Prof. Dr. Timm Grams
Hochschule Fulda
Fachbereich Elektrotechnik
Marquardstraße 35
36039 Fulda

Tel (0661) 78157
(0661) 9640-550
Fax (0661) 9640-559
mailto:Timm.Grams@et.hs-fulda.de
www.hs-fulda.de/~grams

Samstag, 30. Mai 2009
(letzte Änderung: 07.09.2010)

# Java-Projekt: Fahrweisenoptimierung

**Eine beispielhafte Anwendungen von Heuristiken bei der Optimierung**

## Inhalt

Einleitung ... 2
Fahrweisenoptimierung ... 2
*Zug- und Bremskraft ... 2*
*Modell der Fahrdynamik und numerische Integration ... 4*
Pflichtenblatt ... 6
*Eingabe: Daten zur Fahrstrecke und zu den Fahrweisen ... 6*
*Eingabe: Steuerungsparameter ... 7*
*Verarbeitung ... 8*
*Ausgabe ... 8*
*Testdefinition ... 8*
Stationäre Analyse zur Validierung des Modells ... 8
Testfälle ... 10
Entwurf ... 10
*Modellierung der Individuen ... 10*
*Variationsoperatoren ... 11*
*Bewertungsfunktion ... 11*
*Auswahloperator ... 12*
*Ablaufpläne der Optimierung ... 13*
Ablaufplan 1 für das Evolutionsverfahren ... 13
Ablaufplan 2 für das Evolutionsverfahren ... 14
Ablaufplan 3 für das Evolutionsverfahren ... 15
*Interaktive Steuerung ... 16*
*Bedienoberfläche ... 16*
Realisierung ... 17
*Programmsteuerung (Ein- und Ausgabe) ... 17*
*Die Individuen ... 17*
*Die run-Methode der Run-Klasse: Spielzüge ... 17*
Abnahme – Testdurchführung ... 18
*Experiment 0 – Strecke mit Geschwindigkeitsbeschränkungen ... 18*
Die Eingabedaten ... 18
Phase 1: Aus Zufallsfahrten entsteht der erste Lösungsvorschlag ... 19
Phase 2: Verfeinerung ... 19
Phase 3: Teilstreckenreduktion ... 19
*Experiment 1 – Straffe Fahrt ... 19*
*Experiment 2 – Energiesparende Fahrt ... 19*
Anhang: Numerische Integration von Differentialgleichungen ... 20
Literatur ... 22

## Einleitung

Ziel des Projekts ist weitestgehende Übertragung der KoopeEgo-Strategie (Grams, 2007) auf ein technisch-wirtschaftliches Optimierungsproblem.

Die Aufgabenstellung: Das Programm soll für einen Eisenbahnzug die optimale Fahrweise bestimmen. Der Zug soll eine vorgegebene gerade Strecke in möglichst kurzer Zeit und mit möglichst geringem Energieverbrauch zurücklegen. Dabei sind abschnittsweise gegebene Geschwindigkeitsbeschränkungen einzuhalten. Die Fahrt sollte mit möglichst wenigen Fahrstufenumschaltungen auskommen.

Es handelt sich um ein Problem der *Nichtlinearen Programmierung* mit einer unbestimmten Anzahl von Parametern. Die Optimierung geschieht mit einem *Evolutionsverfahren* in Anlehnung an KoopEgo, das mit dem *Simulated-Annealing-Verfahren* ("simulierte Abkühlung") kombiniert wird.

Grundidee: Es wird eine schachbrettartige Welt gebildet. Jedes Feld ist mit einem Individuum besetzt, das eine Fahrweise des Zuges repräsentiert. Die Fahrweise ist sozusagen das Gen des Individuums.

Je Spielzug wird ein Platz der Welt rein zufällig als Zentrum bestimmt. Aus der Umgebung des Zentrums wiederum werden zwei Individuen gewählt. Das sind die Eltern eines neuen Individuums. Das Gen – die Fahrweise – des neuen Individuums ist Ergebnis einer Crossing-Over-Operation, angewandt auf die Gene der Eltern, und einer anschließenden Mutationsoperation.

Ergibt sich dabei eine günstigere Fahrweise als diejenige des zentralen Individuums, so wird letzteres durch das neue Individuum ersetzt. Bei der simulierten Abkühlung wird die Ersetzung nur mit einer gewissen Wahrscheinlichkeit durchgeführt. Diese Wahrscheinlichkeit hängt vom Grad der Verbesserung ab.

Die hier vorgeschlagene Optimierungsmethode ist grundsätzlich auf alle Probleme anwendbar, bei denen die Lösung von einer Vielzahl reeller Parameter abhängt und deren Anzahl sich im Verlaufe der Optimierung sogar noch ändern kann.

## Fahrweisenoptimierung

Das Optimierungsproblem: Ein Eisenbahnzug soll möglichst schnell und energiesparend über eine Strecke bestimmter Länge mit abschnittsweise definierten Höchst- und Zielgeschwindigkeiten gefahren werden.

Als konkretes Beispiel dienen die Daten eines ICE 1 mit 12 Mittelwagen. Für die Demonstration des Optimierungsverfahrens genügt hier ein vereinfachtes Modell für die Bewegung des Zuges. Die wesentlichen Zusammenhänge, Formeln und Daten sind der Dissertation von Ulrich Lindner (2004) entnommen.

### *Zug- und Bremskraft*

Die maximale Zugkraft $F_z$ hängt von der verfügbaren mechanischen Antriebsleistung $P_{z,\,mech}$, von der Geschwindigkeit $v$ und der Zugkraftbegrenzung $F_{z,\,max}$ ab:

$F_z = F_{z,\,max}$ für $v < P_{z,\,mech}/F_{z,\,max}$ und

$F_z = P_{z,\,mech}/v$ sonst.

Der Zugkraft wirkt die geschwindigkeitsabhängige Laufwiderstandskraft $W_l$ entgegen. Die geschwindigkeitsabhängige Laufwiderstandkraft wird näherungsweise als Polynom zweiten

Grades dargestellt, deren Koeffizienten $a$, $b$, und $c$ wegen der vielen Einflussfaktore für jedes Fahrzeug messtechnisch ermittelt werden müssen:

$$W_l = a + bv + cv^2$$

Die den Zug beschleunigende resultierende Kraft ist

$$F = F_z - W_l.$$

Für den Bremsvorgang gelten entsprechende Beziehungen. $F_z$ nimmt hier einen negativen Wert an, der gegeben ist durch $-F_{b,\,max}$ für $v < P_{b,\,mech}/F_{b,\,max}$ und $-P_{b,\,mech}/v$ sonst.

Zug- bzw. Bremskraft nehmen nicht nur diese maximalen Werte an. Diese Werte reduzieren sich um einen von der gewählten Fahrstufe abhängigen Proportionalitätsfaktor $\alpha$. Es gilt $0 \le \alpha \le 1$. Eine Reduktion der geschwindigkeitsabhängigen Kraft um den Faktor $\alpha$ wird erreicht durch eine entsprechende Reduktion der Maximalwerte für Kraft und Leistung.

Ob der Zug gerade angetrieben oder gebremst wird, legen wir durch die binäre Variable $\beta$ fest: $\beta = 0$ steht für Antreiben und $\beta = 1$ für Bremsen. Das Parameterpaar $(\alpha, \beta)$ ist der *Fahrzustand* des Zuges.

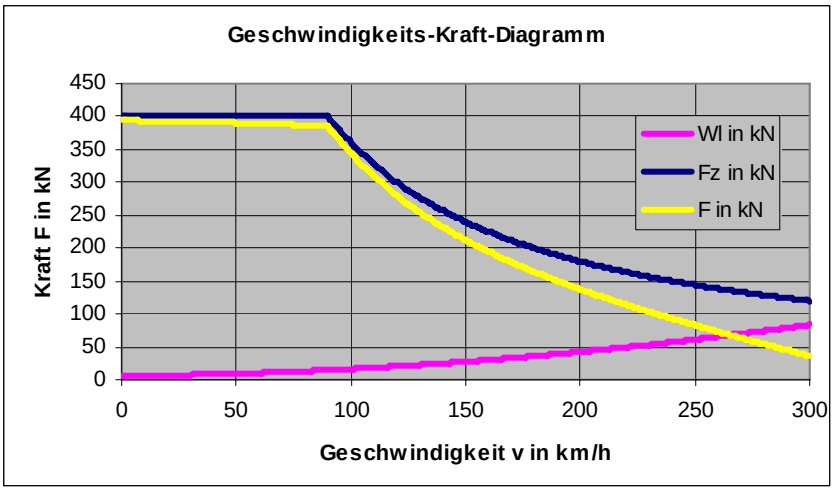
Bei Überschreitung der Höchstgeschwindigkeit oder einer Geschwindigkeitsbegrenzung wird die Antriebskraft auf null gesetzt.

*Beispiel:* Für den ICE 1 mit zwei Triebköpfen und 12 Mittelwagen gehen wir von den folgenden Daten aus.

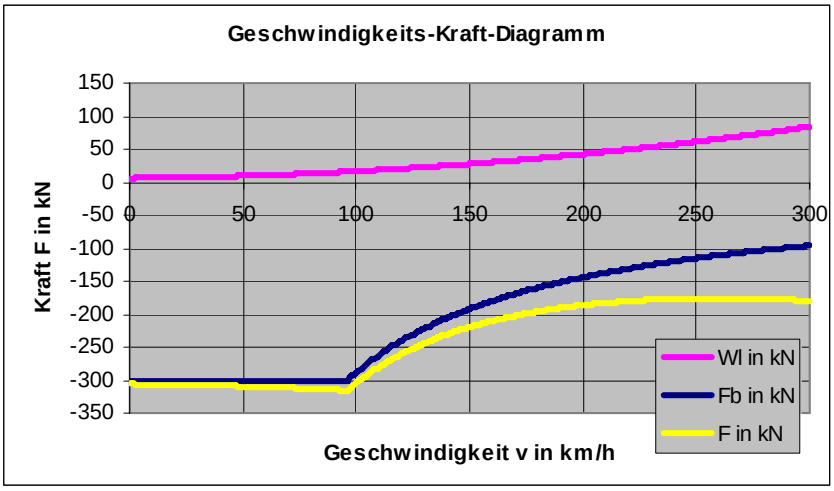
$F_{z,\,max} = 400$ kN, $P_{mech} = 10000$ kW,

$a = 6$ kN, $b = 0.1$ kNs/m und $c = .01$ kNs$^2$/m$^2$.

Für den Antrieb ergeben sich die Geschwindigkeits-Kraft-Abhängigkeiten der folgenden Grafik.




