure_int
end interface
interface operator(<)
   module procedure measurable_less_measurable
   module procedure measurable_less_double
end interface
interface operator(<=)
   module procedure measurable_less_or_equal_measurable
   module procedure measurable_less_or_equal_double
end interface
interface operator(==)
   module procedure measurable_equal_measurable
```

```
      module procedure measurable_equal_double
   end interface
   interface operator(>=)
      module procedure measurable_equal_or_greater_measurable
      module procedure measurable_equal_or_greater_double
   end interface
   interface operator(>)
      module procedure measurable_greater_measurable
      module procedure measurable_greater_double
   end interface
   abstract interface
      subroutine ser_write_if(this,marker,status)
        import serializable_class
        import marker_type
        import dik
        class(serializable_class),intent(in)::this
        class(marker_type),intent(inout)::marker
        integer(kind=dik),intent(out)::status
      end subroutine ser_write_if
   end interface
   abstract interface
      subroutine ser_read_if(this,marker,status)
        import serializable_class
        import marker_type
        import dik
        class(serializable_class),intent(out)::this
        class(marker_type),intent(inout)::marker
        integer(kind=dik),intent(out)::status
      end subroutine ser_read_if
   end interface
   abstract interface
      subroutine ser_unit(this,unit,parents,components,peers)
        import serializable_class
        import dik
        class(serializable_class),intent(in)::this
        integer,intent(in)::unit
        integer(kind=dik),intent(in)::parents,components,peers
      end subroutine ser_unit
   end interface
   abstract interface
      pure subroutine ser_type(type)
        character(:),allocatable,intent(out)::type
      end subroutine ser_type
   end interface
   interface page_ring_position_is_before
      module procedure &
           page_ring_position_is_before_int_pos,&
           page_ring_position_is_before_pos_pos,&
           page_ring_position_is_before_pos_int
   end interface
```

13.5 Operators

```
public operator(<),operator(<=),operator(>=),operator(>)
public serialize_print_comp_pointer,serialize_print_peer_pointer&
     &,serialize_print_allocatable
public identified_initialize,identified_print_to_unit&
     &,identified_read_from_marker,identified_write_to_marker

public serializable_deserialize_from_marker
public ilog2,generate_unit,integer_with_leading_zeros
```

13.6 Implementierung der Prozeduren

13.6.1 Methoden für serializable_class

serializable_verify_type ↑

```
elemental logical function serializable_verify_type(type) result(match)
  character(*),intent(in)::type
  match=type=="serializable_class"
end function serializable_verify_type
```

serializable_read_target_from_marker ↑

```
subroutine serializable_read_target_from_marker(this,marker,status)
  ! This is a dummy procedure. Usually, you dont't need to deserialize targets,
  ! so by implementing this dummy we don"t force all descendants to override this
  ! procedure. Then again this is the only way to read targets from markers.
  class(serializable_class),target,intent(out) :: this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  print *,"serializable_read_target_from_marker:"
  print *,"This is a dummy procedure. Usually, this message indicates a missing overridden &
       &read_target_from_marker TPB for "
  call this%write_type(output_unit)
  print *,""
  call this%read_from_marker(marker,status)
end subroutine serializable_read_target_from_marker
```

serializable_write_type ↑

```
subroutine serializable_write_type(this,unit)
  class(serializable_class),intent(in)::this
  integer,intent(in)::unit
  character(:),allocatable::this_type
  call this%get_type(this_type)
  write(unit,fmt='(a)',advance="no")this_type
end subroutine serializable_write_type
```

serializable_print ↑

```
recursive subroutine serializable_print(this,parents,components,peers,unit)
  class(serializable_class),intent(in)::this
  integer(kind=dik),intent(in)::parents,components,peers
  integer,optional::unit
  if(present(unit))then
     write(unit,'("")')
     write(unit,'("Instance of type: ")',advance="no")
     call this%write_type(unit)
     write(unit,fmt='("")')
     call this%print_to_unit(unit,parents,components,peers)
  else
     write(output_unit,'("")')
     write(output_unit,'("Instance of type: ")',advance="no")
     call this%write_type(output_unit)
     write(output_unit,fmt='("")')
     call this%print_to_unit(output_unit,parents,components,peers)
  end if
end subroutine serializable_print
```

serializable_print_all ↑

```
recursive subroutine serializable_print_all(this,unit)
  class(serializable_class),intent(in)::this
  integer,optional::unit
  if(present(unit))then
     write(unit,'("")')
     write(unit,'("Instance of type: ")',advance="no")
     call this%write_type(unit)
     write(unit,fmt='("")')
     call this%print_to_unit(unit,huge(one),huge(one),huge(one))
  else
     write(output_unit,'("")')
     write(output_unit,'("Instance of type: ")',advance="no")
     call this%write_type(output_unit)
     write(output_unit,fmt='("")')
     call this%print_to_unit(output_unit,huge(one),huge(one),huge(one))
  end if
end subroutine serializable_print_all
```

serializable_print_little ↑

```
recursive subroutine serializable_print_little(this,unit)
  class(serializable_class),intent(in)::this
  integer,optional::unit
  if(present(unit))then
     write(unit,'("")')
     write(unit,'("Instance of type: ")',advance="no")
     call this%write_type(unit)
     write(unit,fmt='("")')
     call this%print_to_unit(unit,zero,zero,zero)
  else
     write(output_unit,'("")')
```

```
      write(output_unit,'("Instance of type: ")',advance="no")
      call this%write_type(output_unit)
      write(output_unit,fmt='("")')
      call this%print_to_unit(output_unit,zero,zero,zero)
    end if
  end subroutine serializable_print_little
```

serializable_print_parents ↑

```
  recursive subroutine serializable_print_parents(this)
    class(serializable_class),intent(in)::this
    write(output_unit,'("")')
    write(output_unit,'("Instance of type: ")',advance="no")
      call this%write_type(output_unit)
      write(output_unit,fmt='("")')
    call this%print_to_unit(output_unit,huge(one),zero,zero)
  end subroutine serializable_print_parents
```

serializable_print_components ↑

```
  recursive subroutine serializable_print_components(this)
    class(serializable_class),intent(in)::this
    write(output_unit,'("")')
    write(output_unit,'("Instance of type: ")',advance="no")
      call this%write_type(output_unit)
      write(output_unit,fmt='("")')
    call this%print_to_unit(output_unit,zero,huge(one),zero)
  end subroutine serializable_print_components
```

serializable_print_peers ↑

```
  recursive subroutine serializable_print_peers(this)
    class(serializable_class),intent(in)::this
    write(output_unit,'("")')
    write(output_unit,'("Instance of type: ")',advance="no")
      call this%write_type(output_unit)
      write(output_unit,fmt='("")')
    call this%print_to_unit(output_unit,zero,zero,huge(one))
  end subroutine serializable_print_peers
```

serializable_print_error ↑

```
  recursive subroutine serializable_print_error(this)
    class(serializable_class),intent(in)::this
    call this%print_to_unit(error_unit,zero,zero,zero)
  end subroutine serializable_print_error
```

serializable_serialize_to_unit ↑

```
  subroutine serializable_serialize_to_unit(this,unit,name)
    class(serializable_class),intent(in)::this
    integer, intent(in) :: unit
    character (len=*), intent(in) :: name
```

```
    logical::opened
    character(32)::file
    !    gfortran bug
    !    character::stream
    character::write
    type(marker_type)::marker
    !    inquire(unit=unit,opened=opened,stream=stream,write=write)
    inquire(unit=unit,opened=opened,write=write)
    if(opened)then
!       if(stream=="Y")then
          if(write=="Y")then
             print *,"dummy: serializable_serialize_to_unit"
             stop
          else
             print *,"serializable_serialize_to_unit: cannot write to read-only unit."
          end if
!       else
!          print *,"serializable_serialize_to_unit: access kind of unit is not 'stream'."
!       end if
    else
       print *,"serializable_serialize_to_unit: file is not opened."
    end if
  end subroutine serializable_serialize_to_unit
```

serializable_serialize_to_file ↑

```
  subroutine serializable_serialize_to_file(this,name,file)
    class(serializable_class),intent(in)::this
    character (len=*), intent(in) :: file,name
    type(marker_type)::marker
    call marker%open_for_write_access(file)
    print *,"serializable_serialize_to_file: writing xml preamble to ",file
    call marker%activate_next_page()
    call marker%push('<?xml version="1.0"?>')
    call marker%mark_begin(tag="file",name=file)
    flush(marker%unit)
    call this%serialize_to_marker(marker,name)
    call marker%mark_end("file")
    call marker%close()
    call marker%finalize()
  end subroutine serializable_serialize_to_file
```

serializable_serialize_to_marker ↑

```
  recursive subroutine serializable_serialize_to_marker(this,marker,name)
    class(serializable_class),intent(in)::this
    class(marker_type),intent(inout)::marker
    character (len=*), intent(in) :: name
    if(marker%action==1)then
       call marker%mark_instance(this,name)
    else
       print *,"serializable_serialize_to_marker: Marker is not ready for write access. STOP
```

```
      stop
    end if
  end subroutine serializable_serialize_to_marker
```

serializable_deserialize_from_unit ↑

```
 subroutine serializable_deserialize_from_unit(this,unit,name)
    class(serializable_class),intent(inout)::this
    integer, intent(in) :: unit
    character (len=*), intent(in) :: name
    logical::opened
    !    gfortran bug
    !    character::stream
    character::read
    type(marker_type)::marker
    !    inquire(unit=unit,opened=opened,stream=stream,read=read)
    inquire(unit=unit,opened=opened,read=read)
    if(opened)then
!       if(stream=="Y")then
          if(read=="Y")then
             print *,"dummy: serializable_serialize_from_unit"
             stop
          else
             print *,"serializable_serialize_from_unit: cannot write from read-only unit."
          end if
!       else
!          print *,"serializable_serialize_from_unit: access kind of unit is not 'stream'."
!       end if
    else
       print *,"serializable_serialize_from_unit: file is not opened."
    end if
  end subroutine serializable_deserialize_from_unit
```

serializable_deserialize_from_marker ↑

```
  subroutine serializable_deserialize_from_marker(this,name,marker)
    class(serializable_class),intent(out)::this
    character(*),intent(in)::name
    class(marker_type),intent(inout)::marker
    integer(kind=dik)::status
    if(marker%action==2)then
       call marker%pick_instance(name,this,status)
    else
       print *,"serializable_deserialize_from_ring: Ring is not ready for read access. STOP."
       stop
    end if
  end subroutine serializable_deserialize_from_marker
```

serializable_deserialize_from_file ↑

```
  subroutine serializable_deserialize_from_file(this,name,file)
    class(serializable_class),intent(out)::this
```

```
    character(*),intent(in)::name,file
    type(marker_type)::marker
    integer(kind=dik),dimension(2)::p1,p2
    call marker%open_for_read_access(file,"</file>")
    marker%eof_int=huge(one)
    marker%eof_pos=page_ring_position(marker%eof_int)
    call marker%read_page()
    call marker%find('<?',skip=2,proceed=.true.,pos=p1)
    call marker%find('?>',skip=3,proceed=.false.,pos=p2)
    if((p1(2)<=0).or.(p2(2)<=0))then
       print *,"no version substring found."
    end if
    call marker%set_position(p2)
    call marker%find('<file ',skip=4,proceed=.true.,pos=p1)
    call marker%find('>',skip=1,proceed=.false.,pos=p2)
    if((p1(2)>0).and.(p2(2)>0))then
       call marker%push_position(p2)
       call marker%find('name="',skip=4,proceed=.true.,pos=p1)
       call marker%find('"',skip=1,proceed=.false.,pos=p2)
       call marker%pop_position()
    else
       print *,"no file header found. STOP."
       STOP
    end if
    call this%deserialize_from_marker(name,marker)
    call marker%close()
    call marker%finalize()
  end subroutine serializable_deserialize_from_file
```

13.6.2 Methoden für identified_type

Überschriebene serializable_class Methoden

identified_write_to_marker ↑

