IOMath

Was ist IOMath?

IOMath ist konzipiert als *kleines*, unkompliziertes Werkzeug, um Graphen und andere Plots mathematischer Funktionen/Kurven etc. zu erzeugen, die dann beispielsweise im Schulalltag verwendet werden können.

Was kann IOMath, was kann es nicht?

IOMath ist *nicht* geeignet zur Messwertvisualisierung (dafür gibt es Excel, Calc, Origin, CoPlot etc.). IOMath kann lediglich Graphen zeichnen, die sich aus mathematischen Ausdrücken ergeben.

IOMath ist *nicht* in der Lage mit komplexen Zahlen oder mehrdimensional zu rechnen. Es können lediglich reellwertige Funktionen verarbeitet werden.

IOMath kann *nicht* algebraisch korrekt rechnen. Werte wie „Wurzel aus zwei“ werden als Fließkommazahlen genähert. IOMath arbeitet intern mit *double*-Variablen (ca. 15 Nachkommastellen)

IOMath ist *nicht* so konzipiert, dass der Benutzer das Aussehen der Funktionsgraphen *völlig* frei nach seinen Wünschen verändern kann. Es gibt jedoch etliche Einstellungsmöglichkeiten und Farbschemata, so dass nicht all zu viele Wünsche offen bleiben sollten.

IOMath ist in der Lage in hoher Auflösung (mehrere Megapixel) Graphen zu erzeugen und zu speichern, die beispielsweise auch für großformatigen Druck geeignet sind.

IOMath kann praktisch beliebig komplizierte Funktionen verarbeiten. Auch zusammengesetzte Funktionen sind möglich.

IOMath kann beliebig viele Funktionen in ein Koordinatensystem zeichnen. So lassen sich prinzipiell auch ganze Funktionsscharen visualisieren, oder beispielsweise Funktion und Ableitung in *einer* Grafik betrachten.

IOMath beinhaltet einen „Rechner“, mit dem Sie umfangreiche Ausdrücke wirtschaftlicher Auswerten können, als mit einem Taschenrechner, sowie die Möglichkeit eine Wertetabelle einer Funktion auszugeben.

IOMath beinhaltet zudem die Möglichkeit, Steigungs- bzw. Vektorfelder in der Ebene zu skizzieren, sowie parametrisierte Kurven in der Ebene und zweidimensionale Skalarfelder.

Welche Funktionen und Befehle kennt IOMath?

- +, -, *, /: Addition, Subtraktion, Multiplikation, Division. Bsp: „5+2*x“
- ^: Potenz. Bsp: „x^3“
- %: Modulo, d.h. Rest der Division. Bsp: „x%1“
- pi, e: Kreiszahl π und Eulersche Zahl *e*. Bsp: „e^x“

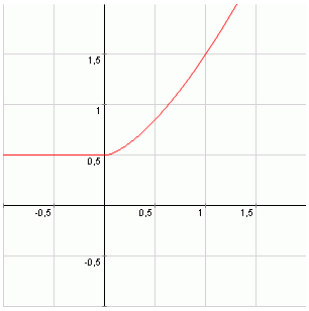
- sin(), cos(), tan(), asin(), acos(), atan(): Die bekannten trigonometrischen Funktionen und ihre Umkehrfunktionen. Bsp: „cos(2*pi*x)“
- sinh(), cosh(), asinh(), acosh(): Die hyperbolisch-trigonometrischen Funktionen und ihre Umkehrfunktionen. Bsp: „cosh(x)“
- exp(), log(): Die Exponentialfunktion und der Logarithmus. Bsp: „log(sin(x)+2)“
- sqrt(): Die Quadratwurzel. Bsp: „sqrt(x)“
- abs(): Der Absolutwert, d.h. der Betrag des Arguments. Bsp: „log(abs(x)+1)“
- sgn(): Das Vorzeichen des Arguments. Bsp: „sgn(sin(x))“
- int(): Der Integerwert, genauer: die größte ganze Zahl unterhalb des angegebenen Arguments. Bsp: „int(x^2)“
- !: Fakultät: Bitte bedenken Sie, dass diese Funktion auf den Wert *vor* dem Ausrufezeichen wirkt und nur dann sinnvolle Ergebnisse liefert, wenn dieser Wert eine natürliche Zahl ist. Bsp: „1/(3!)“
- ?(): Bedingung: Nimmt den Wert 1 an, wenn die Bedingung in der Klammer erfüllt ist. Nimmt den Wert 0 an, wenn die Bedingung nicht erfüllt ist. Mögliche Bedingungen sind beispielsweise „?(x>1)“ oder „?(abs(x)>=1)“ oder „?(x=0)“ oder „?(0<sin(x))“ usw…

Wie funktioniert der Funktionsplotter von IOMath?