Für die Bremsung mittels regenerativer Bremse setzen wir anstelle der maximalen Zugkraft die maximale Widerstandkraft von 300 kN und eine maximale Bremsleistung von 8000 kW ein. Die Grafik zeigt die sich aus diesen Annahmen ergebende resultierende Kraft $F$.




Ich fasse zusammen: Für jeden Fahrzustand ist die auf das Fahrzeug einwirkende beschleunigende oder bremsende Kraft eine Funktion der Geschwindigkeit.

$$F = F(v).$$

Soll auch die Abhängigkeit vom Fahrzustand ($\alpha$, $\beta$) zum Ausdruck kommen, wird ausführlicher

$$F = F(v, \alpha, \beta)$$

geschrieben.

### *Modell der Fahrdynamik und numerische Integration*

Die elektrische Leistung $P_{el}$ geht nur mit einem Wirkungsgrad von $\eta_z$ in mechanische Leistung über. Die für die Geschwindigkeit $v$ erforderliche elektrische Leistung zur Erzielung der maximalen Zugkraft ist gleich

$$P = F_z \cdot v / \eta_z$$

Entsprechend unvollkommen ist die Rückspeisung mittels regenerativer Bremsung. Nur ein Anteil $\eta_b$ der mechanischen Leistung wird in elektrische Leistung umgesetzt:

$$P = F_z \cdot v \cdot \eta_b.$$

Hier wird der Einfachheit halber mit durchschnittlichen Wirkungsgraden gerechnet, und zwar im Allgemeinen mit $\eta_z = \eta_b = 0.9$.

Die verbrauchte bzw. zurückgewonnene Energie $E$ ergibt sich durch Integration der Leistung über die Zeit.

Kraft $F$, Ort $x$, Geschwindigkeit $v$, elektrische Leistung $P$ und elektrische Energie $E$ sind abhängig von der Zeit $t$. Diese Zeitabhängigkeit wird hier zugunsten einer knappen Darstellung nicht immer mit hingeschrieben. Es gelten die *Bewegungsgleichungen:*

$$\dot{x} = v\,,\ \dot{v} = F(v)/m\,,\ \dot{E} = P(v)\,.$$

Darin steht der Punkt über einer Variablen für deren zeitliche Ableitung, z.B. $\dot{x} = \dfrac{\mathrm{d}x}{\mathrm{d}t}$.

Der dynamische Zustand $\mathbf{z}$ zum Zeitpunkt $t$ ist definiert durch das Zahlentripel aus Ort, Geschwindigkeit und verbrauchter elektrischer Energie:

$$\mathbf{z}(t) = (x(t), v(t), E(t)), \text{ kurz } \mathbf{z} = (x, v, E).$$

Die Bewegungsgleichungen haben mit der Übergangsfunktion

$$\boldsymbol{f}(\boldsymbol{z}) = (v, F(v)/m, P)$$

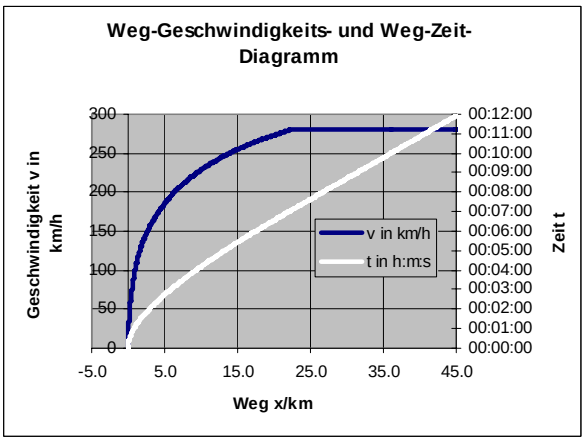
die Gestalt

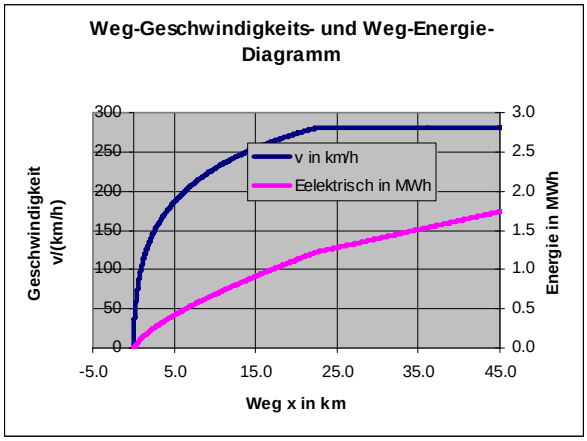
$$\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z}) .$$

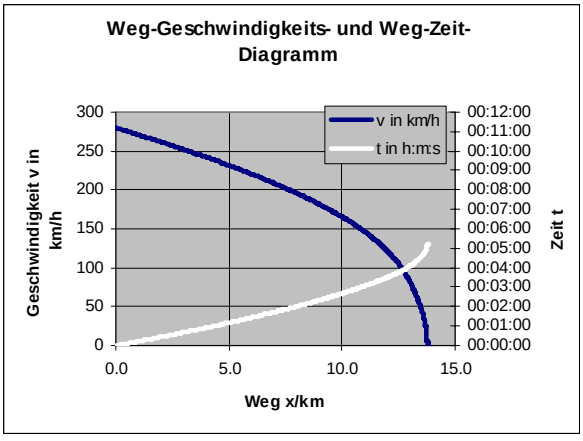
Zu Beginn einer Fahrt gelten die Anfangsbedingungen $\boldsymbol{z}(0) = (0, 0, 0)$, also $x(0) = 0$, $v(0) = 0$ und $E(0) = 0$.

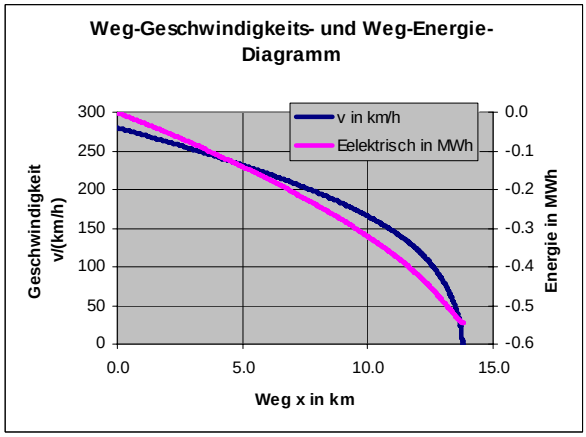
Die Masse $m$ setzt sich zusammen aus der Masse des Zuges zuzüglich eines prozentualen Aufschlags von etwa 9%, der die Trägheitsmomente der rotierenden Teile berücksichtigt. Im Falle des ICE 1 mit 12 Mittelwagen ist der Wert gleich 925410 kg.

Das Tabellenkalkulationsblatt Fahrdynamik.xls gibt einen ersten Einblick in die Fahrdynamik bei Wahl der höchsten Fahr- bzw. Bremsstufe. Für die Integration der Differentialgleichungen wurde das Euler-Cauchy-Verfahren mit einer Zeitschrittweite von 1s gewählt.




Im Programm werden die Bewegungsgleichungen mit dem Verfahren von Heun (Hn) oder mit dem Runge-Kutta-Verfahren (RK) numerisch integriert. Als Integrationsschrittweite $\Delta t$ wird ein Bruchteil $\rho$ der Zeit genommen, die gebraucht wird um das Fahrzeug unter Einsatz der maximalen Kraft von 0 auf die Geschwindigkeit $v_{\max}$ zu bringen. In Formelzeichen:

$$\Delta t = \rho \cdot \frac{v_{\max} \cdot m}{F_{z,\max}} .$$

Für die Zahlenwerte des Beispiels und mit einem Faktor $\rho$=0.1 ergibt sich der Wert $\Delta t$ = 0.005h =19s. Der Parameter $\rho$ bestimmt also die Genauigkeit (Fineness) der Integration.

## Pflichtenblatt

### *Eingabe: Daten zur Fahrstrecke und zu den Fahrweisen*

Die Syntax der Eingabesprache wird mittels erweiterter Backus-Naur-Form (EBNF) definiert. Erläuterung der EBNF: Das Gleichheitszeichen wird gelesen als „ist definiert als“, nichtterminale Symbole sind kursiv gesetzt. Die eckige Klammer steht für optionale Textteile und die geschweifte für optionale und beliebig oft wiederholbare Teile. Der senkrechte Strich trennt Alternativen. Terminale Symbole werden fett geschrieben.

*SystemData* = { *Def* **;** }
*Def* = *RouteData* | *DrivingForceMax* | *BrakingForceMax* | *Resistance* |
*VelocityMax* | *DrivingPowerMax* | *BrakingPowerMax* | *DrivingEffectiveness* |
*BrakingEffectiveness* | *Lmax* | *Mass* | *MaxTime* | *NumericalMethod* | *Fineness*
*RouteData* = **Route** *Length* { **,** *Restriction* }
*Restriction* = *Begin End Velocity*
*Length* = *Float*
*Begin* = *Float*
*End* = *Float*
*Velocity* = *Float*
*DrivingForceMax* = **FzMax** *Float*
*BrakingForceMax* = **FbMax** *Float*
*Resistance* = **abc** *Float Float Float*
*VelocityMax* = **vMax** *Float*
*DrivingPowerMax* = **PzMax** *Float*
*BrakingPowerMax* = **PbMax** *Float*
*DrivingEffectiveness* = z**Eta** *Float*
*BrakingEffictiveness* = **bEta** *Float*
*Lmax* = **nLevel** *Integer*
*Mass* = **Mass** *Float*
*MaxTime* = **tMax** *Float*
*NumericalMethod* = **Hn** | **RK**
*Fineness* = **rho** *Float*

Startsymbol ist das Symbol *SystemData.* Die meisten Schlüsselwörter sind selbsterklärend. Die maximale Zeit tMax ist die Obergrenze für die Fahrtdauer. Erreicht das Fahrzeug innerhalb dieser Zeit nicht den Zielort, bleibt die Fahrt unberücksichtigt.

Kommentare beginnen mit der Zeichenfolge /* und enden mit der Zeichenfolge */.