```
  subroutine identified_write_to_marker(this,marker,status)
    class(identified_type),intent(in)::this
    class(marker_type),intent(inout)::marker
    integer(kind=dik),intent(out)::status
    call marker%mark_begin("identified_type")
    call marker%mark("name",this%get_name())
    call marker%mark("id",this%get_id())
    call marker%mark_end("identified_type")
  end subroutine identified_write_to_marker
```

identified_read_from_marker ↑

```
  subroutine identified_read_from_marker(this,marker,status)
    class(identified_type),intent(out)::this
    class(marker_type),intent(inout)::marker
```

```
    integer(kind=dik),intent(out)::status
    character(:),allocatable::name
    call marker%pick_begin("identified_type",status=status)
    call marker%pick("name",name,status)
    call marker%pick("id",this%id,status)
    call marker%pick_end("identified_type",status=status)
    this%name=name
  end subroutine identified_read_from_marker
```

identified_print_to_unit ↑

```
  subroutine identified_print_to_unit(this,unit,parents,components,peers)
    class(identified_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    write(unit,'("Components of identified_type:")')
    write(unit,'("Name:              ",a)')this%get_name()
    write(unit,'("ID:                ",I10)')this%get_id()
  end subroutine identified_print_to_unit
```

identified_get_type ↑

```
  pure subroutine identified_get_type(type)
    character(:),allocatable,intent(out)::type
    allocate(type,source="identified_type")
  end subroutine identified_get_type
```

identified_verify_type ↑

```
  elemental logical function identified_verify_type(string)
    character(len=*),intent(in)::string
    identified_verify_type=(string=="identified_type")
  end function identified_verify_type
```

Originäre identified_type Methoden

identified_initialize ↑

```
  subroutine identified_initialize(this,id,name)
    class(identified_type),intent(out)::this
    integer(kind=dik),intent(in)::id
    character(len=*),intent(in)::name
    this%name=name
    this%id=id
  end subroutine identified_initialize
```

identified_get_id ↑

```
  elemental function identified_get_id(this) result(id)
    class(identified_type),intent(in)::this
    integer(kind=dik)::id
    id=this%id
  end function identified_get_id
```

identified_get_name ↑

```
pure function identified_get_name(this)
  class(identified_type),intent(in)::this
  character(len(this%name))::identified_get_name
  identified_get_name=char(this%name)
end function identified_get_name
```

13.6.3 Methoden für unique_type

Überschriebene serializable_class Methoden

unique_print_to_unit ↑

```
subroutine unique_print_to_unit(this,unit,parents,components,peers)
  class(unique_type),intent(in)::this
  integer,intent(in)::unit
  integer(kind=dik),intent(in)::parents,components,peers
  if(parents>0)call identified_print_to_unit(this,unit,parents-1,components&
       &,peers)
  write(unit,'("Unique ID:         ",I10)')this%get_unique_id()
end subroutine unique_print_to_unit
```

unique_get_type

```
pure subroutine unique_get_type(type)
  character(:),allocatable,intent(out)::type
  allocate(type,source="unique_type")
end subroutine unique_get_type
```

unique_verify_type

```
elemental logical function unique_verify_type(string)
  character(len=*),intent(in)::string
  unique_verify_type=(string=="unique_type")
end function unique_verify_type
```

unique_write_to_marker ↑

```
subroutine unique_write_to_marker(this,marker,status)
  class(unique_type),intent(in)::this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  call marker%mark_begin("unique_type")
  call identified_write_to_marker(this,marker,status)
  call marker%mark("unique_id",this%get_unique_id())
  call marker%mark_end("unique_type")
end subroutine unique_write_to_marker
```

unique_read_from_marker ↑

```
subroutine unique_read_from_marker(this,marker,status)
  class(unique_type),intent(out)::this
  class(marker_type),intent(inout)::marker
  integer(kind=dik),intent(out)::status
  call marker%pick_begin("unique_type",status=status)
  call identified_read_from_marker(this,marker,status)
  call marker%pick("unique_id",this%unique_id,status)
  call marker%pick_end("unique_type",status)
end subroutine unique_read_from_marker
```

Originäre unique_type Methoden

unique_initialize ↑

```
subroutine unique_initialize(this,id,name)
  class(unique_type),intent(out)::this
  integer(kind=dik),intent(in)::id
  character(len=*),intent(in)::name
  call identified_initialize(this,id,name)
  last_id=last_id+1
  this%unique_id=last_id
end subroutine unique_initialize
```

unique_get_unique_id ↑

```
elemental function unique_get_unique_id(this)
  class(unique_type),intent(in)::this
  integer(kind=dik)::unique_get_unique_id
  unique_get_unique_id=this%unique_id
end function unique_get_unique_id
```

13.6.4 Methoden für serializable_ref_type

serializable_ref_finalize ↑

```
subroutine serializable_ref_finalize(this)
  class(serializable_ref_type),intent(inout)::this
  class(serializable_ref_type),pointer::next
  do while (associated(this%next))
     next=>this%next
     this%next=>next%next
     nullify(next%ref)
     deallocate(next)
  end do
  if(associated(this%ref))nullify(this%ref)
end subroutine serializable_ref_finalize
```

13.6.5 Methoden für position_stack_type

position_stack_push_head ↑

```
subroutine position_stack_push_head(this)
  class(position_stack_type)::this
  class(position_stack_type),pointer::new
  allocate(new)
  new%next=>this%next
  new%position=this%position
  this%next=>new
end subroutine position_stack_push_head
```

position_stack_push_given ↑

```
subroutine position_stack_push_given(this,position)
  class(position_stack_type)::this
  integer(kind=dik),dimension(2),intent(in)::position
  class(position_stack_type),pointer::new
  allocate(new)
  new%next=>this%next
  new%position=position
  this%next=>new
end subroutine position_stack_push_given
```

position_stack_pop ↑

```
subroutine position_stack_pop(this)
  class(position_stack_type)::this
  class(position_stack_type),pointer::old
  if(associated(this%next))then
     old=>this%next
     this%next=>old%next
     this%position=old%position
     deallocate(old)
  end if
end subroutine position_stack_pop
```

position_stack_drop ↑

```
subroutine position_stack_drop(this,position)
  class(position_stack_type)::this
  integer(kind=dik),dimension(2),intent(out)::position
  class(position_stack_type),pointer::old
  if(associated(this%next))then
     old=>this%next
     this%next=>old%next
     position=old%position
     deallocate(old)
  else
     position=[0,0]
  end if
end subroutine position_stack_drop
```

position_stack_nth_position ↑

```
function position_stack_nth_position(this,n) result(position)
  class(position_stack_type),intent(in)::this
  integer(kind=dik),intent(in)::n
  integer(kind=dik),dimension(2)::position
  class(position_stack_type),pointer::tmp
  integer(kind=dik)::pos
  tmp=>this%next
  pos=n
  do while(associated(tmp).and.pos>0)
     tmp=>tmp%next
     pos=pos-1
  end do
  if(associated(tmp))then
     position=tmp%position
  else
     position=[0,0]
  end if
end function position_stack_nth_position
```

position_stack_first ↑

```
function position_stack_first(this) result(position)
  class(position_stack_type),intent(in)::this
  integer(kind=dik),dimension(2)::position,tmp_position
  class(position_stack_type),pointer::tmp_stack
  tmp_position=this%position
  tmp_stack=>this%next
  do while(associated(tmp_stack))
     if(page_ring_position_is_before(tmp_stack%position,tmp_position))then
        tmp_position=tmp_stack%position
     end if
     tmp_stack=>tmp_stack%next
  end do
end function position_stack_first
```

position_stack_last ↑

```
function position_stack_last(this) result(position)
  class(position_stack_type),intent(in)::this
  integer(kind=dik),dimension(2)::position,tmp_position
  class(position_stack_type),pointer::tmp_stack
  tmp_position=this%position
  tmp_stack=>this%next
  do while(associated(tmp_stack))
     if(page_ring_position_is_before(tmp_position,tmp_stack%position))then
        tmp_position=tmp_stack%position
     end if
     tmp_stack=>tmp_stack%next
  end do
end function position_stack_last
```

position_stack_range ↑

```
pure function position_stack_range(this) result(position)
  class(position_stack_type),intent(in)::this
  integer(kind=dik),dimension(2)::position
  class(position_stack_type),pointer::tmp
end function position_stack_range
```

13.6.6 Methoden für page_ring_type

page_ring_open_for_read_access ↑

```
subroutine page_ring_open_for_read_access(this,file,eof_string,asynchronous)
  class(page_ring_type),intent(inout)::this
  character(*),intent(in)::file,eof_string
  logical,intent(in),optional::asynchronous
  logical::exist
  this%eof_string=eof_string
  inquire(file=file,exist=exist)
  if(exist)then
     this%action=2
  else
     print *,"page_ring_open: File ",file," is opened for read access but &
     &does not exist. STOP."
     STOP
  end if

  if(present(asynchronous))this%asynchronous=asynchronous
  if(this%unit<0)call generate_unit(this%unit,100,1000)
  if(this%unit<0)then
     print *,"page_ring_open: No free unit found. STOP."
     STOP
  end if
  this%ring_size=2
  call this%set_position([zero,one])
  this%active_pages=[zero,-one]
  if(allocated(this%ring))deallocate(this%ring)
  allocate(this%ring(zero:this%ring_size-one))
  if(this%asynchronous)then
     open(this%unit,&
          file=file,&
          access="stream",&
          action="read",&
          asynchronous="yes",&
          status="old")
  else
     open(this%unit,&
          file=file,&
          access="stream",&
          action="read",&
```

```
          asynchronous="no",&
          status="old")
    end if
    call this%read_page()
  end subroutine page_ring_open_for_read_access
```

page_ring_open_for_write_access ↑

```
  subroutine page_ring_open_for_write_access(this,file,asynchronous)
    class(page_ring_type),intent(inout)::this
    character(*),intent(in)::file
    logical,intent(in),optional::asynchronous
    this%action=1

    if(present(asynchronous))this%asynchronous=asynchronous
    if(this%unit<0)call generate_unit(this%unit,100,1000)
    if(this%unit<0)then
       print *,"page_ring_open: No free unit found. STOP."
       STOP
    end if
    this%ring_size=2
    call this%set_position([zero,one])
    this%active_pages=[zero,-one]
    if(allocated(this%ring))deallocate(this%ring)
    allocate(this%ring(zero:this%ring_size-one))

    if(this%asynchronous)then
       open(this%unit,&
            file=file,&
            access="stream",&
            action="write",&
            asynchronous="yes",&
            status="replace")
    else
       open(this%unit,&
            file=file,&
            access="stream",&
            action="write",&
            asynchronous="no",&
            status="replace")
    end if
  end subroutine page_ring_open_for_write_access
```

page_ring_close ↑

```
  subroutine page_ring_close(this)
    class(page_ring_type),intent(inout)::this
    if(this%action==1)then
       call this%flush()
       !call this%print_position()
       if(this%asynchronous)then
          write(this%unit,asynchronous="yes")&
```

