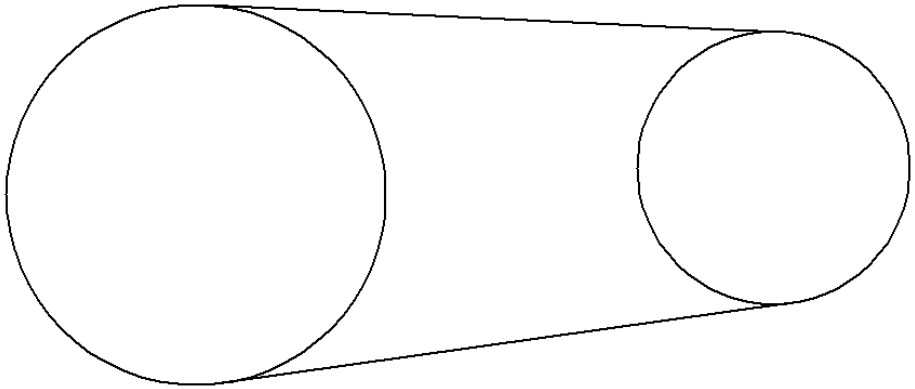
Vortrag Software AG 2013

Constraint basiertes CAD

Jedes bessere CAD System erlaubt es eine Linie als Tangente an 2 Kreise zu konstruieren:



VTangente2kreise

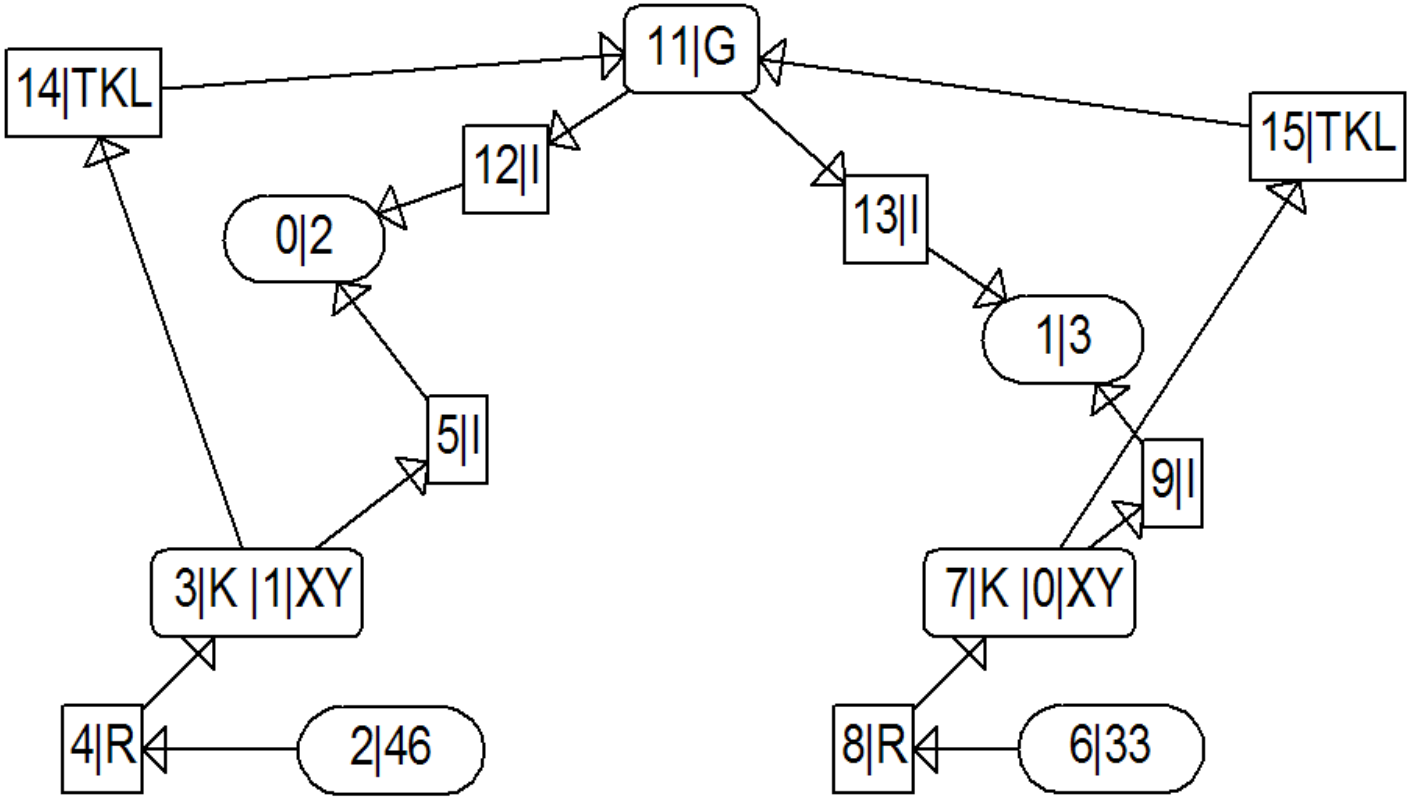
Wird jedoch ein Kreis verschoben, gehen die Linien nicht mit.