Die Eingabe für die Fahrt eines ICE 1 mit 12 Mittelwagen über eine Strecke von 100 km Länge mit zwei geschwindigkeitsbeschränkten Teilstrecken könnte so aussehen:

```
Route 100.0, 30.0 50.0 150.0, 70.0 90.0 180.0;
                        /*Streckenlaenge und Restriktionen. Zahlenangaben in km bzw. km/h*/
FzMax  400.0; /*Maximale Zugkraft in kN*/
FbMax  300.0; /*Maximale Bremskraft in kN*/
abc
  6.0 /*Einheit: kN*/
  0.1 /*Einheit: kN/(m/s)*/
  0.01; /*Einheit: kN/(m/s)^2*/
vMax  280.0; /*Hoechstgeschwindigkeit in km/h*/
PzMax  10000.0; /*Maximale Antriebsleistung in kW*/
PbMax  8000.0; /*Maximale Bremsleistung in kW*/
zEta  0.9; /*Antriebswirkungsgrad*/
bEta  0.9; /*Bremswirkungsgrad*/
nLevel  10; /*Anzahl der Fahr- bzw. Bremsstufen*/
Mass  925410.0; /*Wirksame Masse in kg*/
tMax  60.0; /*Obergrenze der Fahrtdauer in min*/
RK; /*Integration mit einem Runge-Kutta-Verfahren*/
rho 0.1; /*Feinheit der Integration (Genauigkeit)*/
```

### *Eingabe: Steuerungsparameter*

Auch für jeden Steuerungsparameter ist ein Default-Wert voreingestellt. Durch die Parametereingabe kann diese Standardeinstellung verändert werden. Nach Drücken des Eingabe-Knopfs (Button) werden die Verfahrensparameter und die Systemdaten von einer Hintergrunddatei geladen. Die Liste der Verfahrensparameter:

| | |
|---|---|
| $n$ | Größe der Welt. Die Welt ist ein Spielfeld aus n-mal-n Feldern. |
| $k$ | Umgebungsparameter. Legt die k-Umgebung fest. |
| $h$ | Anzahl der Spielzüge bis zum Halt. |
| $p$ | Mutationswahrscheinlichkeit. |
| $q$ | Wahrscheinlichkeit für Crossing-Over. |
| $T$ | Temperatur (Simulated Annealing). Parameter des Auswahloperators. |
| $k_S$ | Gewicht für die Bewertung der Anzahl von Teilstrecken in min/Teilstrecke |
| $k_E$ | Gewicht für die Bewertung des Energieverbrauchs in min/MWh |
| $k_v$ | Gewicht für die Geschwindigkeitsüberschreitungen in min |
| $v_T$ | Toleranz für Geschwindigkeitsüberschreitungen in km/h |

Die Bedeutung der Parameter wird weiter unten erläutert. Alle Bewertungen werden in äquivalente Fahrtzeiten umgerechnet und in Minuten angegeben. Die Einheiten für die Gewichte der Zielgrößen enthalten demzufolge im Zähler die Basiseinheit der Bewertungsfunktion (min) und im Nenner die Einheit der zu bewertenden Größe.

Die Grammatik für die Eingabe der Verfahrensparameter ist folgendermaßen definiert:

*Parameter* = { *ParameterNumber* }
*ParameterNumber = IntParam Integer* | *RealParam Float*
*IntParam* = **n** | **k** | **h**
*RealParam* = **p** | **q** | **T** | **kS** | **kE** | **kv** | **vT**

Anmerkung zur Schreibweise: Programmvariablen werden nicht kursiv geschrieben. Kommen diese Variablen in mathematischen Ausdrücken vor, dann wird zur Kursivschreibweise übergegangen.

Einige der Verfahrensparameter lassen sich während des Programmlaufs über Textfelder verändern. Das erlaubt eine interaktive Steuerung des Optimierungsverfahrens.

Das Format der gesamten Eingabedatei (Steuerungsparameter plus Systemdaten):

*InputText = Parameter* **System** *SystemData* [*Run* ]

Die optionale Erweiterung um eine Musterfahrt (Run) beinhaltet eine Folge von Fahrtabschnitten, von denen jeweils der Endpunkt, gemessen von Streckenanfang, und die Fahrstufe angegeben wird.

*Run* = **SampleRun** *Sect* { **,** *Sect* } **;**
*Sect = Position Level*
*Position = Float*
*Level = Integer*

Die Musterfahrt wird unmittelbar nach Drücken des Startknopfes im Fenster für die Fahrtgrafik dargestellt. Auf diese Weise kann man sich ein Weg-Geschwindigkeitsdiagramm einer frei definierten Musterfahrt verschaffen. Außerdem wird das Spielfeld mit Kopien der Musterfahrt vorbelegt, so dass man die Optimierung mit lauter Klonen der Musterfahrt beginnen kann.

Eine vollständige Eingabe:

```
n  40 /*Spielfeldgroesse*/
k  1 /*Umgebungsparameter*/
h  10000 /*Anzahl der Spielzuege je Lauf*/
p  0.2 /*Mutationswahrscheinlichkeit*/
q  0.5 /*Crossing-Over-Wahrscheinlichkeit*/
```

```
T  1.4E-45 /*Temperatur*/
kS  1.0 /*Gewichtsfaktor fuer Anzahl der Teilstrecken in min*/
kE  1.0 /*Gewichtsfaktor fuer den Energieverbrauch in min/MWh*/
kv  1.0 /*Gewichtsfaktor fuer Geschwindigkeitsueberscheitungen in min*/
vT  1.0 /*Toleranz fuer Geschwindigkeitsueberscheitungen in km/h*/
System
Route 100.0; /*Streckenlaenge und Restriktionen. Zahlenangaben in km bzw. km/h*/
FzMax  400.0; /*Maximale Zugkraft in kN*/
FbMax  300.0; /*Maximale Bremskraft in kN*/
abc
  6.0 /*Einheit: kN*/
  0.1 /*Einheit: kN/(m/s)*/
  0.01; /*Einheit: kN/(m/s)^2*/
vMax  280.0; /*Hoechstgeschwindigkeit in km/h*/
PzMax  10000.0; /*Maximale Antriebsleistung in kW*/
PbMax  8000.0; /*Maximale Bremsleistung in kW*/
zEta  0.9; /*Antriebswirkungsgrad*/
bEta  0.9; /*Bremswirkungsgrad*/
nLevel  10; /*Anzahl der Fahr- bzw. Bremsstufen*/
Mass  925410.0; /*Wirksame Masse in kg*/
tMax  60.0; /*Obergrenze der Fahrtdauer in min*/
RK; /*Integration mit einem Runge-Kutta-Verfahren*/
rho 0.1; /*Feinheit der Integration (Genauigkeit)*/
SampleRun 86.5 10, 100.0 -10;
```

Abgesehen von den Angaben zu SampleRun sind alle Parameter mit den hier angegebenen Werten vorbelegt. Denselben Effekt wie die hier gezeigte vollständige Eingabe hätte die folgende Eingabe.

```
System
SampleRun 86.5 10, 100.0 -10;
```

### *Verarbeitung*

Die Strecke der Länge $L$ wird in Teilstrecken mit jeweils konstanter Fahrstufe aufgeteilt. Optimierungsziele dieser Aufteilung sind (1) eine möglichst kurze Zeit bis zur Erreichung des Ziels, (2) eine möglichst geringe Anzahl von Fahrstufenwechseln (Teilstrecken), (3) ein möglichst geringer Verbrauch an elektrischer Energie sowie (4) die Einhaltung der Geschwindigkeitsbeschränkungen und die Erreichung der Ankunftsgeschwindigkeit null.

### *Ausgabe*

Optimierungsverlauf und –ergebnis werden in zwei quadratischen Panels dargestellt. Eines davon zeigt die Welt. Jedes der $n \times n$ Felder der Welt repräsentiert einen Lösungsvorschlag für die Fahrt. Die Farbe des Feldes gibt die Güte der Lösung wieder. Dabei steht die Farbe Rot für eine schlechtes und die Farbe Blau für ein gutes Ergebnis. Dazwischen liegen weitere Farben des Regenbogens.

Klickt man ein Feld der Welt an, erscheint eine Kurzinformation zur Fahrt, insbesondere die Bewertung nach Zeitbedarf, Energieverbrauch und Anzahl der Teilstrecken. Gleichzeitig wird die Fahrt als Musterfahrt definiert und im nebenstehenden Panel mit ihrem Weg-Geschwindigkeitsdiagramm samt einzuhaltenden Geschwindigkeitsbeschränkungen grafisch dargestellt.

Die aktuell gültigen Steuerungsparameter, die Systemdaten und die Musterfahrt können im Eingabeformat abgespeichert werden.

### *Testdefinition*

#### Stationäre Analyse zur Validierung des Modells

Für Validierungszwecke wird für den stationären Fall die ideale Geschwindigkeit ermittelt. Das Weg-Geschwindigkeitsdiagramm einer optimalen Fahrt sollte wo möglich dieser Geschwindigkeit nahe kommen.

Für das Durchfahren einer Strecke mit konstanter Geschwindigkeit $v$, unter Beiseitelassung der Strafen für Geschwindigkeitsüberschreitungen und für Streckenstückelung, sieht die Bewertungsfunktion so aus:

$$f = t + k_E \cdot \int_0^t \frac{W_l(v) \cdot v}{\eta_z} \cdot \mathrm{d}t \,.$$

Bewertungsmaßstab ist die Zeit $t$ für das Durchfahren der Strecke. Dazu kommt der mit dem Faktor $k_E$ gewichtete Verbrauch an elektrischer Energie. Im Integranden steht der elektrische Leistungsbedarf für die Überwindung des Laufwiderstands. Der Energiebedarf für das Aufbauen der kinetischen Energie des Zuges ebenso wie die Rückspeisung werden nicht berücksichtigt: Wir gehen davon aus, dass der Zug mit immer derselben Geschwindigkeit sehr weit fährt, so dass diese Anteile keine Rolle spielen.