```
              &this%ring(this%actual_index())(:this%actual_offset()-1)
        else
           write(this%unit,asynchronous="no")&
              &this%ring(this%actual_index())(:this%actual_offset()-1)
        end if
     end if
     close(this%unit)
  end subroutine page_ring_close
```

page_ring_read_page ↑

```
  subroutine page_ring_read_page(this)
    class(page_ring_type),intent(inout)::this
    integer(kind=dik)::iostat
    character(8)::iomsg
    if(.not.this%eof_reached)then
       call page_ring_activate_next_page(this)
       read(this%unit,iostat=iostat)this%ring(this%last_index())
       if(iostat==iostat_end)then
          this%eof_reached=.true.
          this%eof_pos(1)=this%last_page()
          this%eof_pos(2)=index(this%ring(this%last_index()),char(this%eof_string))
          this%eof_pos(2)=this%eof_pos(2)+len(this%eof_string)-1
          this%eof_int=page_ring_ordinal(this%eof_pos)
       end if
    end if
  end subroutine page_ring_read_page
```

page_ring_enlarge ↑

```
  subroutine page_ring_enlarge(this)
    class(page_ring_type),intent(inout)::this
    character(serialize_page_size),dimension(:),allocatable::tmp_ring
    integer(kind=dik)::n
    call move_alloc(this%ring,tmp_ring)
    allocate(this%ring(0:this%ring_size*2-1))
    do n=this%active_pages(1),this%active_pages(2)
       this%ring(mod(n,this%ring_size*2))=tmp_ring(mod(n,this%ring_size))
    end do
    this%ring_size=this%ring_size*2
  end subroutine page_ring_enlarge
```

page_ring_print_to_unit ↑

```
  subroutine page_ring_print_to_unit(this,unit,parents,components,peers)
    class(page_ring_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik),intent(in)::parents,components,peers
    write(unit,'("Components of page_ring_type:")')
    print *,"asynchronous: ",this%asynchronous
    print *,"eof reached:  ",this%eof_reached
    print *,"ring_size:    ",this%ring_size
```

```
    print *,"unit:          ",this%unit
    print *,"action:        ",this%action
    print *,"position:      ",this%position_stack%position
    print *,"active_pages: ",this%active_pages
    print *,"file size:     ",this%eof_int
    print *,"eof position: ",this%eof_pos
    print *,"eof string:    ",char(this%eof_string)
    if(allocated(this%ring))then
       print *,"Ring is allocated."
       if(components>0)call this%print_ring(unit)
    else
       print *,"Ring is not allocated."
    end if
  end subroutine page_ring_print_to_unit
```

page_ring_print_ring ↑

```
  subroutine page_ring_print_ring(this,unit)
    class(page_ring_type),intent(in)::this
    integer,intent(in)::unit
    integer(kind=dik)::n
    write(unit,fmt=*)"Begin of page ring"
    do n=this%active_pages(1),this%active_pages(2)
       write(unit=unit,fmt="('(',I0,')',a)")n,this%ring(mod(n,this%ring_size))
    end do
    write(unit,fmt=*)"End of page ring"
  end subroutine page_ring_print_ring
```

page_ring_push_string ↑

```
  recursive subroutine page_ring_push_string(this,string)
    class(page_ring_type),intent(inout)::this
    character(*),intent(in)::string
    integer(kind=dik)::cut,l
    l=len(string)
    if(l<=serialize_page_size-this%actual_offset()+1)then
       this%ring(this%actual_index())(this%actual_offset():this%actual_offset()+l-1)=string
       if(l==serialize_page_size-this%actual_offset()+1)then
          call this%break()
          call this%flush()
       else
          call this%proceed(l)
       end if
    else
       cut=serialize_page_size-this%actual_offset()+1
       call this%push_string(string(:cut))
       call this%push_string(string(cut+1:))
    end if
  end subroutine page_ring_push_string
```

page_ring_push_integer_dik ↑

```
recursive subroutine page_ring_push_integer_dik(this,int)
  class(page_ring_type),intent(inout)::this
  integer(kind=dik),intent(in)::int
  integer(kind=dik)::int1
  if(int<0)then
     call this%push("-")
     call page_ring_push_integer_dik(this,-int)
  else
     if(int>9)call this%push(int/10)
     int1=mod(int,10*one)
     select case (int1)
     case (0)
        call this%push("0")
     case (1)
        call this%push("1")
     case (2)
        call this%push("2")
     case (3)
        call this%push("3")
     case (4)
        call this%push("4")
     case (5)
        call this%push("5")
     case (6)
        call this%push("6")
     case (7)
        call this%push("7")
     case (8)
        call this%push("8")
     case (9)
        call this%push("9")
     end select
  end if
end subroutine page_ring_push_integer_dik
```

page_ring_push_integer ↑

```
subroutine page_ring_push_integer(this,in)
  class(page_ring_type),intent(inout)::this
  integer,intent(in)::in
  call page_ring_push_integer_dik(this,int(in,kind=dik))
end subroutine page_ring_push_integer
```

page_ring_pop_integer ↑

```
subroutine page_ring_pop_integer(this,in)
  class(page_ring_type),intent(inout)::this
  integer,intent(out)::in
  integer(kind=dik)::in_dik
  call page_ring_pop_integer_dik(this,in_dik)
  in=int(in_dik)
end subroutine page_ring_pop_integer
```

page_ring_pop_integer_dik ↑

```
subroutine page_ring_pop_integer_dik(this,int)
  class(page_ring_type),intent(inout)::this
  integer(kind=dik),intent(out)::int
  integer(kind=dik)::int1
  integer(kind=dik)::sign
  character::c
  int=0
  sign=1
  c=" "
  do while(scan(c,serialize_integer_characters)==0)
     call this%pop_character(c)
  end do
  if(c=="-")then
     sign=-1
     call this%pop_character(c)
  end if
  do while(scan(c,serialize_integer_characters)>0)
     int=int*10
     select case (c)
     case ("1")
        int=int+1
     case ("2")
        int=int+2
     case ("3")
        int=int+3
     case ("4")
        int=int+4
     case ("5")
        int=int+5
     case ("6")
        int=int+6
     case ("7")
        int=int+7
     case ("8")
        int=int+8
     case ("9")
        int=int+9
     end select
     call this%pop_character(c)
  end do
  int=int*sign
  if(c=="<")call this%proceed(-one)
end subroutine page_ring_pop_integer_dik
```

page_ring_pop_logical ↑

```
subroutine page_ring_pop_logical(this,l)
  class(page_ring_type),intent(inout)::this
  logical,intent(out)::l
```

```
    character(1)::lc
    call this%pop(lc)
    do while(scan(lc,"tTfF")==0)
       call this%pop(lc)
    end do
    read(lc,fmt="(l1)")l
  end subroutine page_ring_pop_logical
```

page_ring_push_integer_array_dik ↑

```
  subroutine page_ring_push_integer_array_dik(this,int)
    class(page_ring_type),intent(inout)::this
    integer(kind=dik),dimension(:),intent(in)::int
    integer(kind=dik)::n
    do n=1,size(int)
       call this%push(int(n))
       call this%push(" ")
    end do
  end subroutine page_ring_push_integer_array_dik
```

page_ring_push_integer_array ↑

```
  subroutine page_ring_push_integer_array(this,int)
    class(page_ring_type),intent(inout)::this
    integer,dimension(:),intent(in)::int
    integer::n
    do n=1,size(int)
       call this%push(int(n))
       call this%push(" ")
    end do
  end subroutine page_ring_push_integer_array
```

page_ring_pop_integer_array ↑

```
  subroutine page_ring_pop_integer_array(this,int)
    class(page_ring_type),intent(inout)::this
    integer,dimension(:),intent(out)::int
    integer::n
    do n=1,size(int)
       call this%pop(int(n))
    end do
  end subroutine page_ring_pop_integer_array
```

page_ring_pop_integer_array_dik ↑

```
   subroutine page_ring_pop_integer_array_dik(this,int)
    class(page_ring_type),intent(inout)::this
    integer(kind=dik),dimension(:),intent(out)::int
    integer(kind=dik)::n
    do n=1,size(int)
       call this%pop(int(n))
    end do
  end subroutine page_ring_pop_integer_array_dik
```

page_ring_push_double ↑

```
  subroutine page_ring_push_double(this,dou)
    class(page_ring_type),intent(inout)::this
    real(kind=drk),intent(in)::dou
    integer(kind=dik)::f
!    print *,"page_ring_push_double: ",dou
    if(dou==0D0)then
       call this%push("0")
    else
       f=int(scale(fraction(dou),digits(dou)),kind=dik)
       call this%push(digits(dou))
       call this%push(":")
       call this%push(f)
       call this%push(":")
       call this%push(exponent(dou))
    end if
    call this%push(" ")
  end subroutine page_ring_push_double
```

page_ring_push_double_array ↑

```
  subroutine page_ring_push_double_array(this,dou)
    class(page_ring_type),intent(inout)::this
    real(kind=drk),dimension(:),intent(in)::dou
    integer(kind=dik)::n
    do n=1,size(dou)
       call this%push(dou(n))
    end do
  end subroutine page_ring_push_double_array
```

page_ring_pop_double ↑

```
  subroutine page_ring_pop_double(this,dou,skip)
    class(page_ring_type),intent(inout)::this
    real(kind=drk),intent(out)::dou
    logical,optional,intent(in)::skip
    integer(kind=dik)::d,f,e
    call this%pop(d)
    if(d==zero)then
       dou=0D0
    else
       call this%pop(f)
       call this%pop(e)
       dou=set_exponent(scale(real(f,kind=double),-d),e)
    end if
    if(present(skip))then
       if(.not.skip)call this%proceed(-one)
    end if
  end subroutine page_ring_pop_double
```

page_ring_pop_double_array ↑

```
subroutine page_ring_pop_double_array(this,dou,skip)
  class(page_ring_type),intent(inout)::this
  real(kind=drk),dimension(:),intent(out)::dou
  logical,optional,intent(in)::skip
  integer(kind=dik)::n
  call this%pop_double(dou(1))
  do n=2,size(dou)
     call this%pop_double(dou(n))
  end do
  if(present(skip))then
     if(.not.skip)call this%proceed(-one)
  end if
end subroutine page_ring_pop_double_array
```

page_ring_pop_character ↑

```
subroutine page_ring_pop_character(this,c)
  class(page_ring_type),intent(inout)::this
  character,intent(out)::c
  c=this%ring(this%actual_index())(this%actual_offset():this%actual_offset())
  if(this%actual_offset()==serialize_page_size)call this%read_page
  call this%proceed(one)
end subroutine page_ring_pop_character
```

page_ring_pop_string ↑

```
recursive subroutine page_ring_pop_string(this,res)
  class(page_ring_type),intent(inout)::this
  character(len=*),intent(out)::res
  integer(kind=dik)::n,cut
  n=len(res)
  cut=serialize_page_size-this%actual_offset()+1
  if(n<=cut)then
     res=this%ring(this%actual_index())(this%actual_offset():this%actual_offset()+n)
     if(n==cut)then
        call this%read_page
     end if
     call this%proceed(n)
  else
     call page_ring_pop_string(this,res(:cut))
     call page_ring_pop_string(this,res(cut+1:))
  end if
end subroutine page_ring_pop_string
```

page_ring_substring2 ↑

```
pure function page_ring_substring2(this,i1,i2) result(res)
  class(page_ring_type),intent(in)::this
  integer(kind=dik),dimension(2),intent(in)::i1,i2
  character(ring_position_metric2(i1,i2))::res
  integer(kind=dik)::page,pos
  if(i1(1)==i2(1))then
```