Der zu plottende Funktionsausdruck muss in Abhängigkeit von *x* geschrieben werden, wobei *x* die an der horizontalen Achse abgetragene Variable ist. Beispiel: für die Funktion $f\colon \mathbb{R} \to \mathbb{R}, x \mapsto (\sin x)^2$ schreibt man den Ausdruck „sin(x)^2“.

Wie realisiere ich eine bestimmte Funktion?

Zusammengesetzte Funktionen



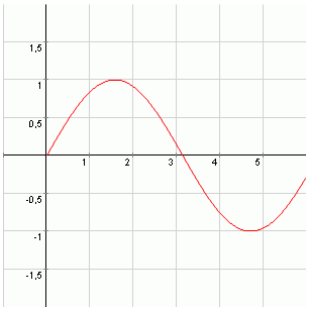
Zusammengesetzte Funktionen lassen sich mit *Bedingungen* realisieren.

Beispiel: $f(x) = \begin{cases} \frac{1}{2}, & x < 0 \\ \sqrt{x^3} - 0.5, & x \geq 0 \end{cases}$

Ausdruck: „?(x<0)*1/2+?(x>=0)*(sqrt(x^3)+ 0.5)“

Funktionen mit eingeschränktem Definitionsbereich

Ist der Funktionsausdruck von Vornherein nicht für alle reellen Zahlen definiert, so wird der Funktionsgraph auch nur in diesem Bereich gezeichnet. Beispielsweise wird für den Ausdruck „sqrt(x)“ oder „log(x)“ im negativen Bereich automatisch nichts gezeichnet.



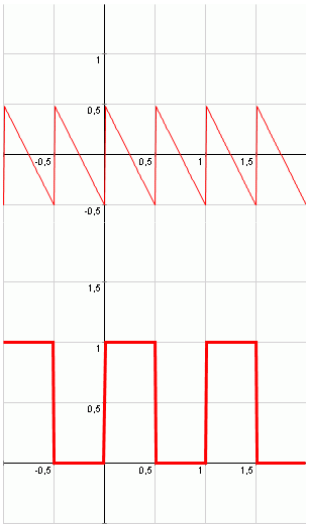
Ist der Ausdruck an sich überall definiert, soll aber nicht überall gezeichnet werden, muss man zu einem kleinen Trick greifen:

Beispiel: $f\colon [0;\infty[\to \mathbb{R}, x \mapsto \sin x$

Ausdruck: „sin(x)+?(x<0)*1/0“

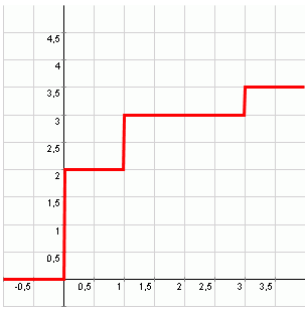
Im negativen Bereich wird der nicht definierte Ausdruck „1/0“ ausgewertet, was dazu führt, dass hier keine Funktionswerte gezeichnet werden.

Sägezahn, Rechteck und Konsorten

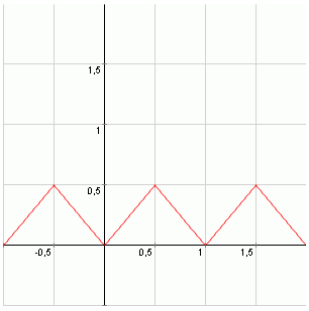


Eine schöne Sägezahnfunktion erreicht man mit dem *Modulo*-Operator: „x%1“ ergibt eine Sägezahnfunktion mit ansteigender Flanke zwischen y=0 und y=1. Streckungen, Verschiebungen etc. lassen sich einfach erreichen. Beispielsweise ergibt „(-2*x)%1 – 0,5“ eine Sägezahnfunktion mit steilerer fallender Flanke zwischen y=-0,5 und y=+0,5.

