

1 Datenstruktur

1.1 Allgemeine Konstanten und Steuerungsparameter (tConst)

Die relevanten Konstanten für die zu lösenden Gleichungen sind:

Variable	Name im Code	Datentyp
Adiabatenexponent γ	gamma	REAL
$\gamma - 1$	gamma1	REAL
$\gamma - 2$	gamma2	REAL
$\frac{1}{\gamma-1}$	gamma1q	REAL

Die Steuerungsparameter für die Diskretisierung sind:

Variable	Name im Code	Datentyp
CFL-Zahl	CFL	REAL
Zeitdiskretisierung	TimeSteppingMethod	INTEGER
1: STE		
2: Runge-Kutta		
Zeitordnung	TimeOrder	INTEGER
Art des Zeitschritts	TimeStepping	INTEGER
0: globale Zeitschritte		
1: lokale Zeitschritte		
Anzahl der Runge-Kutta-Stages	nRKStages	INTEGER
Runge-Kutta-Konstanten	RKa(5), RKb(5), RKc(5)	REAL
(Verwendung hängt von RK-Variante ab)		
Raumordnung	SpaceOrder	INTEGER
Flussfunktion	iFlux	INTEGER
1: Godunov		
2: Roe		
3: HLLC		
4: Local Lax-Friedrichs (Rusanov)		
5: Steger-Warming		
6: Zentral		
7: HLLC		
8: AUSMD		
Limiter	Limiter	INTEGER
1: Barth und Jespersen		
2: Venkatakrishnan		
Exakte Lösung	ExactFunc	INTEGER
Quellterm	SourceFunc	INTEGER

1.2 Netzinformationen (tMesh)

1.2.1 Variablen der Köpfe der verlinkten Listen:

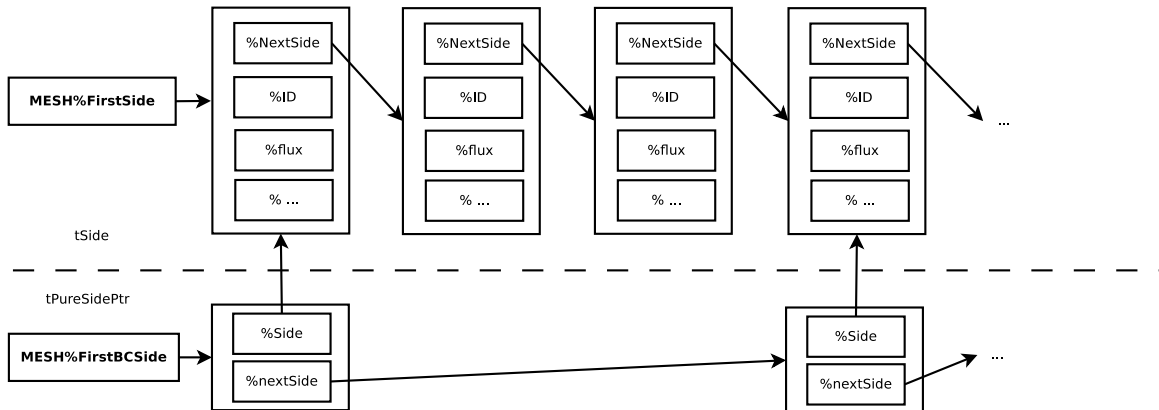
Es gibt drei für unsere Zwecke relevante verlinkte Listen.

Variable	Name im Code	Datentyp
Listenkopf der Elementliste	FirstElem	TYPE(tElem), POINTER
Listenkopf der Seitenliste	FirstSide	TYPE(tSide), POINTER
Listenkopf der Boundary Condition Seitenliste	FirstBCSide	TYPE(tPureSidePtr), POINTER

Im gegensatz zur FirstSide-Liste greift die FirstBCSide-Liste auf einen anderen type zu, TYPE(tPureSidePtr). Dieser Type enthält folgende pointer:

Type	Pointer im array	Datentyp
tPureSidePtr	Side	TYPE(tSide), POINTER
tPureSidePtr	nextSide	TYPE(tPureSidePtr), POINTER

Damit verbindet dieser Type die FirstBCSide-Liste mit der regulären FirstSide-Liste an den Seiten, an denen eine BC Seite vorliegt. Mit dem nextSide Pointer kann auf die nächste BC Seite gesprungne werden, welche unter Umständen mehrere reguläre Seiten in der FirstSide-Liste überspringt. Der tPureSidePtr Pointer ermöglicht somit eine einfache Definition der FirstBCSide-Liste ohne die Variablen des tSide Types für die BC Seiten doppelt zu definieren. Dieses Konstrukt ist in der unterstehenden Graphik noch einmal verdeutlichen.