```
      res=this%ring(mod(i1(1),this%ring_size))(i1(2):i2(2))
    else
      pos=serialize_page_size-i1(2)
      res(1:pos+1)=this%ring(mod(i1(1),this%ring_size))(i1(2):)
      do page=i1(1)+1,i2(1)-1
        res(pos+2:pos+2+serialize_page_size)=this%ring(mod(page,this%ring_size))
        pos=pos+serialize_page_size
      end do
      res(pos+2:pos+1+i2(2))=this%ring(mod(page,this%ring_size))(1:i2(2))
    end if
  end function page_ring_substring2
```

page_ring_substring1 ↑

```
  pure function page_ring_substring1(this,i) result(res)
    class(page_ring_type),intent(in)::this
    integer(kind=dik),dimension(2,2),intent(in)::i
    character(ring_position_metric1(i))::res
    integer(kind=dik)::page,pos
    if(i(1,1)==i(1,2))then
      res=this%ring(mod(i(1,1),this%ring_size))(i(2,1):i(2,2))
    else
      pos=serialize_page_size-i(2,1)
      res(1:pos+1)=this%ring(mod(i(1,1),this%ring_size))(i(2,1):)
      do page=i(1,1)+1,i(1,1)-1
        res(pos+2:pos+2+serialize_page_size)=this%ring(mod(page,this%ring_size))
        pos=pos+serialize_page_size
      end do
      res(pos+2:pos+1+i(2,2))=this%ring(mod(page,this%ring_size))(1:i(2,2))
    end if
  end function page_ring_substring1
```

page_ring_allocate_substring ↑

```
  subroutine page_ring_allocate_substring(this,p1,p2,string)
    class(page_ring_type),intent(in)::this
    integer(kind=dik),dimension(2),intent(in)::p1,p2
    character(:),allocatable,intent(out)::string
    string=page_ring_substring2(this,p1,p2)
  end subroutine page_ring_allocate_substring
```

page_ring_find_default ↑

```
  subroutine page_ring_find_default(this,exp,skip,proceed,pos)
    class(page_ring_type),intent(inout)::this
    character(*),optional,intent(in)::exp
    integer,intent(in)::skip
    logical,intent(in)::proceed
    integer(kind=dik),dimension(2),intent(out)::pos
    call page_ring_find(this,exp,this%position_stack%position,this%eof_pos,skip,proceed,pos)
  end subroutine page_ring_find_default
```

page_ring_find ↑

```
recursive subroutine page_ring_find(this,exp,start,limit,skip,proceed,pos)
  class(page_ring_type),intent(inout)::this
  integer(kind=dik),dimension(2),intent(in)::start
  integer(kind=dik),dimension(2),intent(in)::limit
  character(*),intent(in)::exp
  integer,intent(in)::skip
  logical,intent(in)::proceed
  integer(kind=dik),dimension(2),intent(out)::pos
  integer(kind=dik)::page,page2,ind
  page=this%ring_index(start(1))
  if(limit(1)==start(1))then
     ind=index(this%ring(page)(start(2):limit(2)),exp)
     if(ind>0)then
        select case (skip)
        case(1)
           pos=[start(1),start(2)+ind-2]
           if(pos(2)==0)then
              pos(1)=pos(1)-1
              pos(2)=serialize_page_size
           end if
        case(2)
           pos=[start(1),start(2)+ind-1]
        case(3)
           pos=[start(1),start(2)+ind+len(exp)-2]
        case(4)
           pos=[start(1),start(2)+ind+len(exp)-1]
           if(pos(1)==this%last_page())call this%read_page()
           if(pos(2)>serialize_page_size)then
              pos(1)=pos(1)+1
              pos(2)=pos(2)-serialize_page_size
           end if
        end select
        if(proceed)call this%set_position(pos)
     else
        print *,"page_ring_find: limit reached."
        pos=[-1,-1]
     end if
  else
     ind=index(this%ring(page)(start(2):),exp)
     if(ind>0)then
        select case (skip)
        case(1)
           pos=[start(1),start(2)+ind-2]
           if(pos(2)==0)then
              pos(1)=pos(1)-1
              pos(2)=serialize_page_size
           end if
        case(2)
           pos=[start(1),start(2)+ind-1]
        case(3)
           pos=[start(1),start(2)+ind+len(exp)-2]
```

```
        case(4)
           pos=[start(1),start(2)+ind+len(exp)-1]
           if(pos(1)==this%last_page())call this%read_page()
           if(pos(2)>serialize_page_size)then
              pos(1)=pos(1)+1
              pos(2)=one
           end if
        end select
        if(proceed)call this%set_position(pos)
     else
        if(start(1)+1>this%active_pages(2))then
           call this%read_page()
           page=this%ring_index(start(1))
        end if
        page2=this%ring_index(start(1)+1)
        ind=index(this%ring(page)(serialize_page_size-len(exp)+1:)&
                  //this%ring(page2)(:len(exp)),exp)
        if(ind>0)then
           select case (skip)
           case(1)
              pos=[start(1),serialize_page_size-len(exp)+ind-1]
           case(2)
              pos=[start(1),serialize_page_size-len(exp)+ind]
           case(3)
              pos=[start(1)+1,ind-1]
           case(4)
              pos=[start(1)+1,ind]
           end select
           if(pos(2)>serialize_page_size)then
              pos(1)=pos(1)+1
              pos(2)=pos(2)-serialize_page_size
           else
              if(pos(2)<0)then
                 pos(1)=pos(1)-1
                 pos(2)=pos(2)+serialize_page_size
              end if
           end if
           if(proceed)call this%set_position(pos)
        else
           if(proceed)this%active_pages(1)=this%active_pages(2)
           call page_ring_find(this,exp,[start(1)+one,one],limit,skip,proceed,pos)
        end if
     end if
   end if
 end subroutine page_ring_find
```

page_ring_str_equal ↑

```
 pure logical function page_ring_str_equal(this,string,pos)
   class(page_ring_type),intent(in)::this
   character(*),intent(in)::string
```

```
    integer(kind=dik),dimension(2,2),intent(in)::pos
    page_ring_str_equal=string==this%substring(pos)
  end function page_ring_str_equal
```

page_ring_find_pure ↑

```
  pure recursive function page_ring_find_pure(this,exp,start,limit,skip) result(pos)
    class(page_ring_type),intent(in)::this
    integer(kind=dik),dimension(2),intent(in)::start
    integer(kind=dik),dimension(2),intent(in)::limit
    character(*),intent(in)::exp
    integer,optional,intent(in)::skip
    integer(kind=dik),dimension(2)::pos
    integer(kind=dik)::page,page2,ind,actual_skip
    ! Is the starting point before limit?
    if(start(1)<=limit(1))then
       ! Default skip is what you expect from the build-in index function
       if(present(skip))then
          actual_skip=skip
       else
          actual_skip=2
       end if
       page=mod(start(1),this%ring_size)
       ! Does the scanning region end on the page?
       if(start(1)==limit(1))then
          ind=index(this%ring(page)(start(2):limit(2)),exp)
       else
          ind=index(this%ring(page)(start(2):),exp)
       end if
       if(ind>0)then
          ! substring found on first page
          select case (actual_skip)
          case(1)
             pos=[start(1),start(2)+ind-2]
             if(pos(2)==0)then
                pos(1)=pos(1)-1
                pos(2)=serialize_page_size
             end if
          case(2)
             pos=[start(1),start(2)+ind-1]
          case(3)
             pos=[start(1),start(2)+ind+len(exp)-2]
          case(4)
             pos=[start(1),start(2)+ind+len(exp)-1]
             if(pos(2)>serialize_page_size)then
                pos(1)=pos(1)+1
                pos(2)=pos(2)-serialize_page_size
             end if
          end select
       else
          ! Substring not found on first page. Is the next page already read?
```

```
      if((start(1)>=limit(1)).or.(start(1)+1>this%active_pages(2)))then
         ! Either the limit is reached or the next page is not ready.
         pos=[0,0]
      else
         ! The next page is available.
         page2=mod(start(1)+1,this%ring_size)
         ! We concatenate the edges. When l is the length of exp, then we want to concat
         ! the l-1 last characters of page one and the first l characters of page two.
         ind=index(this%ring(page)(serialize_page_size-len(exp)+2:)&
                 //this%ring(page2)(:len(exp)),exp)
         if(ind>0)then
            select case (actual_skip)
            case(1)
               pos=[start(1),serialize_page_size-len(exp)+ind]
            case(2)
               pos=[start(1),serialize_page_size-len(exp)+ind+1]
            case(3)
               pos=[start(1)+1,ind]
            case(4)
               pos=[start(1)+1,ind+1]
            end select
         else
            ! EXP is not found in the overlap region. We recursively search the next pages.
            pos=page_ring_find_pure(this,exp,[start(one)+one,one],limit,skip)
         end if
      end if
   end if
else
   ! limit is before start
   pos=[0,0]
end if
end function page_ring_find_pure
```

page_ring_positions_by_keys ↑

```
pure recursive subroutine page_ring_positions_by_keys&
  (this,exp1,exp2,start,limit,inclusive,length,pos)
  class(page_ring_type),intent(in)::this
  character(*),intent(in)::exp1,exp2
  integer(kind=dik),dimension(2),intent(in)::start,limit
  logical,optional,intent(in)::inclusive
  integer(kind=dik),intent(out),optional::length
  integer(kind=dik),dimension(2,2),intent(out)::pos
  if(inclusive)then
     pos(1:2,1)=this%find_pure(exp1,start,limit,2)
  else
     pos(1:2,1)=this%find_pure(exp1,start,limit,4)
  end if
  !print *,pos1
  if(present(length))then
     length=0
```

```
    end if
    if(pos(2,1)>0)then
       if(inclusive)then
          pos(1:2,2)=this%find_pure(exp2,pos(1:2,1),limit,3)
       else
          pos(1:2,2)=this%find_pure(exp2,pos(1:2,1),limit,1)
       end if
       !print *,pos2
       if(pos(2,2)>0)then
          if(present(length))then
             length=ring_position_metric1(pos)
          end if
       end if
    end if
  end subroutine page_ring_positions_by_keys
```

page_ring_character_by_keys ↑

```
  pure recursive subroutine page_ring_character_by_keys&
    (this,exp1,exp2,start,limit,inclusive,length,string)
    class(page_ring_type),intent(in)::this
    character(*),intent(in)::exp1,exp2
    integer(kind=dik),dimension(2),intent(in)::start,limit
    logical,optional,intent(in)::inclusive
    integer(kind=dik),intent(out),optional::length
    character(:),allocatable,intent(out)::string
    integer(kind=dik),dimension(2,2)::pos
    call this%substring_by_keys(exp1,exp2,start,limit,inclusive,length,pos)
    string=this%substring(pos(:,1),pos(:,2))
  end subroutine page_ring_character_by_keys
```

page_ring_pop_by_keys ↑

```
  subroutine page_ring_pop_by_keys(this,start,stop,inclusive,res)
    class(page_ring_type),intent(inout)::this
    character(*),intent(in),optional::start
    character(*),intent(in)::stop
    logical,optional,intent(in)::inclusive
    character(len=*),intent(out)::res
    integer(kind=dik),dimension(2)::i1,i2
    if(inclusive)then
       call this%find(start,2,.true.,i1)
       call this%find(stop,3,.false.,i2)
    else
       call this%find(start,4,.true.,i1)
       call this%find(stop,1,.false.,i2)
    end if
    res=this%substring(i1,i2)
    call this%set_position(i2)
  end subroutine page_ring_pop_by_keys
```

page_ring_get_character ↑