Der Energiebedarf bezogen auf die Streckeneinheit ist konstant und ergibt sich aus der Ableitung der Bewertungsfunktion $f$ nach dem Ort $x$:

$$\mathrm{d}f/\mathrm{d}x = 1/v + k_E \cdot W_l(v) / \eta_z.$$

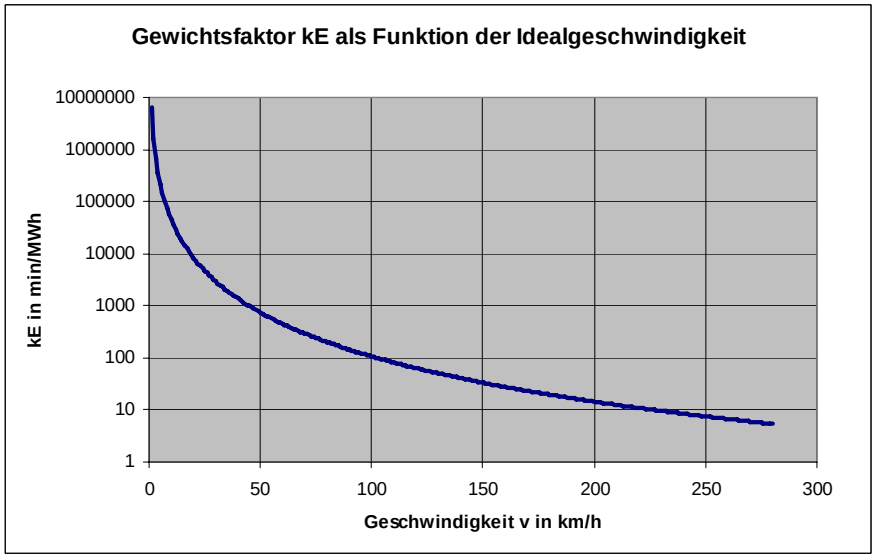
Gesucht ist die Geschwindigkeit, für die dieser Wert minimal wird. Eine notwendige Bedingung für die Minimalität ist, dass die Ableitung des streckenbezogenen Energiebedarfs nach der Geschwindigkeit gleich null ist:

$$\frac{1}{v^2} = \frac{k_E}{\eta_z} \cdot \frac{\mathrm{d}W_l(v)}{\mathrm{d}v}$$

Idealgeschwindigkeit $v$ und Gewichtsfaktor $k_E$ hängen also folgendermaßen zusammen:

$$k_E = \frac{\eta_z}{v^2 \cdot \dfrac{\mathrm{d}W_l(v)}{\mathrm{d}v}} = \frac{\eta_z}{bv^2 + 2cv^3} \,.$$

Für die Parameterwerte $a$ = 6 kN, $b$=0.1 kN/(m/s) und $c$ = 0.01 kN/(m/s)$^2$ und den Wirkungsgrad $\eta_z$ = 0.9 ergibt sich die im folgenden Diagramm dargestellte Abhängigkeit.




*Weiterführung des Beispiels:* Der Zeitbedarf bildet den Bewertungsmaßstab. Der Energiebedarf wird in ein Zeitäquivalent umgerechnet, so dass er auf derselben Skala gemessen werden kann. Wie viele Minuten also kostet eine Kilowattstunde? Zunächst errechnen wir die monetären Kosten der Minute und monetären Kosten einer Kilowattstunde.

Zeitkosten: Typischerweise ist die Hälfte der etwa 700 Sitzplätze eines ICE belegt. Bei einem fiktiven Stundenlohn für jeden Reisenden von 20 € kostet also jede Reiseminute (entgangene Arbeitszeit) etwa 100 €. Da die Reisezeit aber auch genutzt oder gar als Erholung angesehen werden kann, wollen wir von der Hälfte dieses Betrags ausgehen. Jede Reiseminute wird mit Kosten von 50 € bewertet.

Energiekosten: Zurzeit kostet eine kWh etwa 13 Cent. Für eine MWh macht das 130 €. Da dieser Preis die Endlichkeit der genutzten Ressourcen nicht ausreichend berücksichtigt, wollen wir der Umwelt zuliebe von deutlich höheren Kosten ausgehen: Wir veranschlagen die Energiekosten auf 500 € je Megawattstunde.

Zeitäquivalent der Energie: In unserem ICE-Beispiel verursacht der Verbrauch von einer MWh dieselben Kosten wie eine Reiszeit von 10 Minuten. Das legt nahe, den Gewichtsfaktor für den Energieverbrauch auf den Wert $k_E$ = 10 min/MWh festzulegen. Der Grafik ist zu entnehmen, dass für diesen Wert die Idealgeschwindigkeit gleich 225 km/h ist.

### Testfälle

*Experiment 0:* Die Streckenlänge beträgt 100 km. Von Kilometer 30 bis 50 besteht eine Geschwindigkeitsbeschränkung auf 150 km/h. Von Kilometer 70 bis 90 ist die Geschwindigkeitsbeschränkung gleich 180 km/h. Optimierungsziel: Möglichst kurze Fahrzeit und geringe Zahl von Fahrstufenwechseln. Energiesparsame Fahrweise ist nicht gefordert.

*Prognose:* Erwartet wird ein straffe bzw. spitze Fahrweise, bei der der Zug durchgehend – abgesehen von wirksamen Geschwindigkeitsbeschränkungen – mit maximaler Zugkraft beschleunigt bzw. mit. maximaler Betriebsbremskraft abbremst.

*Experiment 1:* Die Strecke der Länge 100 km ist frei von Geschwindigkeitsbeschränkungen. Optimierungsziel: Möglichst kurze Fahrzeit und geringe Zahl von Fahrstufenwechseln. Energiesparsame Fahrweise ist nicht gefordert.

*Prognose:* Erwartet wird ein straffe Fahrweise. Der Zug wird nach einer Phase maximaler Beschleunigung die maximale Geschwindigkeit erreichen. Dieser Antriebsphase schließt sich die Bremsphase an, in der mit maximaler Betriebsbremskraft der Zug im Zielpunkt zum Stillstand gebracht wird.

*Experiment 2:* Die Strecke der Länge 100 km ist frei von Geschwindigkeitsbeschränkungen. Optimierungsziel: Möglichst kurze Fahrzeit, geringe Zahl von Fahrstufenwechseln und energiesparsame Fahrweise. Der Gewichtsfaktor $k_E$ wird auf den Wert 1000 min/MWh festgelegt.

*Prognose:* Zu erwarten ist eine Fahrt mit einer Geschwindigkeit um 50 km/h. Für die hundert Kilometer werden etwa zwei Stunden gebraucht.

## Entwurf

### *Modellierung der Individuen*

Die komplette *Fahrt* setzt sich aus *Teilfahrten* mit jeweils konstanter Fahrstufe zusammen. Jede Teilfahrt wird durch den Fahrzustand und die Streckenlänge $\lambda$ charakterisiert: $(\alpha, \beta, \lambda)$. Wir schreiben eine Fahrt als Folge der sukzessiv durchlaufenen Teilfahrten: $(\alpha_1, \beta_1, \lambda_1)$, $(\alpha_2, \beta_2, \lambda_2)$, …, $(\alpha_N, \beta_N, \lambda_N)$.

Die simulierte Evolution findet auf einer Matrix mit $n \times n$ Plätzen statt. Das ist die Welt, in der sich die Evolution abspielt. Jeder Platz beherbergt ein Individuum, das eine Fahrt repräsentiert (das „Gen“).

Anfangs werden alle Plätze der Welt mit Fahrten aus zwei Teilfahrten belegt: ($\alpha_1$, 0, $rL$), ($\alpha_2$, 1, (1-$r$)$L$). Hierin sind $\alpha_1$, $\alpha_2$ und $r$ Realisierungen einer Zufallszahl aus dem Intervall [0, 1). Gleichverteilung wird vorausgesetzt. Bei den Fahrzuständen wird außerdem von einer Rasterung nach Maßgabe der Fahrstufen ausgegangen. Falls eine Musterfahrt eingegeben worden ist, werden alle Plätze der Welt mit Klonen dieser Musterfahrt vorbelegt.

### *Variationsoperatoren*

Für jeden Spielzug wird ein Feld der Welt rein zufällig ausgewählt. Das ist das Zentrum. Die Fahrt des Zentrums kann nach folgender Vorschrift ersetzt werden.

Der *Crossing-Over-Operator* wird mit der Wahrscheinlichkeit $q$ angewendet. Er arbeitet so: Aus der Umgebung des Zentrums werden zwei Eltern zufällig ausgewählt. Eine neue Fahrt wird erzeugt, die sich durch Zusammensetzung der Fahrten der Eltern ergibt. Dazu wird die Gesamtstrecke der Länge $L$ durch Wahl einer auf dem Intervall [0, 1) gleichverteilten Zufallszahl $r$ in ein Anfangsstück der Länge $rL$ und ein Endstück der Länge (1-$r$)$L$ aufgeteilt. Für das erste Teilstück wird die Fahrt des ersten Elters übernommen und für das zweite Teilstück die des zweiten Elters. Die Streckenlängen der Teilfahrten im Übergangsbereich werden geeignet verkürzt.

Falls kein Crossing-Over stattfindet, wird eine neue Fahrt dadurch gebildet, dass ein zufällig ausgewählter Nachbar geklont wird.

Der *Mutationsoperator* wird auf jede Teilfahrt der neuen Fahrt mit der Wahrscheinlichkeit $p$ angewendet. Bei jeder Mutation wird eine der folgenden Aktionen ausgeführt:

1. Die Fahrstufe wird in 40% der Fälle um 1 verringert, falls möglich.
2. Die Fahrstufe wird in 40% der Fälle um 1 vergrößert, falls möglich.
3. Die Teilfahrt wird in 10% der Fälle auf zwei Teilfahrten aufgeteilt. Die Aufteilung der Teilstrecke geschieht mittels gleichverteilter Zufallszahl. Auf einer der beiden Teilstrecken wird die Fahrstufe um eins erhöht und auf der anderen um eins erniedrigt. Wieder legt eine Fifty-fifty-Zufallsauswahl fest, wo was passiert.
4. Die Teilfahrt wird in 10% der Fälle mit der folgenden Teilfahrt verschmolzen und als Fahrstufe wird mit jeweils 50-prozentiger Wahrscheinlichkeit die der einen oder die der anderen Teilfahrt genommen.

Grundsätzlich wird zunächst entschieden, ob Crossing-Over stattfindet oder nicht. Diese Crossing-Over-Operation wird gegebenenfalls zuerst ausgeführt. Dann tritt der Mutationsoperator in Kraft, und zwar unabhäng davon, ob Crossing-Over stattgefunden hat oder nicht.

Die zum Zentrum gehörige Fahrt wird nach Maßgabe der simulierten Abkühlung durch die neue Fahrt ersetzt: Je besser die Bewertung der neuen Fahrt ist, desto wahrscheinlicher ersetzt sie die Fahrt im Zentrum (siehe den Unterabschnitt „Auswahloperator“).

### *Bewertungsfunktion*

Die zu minimierende Bewertungsfunktion ergibt sich nach einem zweistufigen Verfahren.

Die erste Stufe: Im Rahmen der Simulation werden für jede Fahrweise die folgenden Rohdaten ermittelt: Die Fahrtzeit $f_T$, gemessen in min, die Anzahl $f_S$ der Teilstrecken, der Energieverbrauch $f_E$ in kWh und Maximalwert der Geschwindigkeitsüberschreitungen $f$v, gemessen in km/h. Wird das Ende einer der Teilstrecken nicht erreicht, wird die Fahrtzeit auf den Wert unendlich gesetzt.