1.2.2 Variablen der Pointer Arrays:

Aus den verlinkten Listen werden Pointer Arrays erstellt welche in allen Schleifenbasierten Operation verwendet werden. Die wichtigsten Arrays sind die Seiten Arrays, weil sie zur Flussberechnung und zur Rekonstruktion verwendet werden. Das Elementen Array spielt eine weniger vitale Rolle, wird jedoch z. Bsp. für die CauchyKovalevskaya Prozedur verwendet.

Variable	Name im Code	Datentyp
Elementen pointer array	Elems(:)	TYPE(tElemPtr), POINTER
Seiten pointer array	Sides(:)	TYPE(tSidePtr), POINTER
BC Seiten pointer array	BCSides(:)	TYPE(tSidePtr), POINTER

Um die Pointer Arrays zu erzeugen wurden eine Reihe von neuen Types definiert, welche im Folgenden kurz erleutert werden sollen.

Type	Pointer im array	Datentyp
tElemPtr	Elem	TYPE(tElem), POINTER
tSidePtr	Side	TYPE(tSide), POINTER

Die BCSides(:) sind vom TYPE(tSidePtr), POINTER und werden damit von den Side pointern abgedeckt. Da aus dem Pointer Array direkt auf die BC Seiten gezeigt wird ist der Umweg über den TYPE(tPureSidePtr) nicht nötig, wie er bei der FirstBCSide-Liste gegangen wird.

1.3 Die Zelle (tElem)

Die Zellen bzw. Elemente sind die Kontrollvolumina des FV-Verfahrens:

Variable	Name im Code	Datentyp
Fläche	Area	REAL
Inverse der Fläche	Area_q	REAL
Baryzentrum (x,y)	Bary(2)	REAL
Projektion der Fläche auf die x- bzw. y-Achse (S_x, S_y)	Sx, Sy	REAL
Zeitschritt	dt	REAL
Zeitschrittvariable für STE und Quellterme	dt_loc	REAL
konservative Variablen (ρ, m_1, m_2, E)	cvar(4)	REAL
konservative Variablen an $u^{(0)}$ für RK-Verfahren	cvar_old(4)	REAL
primitive Variablen (ρ, v_1, v_2, p)	pvar(4)	REAL
x-Komponenten des Gradienten der primitiven Variablen	u_x(4)	REAL
y-Komponenten des Gradienten der primitiven Variablen	u_y(4)	REAL
Zeitableitung der primitiven Variablen	u_t(4)	REAL
Konstante ϵ^2 für den Venkatakrishnan-Limiter	venk_epsilon_sq	REAL
Volumenintegral: Anzahl der Gausspunkte	nGP	INTEGER
Volumenintegral: Gausspunkte	xGP(nGP,2)	REAL
Volumenintegral: Gaussgewichte	wGP(nGP)	REAL
Pointer auf die erste Elementseite	firstSide	TYPE(tSide), POINTER
Pointer auf das nächste Element	nextElem	TYPE(tElem), POINTER

1.4 Die Seite eines Elements (tSide)

In der Seite sind alle Informationen gespeichert die für die Flussberechnung und Rekonstruktion benötigt werden. Da bis auf die Randseiten alle Seiten zweimal vorkommen, da sie aneinander liegen wird vom Sides Pointer Array immer nur auf eine der beiden Seiten gezeigt. Um die andere Seite zu erreichen muss man den `connection` Pointer verwenden.