```
elemental function  page_ring_get_character(this)
  class(page_ring_type),intent(in)::this
  character::page_ring_get_character
  page_ring_get_character=&
   this%ring(this%actual_index())(this%actual_offset():this%actual_offset())
end function page_ring_get_character
```

page_ring_break ↑

```
subroutine page_ring_break(this)
  class(page_ring_type),intent(inout)::this
  if(this%actual_page()>=this%active_pages(2))call this%activate_next_page()
  call this%turn_page()
end subroutine page_ring_break
```

page_ring_turn_page ↑

```
subroutine page_ring_turn_page(this)
  class(page_ring_type),intent(inout)::this
  this%position_stack%position(1)=this%position_stack%position(1)+1
  this%position_stack%position(2)=1
end subroutine page_ring_turn_page
```

page_ring_flush ↑

```
subroutine page_ring_flush(this)
  class(page_ring_type),intent(inout)::this
  integer(kind=dik)::page
  do while(this%active_pages(1)<this%actual_page())
     if(this%asynchronous)then
        write(this%unit,asynchronous="yes")this%ring(mod(this%active_pages(1),this%ring_size))
     else
        write(this%unit,asynchronous="no")this%ring(mod(this%active_pages(1),this%ring_size))
     end if
     this%active_pages(1)=this%active_pages(1)+1
  end do
end subroutine page_ring_flush
```

page_ring_activate_next_page ↑

```
subroutine page_ring_activate_next_page(this)
  class(page_ring_type),intent(inout)::this
  if(this%active_pages(2)-this%active_pages(1)+1>=this%ring_size)call this%enlarge
  this%active_pages(2)=this%active_pages(2)+1
end subroutine page_ring_activate_next_page
```

page_ring_set_position ↑

```
subroutine page_ring_set_position(this,pos)
  class(page_ring_type),intent(inout)::this
  integer(kind=dik),dimension(2),intent(in)::pos
  this%position_stack%position=pos
end subroutine page_ring_set_position
```

page_ring_put ↑

```
subroutine page_ring_put(this)
  class(page_ring_type),intent(inout)::this
end subroutine page_ring_put
```

page_ring_proceed ↑

```
subroutine page_ring_proceed(this,n,deactivate)
  class(page_ring_type),intent(inout)::this
  integer(kind=dik),intent(in)::n
  logical,intent(in),optional::deactivate
  integer(kind=dik)::offset
  offset=this%position_stack%position(2)+n
  do while (offset>serialize_page_size)
     if(this%position_stack%position(1)&
        >=this%active_pages(2))call this%activate_next_page()
     this%position_stack%position(1)=this%position_stack%position(1)+1
     offset=offset-serialize_page_size
  end do
  this%position_stack%position(2)=offset
  if(present(deactivate))then
     if(deactivate)this%active_pages(1)=this%actual_page()
  end if
end subroutine page_ring_proceed
```

page_ring_print_position ↑

```
subroutine page_ring_print_position(this)
  class(page_ring_type),intent(inout)::this
  print *,&
       this%actual_position(),&
       this%ring(this%actual_index())(:this%actual_offset()-1),&
       "|",&
       this%ring(this%actual_index())(this%actual_offset():)
end subroutine page_ring_print_position
```

page_ring_ring_index ↑

```
elemental integer(kind=dik) function page_ring_ring_index(this,n)
  class(page_ring_type),intent(in)::this
  integer(kind=dik),intent(in)::n
  page_ring_ring_index=mod(n,this%ring_size)
end function page_ring_ring_index
```

page_ring_ring_push_given_position ↑

```
subroutine page_ring_ring_push_given_position(this,pos)
  class(page_ring_type),intent(inout)::this
  integer(kind=dik),dimension(2),intent(in)::pos
  call this%position_stack%push(pos)
end subroutine page_ring_ring_push_given_position
```

page_ring_ring_pop_actual_position ↑

```
subroutine page_ring_ring_pop_actual_position(this)
  class(page_ring_type),intent(inout)::this
  call this%position_stack%pop()
end subroutine page_ring_ring_pop_actual_position
```

page_ring_ring_push_actual_position ↑

```
subroutine page_ring_ring_push_actual_position(this)
  class(page_ring_type),intent(inout)::this
  call this%position_stack%push()
end subroutine page_ring_ring_push_actual_position
```

page_ring_ring_pop_given_position ↑

```
subroutine page_ring_ring_pop_given_position(this,pos)
  class(page_ring_type),intent(inout)::this
  integer(kind=dik),dimension(2),intent(out)::pos
  call this%position_stack%pop(pos)
end subroutine page_ring_ring_pop_given_position
```

page_ring_get_position1 ↑

```
pure subroutine page_ring_get_position1(this,pos)
  class(page_ring_type),intent(in)::this
  integer(kind=dik),intent(out)::pos
  pos=page_ring_ordinal(this%position_stack%position)
end subroutine page_ring_get_position1
```

page_ring_get_position2 ↑

```
pure subroutine page_ring_get_position2(this,pos)
  class(page_ring_type),intent(in)::this
  integer(kind=dik),dimension(2),intent(out)::pos
  pos=this%position_stack%position
end subroutine page_ring_get_position2
```

page_ring_actual_index ↑

```
elemental integer(kind=dik) function page_ring_actual_index(this)
  class(page_ring_type),intent(in)::this
  page_ring_actual_index=mod(this%position_stack%position(1),this%ring_size)
end function page_ring_actual_index
```

page_ring_actual_page ↑

```
elemental integer(kind=dik) function page_ring_actual_page(this)
  class(page_ring_type),intent(in)::this
  page_ring_actual_page=this%position_stack%position(1)
end function page_ring_actual_page
```

page_ring_actual_offset ↑

```
  elemental integer(kind=dik) function page_ring_actual_offset(this)
    class(page_ring_type),intent(in)::this
    page_ring_actual_offset=this%position_stack%position(2)
  end function page_ring_actual_offset
```

page_ring_actual_position ↑

```
  pure function page_ring_actual_position(this)
    class(page_ring_type),intent(in)::this
    integer(kind=dik),dimension(2)::page_ring_actual_position
    page_ring_actual_position=this%position_stack%position
  end function page_ring_actual_position
```

page_ring_first_index ↑

```
  elemental integer(kind=dik) function page_ring_first_index(this)
    class(page_ring_type),intent(in)::this
    page_ring_first_index=mod(this%active_pages(1),this%ring_size)
  end function page_ring_first_index
```

page_ring_first_page ↑

```
  elemental integer(kind=dik) function page_ring_first_page(this)
    class(page_ring_type),intent(in)::this
    page_ring_first_page=this%active_pages(1)
  end function page_ring_first_page
```

page_ring_last_index ↑

```
  elemental integer(kind=dik) function page_ring_last_index(this)
    class(page_ring_type),intent(in)::this
    page_ring_last_index=mod(this%active_pages(2),this%ring_size)
  end function page_ring_last_index
```

page_ring_last_page ↑

```
  elemental integer(kind=dik) function page_ring_last_page(this)
    class(page_ring_type),intent(in)::this
    page_ring_last_page=this%active_pages(2)
  end function page_ring_last_page
```

13.6.7 Methoden für marker_type

marker_mark_begin ↑

```
  subroutine marker_mark_begin(this,tag,type,name,target,pointer,shape)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::tag
    character(*),intent(in),optional::type,name
    integer(kind=dik),intent(in),optional::target,pointer
    integer,intent(in),dimension(:),optional::shape
    call this%indent()
    call this%push("<")
```

```
    call this%push(tag)
    if(present(type))call this%push(' type="'//type//'"')
    if(present(name))call this%push(' name="'//name//'"')
    if(present(target))then
       call this%push(' target="')
       call this%push(target)
       call this%push('"')
    end if
    if(present(pointer))then
       call this%push(' pointer="')
       call this%push(pointer)
       call this%push('"')
    end if
    if(present(shape))then
       call this%push(' shape="')
       call this%push(shape)
       call this%push('"')
    end if
    call this%push(">")
    this%indentation=this%indentation+1
  end subroutine marker_mark_begin
```

marker_mark_instance_begin ↑

```
  subroutine marker_mark_instance_begin(this,ser,name,target,pointer,shape)
    class(marker_type),intent(inout)::this
    class(serializable_class),intent(in)::ser
    character(*),intent(in)::name
    integer(kind=dik),intent(in),optional::target,pointer
    integer,intent(in),dimension(:),optional::shape
    character(:),allocatable::this_type
    call ser%get_type(this_type)
    call this%mark_begin("ser",this_type,name,target,pointer,shape)
  end subroutine marker_mark_instance_begin
```

marker_mark_end ↑

```
  subroutine marker_mark_end(this,tag)
    class(marker_type),intent(inout)::this
    character(*),intent(in),optional::tag
    this%indentation=this%indentation-1
    call this%indent()
    if(present(tag))then
       call this%push("</"//tag//">")
    else
       call this%push("</ser>")
    end if
  end subroutine marker_mark_end
```

marker_mark_instance_end ↑

```
subroutine marker_mark_instance_end(this)
  class(marker_type),intent(inout)::this
  call this%mark_end("ser")
end subroutine marker_mark_instance_end
```

marker_mark_logical ↑

```
subroutine marker_mark_logical(this,name,content)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::name
  logical,intent(in)::content
  call this%indent()
  call this%push("<"//name//">")
  if(content)then
     call this%push("T")
  else
     call this%push("F")
  end if
  call this%push("</"//name//">")
end subroutine marker_mark_logical
```

marker_mark_integer ↑

```
subroutine marker_mark_integer(this,name,content)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::name
  integer,intent(in)::content
  call this%indent()
  call this%push("<"//name//">")
  call this%push(content)
  call this%push("</"//name//">")
end subroutine marker_mark_integer
```

marker_mark_integer_array ↑

```
subroutine marker_mark_integer_array(this,name,content)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::name
  integer,dimension(:),intent(in)::content
  call this%indent()
  call this%push("<"//name//">")
  call this%push(content)
  call this%push("</"//name//">")
end subroutine marker_mark_integer_array
```

marker_mark_integer_matrix ↑

```
subroutine marker_mark_integer_matrix(this,name,content)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::name
  integer,dimension(:,:),intent(in)::content
  integer::n
  integer,dimension(2)::s
```

```
    s=shape(content)
    call this%indent()
    call this%push("<"//name//">")
    do n=1,s(2)
       call this%push(content(:,n))
       call this%push(" ")
    end do
    call this%push("</"//name//">")
  end subroutine marker_mark_integer_matrix
```

marker_mark_integer_dik ↑

```
  subroutine marker_mark_integer_dik(this,name,content)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    integer(kind=dik),intent(in)::content
    call this%indent()
    call this%push("<"//name//">")
    call this%push(content)
    call this%push("</"//name//">")
  end subroutine marker_mark_integer_dik
```

marker_mark_integer_array_dik ↑

```
  subroutine marker_mark_integer_array_dik(this,name,content)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    integer(kind=dik),dimension(:),intent(in)::content
    call this%indent()
    call this%push("<"//name//">")
    call this%push(content)
    call this%push("</"//name//">")
  end subroutine marker_mark_integer_array_dik
```

marker_mark_integer_matrix_dik ↑