In der zweiten Stufe werden diese Rohdaten zur Bewertungsgröße $f$ nach folgender Formel zusammengefasst:

$$f = f_T + k_S \cdot f_S + k_E \cdot f_E + k_v \cdot \left( \frac{f_v}{v_T} \right)^2 .$$

Die Ermittlung der Rohdaten $f_T$, $f_S$, $f_E$ und $fv$ im Rahmen der ersten Stufe ist aufwendig und erfolgt deshalb für die allermeisten Fahrweisen nur einmal. Die Bewertungsfunktion $f$ wird bei jeder Grafikaktualisierung für jedes Individuum der Welt neu ermittelt. Für diese häufig vorkommenden Neuberechnungen ist nur die Formel der zweiten Stufe auszuwerten.

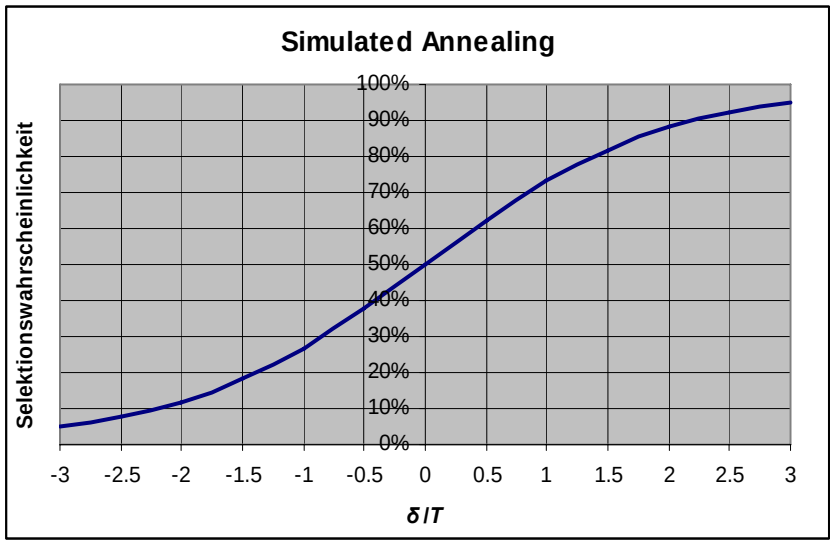
Jede Änderung der interaktiv beeinflussbaren Gewichtsfaktoren $k_S$, $k_E$ , $k$v und $v_T$ wird auf diese Weise in ihren Konsequenzen sofort wirksam und auf dem Bildschirm sichtbar.

### *Auswahloperator*

Das Verfahren der simulierten Abkühlung (Simulated Annealing) ist eine Fortentwicklung bzw. Variante des Mutations-Selektionsverfahrens (Rechenberg, 1973).

So funktionieren die Mutations-Selektions-Verfahren: Man beginnt mit einem Startpunkt $v_c$ für die Lösung. Das ist der Aufenthaltspunkt (Current Point), die momentan gültigen Lösung. Durch zufällige Variation wird ein Mutationspunkt $v_n$ (New Point) ermittelt. Mit $\delta$ wird die Verbesserung der Zielgröße bezeichnet, die durch den Mutationspunkt erreicht wird: $\delta = f(v_c) - f(v_n)$. Ergibt sich eine Verbesserung der Zielgröße ($0< \delta$), wird der Mutationspunkt zum neuen Aufenthaltspunkt, ansonsten wird der Mutationspunkt verworfen (Selektion). Nach mehreren vergeblichen Verbesserungsversuchen wird der Prozess abgebrochen.

Bei der simulierten Abkühlung wird die streng deterministische Selektion durch eine stochastische ersetzt. Auf diese Weise haben auch weniger gute Lösungen eine „Überlebenschance“ und es ergibt sich die Möglichkeit, einer Einengung der Suche auf ein schwaches lokales Optimum zu entgehen.




Die Selektionswahrscheinlichkeit für den Mutationspunkt wird auf den Wert $1/(1+e^{-\delta/T})$ gesetzt. Das heißt: Ist die Mutation neutral ($\delta$=0), dann wird eine Ersetzung mit der Wahrscheinlichkeit ½ durchgeführt. Ergibt sich tatsächlich eine Verbesserung ($0<\delta$), ist die Wahrscheinlichkeit für Ersetzung des Aufenthaltspunkts durch den Mutationspunkt größer als ½. Bei einer Verschlechterung ($\delta<0$) wird die Ersetzung mit einer geringeren Wahrscheinlichkeit als ½ durchgeführt.

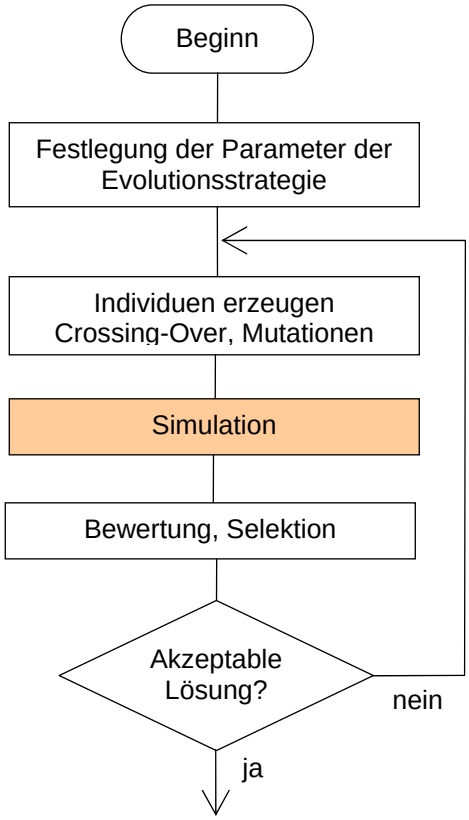
Das Verfahren wird über die Wahl des Parameters $T$ („Temperatur“) gesteuert. Ist die Verbesserung so groß wie diese Temperatur, dann erhöht sich die Selektionswahrscheinlichkeit für den Mutationspunkt auf etwa 73%. Bei einer entsprechenden Verschlechterung sinkt die Selektionswahrscheinlichkeit auf etwa 27%. Genaueres zeigt die Grafik:

### *Ablaufpläne der Optimierung*

Ablaufplan 1 für das Evolutionsverfahren

Es werden die Bewegungen für verschiedene Fahrweisensteuerungen mittels Simulation ermittelt und bewertet. Diese Simulationen werden eingebettet in das Evolutionsverfahren. Das ist das übliche Vorgehen im Rahmen von Simulationssoftware: Die Evolutionsverfahren werden den Simulationsprogrammen quasi übergestülpt.

Wegen der Dauer der Simulation und weil für die Evolutionsverfahren Millionen solcher Simulationen fällig werden, wird auf eine interaktiven Beeinflussung des Optimierungsprozesses verzichtet. Alle Parameter der Evolutionsstrategie werden vorab festgelegt, so dass die Optimierung gemäß Ablaufplan 1 autonom ablaufen kann.




Ablaufplan 1

### Ablaufplan 2 für das Evolutionsverfahren

Für die Evolutionsverfahren müssen Millionen von Lösungskandidaten erzeugt und bewertet werden. Der kritische Punkt beim Verfahren 1 ist die eingebettete Simulation, die für jedes Individuum durchgeführt werden muss.

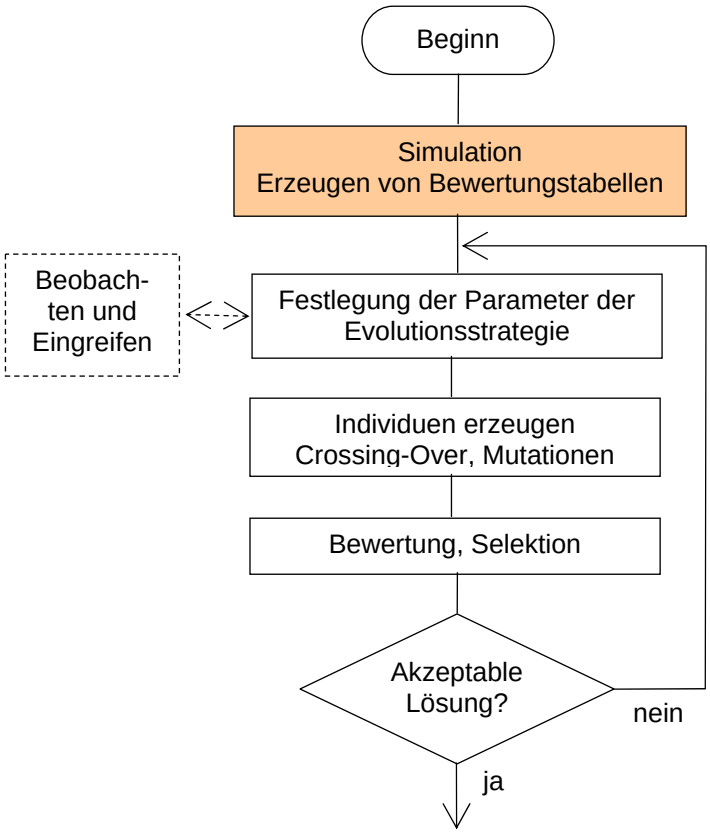
Um eine interaktive Beeinflussung des Optimierungsprozesses zu ermöglichen, wird eine Reduktion des Simulationsaufwands angestrebt.

Die aufwendige Simulation wird aus der inneren Schleife des Evolutionsverfahrens herausgezogen. Zu diesem Zweck werden die Zeitfunktionen für alle relevanten elementaren Fahrweisen vorab berechnet und in Bewegungstabellen erfasst. Dem kommt entgegen, dass die Bewegungsgleichungen ein *nichtlineares*, *zeitinvariantes* und *autonomes* dynamisches System beschreiben und dass sie invariant gegen Ortsverschiebungen sind.

Innerhalb der Optimierungsschleife werden die Bewegungen für die zusammengesetzten Fahrweisen der Individuen mithilfe der Tabelle ermittelt: Für jeden Streckenabschnitt wird aus dem Bewegungszustand am Anfang der Teilstrecke, aus der Fahrstufe und aus der Teilstreckenlänge der Bewegungszustand am Streckenende ermittelt. Das Durchlaufen der Gesamtstrecke von vorn nach hinten liefert auf diese Weise sämtliche für die Optimierung relevanten Dynamikdaten.

Da die Bewegungsgleichungen invariant gegenüber Streckenverschiebungen sind, müssen die Tabellen nur für die verschiedenen Fahrstufen und für extreme Anfangsgeschwindigkeiten (vMax und 0) angelegt werden.