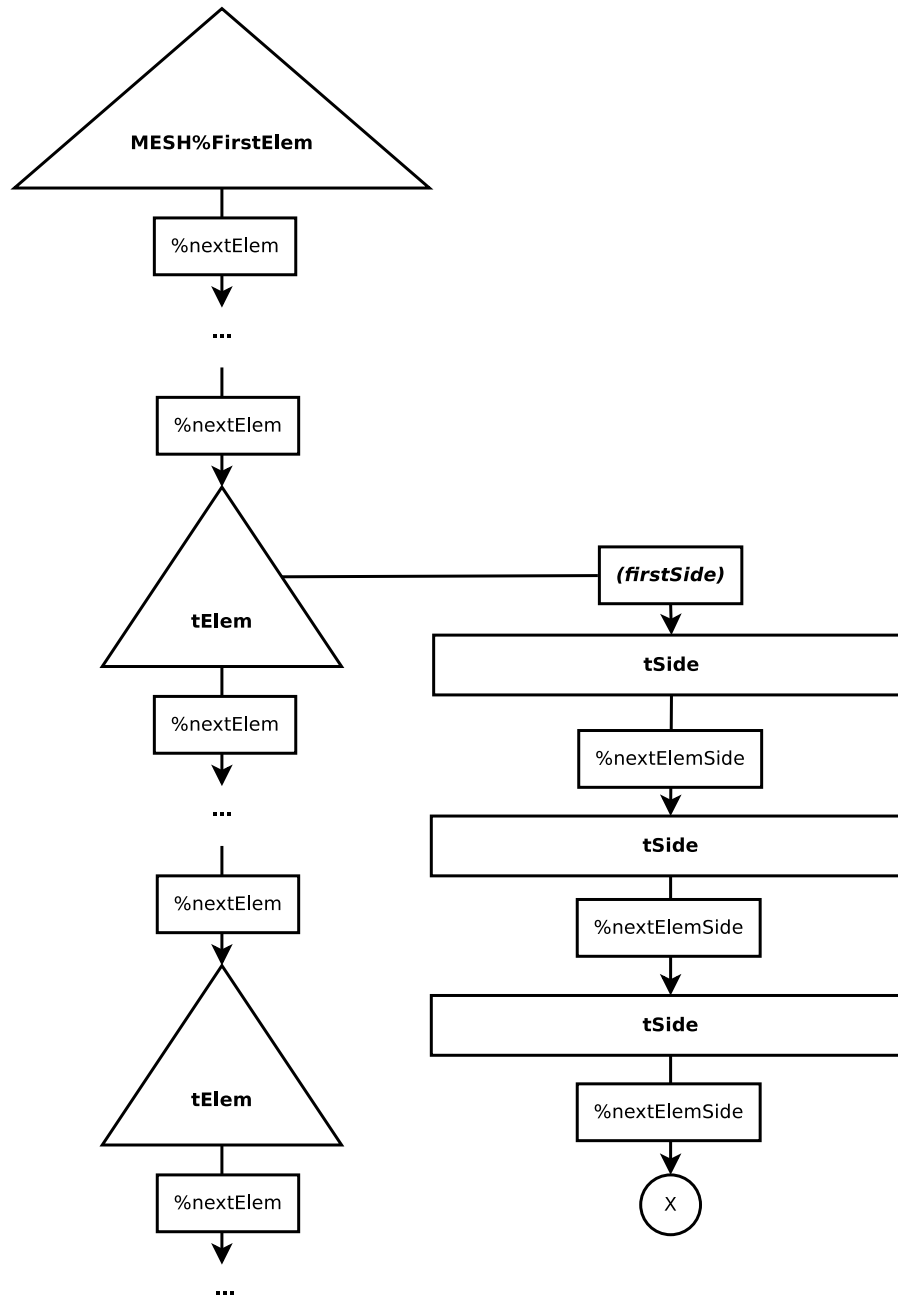
Variable	Name im Code	Datentyp
Vektor vom Baryzentrum der Zelle zum Seitenmittelpunkt (Gausspunkt für die numerische Integration über die Seite)	GP(2)	REAL
Vektor vom Baryzentrum der Zelle zum Baryzentrum der Nachbarzelle	BaryBaryVec(2)	TYPE(tBoundary), Pointer
Länge von Vektor vom Baryzentrum der Zelle zum Baryzentrum der Nachbarzelle	BaryBaryDist	REAL
Vektor \vec{w} für die Least-Squares-Rekonstruktion	w(2)	REAL
Rekonstruierte primitive Variablen am Seitenmittelpunkt	pvar(4)	REAL
Normalenvektor	n(2)	REAL
Seitenlänge	length	REAL
Pointer auf die Informationen für die Randbedingung (Dieser Pointer ist nur für Randseiten allokiert)	BC	TYPE(tBoundary), POINTER
Pointer auf die nächste Seite in der globalen Liste	nextSide	TYPE(tSide), POINTER
Pointer auf die nächste Seite in der elementlokalen Liste	nextElemSide	TYPE(tSide), POINTER
Pointer auf das Element zu dem die Seite gehört	Elem	TYPE(tElem), POINTER
Nachbarseite	connection	TYPE(tSide), POINTER
Flussvektor	flux(NVAR)	REAL