```
  subroutine marker_mark_integer_matrix_dik(this,name,content)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    integer(kind=dik),dimension(:,:),intent(in)::content
    integer::n
    integer,dimension(2)::s
    call this%indent()
    call this%push("<"//name//">")
    do n=1,s(2)
       call this%push(content(:,n))
       call this%push(" ")
    end do
    call this%push("</"//name//">")
  end subroutine marker_mark_integer_matrix_dik
```

marker_mark_double ↑

```
subroutine marker_mark_double(this,name,content)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::name
  real(kind=drk),intent(in)::content
  call this%indent()
  call this%push("<"//name//">")
  call this%push(content)
  call this%push("</"//name//">")
end subroutine marker_mark_double
```

marker_mark_double_array ↑

```
subroutine marker_mark_double_array(this,name,content)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::name
  real(kind=drk),dimension(:),intent(in)::content
  call this%indent()
  call this%push("<"//name//">")
  call this%push(content)
  call this%push("</"//name//">")
end subroutine marker_mark_double_array
```

marker_mark_double_matrix ↑

```
subroutine marker_mark_double_matrix(this,name,content)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::name
  real(kind=drk),dimension(:,:),intent(in)::content
  integer::n
  integer,dimension(2)::s
  s=shape(content)
  call this%indent()
  call this%push("<"//name//">")
  do n=1,s(2)
     call this%push(content(:,n))
     call this%push(" ")
  end do
  call this%push("</"//name//">")
end subroutine marker_mark_double_matrix
```

marker_mark_string ↑

```
subroutine marker_mark_string(this,name,content)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::name,content
  call this%indent()
  call this%push("<"//name//">")
  call this%push(content)
  call this%push("</"//name//">")
end subroutine marker_mark_string
```

marker_mark_instance ↑

```
recursive subroutine marker_mark_instance(this,ser,name,target,pointer)
  class(marker_type),intent(inout)::this
  class(serializable_class),intent(in)::ser
  character (len=*), intent(in)::name
  integer(kind=dik),intent(in),optional::target,pointer
  integer(kind=dik)::status
  call this%mark_instance_begin(ser,name,target,pointer)
  call ser%write_to_marker(this,status)
  call this%mark_end("ser")
end subroutine marker_mark_instance
```

marker_mark_target ↑

```
recursive subroutine marker_mark_target(this,name,ser)
  class(marker_type),intent(inout)::this
  class(serializable_class),target,intent(in)::ser
  character (len=*), intent(in)::name
  this%n_instances=this%n_instances+1
  call this%push_heap(ser,this%n_instances)
  call this%mark_instance(ser,name,target=this%n_instances)
end subroutine marker_mark_target
```

marker_mark_allocatable ↑

```
subroutine marker_mark_allocatable(this,name,ser)
  class(marker_type),intent(inout)::this
  class(serializable_class),allocatable,intent(in)::ser
  character (len=*), intent(in)::name
  if(allocated(ser))then
     call this%mark_instance(ser,name)
  else
     call this%mark_null(name)
  end if
end subroutine marker_mark_allocatable
```

marker_mark_pointer ↑

```
recursive subroutine marker_mark_pointer(this,name,ser)
  class(marker_type),intent(inout)::this
  class(serializable_class),pointer,intent(in)::ser
  character(len=*),intent(in)::name
  character(:),allocatable::type
  integer(kind=dik)::p
  if(associated(ser))then
     call this%search_heap(ser,p)
     if(p>0)then
        call ser%get_type(type)
        call this%push('<ser type="')
        call this%push(type)
        call this%push('" name="')
        call this%push(name)
        call this%push('" pointer="')
```

```
          call this%push(p)
          call this%push('"/>')
       else
          call this%mark_target(name,ser)
       end if
    else
       call this%mark_null(name)
    end if
  end subroutine marker_mark_pointer
```

marker_mark_null ↑

```
  subroutine marker_mark_null(this,name)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    call this%indent()
    call this%push('<ser type="null" name="')
    call this%push(name)
    call this%push('"/>')
  end subroutine marker_mark_null
```

marker_mark_nothing ↑

```
  subroutine marker_mark_nothing(this,name)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    call this%indent()
    call this%push('<')
    call this%push(name)
    call this%push('/>')
  end subroutine marker_mark_nothing
```

marker_mark_empty ↑

```
  subroutine marker_mark_empty(this,tag,type,name,target,pointer,shape)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::tag
    character(*),intent(in),optional::type,name
    integer(kind=dik),intent(in),optional::target,pointer
    integer,intent(in),dimension(:),optional::shape
    call this%push("<")
    call this%push(tag)
    if(present(type))call this%push(' type="'//type//'"')
    if(present(name))call this%push(' name="'//name//'"')
    if(present(target))then
       call this%push(' target="')
       call this%push(target)
       call this%push('"')
    end if
    if(present(pointer))then
       call this%push(' pointer="')
       call this%push(pointer)
```

```
      call this%push('"')
   end if
   if(present(shape))then
      call this%push(' shape="')
      call this%push(shape)
      call this%push('"')
   end if
   call this%push("/>")
 end subroutine marker_mark_empty
```

marker_pick_begin ↑

```
 subroutine marker_pick_begin(this,tag,type,name,target,pointer,shape,status)
   class(marker_type),intent(inout)::this
   character(*),intent(in)::tag
   integer(kind=dik),dimension(2,2),intent(out),optional::type,name
   integer(kind=dik),intent(out),optional::target,pointer
   integer,dimension(:),allocatable,optional,intent(out)::shape
   integer(kind=dik),intent(out)::status
   integer(kind=dik),dimension(2)::p1,p2,p3
   integer(kind=dik)::l
   call this%find("<",skip=4,proceed=.true.,pos=p1)
   call this%find(">",skip=1,proceed=.false.,pos=p2)
   p3=this%find_pure(" ",p1,p2,skip=1)
   if(p3(2)>0)then
      if(this%substring(p1,p3)==tag)then
         status=serialize_ok
         if(present(type))then
            call this%substring_by_keys('type="','"',p3,p2,.false.,l,type)
            if(l<=0)then
               print *,"marker_pick_begin: No type found"
               status=serialize_wrong_type
            end if
         end if
         if(present(name))then
            call this%substring_by_keys('name="','"',p3,p2,.false.,l,name)
            if(l<=0)then
               print *,"marker_pick_begin: No name found"
               status=serialize_wrong_name
               call this%print_position()
               stop
            end if
         end if
         if(present(target))then
            p1=this%find_pure('target="',p3,p2,4)
            if(p1(2)>0)then
               call this%set_position(p1)
               call this%pop(target)
            else
               target=-1
               status=serialize_ok
```

```
          end if
       end if
       if(present(pointer))then
          p1=this%find_pure('pointer="',p3,p2,4)
          if(p1(2)>0)then
             call this%set_position(p1)
             call this%pop(pointer)
          else
             pointer=-1
             status=serialize_ok
          end if
       end if
       if(present(shape))then
          p1=this%find_pure('shape="',p3,p2,4)
          if(p1(2)>0)then
             call this%set_position(p1)
             call this%pop(shape)
          else
             status=serialize_ok
          end if
       end if
    else
       print *,"marker_pick_begin: Wrong tag. Expected: "&
       ,tag," Found: ",this%substring(p1,p3)
       status=serialize_wrong_tag
       call this%print_position()
    end if
  else
    if(this%substring(p1,p2)==tag)then
       status=serialize_ok
    else
       print *,"marker_pick_begin: Wrong tag. Expected: "&
       ,tag," Found: ",this%substring(p1,p2)
       status=serialize_wrong_tag
    end if
  end if
  call this%set_position(p2)
  call this%proceed(one*2,.true.)
end subroutine marker_pick_begin
```

marker_query_instance_begin ↑

```
subroutine marker_query_instance_begin(this,type,name,target,pointer,shape,status)
  class(marker_type),intent(inout)::this
  integer(kind=dik),dimension(2,2),intent(out),optional::type,name
  integer(kind=dik),intent(out),optional::target,pointer
  integer,dimension(:),allocatable,optional,intent(out)::shape
  integer(kind=dik),intent(out)::status
  call this%pick_begin("ser",type,name,target,pointer,shape,status)
end subroutine marker_query_instance_begin
```

marker_pick_instance_begin ↑

```
subroutine marker_pick_instance_begin(this,name,type,target,pointer,shape,status)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::name
  integer(kind=dik),dimension(2,2),intent(out),optional::type
  integer(kind=dik),intent(out),optional::target,pointer
  integer,dimension(:),allocatable,optional,intent(out)::shape
  integer(kind=dik),intent(out)::status
  integer(kind=dik),dimension(2,2)::read_name
  call this%query_instance_begin(type,read_name,target,pointer,shape,status)
  if(status==serialize_ok)then
     if(.not.this%str_equal(name,read_name))status=serialize_wrong_name
  end if
end subroutine marker_pick_instance_begin
```

marker_pick_end ↑

```
subroutine marker_pick_end(this,tag,status)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::tag
  integer(kind=dik),intent(out)::status
  integer(kind=dik),dimension(2)::p1,p2
  call this%find("</",skip=4,proceed=.true.,pos=p1)
  call this%find(">",skip=1,proceed=.false.,pos=p2)
  if(tag==this%substring(p1,p2))then
     status=serialize_ok
  else
     print *,"marker_pick_end: Wrong tag. Expected: ",tag," Found: ",this%substring(p1,p2)
     print *,"p1=",p1,"p2=",p2
     call this%print_position()
  end if
  call this%set_position(p2)
  call this%proceed(one*2,.true.)
end subroutine marker_pick_end
```

marker_pick_instance_end ↑

```
subroutine marker_pick_instance_end(this,status)
  class(marker_type),intent(inout)::this
  integer(kind=dik),intent(out)::status
  call this%pick_end("ser",status)
end subroutine marker_pick_instance_end
```

marker_pick_instance ↑

```
subroutine marker_pick_instance(this,name,ser,status)
  class(marker_type),intent(inout)::this
  class(serializable_class),intent(out)::ser
  character(*),intent(in)::name
  integer(kind=dik),intent(out)::status
  integer(kind=dik),dimension(2,2)::type,r_name
  call this%pick_begin("ser",type,r_name,status=status)
  if(status==serialize_ok)then
```

```
      if(ser%verify_type(this%substring(type)))then
         if(this%str_equal(name,r_name))then
            call ser%read_from_marker(this,status)
            call this%pick_end("ser",status)
         else
            print *,"marker_pick_instance: Name mismatch: Expected: "&
            ,name," Found: ",r_name
            status=serialize_wrong_name
            call this%print_position
         end if
      else
         print *,"marker_pick_instance: Type mismatch: ",type
         call ser%write_type(output_unit)
         print *,""
         status=serialize_wrong_type
         call this%print_position
      end if
   end if
 end subroutine marker_pick_instance
```

marker_pick_target ↑

```
 subroutine marker_pick_target(this,name,ser,status)
   class(marker_type),intent(inout)::this
   class(serializable_class),target,intent(out)::ser
   character(*),intent(in)::name
   integer(kind=dik),intent(out)::status
   integer(kind=dik),dimension(2,2)::type,r_name
   integer(kind=dik)::target
   call this%pick_begin("ser",type,r_name,target,status=status)
   if(status==serialize_ok)then
      if(ser%verify_type(this%substring(type)))then
         if(this%str_equal(name,r_name))then
            call ser%read_target_from_marker(this,status)
            if(target>0)call this%push_heap(ser,target)
         else
            print *,"marker_pick_instance: Name mismatch: Expected: "&
            ,name," Found: ",r_name
            status=serialize_wrong_name
         end if
      else
         print *,"marker_pick_instance: Type mismatch: ",type
         status=serialize_wrong_type
      end if
   end if
   call this%pick_end("ser",status)
 end subroutine marker_pick_target
```

marker_pick_allocatable ↑