Bei Stecken mit Steigungen und Gefälle geht diese Eigenschaft verloren. Dann werden die Tabellen möglicherweise zu umfangreich für eine Verwirklichung des Ablaufplans 2.



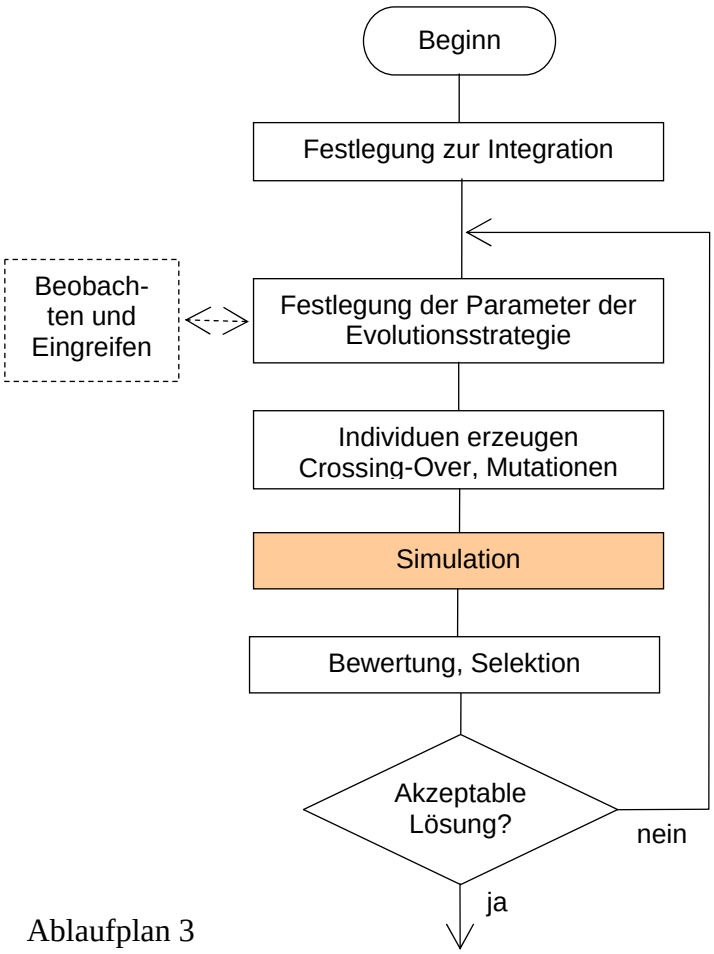

Ablaufplan 2

### Ablaufplan 3 für das Evolutionsverfahren

Da hier kein universelles Simulationsprogramm angewendet werden soll, ist ein guter Kompromiss zwischen Ablaufplan 1 und Ablaufplan 2 möglich: Die Simulation wird explizit in Java programmiert. Das Integrationsverfahren und die Schrittweite sind vorab wählbar. Dadurch können die Schnelligkeits- und Genauigkeitsforderungen gegeneinander ausbalanciert werden.

Die eingebettet Simulation wird durch Wahl einer großen Schrittweite schnell genug für eine interaktive Beeinflussung des Optimierungsprozesses (Ablaufplan 3). Hat man so eine gute Lösung gefunden, kann in einem anschließenden Optimierungslauf die Feinabstimmung mit höherer Genauigkeit erfolgen.

Die Entwurfsentscheidung fällt für den Ablaufplan 3.




Ablaufplan 3

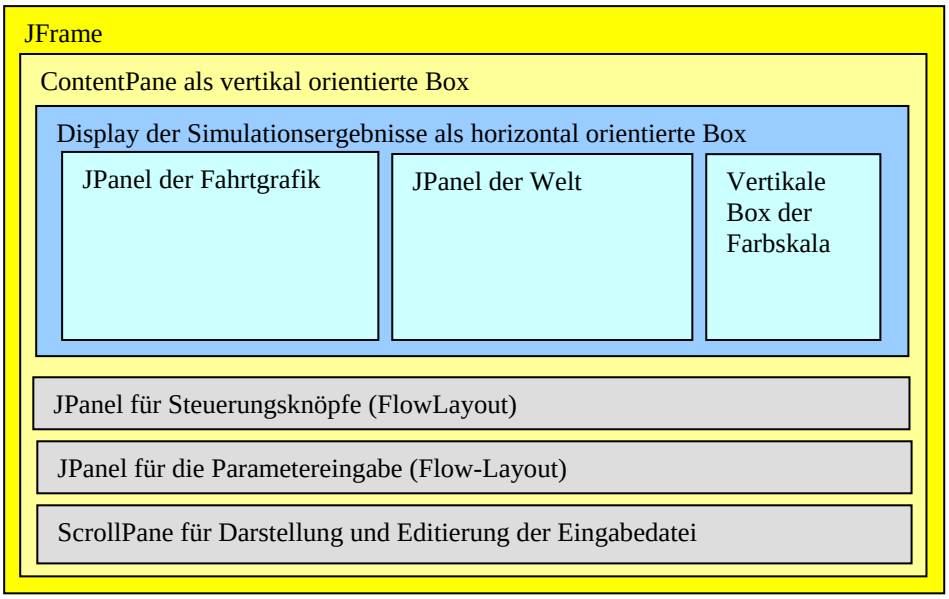
### *Interaktive Steuerung*

Nach jedem Lauf aus *h* Spielzügen wird die Simulation automatisch angehalten. In diesem Moment wird die Grafik aktualisiert und es besteht die Möglichkeit, Parameter neu zu setzen und den nächsten Lauf zu starten.

### *Bedienoberfläche*

Anders als bei KoopEgo wird das Steuerpult nicht separat angeordnet. Es gibt für die grafische Darstellung nur einen Frame und ansonsten nur Dialog-Elemente. Die Strukturierung des Layouts geschieht mittels Box-Klassen und BoxLayout. Boxen erlauben ein gut definiertes Layout und sie haben den Vorteil, dass sie sich bei Änderung der Frame-Dimensionen (z. B. beim Wechsel von der Normal- zur Vollbilddarstellung) automatisch an die eingelagerten Elemente vom Typ JPanel oder Box anpassen.

Bei Änderung der Frame Dimensionen wird eine Anpassung des Maßstabs für die Modelldarstellung vorgenommen: Bei Vergrößerung des Frames werden die Elemente der Welt und der Fahrtgrafik entsprechend größer dargestellt.



## Realisierung

Der Programmtext ist im Java-Archiv FahrweisenOptimierung.jar enthalten[1]. Hier sollen nur einige der besonders erläuterungsbedürftigen Abschnitte besprochen werden.

### *Programmsteuerung (Ein- und Ausgabe)*

Die Eingabeparameter und die Daten zur Strecke und Fahrdynamik werden nach Drücken des Eingabe-Knopfs aus einer Textdatei in ein Textfenster geladen. Der Text ist editierbar.

Nach Drücken des Start-Knopfs wird der Text des Textfensters in die Eingabedatei zurückgespeichert. Die Welt wird initialisiert und mit Individuen vorbelegt.

Drücken des Stop-and-Go-Knopfs startet die Simulation oder setzt sie fort. Je Spielzug wird ein neuer Bewohner der Welt (Individuum, Fahrt) erzeugt. Nach einem Lauf von $h$ Spielzügen wird die Simulation angehalten und die grafische Darstellung des Zustands der Welt aktualisiert.

Der Weiter-Knopf startet eine Folge von Läufen, die sich mit dem Halt-Knopf unterbrechen lässt.

Der Ende-Knopf beendet die Simulation und schließt den Programmlauf.

Immer wenn die Simulation steht, können gewisse Parameter in den dafür vorgesehenen Textfenstern geändert werden. Eine Änderung wird nur wirksam, wenn sie mit dem Return-Knopf abgesendet wird.

Jeder Farbpunkt repräsentiert einen Bewohner der Welt (eine Fahrt). Durch Mausklick auf einen solchen Farbpunkt wird ein Fenster mit den Informationen über die jeweilige Fahrt geöffnet. Gleichzeitig wird das zugehörige Weg-Geschwindigkeits-Energie-Diagramm im Fenster für die Fahrtgrafik angezeigt.

Drücken des „Speichern“-Knopfes bewirkt, dass die aktuellen Eingabeparameter, Steuerungsdaten und die Fahrtdaten des zuletzt angeklickten Individuums (Musterfahrt) abgespeichert werden. Die Tabelle des zugehörigen Fahrtverlaufs wird als Kommentar hinten angehängt. Die so erzeugte Datei kann mit einem Tabellenkalkulationsblatt weiter verarbeitet und grafisch aufbereitetet werden. Da die Abspeicherung in Eingabeformat geschieht, ist eine solche Datei als Eingabedatei für weitere Optimierungsläufe geeignet.

### *Die Individuen*

Die Individuen sind Objekte der Run-Klasse. Die Fahrt eines Run-Objekts wird durch eine lineare Liste von Fahrtabschnitten definiert. Ein Fahrtabschnitt ist eine Teilstrecke mit konstanter Fahrstufe. Die Fahrtabschnitte werden als Objekte der Section-Klasse realisiert. Ein Section-Objekt c hat zwei Attribute zur Festlegung der Fahrt: c.x markiert den Endpunkt der Teilstrecke und c.level ist die Fahrstufe für diese Teilstrecke. Außerdem kann über die next()-Methode auf das in der linearen Liste folgende Objekt zugegriffen werden.

### *Die run-Methode der Run-Klasse: Spielzüge*

Je Spielzug wird in der Welt ein Feld (x0, y0) – das aktuelle Zentrum – zufällig ausgewählt. Dann wird mit der Wahrscheinlichkeit $q$ eine Crossing-Over-Operation ausgeführt. Dazu werden aus der $k$-Umgebung des Zentrums zwei Eltern rein zufällig ausgewählt. Diese Individuen können mit dem im Zentrum aber auch miteinander identisch sein. Mittels Crossing-

[1] Es ist eines der Programme zum Thema Problemlösen und im Internet frei erhältlich.