2 Datenzugriff

2.1 Zugriff von einem Element auf dessen Seiten

Der Zugriff auf die Elementliste erfolgt über den Listenkopf `aElem => MESH%FirstElem`. Vom Element aus gibt es über den `aSide => aElem%firstSide` Pointer Zugriff auf die erste Seite des Elements. Über den `aSide => aElem%nextElemSide` Pointer kommt man zur nächsten Seite im Element

2.1.1 Schematische Darstellung



2.1.2 Codebeispiel: Verlinkte Liste

Hier wird in jedem Element die gesamte Seitenliste abgearbeitet:

```
...
TYPE(tElem), POINTER      :: aElem
TYPE(tSide), POINTER     :: aSide
...
aElem => MESH%FirstElem
DO WHILE(ASSOCIATED(aElem))
  aSide => aElem%firstSide
  DO WHILE(ASSOCIATED(aSide))
    ...
    aSide => aSide%nextElemSide
  END DO
  aElem => aElem%nextElem
END DO
```

Hier wird mit allen BC Randseiten etwas getan:

```
...
TYPE(tPureSidePtr), POINTER :: aBCSide
TYPE(tSide), POINTER :: aSide
...
aBCSide => MESH%FirstBCSide
DO WHILE (ASSOCIATED(aBCSide))
  aSide => aBCSide%Side
  ...
  aBCSide => aBCSide%nextSide
END DO
```

Es ist auch möglich das ganze über die Elemente zu machen und den BC Pointer zu nutzen.

```
...
TYPE(tElem), POINTER      :: aElem
TYPE(tSide), POINTER     :: aSide
...
aElem => MESH%FirstElem
DO WHILE(ASSOCIATED(aElem))
  aSide => aElem%firstSide
  DO WHILE (ASSOCIATED(aSide))
    IF (ASSOCIATED(aSide%BC) THEN
      ...
    END IF
    aSide => aSide%nextElemSide
  END DO
  aElem => aElem%nextElem
END DO
```

2.1.3 Codebeispiel: Pointer Array

Hier wird in jedem Element die gesamte Seitenliste abgearbeitet:

```

...
TYPE(tElem), POINTER      :: aElem
TYPE(tSide), POINTER     :: aSide
INTEGER                  :: iElem
...
DO iElem = 1, MESH%nElems
  aElem => MESH%Elems(iElem)Elem
  aSide => aElem%firstSide
  DO WHILE(ASSOCIATED(aSide))
    ...
    aSide => aSide%nextElemSide
  END DO
END DO

```

Hier wird mit allen BC Randseiten etwas getan:

```

...
TYPE(tSidePtr), POINTER :: aSide
INTEGER             :: iSide
...
DO iSide = 1, MESH%nBCSides
  aSide => MESH%BCSides(iSide)%Side
  ...
END DO

```

2.2 Zugriff von einer Seite auf deren Elemente

Das Pointer Array `MESH%Sides` enthält alle Seiten nur einmal. Für die Flussberechnung ist dies essentiell. Deswegen wird geraten immer das Pointer Array zu verwenden und dann von jeder Seite über den `aSide%connection` Pointer auf die Nachbarseite zuzugreifen.

2.2.1 Codebeispiel

Hier werden alle Seiten abgearbeitet und auf die jeweiligen Nachbarseiten zugegriffen.

```

...
TYPE(tSide), POINTER      :: aSide
TYPE(tSide), POINTER     :: aNeighborSide
INTEGER                  :: iSide
...
DO iSide = 1, MESH%nSides
  aSide => MESH%Sides(iSide)%Side
  aNeighborSide => aSide%connection
  ...
END DO

```

2.2.2 Schematische Darstellung