```
 subroutine marker_pick_allocatable(this,name,ser)
   class(marker_type),intent(inout)::this
```

```
    character(*),intent(in)::name
    class(serializable_class),allocatable,intent(out)::ser
    class(serializable_class),pointer::ref
    integer(kind=dik),dimension(2,2)::type,r_name
    integer(kind=dik)::status
    call this%pick_begin("ser",type,r_name,status=status)
    if(status==serialize_ok)then
       if(ser%verify_type(this%substring(type)))then
          if(this%str_equal(name,r_name))then
             call this%search_reference(type,ref)
             if(associated(ref))then
                allocate(ser,source=ref)
                call ser%read_from_marker(this,status)
             else
                print *,"marker_pick_allocatable:&
                     & Type ",type," not found on reference stack."
             end if
          else
             print *,"marker_pick_instance: Name mismatch: Expected: ",&
             name," Found: ",r_name
             status=serialize_wrong_name
          end if
       else
          print *,"marker_pick_instance: Type mismatch: ",type
          status=serialize_wrong_type
       end if
    end if
    call this%pick_end("ser",status)
  end subroutine marker_pick_allocatable
```

marker_pick_pointer ↑

```
  recursive subroutine marker_pick_pointer(this,name,ser)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    class(serializable_class),pointer,intent(out)::ser
    class(serializable_class),pointer::ref
    integer(kind=dik),dimension(2,2)::type,r_name
    integer(kind=dik)::status,t,p
    nullify(ser)
    call this%pick_begin("ser",type,r_name,target=t,pointer=p,status=status)
    if(status==serialize_ok)then
       if(.not.this%str_equal("null",type))then
          if(p>0)then
             call this%search_heap(p,ser)
          else
             call this%search_reference(type,ref)
             if(associated(ref))then
                allocate(ser,source=ref)
                call ser%read_target_from_marker(this,status)
                call this%pick_end("ser",status)
```

```
                if(t>0)call this%push_heap(ser,t)
             else
                print *,"marker_pick_pointer:&
                     & Type ",type," not found on reference stack."
             end if
          end if
       end if
    end if
  end subroutine marker_pick_pointer
```

marker_pick_logical ↑

```
  subroutine marker_pick_logical(this,name,content,status)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    logical,intent(out)::content
    integer(kind=dik),intent(out)::status
    call this%pick_begin(name,status=status)
    if(status==serialize_ok)then
       call this%pop(content)
       call this%pick_end(name,status)
    end if
  end subroutine marker_pick_logical
```

marker_pick_integer ↑

```
  subroutine marker_pick_integer(this,name,content,status)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    integer,intent(out)::content
    integer(kind=dik),intent(out)::status
    call this%pick_begin(name,status=status)
    if(status==serialize_ok)then
       call this%pop(content)
       call this%pick_end(name,status)
    end if
  end subroutine marker_pick_integer
```

marker_pick_integer_array ↑

```
  subroutine marker_pick_integer_array(this,name,content,status)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    integer,dimension(:),intent(out)::content
    integer(kind=dik),intent(out)::status
    call this%pick_begin(name,status=status)
    if(status==serialize_ok)then
       call this%pop(content)
       call this%pick_end(name,status)
    end if
  end subroutine marker_pick_integer_array
```

marker_pick_integer_matrix ↑

```
subroutine marker_pick_integer_matrix(this,name,content,status)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::name
  integer,dimension(:,:),intent(out)::content
  integer(kind=dik),intent(out)::status
  integer::n
  integer,dimension(2)::s
  s=shape(content)
  call this%pick_begin(name,status=status)
  if(status==serialize_ok)then
     do n=1,s(2)
        call this%pop(content(:,n))
     end do
     call this%pick_end(name,status)
  end if
end subroutine marker_pick_integer_matrix
```

marker_pick_integer_dik ↑

```
subroutine marker_pick_integer_dik(this,name,content,status)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::name
  integer(kind=dik),intent(out)::content
  integer(kind=dik),intent(out)::status
  call this%pick_begin(name,status=status)
  if(status==serialize_ok)then
     call this%pop(content)
     call this%pick_end(name,status)
  end if
end subroutine marker_pick_integer_dik
```

marker_pick_integer_array_dik ↑

```
subroutine marker_pick_integer_array_dik(this,name,content,status)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::name
  integer(kind=dik),dimension(:),intent(out)::content
  integer(kind=dik),intent(out)::status
  call this%pick_begin(name,status=status)
  if(status==serialize_ok)then
     call this%pop(content)
     call this%pick_end(name,status)
  end if
end subroutine marker_pick_integer_array_dik
```

marker_pick_integer_matrix_dik ↑

```
subroutine marker_pick_integer_matrix_dik(this,name,content,status)
  class(marker_type),intent(inout)::this
  character(*),intent(in)::name
  integer(kind=dik),dimension(:,:),intent(out)::content
  integer(kind=dik),intent(out)::status
```

```
    integer::n
    integer,dimension(2)::s
    s=shape(content)
    call this%pick_begin(name,status=status)
    if(status==serialize_ok)then
       do n=1,s(2)
          call this%pop(content(:,n))
       end do
       call this%pick_end(name,status)
    end if
  end subroutine marker_pick_integer_matrix_dik
```

marker_pick_double ↑

```
  subroutine marker_pick_double(this,name,content,status)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    real(kind=drk),intent(out)::content
    integer(kind=dik),intent(out)::status
    call this%pick_begin(name,status=status)
    if(status==serialize_ok)then
       call this%pop(content)
       call this%pick_end(name,status)
    end if
  end subroutine marker_pick_double
```

marker_pick_double_array ↑

```
  subroutine marker_pick_double_array(this,name,content,status)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    real(kind=drk),dimension(:),intent(out)::content
    integer(kind=dik),intent(out)::status
    call this%pick_begin(name,status=status)
    if(status==serialize_ok)then
       call this%pop(content)
       call this%pick_end(name,status)
    end if
  end subroutine marker_pick_double_array
```

marker_pick_double_matrix ↑

```
  subroutine marker_pick_double_matrix(this,name,content,status)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    real(kind=drk),dimension(:,:),intent(out)::content
    integer(kind=dik),intent(out)::status
    integer::n
    integer,dimension(2)::s
    s=shape(content)
    call this%pick_begin(name,status=status)
    if(status==serialize_ok)then
```

```
      do n=1,s(2)
         call this%pop(content(:,n))
      end do
      call this%pick_end(name,status)
    end if
  end subroutine marker_pick_double_matrix
```

marker_pick_string ↑

```
  subroutine marker_pick_string(this,name,content,status)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    character(:),allocatable,intent(out)::content
    integer(kind=dik),intent(out)::status
    call this%pick_begin(name,status=status)
    if(status==serialize_ok)then
      call this%pop(content)
      call this%pick_end(name,status)
    end if
  end subroutine marker_pick_string
```

marker_verify_nothing ↑

```
  subroutine marker_verify_nothing(this,name,status)
    class(marker_type),intent(inout)::this
    character(*),intent(in)::name
    integer(kind=dik),intent(out)::status
    integer(kind=dik),dimension(2)::p1,p2
    call this%find("<",skip=4,proceed=.false.,pos=p1)
    call this%find(">",1,.false.,p2)
    if(name//"/"==this%substring(p1,p2))then
      status=serialize_nothing
      call this%set_position(p2)
      call this%proceed(one*3,.true.)
    else
      if(name==this%substring(p1,p2))then
        status=serialize_ok
      else
        status=serialize_wrong_tag
      end if
    end if
  end subroutine marker_verify_nothing
```

marker_indent ↑

```
  subroutine marker_indent(this,step)
    class(marker_type),intent(inout)::this
    integer(kind=dik),optional::step
    if(this%do_break)call this%push(new_line(" "))
    if(this%do_indent)then
      if(present(step))this%indentation=this%indentation+step
      call this%push(repeat(" ",this%indentation))
```

```
    end if
    this%active_pages(1)=this%actual_page()
  end subroutine marker_indent
```

marker_push_heap ↑

```
  subroutine marker_push_heap(this,ser,id)
    class(marker_type),intent(inout)::this
    class(serializable_class),target,intent(in)::ser
    integer(kind=dik),intent(in)::id
    class(serializable_ref_type),pointer::new_ref
    allocate(new_ref)
    new_ref%next=>this%heap
    new_ref%ref=>ser
    new_ref%id=id
    this%heap=>new_ref
  end subroutine marker_push_heap
```

marker_pop_heap ↑

```
  subroutine marker_pop_heap(this,ser)
    class(marker_type),intent(inout)::this
    class(serializable_class),pointer,intent(out)::ser
    class(serializable_ref_type),pointer::old_ref
    if(associated(this%heap))then
       old_ref=>this%heap
       ser=>old_ref%ref
       this%heap=>this%heap%next
       deallocate(old_ref)
    else
       print('("marker_pop_heap: heap_stack is not associated.")')
    end if
  end subroutine marker_pop_heap
```

marker_search_heap_by_id ↑

```
  subroutine marker_search_heap_by_id(this,id,ser)
    class(marker_type),intent(in)::this
    integer(kind=dik),intent(in)::id
    class(serializable_class),pointer,intent(out)::ser
    class(serializable_ref_type),pointer::ref
    ref=>this%heap
    do while(associated(ref))
       if(id==ref%id)then
          ser=>ref%ref
          exit
       end if
       ref=>ref%next
    end do
  end subroutine marker_search_heap_by_id
```

marker_search_heap_by_ref ↑

```
subroutine marker_search_heap_by_ref(this,ref,id)
  class(marker_type),intent(in)::this
  class(serializable_class),pointer,intent(in)::ref
  integer(kind=dik),intent(out)::id
  class(serializable_ref_type),pointer::ref_p
  ref_p=>this%heap
  id=0
  do while(associated(ref_p))
     if(associated(ref,ref_p%ref))then
        id=ref_p%id
        exit
     end if
     ref_p=>ref_p%next
  end do
end subroutine marker_search_heap_by_ref
```

marker_push_reference ↑

```
subroutine marker_push_reference(this,ser,id)
  class(marker_type),intent(inout)::this
  class(serializable_class),target,intent(in)::ser
  integer(kind=dik),intent(in),optional::id
  class(serializable_ref_type),pointer::new_ref
  allocate(new_ref)
  new_ref%next=>this%references
  new_ref%ref=>ser
  if(present(id))then
     new_ref%id=id
  else
     new_ref%id=-1
  end if
  this%references=>new_ref
end subroutine marker_push_reference
```

marker_pop_reference ↑

```
subroutine marker_pop_reference(this,ser)
  class(marker_type),intent(inout)::this
  class(serializable_class),pointer,intent(out)::ser
  class(serializable_ref_type),pointer::old_ref
  if(associated(this%references))then
     old_ref=>this%references
     ser=>old_ref%ref
     this%references=>this%references%next
     deallocate(old_ref)
  else
     print('("marker_pop_reference: reference_stack is not associated.")')
  end if
end subroutine marker_pop_reference
```

marker_search_reference ↑