Over-Operation wird die Fahrt für ein neues Individuum erzeugt. Alternativ zum Crossing-Over wird ein Klon eines Individuums der Umgebung gebildet. Auf das neue Individuum wird der Mutationsoperator angewendet.

```
//Mit run() wird die Simulation gestartet und wieder aufgenommen.
static public void run() {
  for (int i=0; i<h; i++) synchronized(FahrweisenOptimierung.frame) {
    int xy = rand.nextInt(nn), x0 = xy/n, y0 = xy%n;
    Run newRun;
    if(rand.nextFloat()<q) newRun= neighbour(x0, y0).combine(neighbour(x0, y0));
    else newRun= new Run(neighbour(x0, y0).run.clone());
    newRun.mutate();
    newRun.evaluate(false);
    if (rand.nextFloat()<1/(1+Math.exp((newRun.f()-world[x0][y0].f())/T)))
    world[x0][y0]= newRun;
  }
  redraw();
}
```

Die Evaluierungsfunktion wird unmittelbar nach der Erzeugung des Individuums (Fahrt, Run) aufgerufen. Über den booleschen Parameterwert false wird hier bestimmt, dass die Fahrt nicht als Muster (sample) behandelt wird. Ein Aufruf evaluate(true) behandelt den Lauf als Muster: Es entsteht ein Abbild der gesamten Fahrdynamik in einer Textversion und als Zustandsfolge. Die Fahrtgrafik entsteht aus diesen Daten.

Nach Maßgabe des Simulated Annealing wird das aktuelle Muster im Zentrum durch das neue Individuum ersetzt:

```
    if (rand.nextFloat()<1/(1+Math.exp((newRun.f()-world[x0][y0].f())/T)))
    world[x0][y0]= newRun;
```

## Abnahme – Testdurchführung

### *Experiment 0 – Strecke mit Geschwindigkeitsbeschränkungen*

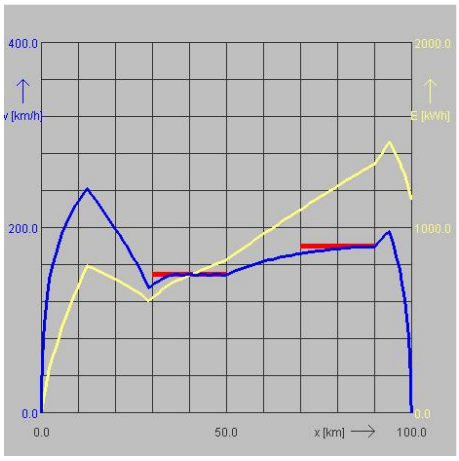
#### Die Eingabedaten

Fett hervorgehoben sind die für die Steuerung der Optimierung wichtigen Parameter. Die anderen Parameter werden während der gesamten Optimierung konstant gelassen

```
n  40 /*Spielfeldgroesse*/
k  1 /*Umgebungsparameter*/
h  10000 /*Anzahl der Spielzuege je Lauf*/
p  0.2 /*Mutationswahrscheinlichkeit*/
q  0.5 /*Crossing-Over-Wahrscheinlichkeit*/
T  1000.0 /*Temperatur*/
kS  1.0 /*Gewichtsfaktor fuer Anzahl der Teilstrecken in min*/
kE  0.0 /*Gewichtsfaktor fuer den Energieverbrauch in min/MWh*/
kv  1.0 /*Gewichtsfaktor fuer Geschwindigkeitsueberscheitungen in min*/
vT  10.0 /*Toleranz fuer Geschwindigkeitsueberscheitungen in km/h*/
System
Route 100.0, 30.0 50.0 150.0, 70.0 90.0 180.0;
/*Streckenlaenge und Restriktionen. Zahlenangaben in km bzw. km/h*/
FzMax  400.0; /*Maximale Zugkraft in kN*/
FbMax  300.0; /*Maximale Bremskraft in kN*/
abc
  6.0 /*Einheit: kN*/
  0.1 /*Einheit: kN/(m/s)*/
  0.01; /*Einheit: kN/(m/s)^2*/
vMax  280.0; /*Hoechstgeschwindigkeit in km/h*/
PzMax  10000.0; /*Maximale Antriebsleistung in kW*/
PbMax  8000.0; /*Maximale Bremsleistung in kW*/
zEta  0.9; /*Antriebswirkungsgrad*/
bEta  0.9; /*Bremswirkungsgrad*/
nLevel  10; /*Anzahl der Fahr- bzw. Bremsstufen*/
Mass  925410.0; /*Wirksame Masse in kg*/
tMax  200.0; /*Obergrenze der Fahrtdauer in min*/
RK; /*Integration mit einem Runge-Kutta-Verfahren*/
rho 0.1; /*Feinheit der Integration (Genauigkeit)*/
```

### Phase 1: Aus Zufallsfahrten entsteht der erste Lösungsvorschlag

Jeweils nach nur wenigen Läufen mit je 10000 Spielzügen werden die Parameter für die Temperatur (T) und die Geschwindigkeitstoleranz (vT) reduziert. Schließlich ist T=0 und vT=1. Damit wird eine passable Lösung herausdestilliert. Es deutet sich die spitze Fahrweise an. Für diese Lösung waren 1 Mio. Spielzüge erforderlich. Die Fahrtgrafik ist nebenan zu sehen.

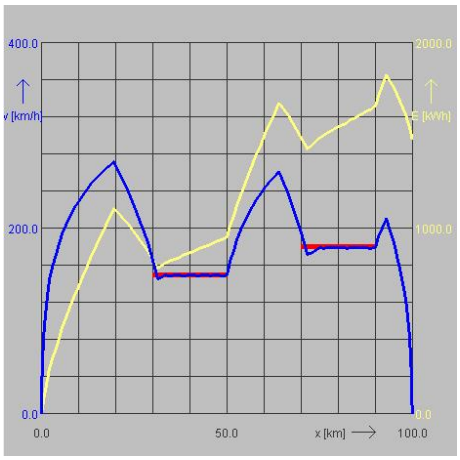



### Phase 2: Verfeinerung

Jetzt wird mittels kS=0 die Aufteilung von Fahrtabschnitten in Teilstrecken gefördert. Die Geschwindigkeitstoleranz bleibt großzügig bei vT=5 km/h. Die Temperatur wird überwiegend bei T=0 gehalten. Im Laufe von über 5 bis 6 Mio. Spielzügen entstehen spitze Fahrten mit etwa 20 Fahrtabschnitten.

### Phase 3: Teilstreckenreduktion

Der Parameter für die Bewertung der Anzahl von Teilstrecken wird heraufgesetzt: kS=0.5. (Ein Wert von kS=1.0 hat sich als zu hoch erwiesen: Hoffnungsvolle Ansätze von spitzen Fahrten wurden vernichtet.) Die Geschwindigkeitstoleranz wird auf 1 km/h reduziert. Nach weniger als 1 Mio. Spielzügen ergibt sich die nebenstehende Fahrt mit einer Fahrtzeit von 34 Minuten. Die acht Fahrtabschnitte:



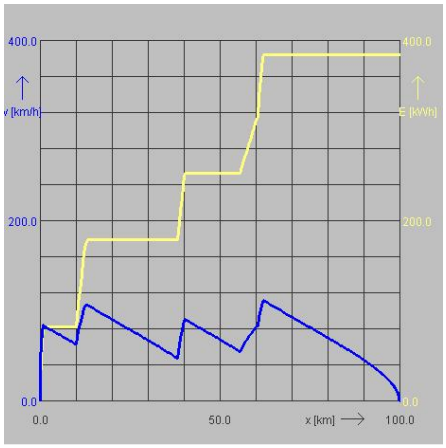

```
19.604391 10, 31.436005 -8, 50.220806 2,
64.19245 10, 71.72621 -10, 90.09105 3,
93.125755 10, 100.0 -10;
```

### *Experiment 1 – Straffe Fahrt*

Mit deutlich weniger Spielzügen als beim Experiment 0 wird die prognostizierte straffe Fahrweise für eine Strecke ohne Geschwindigkeitsbeschränkungen gefunden. Die Fahrtabschnitte: `86.377556 10, 86.90253 -10, 100.0 -10;`

### *Experiment 2 – Energiesparende Fahrt*

Da das Fahren mit fest eingestellter Geschwindigkeit (Cruisen) nicht zu den Optionen der Fahrtauswahl gehört, kommt es hier zu Lösungen mit Ausrollen (Coasten). Es entstehen mehr oder weniger fein gezackte Geschwindigkeitsdiagramme, je nachdem, wie viel Geduld man bei der Lösungsfindung aufbringt. Ein typisches Ergebnis nach Durchlaufen einer ausgiebigen Verfeinerungsphase ist das Nebenstehende. Die Fahrtzeit dauert 100 Minuten. Die mittlere Geschwindigkeit liegt also bei 60 km/h und ist damit etwas höher als prognostiziert.




*Anmerkung:* Auch das Cruisen lässt sich nachbilden: Dazu führt man Geschwindigkeitsbeschränkungen ein.

## Anhang: Numerische Integration von Differentialgleichungen

Die Zustandsraumdarstellung in Normalform nimmt bei kontinuierlichen Systemen die folgende Gestalt an:

$$\mathrm{d}\boldsymbol{z}(t)/\mathrm{d}t = \boldsymbol{f}(\boldsymbol{z}(t), \boldsymbol{x}(t), t)$$

$$\boldsymbol{y}(t) = \boldsymbol{g}(\boldsymbol{z}(t), \boldsymbol{x}(t), t)$$

Dabei ist $\boldsymbol{x}(t)$ der $m$-dimensionale zeitabhängige Vektor der Eingangsfunktionen, $\boldsymbol{y}(t)$ der $n$-dimensionale Vektor der Ausgangsfunktionen und $\boldsymbol{z}(t)$ der $q$-dimensionale Zustandsvektor. Die Ableitung eines Vektors ist der Vektor der abgeleiteten Komponenten: $\mathrm{d}\boldsymbol{z}(t)/\mathrm{d}t = (\mathrm{d}z_1(t)/\mathrm{d}t, \mathrm{d}z_2(t)/\mathrm{d}t, ..., \mathrm{d}z_q(t)/\mathrm{d}t)$.

Die Funktionen $\boldsymbol{f}$ und $\boldsymbol{g}$ heißen *Übergangsfunktion* und *Ausgabefunktion* .

Für jeden Simulationslauf ist der Zeitverlauf des Vektors der Eingangsgröße $\boldsymbol{x}(t)$ fest vorgegeben. Die Eingangsgröße vermittelt nur einen weiteren Einfluss der Zeit auf die Übergangsfunktion. Für den Fall fest gegebener Eingangsgrößen kann man deshalb die Übergangsbeziehung in Form der folgenden *Differentialgleichung* angeben:

$$\mathrm{d}\boldsymbol{z}(t)/\mathrm{d}t = \boldsymbol{f}(\boldsymbol{z}(t), t)$$

Außerdem sei der Anfangsvektor der Zustandsgrößen gegeben

$$\boldsymbol{z}(0) = \boldsymbol{z}_0$$

Die Aufgabe, die Zeitfunktion $\boldsymbol{z}(t)$ so zu bestimmen, dass sie der Differentialgleichung und der Anfangsbedingung genügt, wird auch *Anfangswertproblem* genannt.