Die einfachste Lösung besteht darin, dass bei der sequentiellen Konstruktion nicht nur die Geometriedaten festgehalten werden, sondern auch die jeweiligen Konstruktionsschritte.

Erzeugt man bei diesem **history-based Modeling** z.B. eine Linie als Tangente an zwei Kreisen, so hält man nicht nur die aktuellen Koordinaten fest, sondern auch die Eigenschaft der Linie, Tangente der beiden Kreise zu sein. Das erlaubt es, nach einer Änderung der Kreise die Tangente automatisch nachzuführen. Mit dem gleichen Verfahren kann man auch eine assoziative Bemaßung bzw. Schraffur realisieren.

Das Verfahren versagt jedoch, wenn man etwa einen Tangentialpunkt verschieben möchte.
Dann muss man mit **Constraints** arbeiten.

Ein Constraint beschreibt eine Beziehung zwischen 2 oder mehr Objekten. Das Modell wird durch einen bipartiten Graphen beschrieben.
Im obigen Beispiel etwa:



Der Graph besteht aus 2 Knotenklassen:

* Geometrische Objekte
* Constraints

Es sind jeweils Elemente der beiden Klassen adjazent.

Solche Constraints können u.a. sein:

* Zwei Linien sind parallel
* Zwei Linien bilden einen rechten Winkel
* Zwei Linien sind gleich lang
* Kreis und Gerade sind tangential
* Ein Punkt ist Schnittpunkt von zwei Linien oder Geraden
* Ein Punkt liegt auf einem Kreis
* Zwei Kreise sind tangential

Die Constraints können auch Bemaßungen sein, wie z.B.

* Länge einer Linie
* Radius eines Kreises

Jedes Constraint hat eine Anzahl von Parametern und konsumiert einen oder auch evtl. mehrere Freiheitsgrade.

Beispiel

Eine Linie von (x1, y1) nach (x2, y2) hat die Länge d.

Die fünf Parameter sind x1, y1, x2, y2 und d. Von den fünf Parametern können vier vorgegeben werden, aus denen dann der fünfte berechnet wird. Man kann die Koordinaten vorgeben und berechnet die Länge, man kann aber auch die Länge vorgeben und daraus einen der Koordinatenwerte berechnen. Mit dem Constraint wird ein Freiheitsgrad konsumiert.

Bei einer Konstruktion im 2D hat man z.B. folgende Objekte:

Skalare:	1 Freiheitsgrad
Punkt:	2 Freiheitsgrade
Gerade:	2 Freiheitsgrade
Kreis:	3 Freiheitsgrade

Ein Dreieck hat somit 6 Freiheitsgrade.
Davon werden 3 Freiheitsgrade durch die Lage konsumiert. (X, Y Position und Ausrichtung)
Die restlichen 3 Freiheitsgrade bleiben für die Form des Dreiecks übrig.
Sie können bestimmt werden z.B. durch die Länge von 3 Seiten.

Generell wird im zweidimensionalen Raum bei einer Konstruktion aus n Punkten die Form durch 2*n-3 Constraints, die jeweils einen Freiheitsgrad konsumieren, festgelegt.
Es gibt auch Constraints, die mehr als einen Freiheitsgrad konsumieren, wie z.B. wenn mehr als 2 Punkte horizontal ausgerichtet sind.

Die Techniken zur Lösung dieser geometrischen Constraint-Systeme können in zwei Klassen eingeteilt werden:

* **Gleichungsbasierte Methoden**
* **Graphenbasierte Methoden**

Gleichungsbasierte Methoden

Die Benutzung von Gleichungen zur Geometrischen Modellierung wurde bereits von Sutherland im ersten CAD System Sketchpad 1963 eingeführt.
Fortgeführt wurde sie durch die Arbeiten von Light und Gossard 1982.

Hier werden die Constraints durch i.A. nichtlineare Gleichungen zwischen den Variablen dargestellt. Die Variablen sind die skalaren Größen der Objekte, also Werte wie der Abstand zweier Punkte oder bei Punkt und Kreis x- und y-Koordinaten, beim Kreis zusätzlich der Radius r.

Ein geometrisches System ist überbestimmt, wenn es mehr Gleichungen (Constraints) als freie Variable gibt.
Das System ist vollständig bestimmt, wenn die Anzahl der Variablen gleich der Anzahl der Gleichungen ist.
In dem geometrischen System sind dann alle Freiheitsgrade bestimmt. Dann ist das Gleichungssystem evtl. lösbar.
Hat man weniger Gleichungen als freie Variablen, muss man entweder zusätzliche Gleichungen einführen, oder für die überzähligen Variablen feste Werte annehmen.