Eine Rechtecksfunktion lässt sich über die *Signum*funktion erzeugen. Der Ausdruck „sgn(sin(x))“ ergibt eine Rechtecksfunktion mit Periode 2π zwischen y=-1 und y=+1. Auch hier sind Skalierungen und Verschiebungen natürlich leicht zu erreichen. Beispielsweise „0,5*sgn(sin(2*pi*x))+0,5“ stellt eine Rechtecksfunktion mit Periode 1 zwischen y=0 und y=1 dar.



Eine Darstellung einer Treppenfunktion kann man beispielsweise mit der *Integer*funktion erreichen: „int(x)“ ist eine Treppenfunktion die bei jedem ganzzahligen x um 1 nach oben springt. Auch diese kann man natürlich leicht an die persönlichen Wünsche anpassen. Beispielsweise springt „-0,5*int(0,1*x)“ bei x=0, 10, 20, etc. um je 0,5 nach unten. Kompliziertere Treppenfunktionen mit variabler Schrittweite müssen allerdings von Hand über *Bedingungen* erzeugt werden. Beispiel: „?(x>0)*?(x<1)*2 + ?(x>=1)*?(x<3)*3 + ?(x>=3)*3,5“



Eine Dreiecksfunktion zu erzeugen ist nicht ganz so einfach. Mit einem trickreichen Ausdruck ist aber auch das möglich: „?(x-int(x)<0,5)*x%1 + ?(x-int(x)>=0,5)*(-x)%1“ ergibt eine Dreiecksfunktion zwischen y=0 und y=0.5 mit Periode 1.

Wie kann ich möglichst effizient mit dem Funktionsplotter arbeiten?

Mehrere Funktionen in ein Koordinatensystem zeichnen

Möchten Sie mehrere Funktionen ins selbe Koordinatensystem zeichnen lassen, trennen Sie diese im Eingabefeld einfach durch Semikolons. Bsp: „x^2;x^3;x^4“ plottet vier Potenzfunktionen in ein Koordinatensystem. Die Gestalt der einzelnen Graphen wird von dem „Stil“-Auswahlfeld und der eingestellten Linienstärke beeinflusst. Möchten Sie einen bestimmten Graphenstil, so können Sie dies durch Einfügen geeigneter „leerer Funktionen“ erreichen. Bsp: „;;;x^2“ sorgt dafür, dass die Normalparabel mit dem vierten Graphenstil gezeichnet wird.

Variablen definieren

Im Definitionsfeld von IOMath können Sie beliebig viele Variable definieren, mit denen Sie die gewünschten Funktionen effizient erzeugen können.

Bsp: Definition „a=2*pi“, Funktion „sin(a*x)“.

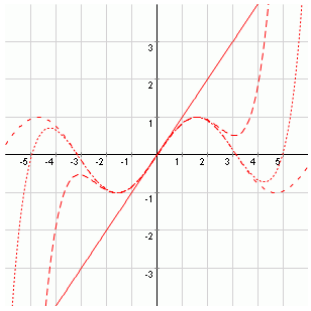
Die von Ihnen gewählten Variablennamen können mehrbuchstabig sein und nach dem ersten Buchstaben auch Ziffern und manche Sonderzeichen enthalten (Bsp: „vari_1=5“). Groß-Klein-Schreibung wird *nicht* ignoriert, d.h. sie können bspw. *unterschiedliche* Variablen „A“ und „a“ definieren. Mehrere Definitionen können durch Semikolons (=Strichpunkte) getrennt oder einfach in verschiedene Zeilen geschrieben werden (Bsp: „a=1; b=2“). Die Definitionen werden von vorne nach

hinten abgearbeitet und können in dieser Reihenfolge auch aufeinander Bezug nehmen. Die Variable „x“ steht schon für die erste Definition zur Verfügung.

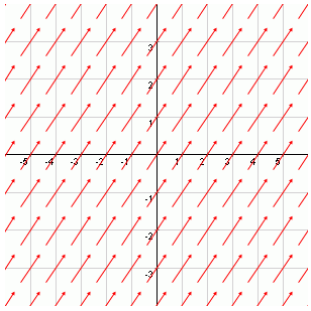
Bsp: Definition „a=2*pi; schwingung=sin(a*x)“, Funktion „schwingung^2“

Dadurch ist es beispielsweise möglich, die Übersicht zu bewahren, wenn mehrere Funktionen ineinander eingesetzt werden sollen, oder wenn kompliziertere Ausdrücke nötig sind.