Für die Lösung des Anfangswertproblems mittels Computer sind die Systemgleichungen zu diskretisieren. Bei Zeitschritten konstanter Schrittweite $h$ werden die Zeitpunkte $t_i$ - ausgehend vom Startzeitpunkt $t_0$ - folgendermaßen rekursiv bestimmt:

$$t_{i+1} = t_i + h$$

Die Diskretisierung der Systemgleichungen lässt sich grundsätzlich so durchführen, dass man linke und rechte Seite die Übergangsbeziehung jeweils über ein Intervall von $t_i$ bis $t_{i+1}$ integriert und auf die rechte Seite der Gleichung die Rechteckregel, die Trapezregel oder irgendeine andere Formel der numerischen Integration anwendet[1].

Die numerischen Verfahren zur Integration der Übergangsbeziehung (Differentialgleichung) gehen für den Fall, dass $\boldsymbol{f}$ eindimensional ist und nicht vom Zustandsvektor abhängt in die bekannten Integrationsformeln für reelle Funktionen über. Genauigkeitsabschätzungen für diese Integrationsformeln waren Gegenstand des vorhergehenden Unterabschnitts. Diese Fehlerabschätzungen geben auch Anhaltspunkte für die Genauigkeit der Integration im Falle der Differentialgleichung $\mathrm{d}\boldsymbol{z}(t)/\mathrm{d}t = \boldsymbol{f}(\boldsymbol{z}(t), t)$. Deshalb wird im Folgenden immer darauf hingewiesen, zu welcher Integrationsformel das Verfahren zur Integration von Differentialgleichungen entartet.

Die Rechteckregel - angewandt auf jede Komponente der Übergangsbeziehung - liefert das sogenannte *Euler-Cauchysche Polygonzugverfahren*:

$$\boldsymbol{z}_{i+1} = \boldsymbol{z}_i + h\,\boldsymbol{f}(\boldsymbol{z}_i, t_i)$$

---

[1] Auch hierzu geben die meisten Bücher der numerischen Mathematik Auskunft: Stoer/Bulirsch (1990), und Issacson/Keller (1973) beispielsweise.

Es handelt sich um Rekursionsformeln, die sich mit dem Digitalrechner direkt lösen lassen. Wenn die Übergangsfunktion nicht vom Zustand $\boldsymbol{z}$ abhängt und eindimensional ist, ergibt sich wieder die ursprüngliche Rechteckregel für die Integration reeler Funktionen.

Ein wesentlich genauerer und stabilerer Algorithmus ergibt sich bei Anwendung der Trapezregel:

$$\boldsymbol{z}_{i+1} = \boldsymbol{z}_i + h\,(\boldsymbol{f}(\boldsymbol{z}_i, t_i) + \boldsymbol{f}(\boldsymbol{z}_{i+1}, t_{i+1}))/2$$

Wenn sich der obige Ausdruck nach $\boldsymbol{z}_{i+1}$ auflösen lässt, erhält man wieder eine Rekursionsformel für die $\boldsymbol{z}_i$. Falls sich der auf einen Zustand $\boldsymbol{z}_i$ folgende Zustand $\boldsymbol{z}_{i+1}$ nicht mehr explizit angeben lässt, liegt ein *implizites Integrationsverfahren* vor.

Die Schwierigkeiten mit dem impliziten Verfahren lassen sich durch eine geringfügige Modifikation umgehen: Für den Zustandsvektor $\boldsymbol{z}_{i+1}$, der auf der rechten Seite vorkommt, liefert das Euler-Cauchy-Verfahren einen Näherungswert: $\boldsymbol{z}_{i+1}^{\star} = \boldsymbol{z}_i + h\,\boldsymbol{f}(\boldsymbol{z}_i, t_i)$. Dieser tritt auf der rechten Seite der Gleichung an die Stelle des $\boldsymbol{z}_{i+1}$.

So ergibt sich das *Verfahren von Heun*. Wir bringen es noch in eine für die Programmierung günstige Form: Der neue Wert des Zustandsvektors $\boldsymbol{z}_{i+1}$ wird über die Hilfsvektoren $\boldsymbol{k}_1$ und $\boldsymbol{k}_2$ ermittelt, die so definiert sind:

$$\boldsymbol{k}_1 = \boldsymbol{f}(\boldsymbol{z}_i, t_\mathrm{i})$$

$$\boldsymbol{k}_2 = \boldsymbol{f}(\boldsymbol{z}_i + h\,\boldsymbol{k}_1, t_i{+}h)$$

Der neue Wert des Zustandsvektors ergibt sich damit zu

$$\boldsymbol{z}_{i+1} = \boldsymbol{z}_i + h\,(\boldsymbol{k}_1 + \boldsymbol{k}_2)/2$$

Offensichtlich soll der Teilausdruck $(\boldsymbol{k}_1{+}\boldsymbol{k}_2)/2$ eine Schätzung des Mittelwerts von $\boldsymbol{f}$ im betrachteten Zeitschritt liefern. Für den Fall, dass $\boldsymbol{f}$ nicht vom Zustandsvektor $\boldsymbol{z}$ abhängt, geht das Verfahren von Heun in die Trapezregel zur Integration reeller Funktionen über.

Beim *Verfahren von Runge-Kutta* wird der neue Wert des Zustandsvektors $\boldsymbol{z}_{i+1}$ über die Hilfsvektoren $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, $\boldsymbol{k}_3$ und $\boldsymbol{k}_4$ ermittelt. Diese sind folgendermaßen definiert:

$$\boldsymbol{k}_1 = \boldsymbol{f}(\boldsymbol{z}_i, t_i)$$

$$\boldsymbol{k}_2 = \boldsymbol{f}(\boldsymbol{z}_i{+}h/2\,\boldsymbol{k}_1, t_i{+}\mathrm{h}/2)$$

$$\boldsymbol{k}_3 = \boldsymbol{f}(\boldsymbol{z}_i{+}h/2\,\boldsymbol{k}_2, t_i{+}\mathrm{h}/2)$$

$$\boldsymbol{k}_4 = \boldsymbol{f}(\boldsymbol{z}_i{+}h\,\boldsymbol{k}_3, t_i{+}\mathrm{h})$$

Der neue Wert des Zustandsvektors ergibt sich damit zu

$$\boldsymbol{z}_{i+1} = \boldsymbol{z}_i + h\,(\boldsymbol{k}_1{+}2\boldsymbol{k}_2{+}2\boldsymbol{k}_3{+}\boldsymbol{k}_4)/6$$

Hier ist der Teilausdruck $(\boldsymbol{k}_1{+}2\boldsymbol{k}_2{+}2\boldsymbol{k}_3{+}\boldsymbol{k}_4)/6$ wiederum eine Schätzung des Mittelwerts von $\boldsymbol{f}$ im betrachteten Zeitschritt. Für den Fall, dass $\boldsymbol{f}$ nicht vom Zustandsvektor $\boldsymbol{z}$ abhängt, geht das Verfahren von Runge-Kutta in die Simpsonsche Regel zur Integration reeller Funktionen über.

Die Verfahren von Euler-Cauchy, Heun und Runge-Kutta sind hier nach wachsender Genauigkeit geordnet. Andersherum: Bei gleicher geforderter Genauigkeit dürfen beim Verfahren von Heun im Allgemeinen größere Schrittweiten gewählt werden als beim Verfahren von Euler-Cauchy. Und das Verfahren von Runge-Kutta kommt mit noch weniger Schritten bei noch größerer Schrittweite aus.

Die Erhöhung der Genauigkeit wird mit einer Erhöhung des Aufwands erkauft: Beim Euler-Cauchy-Verfahren genügt je Schritt eine Auswertung der Übergangsfunktion $\boldsymbol{f}$. Das Verfahren von Heun erfordert bereits zwei Auswertungen, und beim Runge-Kutta-Verfahren sind es gar vier.

Als *lokalen Integrationsfehler* bezeichnet man den Fehler, den man bei der numerischen Integration über einen Schritt macht, wobei vorausgesetzt wird, dass die Integration bis dahin fehlerfrei war. Er lässt sich mit denselben Formeln, die wir zur Fehlerabschätzungen des Integrationsfehlers herangezogen haben, abschätzen. Lokale Fehlerabschätzungen können zur Steuerung einer *automatischen Schrittweitenanpassung* herangezogen werden. Die Einzelheiten dazu und Anleitungen zu weitergehenden Fehler- und Stabilitätsuntersuchungen findet man in der Literatur zur numerischen Mathematik.

## Literatur


Culwin, Fintan: A Java GUI Programmer's Primer. Prentice Hall, Upper Saddle River, New Jersey 1998

Doberenz, Walter, Druckermüller, Uwe: Java. Programmierung interaktiver WWW-Seiten. Hanser, München, Wien 1998

Fischer, Paul: An Introduction to Graphical User Interfaces with Java Swing. Addison-Wesley 2005

Grams, Timm: KoopEgo. Links zur Programmdokumentation und zum Programmtext (2007)
http://www.hs-fulda.de/~grams/OekoSimSpiele/KoopEgoProgramm/KoopEgo.pdf
http://www.hs-fulda.de/~grams/OekoSimSpiele/KoopEgoProgramm/KoopEgo.jar

Grams, Timm: PageRank. Link zur Programmdokumentation (2007)
http://www.hs-fulda.de/~grams/Informatik/PageRank.jar

Holland, John H.: Genetische Algorithmen. Spektrum der Wissenschaft (1992) 9, 44-51

Isaacson, E.; Keller, H. B.: Analyse numerischer Verfahren. Harri Deutsch Frankfurt/M., Zürich 1973

Li, Liwu: Java. Data Structures and Programming. Springer, Berlin, Heidelberg 1998

Lindner, Ulrich: Optimierung von Fahrweisen im spurgeführten Verkehr und deren Umsetzung. Dissertation, genehmigt von der Fakultät V (Verkehrs- und Maschinensysteme) der Technischen Universität Berlin 2004

Michalewicz, Zbigniew; Fogel, David B.: How to Solve It: Modern Heuristics. Springer-Verlag, Berlin, Heidelberg 2000

Rechenberg, Ingo: Evolutionsstrategie. Optimierung technischer System nach den Prinzipien der biologischen Evolution. Friedrich Frommann Verlag, Stuttgart-Bad Cannstatt 1973

Stoer, J.: Numerische Mathematik 1. Springer, Berlin, Heidelberg 1994

Stoer, J.; Bulirsch, R.: Numerische Mathematik 2. Springer-Verlag, Berlin, Heidelberg 1990