```
subroutine marker_search_reference(this,type,ser)
  class(marker_type),intent(in)::this
  integer(kind=dik),dimension(2,2),intent(in)::type
  class(serializable_class),pointer,intent(out)::ser
  class(serializable_class),pointer::tmp_ser!nag bug workaround
  class(serializable_ref_type),pointer::ref
  ref=>this%references
  nullify(ser)
  do while(associated(ref))
     tmp_ser=>ref%ref
     if(tmp_ser%verify_type(this%substring(type)))then
        ser=>tmp_ser
        exit
     end if
     ref=>ref%next
  end do
end subroutine marker_search_reference
```

marker_reset_heap ↑

```
subroutine marker_reset_heap(this)
  class(marker_type),intent(inout)::this
  if(associated(this%heap))then
     call this%heap%finalize()
     deallocate(this%heap)
  end if
end subroutine marker_reset_heap
```

marker_reset_references ↑

```
subroutine marker_reset_references(this)
  class(marker_type),intent(inout)::this
  if(associated(this%references))then
     call this%references%finalize()
     deallocate(this%references)
  end if
end subroutine marker_reset_references
```

marker_finalize ↑

```
subroutine marker_finalize(this)
  class(marker_type),intent(inout)::this
  call this%reset_heap()
  call this%reset_references()
end subroutine marker_finalize
```

13.6.8 Sonstige Prozeduren

serialize_print_comp_pointer

```
recursive subroutine serialize_print_comp_pointer(ser,unit,parents,components,peers,name)
  class(serializable_class),pointer,intent(in)::ser
  integer,intent(in)::unit
  integer(kind=dik),intent(in)::parents,components,peers
  character(len=*),intent(in)::name
  if(associated(ser))then
     write(unit,fmt=*)name," is associated."
     if(components>0)then
        write(unit,fmt=*)"Printing components of ",name
        call ser%print_to_unit(unit,parents,components-one,peers)
     else
        write(unit,fmt=*)"Skipping components of ",name
     end if
  else
     write(unit,fmt=*)name," is not associated."
  end if
end subroutine serialize_print_comp_pointer
```

serialize_print_peer_pointer

```
recursive subroutine serialize_print_peer_pointer(ser,unit,parents,components,peers,name)
  class(serializable_class),pointer,intent(in)::ser
  integer,intent(in)::unit
  integer(kind=dik)::parents,components,peers
  character(len=*),intent(in)::name
  if(associated(ser))then
     write(unit,fmt=*)name," is associated."
     if(peers>0)then
        write(unit,fmt=*)"Printing components of ",name
        call ser%print_to_unit(unit,parents,components,peers-one)
     else
        write(unit,fmt=*)"Skipping components of ",name
     end if
  else
     write(unit,fmt=*)name," is not associated."
  end if
end subroutine serialize_print_peer_pointer
```

serialize_print_allocatable

```
subroutine serialize_print_allocatable(ser,unit,parents,components,peers,name)
  class(serializable_class),allocatable,intent(in)::ser
  integer,intent(in)::unit
  integer(kind=dik),intent(in)::parents,components,peers
  character(len=*),intent(in)::name
  if(allocated(ser))then
     write(unit,fmt=*)name," is allocated."
     if(components>0)then
        write(unit,fmt=*)"Printing components of ",name
        call ser%print_to_unit(unit,parents,components-1,peers)
     else
        write(unit,fmt=*)"Skipping components of ",name
```

```
      end if
    else
      write(unit,fmt=*)name," is not allocated."
    end if
  end subroutine serialize_print_allocatable
```

measurable_less_measurable

```
  elemental function measurable_less_measurable(mea1,mea2)
    class(measurable_class),intent(in)::mea1,mea2
    logical::measurable_less_measurable
    measurable_less_measurable=mea1%measure()<mea2%measure()
  end function measurable_less_measurable
```

measurable_less_double

```
  elemental function measurable_less_double(mea1,dou)
    class(measurable_class),intent(in)::mea1
    real(kind=drk),intent(in)::dou
    logical::measurable_less_double
    measurable_less_double=mea1%measure()<dou
  end function measurable_less_double
```

measurable_less_or_equal_measurable

```
  elemental function measurable_less_or_equal_measurable(mea1,mea2)
    class(measurable_class),intent(in)::mea1,mea2
    logical::measurable_less_or_equal_measurable
    measurable_less_or_equal_measurable=mea1%measure()<=mea2%measure()
  end function measurable_less_or_equal_measurable
```

measurable_less_or_equal_double

```
   elemental function measurable_less_or_equal_double(mea1,dou)
    class(measurable_class),intent(in)::mea1
    real(kind=drk),intent(in)::dou
    logical::measurable_less_or_equal_double
    measurable_less_or_equal_double=mea1%measure()<=dou
  end function measurable_less_or_equal_double
```

measurable_equal_measurable

```
 elemental function measurable_equal_measurable(mea1,mea2)
    class(measurable_class),intent(in)::mea1,mea2
    logical::measurable_equal_measurable
    measurable_equal_measurable=mea1%measure()==mea2%measure()
  end function measurable_equal_measurable
```

measurable_equal_double

```
  elemental function measurable_equal_double(mea1,dou)
    class(measurable_class),intent(in)::mea1
    real(kind=drk),intent(in)::dou
    logical::measurable_equal_double
```

```
    measurable_equal_double=mea1%measure()==dou
  end function measurable_equal_double
```

measurable_equal_or_greater_measurable

```
  elemental function measurable_equal_or_greater_measurable(mea1,mea2)
    class(measurable_class),intent(in)::mea1,mea2
    logical::measurable_equal_or_greater_measurable
    measurable_equal_or_greater_measurable=mea1%measure()>=mea2%measure()
  end function measurable_equal_or_greater_measurable
```

measurable_equal_or_greater_double

```
  elemental function measurable_equal_or_greater_double(mea1,dou)
    class(measurable_class),intent(in)::mea1
    real(kind=drk),intent(in)::dou
    logical::measurable_equal_or_greater_double
    measurable_equal_or_greater_double=mea1%measure()>=dou
  end function measurable_equal_or_greater_double
```

measurable_greater_measurable

```
  elemental function measurable_greater_measurable(mea1,mea2)
    class(measurable_class),intent(in)::mea1,mea2
    logical::measurable_greater_measurable
    measurable_greater_measurable=mea1%measure()>mea2%measure()
  end function measurable_greater_measurable
```

measurable_greater_double

```
  elemental function measurable_greater_double(mea1,dou)
    class(measurable_class),intent(in)::mea1
    real(kind=drk),intent(in)::dou
    logical::measurable_greater_double
    measurable_greater_double=mea1%measure()>dou
  end function measurable_greater_double
```

page_ring_position

```
  pure function page_ring_position(n)
    integer(kind=dik),intent(in)::n
    integer(kind=dik),dimension(2)::page_ring_position
    page_ring_position(2)=mod(n,serialize_page_size)
    page_ring_position(1)=(n-page_ring_position(2))/serialize_page_size
  end function page_ring_position
```

page_ring_ordinal

```
  pure integer(kind=dik) function page_ring_ordinal(pos)
    integer(kind=dik),dimension(2),intent(in)::pos
    page_ring_ordinal=pos(1)*serialize_page_size+pos(2)
  end function page_ring_ordinal
```

page_ring_position_is_before_int_pos

```
pure logical function page_ring_position_is_before_int_pos(m,n)
  integer(kind=dik),intent(in)::m
  integer(kind=dik),dimension(2),intent(in)::n
  if(m<page_ring_ordinal(n))then
     page_ring_position_is_before_int_pos=.true.
  else
     page_ring_position_is_before_int_pos=.false.
  end if
end function page_ring_position_is_before_int_pos
```

page_ring_position_is_before_pos_int

```
pure logical function page_ring_position_is_before_pos_int(m,n)
  integer(kind=dik),dimension(2),intent(in)::m
  integer(kind=dik),intent(in)::n
  if(page_ring_ordinal(m)<n)then
     page_ring_position_is_before_pos_int=.true.
  else
     page_ring_position_is_before_pos_int=.false.
  end if
end function page_ring_position_is_before_pos_int
```

page_ring_position_is_before_pos_pos

```
pure logical function page_ring_position_is_before_pos_pos(m,n)
  integer(kind=dik),dimension(2),intent(in)::m,n
  if(m(1)<n(1))then
     page_ring_position_is_before_pos_pos=.true.
  else
     if(m(1)>n(1))then
        page_ring_position_is_before_pos_pos=.false.
     else
        if(m(2)<n(2))then
           page_ring_position_is_before_pos_pos=.true.
        else
           page_ring_position_is_before_pos_pos=.false.
        end if
     end if
  end if
end function page_ring_position_is_before_pos_pos
```

ring_position_increase

```
subroutine ring_position_increase(pos,n)
  integer(kind=dik),dimension(2),intent(inout)::pos
  integer(kind=dik),intent(in)::n
  pos=page_ring_position(page_ring_ordinal(pos)+n)
end subroutine ring_position_increase
```

ring_position_metric2

```
pure integer(kind=dik) function ring_position_metric2(p1,p2)
  integer(kind=dik),dimension(2),intent(in)::p1,p2
  ring_position_metric2=(p2(1)-p1(1))*serialize_page_size+p2(2)-p1(2)+1
end function ring_position_metric2
```

ring_position_metric1

```
pure integer(kind=dik) function ring_position_metric1(p)
  integer(kind=dik),dimension(2,2),intent(in)::p
  ring_position_metric1=(p(1,2)-p(1,1))*serialize_page_size+p(2,2)-p(2,1)+1
end function ring_position_metric1
```

generate_unit

```
subroutine generate_unit(unit,min,max)
  integer,intent(out) :: unit
  integer,intent(in),optional :: min,max
  integer :: min_u,max_u
  logical :: is_open
  !print *,"generate_unit"
  unit = -1
  if(present(min))then
     min_u=min
  else
     min_u=10
  end if
  if(present(max))then
     max_u=max
  else
     max_u=huge(max_u)
  end if
  do unit=min_u,max_u
     !print *,"testing ",unit
     inquire(unit,opened=is_open)
     if (.not. is_open) then
        exit
     end if
  end do
end subroutine generate_unit
```

ilog2

```
subroutine ilog2(int,exp,rem)
  integer,intent(in) :: int
  integer,intent(out) :: exp,rem
  integer :: count
  count = 2
  exp = 1
  do while (count<int)
     exp=exp+1
     count = ishft(count,1)
  end do
```

```
    if (count>int) then
       rem=(int-ishft(count,-1))
    else
       rem=0
    end if
  end subroutine ilog2
```

character_is_in

```
  pure logical function character_is_in(c,array)
    character,intent(in)::c
    character,dimension(:),intent(in)::array
    integer(kind=dik)::n
    character_is_in=.false.
    do n=1,size(array)
       if(c==array(n))then
          character_is_in=.true.
          exit
       end if
    end do
  end function character_is_in
```

integer_with_leading_zeros

```
  subroutine integer_with_leading_zeros(number,length,string)
    integer,intent(in) :: number,length
    character(len=*),intent(out) :: string
    integer :: zeros
    character::sign
    if(number==0)then
       string = repeat("0",length)
    else
       if(number>0)then
          zeros=length-floor(log10(real(number)))-1
          if(zeros<0)then
             string=repeat("*",length)
          else
             write(string,fmt='(a,I0)') repeat("0",zeros),number
          end if
       else
          zeros=length-floor(log10(real(-number)))-2
          if(zeros<0)then
             string=repeat("*",length)
          else
             write(string,fmt='(a,a,I0)') "-",repeat("0",zeros),abs(number)
          end if
       end if
    end if
  end subroutine integer_with_leading_zeros
```