Bsp: Um die Partialsummen-Funktionen einer durch eine Potenzreihe dargestellten Funktion zu skizzieren, kann man wie folgt vorgehen:
Definitionen
„f0=1/1*x^1
f1=f0 + (-1)/3!*x^3
f2=f1 + 1/5!*x^5
f3=f2+ (-1)/7!*x^7
f4=f3 + 1/9!*x^9“
Funktionen „f0;f2;f4;sin(x)“

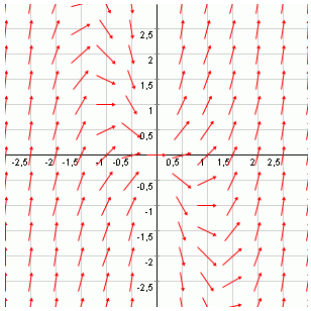


Wie funktionieren Steigungs-/Vektorfelder mit IOMath?



Vektorkomponenten

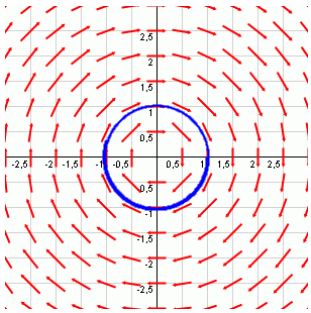
Um ein zweidimensionales Vektorfeld zu zeichnen, geben Sie die x-Komponente und die y-Komponente getrennt durch Semikolon an. Beide Komponenten können auf sämtliche definierten Variablen zugreifen, insbesondere natürlich auf „x“ und „y“. Bsp: „1; 1“ ist ein Vektorfeld, das ortsunabhängig überall in die selbe Richtung weist. Bsp: „y; -x“ ergibt ein Rotationsfeld.



Steigungsfelder

Um das zu einer Differentialgleichung $\frac{dy}{dx} = f(x,y)$ gehörige Steigungsfeld zu skizzieren, geben Sie als x-Komponente des Felds einfach „1“ an, und als y-Komponente den Ausdruck der Funktion f. Bsp: „1; y/x+x^2“ zeichnet das Steigungsfeld der Differentialgleichung $y' = \frac{y}{x} + x^2$.

Trajektorien



Nach den zwei Ausdrücken für x- und y-Komponente des Feldes können Sie durch weitere Semikolons getrennt Koordinaten von Startpunkten angeben, von denen ausgehend IOMath numerisch Trajektorien des Feldes einzeichnet. Bsp: „y; -x; 1; 0“ zeichnet ein Rotationsfeld und eine Trajektorie mit dem Startpunkt $(1;\ 0)$.

Einstellungsmöglichkeiten

Pfeilraster: gibt an, welchen Abstand benachbarte Pfeile des Vektorfeldes haben sollen. Je kleiner der Abstand gewählt wird, umso dichter stehen die Pfeile. Je nach Feld sind unterschiedliche Abstandseinstellungen „gefälliger“ fürs Auge.

Streckfaktor: ermöglicht es, alle Pfeile simultan um diesen Faktor zu verlägern/verkürzen. Das ist insbesondere im Modus „Geschwindigkeiten“ praktisch.

Trajektorien-Schrittweite: Je kleiner dieser Wert gewählt wird, umso präziser folgt eine eingezeichnete Trajektorie tatsächlich dem Feldverlauf. Sehr kleine Werte verursachen allerdings unter Umständen lange Rechenzeit.

Trajektorien-Länge: gibt an, wie weit eine Trajektorie maximal gezeichnet wird. Hält sich bei einem bestimmten Vektorfeld die Trajektorie sehr lange im gezeichneten Bereich auf, kann es ratsam sein, diesen Wert zu variieren.

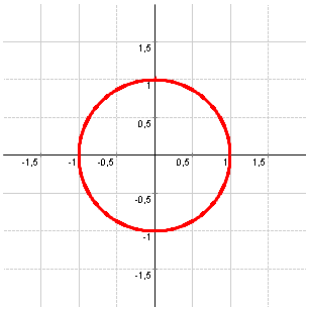
Wegweiser/Kompasse/Geschwindigkeiten: Bei „Geschwindigkeiten“ ist die Pfeillänge proportional zum Betrag des jeweiligen Vektors des Feldes. Das kann zu recht unübersichtlichen Bildern führen. Die anderen beiden Anzeigemodi unterscheiden sich nur im Detail.

Wie funktionieren parametrisierte Kurven mit IOMath?

Aus mathematischer Sicht

Parametrisierte Kurven der Form $c\colon [a,b] \to \mathbb{R}^2$ können beispielsweise als Lösung einer Differentialgleichung entstehen, oder aus der Sicht der Differentialgeometrie interessant sein.

Kurven in IOMath

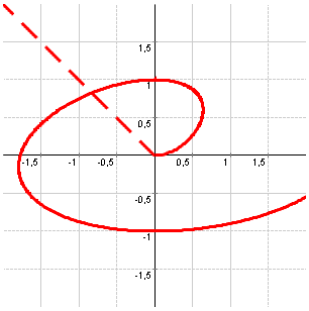


Ähnlich wie bei den Vektorfeldern gibt man die x- und die y-Komponente der Kurve durch einen Strichpunkt getrennt an. Nun allerdings nicht in Abhängigkeit von den Variablen x und y, sondern nur in Abhängigkeit vom Parameter t. Beispielsweise beschreibt „cos(t); sin(t)“ einen Kreis, sofern man den Parameter t das Intervall $[0; 2\pi[$ durchlaufen lässt.

Parametereinstellungen

Der Benutzer kann wählen, welches Intervall der Parameter t durchlaufen soll, und in welcher Schrittweite das geschehen soll. Klar ist: je kleiner die Schrittweite gewählt wird, umso exakter wird die Kurve gezeichnet, aber umso mehr Rechenzeit wird benötigt. Für „normale“ Situationen sollte eine Schrittweite von 0.01 bis 0.001 klein genug für ein gutes Ergebnis sein.

Mehrere Kurven in einem Plot

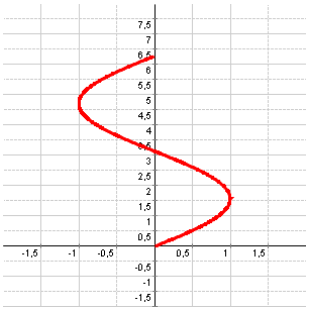


Sie können, durch weitere Strichpunkte oder Zeilenumbrüche getrennt, auch mehrere Kurven in einen Plot zeichnen lassen. Beispielsweise um zu demonstrieren, wo sich zwei Kurven schneiden. Bsp.: „sqrt(t)*cos(t); sin(t); -t; t“ mit Parameter t von 0 bis 2*pi

Graphen durch Kurven realisieren

Unter gewissen Umständen kann es sinnvoll sein, einen Funktionsgraphen als parametrisierte Kurve zu plotten. In IOMath können Sie beim Kurvenplotter nämlich die Schrittweite vorgeben. Beim Funktionsplotter hingegen wählt das Programm automatisch eine Schrittweite, die der Bildauflösung angepasst ist. Z.B. bei schnell oszillierenden Funktionen kann diese zu groß sein. Statt „sin(1/x)“ im Funktionsplotter verwenden Sie dann einfach „t; sin(1/t)“ im Kurvenplotter.

Mit dem Kurvenplotter können Sie auch erreichen, dass ein Graph über der y-Achse statt über der x-Achse gezeichnet wird. Brauchen Sie beispielsweise einen „vertikalen Sinus“, so verwenden Sie „sin(t); t“.



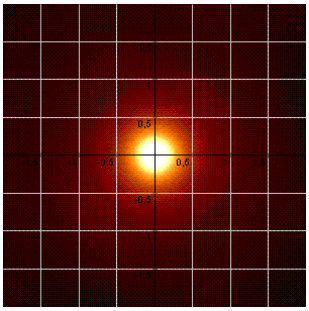
Wie funktionieren Skalarfelder/Potentiale mit IOMath?

Aus mathematischer Sicht

Skalarfelder/Potentiale, die IOMath skizzieren kann, sind aus mathematischer Sicht Abbildungen $f: \mathbb{R}^2 \to \mathbb{R}$. Mit anderen Computerprogrammen kann man solche Funktionen auch als 3D-Gebirge darstellen. Das kann IOMath *nicht*. IOMath füllt hingegen die zweidimensionale Ebene mit unterschiedlichen Farben, die dem Wert der Funktion an der jeweiligen Stelle entsprechen.

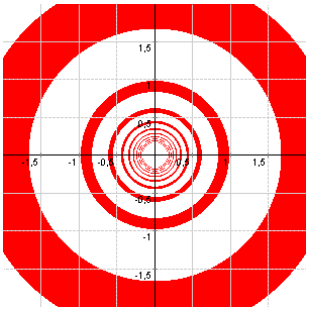
Skalarfelder/Potentiale mit IOMath

Geben Sie im Potentialplotter den Wert des Potentials an der Stelle (x,y) in Abhängigkeit dieser beiden Koordinaten an. Bsp.: „1/sqrt(x^2+y^2)“ ist ein kugelsymmetrisches Potential, wie es z.B. dem Coulombproblem entspricht. Beachten Sie, dass Sie IOMath mitteilen müssen, welchen Bereich der Bildmenge das Programm zeichnen soll. Funktionswerte ausserhalb des angegebenen Bereichs sind dann farblich nicht mehr unterscheidbar. In der Abbildung rechts ist deutlich erkennbar, dass außerhalb eines gewissen Radius keine Unterscheidbarkeit der Funktionswerte mehr gegeben ist.



Verschiedene Sichtweisen

Für jede spezielle Anwendung eignet sich eine andere Darstellungsweise des Potentials. Probieren Sie einfach mal die unter „Rendering-Einstellungen“ verfügbaren Farbschemata aus. Die Namen der Schemata sind nur bedingt aussagekräftig. Machen Sie sich selbst ein Bild! In der Abbildung rechts ist das selbe Potential zu sehen, wie darüber, lediglich in einem anderen Farbschema.



Wie kann ich das Aussehen des Plots beeinflussen?

Einstellmöglichkeiten

Wertebereich: Hier wird festgelegt, in welchem Bereich der Graph bzw. das Vektorfeld gezeichnet wird. In die Textboxen können auch mathematische Ausdrücke eingegeben werden. Bsp: von „-pi“ bis „2*pi“

Gitterlinien-Abstand: Der geplottete Bereich wird mit Gitternetzlinien gerastert. Den Abstand zweier Linien kann man hier einstellen. Sind keine Linien erwünscht, bitte „0“ eintragen.

Achsenbeschriftung: Die Unterpunkte „Abstand“ und „Größe“ sind selbsterklärend. Unter „Format“ kann eingegeben werden, wie die Achsenbeschriftung aussehen soll. Die Symbolsprache ist unter Programmierern bekannt. Es gelten beispielhaft folgende Bedeutungen:

„#.0“	Genau eine Nachkommastelle
„#.000“	Genau drei Nachkommastellen
„000“	Genau drei Stellen, keine Nachkommastelle

„0.##“	Feste Stelle vor dem Komma, maximal zwei Nachkommastellen
„0,000“	Mit Tausendertrennzeichen
„#km“	Mit nachgestellter Einheit „km“

Rendering-Einstellungen

Die Einstellung „Linienstärke“ beeinflusst auch die Stärke, mit der das Koordinatenkreuz gezeichnet wird. Das ist beispielsweise dann von Vorteil, wenn Sie einen Plot in sehr hoher Auflösung rendern möchten. Wählen Sie in diesem Fall eine größere Linienstärke, damit man die Achsen im fertigen Bild noch deutlich erkennen kann.

Das „Farbschema“ beeinflusst bei Funktionenplots/Kurven/Vektorfeldern die Farbe/Linienstärke/Strichelung der Linien, insbesondere auch die Weise, in der mehrere Linien in einem Plot voneinander unterschieden werden. Bei Potentialen beeinflusst es in grober Weise das Aussehen des Ergebnisses. Bitte experimentieren Sie selbst!

Nutzungsbedingungen

Dem Autor ist bewusst, dass IOMath durchaus nicht „perfekt“ ist. Sicherlich lassen sie unzählige Bugs finden. Ich bitte zu berücksichtigen, dass es sich im Endeffekt doch nur um „Laienprogrammierkunst“ handelt, die aus ebendiesem Grund ja kostenlos angeboten wird.

Für Bugreports, Verbesserungsvorschläge und sonstige Kommentare wenden Sie sich bitte an die auf der Homepage http://www.IdeaOverflow.de angegebene eMail-Adresse.