Gleichungen für einige geometrische Bedingungen

Horizontaler Abstand d x2-x1-d=0

Vertikaler Abstand d y2-y1-d=0

Abstand d $(x1-x2)^2+(y1-y2)^2-d^2=0$

Winkel der Linien von P1 nach P2 und P1 nach P3 sei phi

$$((x2-x1)*(x3-x1)+(y2-y1)*(y3-y1))/(L(P1,P2)*L(P1,P3))-cos(phi)=0$$

Ein Punkt x1, y1 liegt auf dem Kreis mit Mittelpunkt x, y und Radius r $(x1-x)^2+(y1-y)^2=r^2$

Die Linie (x1, y1)nach(x2, y2) und die Linie (x3, y3) nach (x4, y4) seien orthogonal. Dann muss das Skalarprodukt 0 sein:

$$(x2-x1)*(x4-x3)+(y2-y1)*(y4-y3)=0$$

Für das nichtlineare Gleichungssystem wird entweder mit Hilfe des Newton- Raphson- Verfahren eine iterative Lösung gesucht oder es wird eine algebraische Methode mit Hilfe von Gröbnerbasen benutzt

Die Lösung eines Gleichungssystems zu finden ist äquivalent dem Problem, die Nullstellen einer Menge von Funktionen mit mehreren Variablen zu finden. Im Folgenden wird vorausgesetzt, dass das Gleichungssystem genauso viele Variable besitzt, wie es Gleichungen hat.

Zunächst wird der Fall betrachtet, die Nullstelle einer allgemeinen nichtlinearen Funktion **f : D -> R** zu finden. Wenn f stetig differenzierbar ist, kann die Stelle f(z) ausgehend von f(x0) über ein Taylor-Polynom approximiert werden.

$$f(z) = f(x0) + f'(x0)(z - x0) + R(z)$$

Unter Fortlassung des Restgliedes R(z) lässt sich mit:

$$f(x0) + f'(x0)(x1 - x0) = 0$$

eine Näherung x1 für eine Nullstelle z von f berechnen:

Aus der Gleichung ergibt sich: $x1 = x0 - f(x0)/f'(x0)$

bzw. allgemein: $x_{i+1} = x_i - f(x_i)/f'(x_i)$

Die Näherung lässt sich natürlich nur berechnen, wenn die Ableitung $f'(x_t) <> 0$ist und der Startwert hinreichend nahe an der Nullstelle liegt. Des Weiteren erhält man von einem Startwert ausgehend nur eine Nullstelle.

Verallgemeinert man dies auf die Lösung eines Gleichungssystems F(X)=0 mit F=(f1,..., fn) und X=(x1,..., xn) so erhält man die Gleichung:

$$F(X_t) + F'(X_t)(X_{t+1} - X_t) = 0$$

über die sich die Näherung X_{t+1}aus $X_t$berechnen lässt.

Die Berechnung einer neuen Näherung erfordert die Lösung des inhomogenen linearen Gleichungssystems:

$$F'(X_t) * \Delta X = -F(X_t)$$

$F'(X_t)$ist die Jacobi-Matrix der Partiellen Ableitungen $\frac{\delta f_i}{\delta x_j}$ für i, j=1...n an der Stelle X_t.

$\Delta X = (\Delta x_1, ..., \Delta x_n)$ist der zu bestimmende Vektor aus dem sich mit $X_{t+1} = X_t + \Delta X$die neue Näherung berechnen lässt. $-F(X_t)$ist der Vektor der Restwerte, d.h. der noch vorhandenen Fehler. Das System besitzt nur dann eine Lösung, wenn die Gleichungen nicht linear abhängig sind, d.h. die Determinante $detF'(X_t) <> 0$ist. Bei komplizierten Funktionen ist es i.A. nicht möglich, die Ableitungen zu berechnen. Dann benutzt man statt der Differentialquotienten $\frac{\delta f_i}{\delta x_j}$die Differenzenquotienten:

$$\frac{\Delta f_i}{\Delta x_j} X = \frac{f_i(x_1, ..., x_n) - f_i(x1, ..., x_j - h, ..., x_n)}{h}$$

um die Jacobimatrix aufzustellen.

Der Differenzenquotient $\frac{\Delta f_i}{\Delta x_j}(X_t)$ kann dadurch erzeugt werden, indem man zunächst

$A = f_i(x_1, ..., x_n)$berechnet wird. Dann wird $B = f_i(x_1, ..., x_j + delta, ..., x_n)$berechnet.

$\frac{B - A}{\Delta}$ ergibt den Differenzenquotienten $\frac{\Delta f_i}{\Delta x_j} X_t$. Die Berechnung von A und B erfolgt durch auswerten des Constraints, dass zu dieser Gleichung gehört.

Damit das Gleichungssystem eine Lösung hat, darf die Determinante der Jacobimatrix nicht 0 sein, da man die inverse Matrix bilden muss. Das bedeutet, die Matrix muss quadratisch sein, d.h. man hat so viele Gleichungen wie Variablen.

Hat man weniger Gleichungen, muss man für einige Variablen den alten Wert benutzen. So hat man weniger Variable.

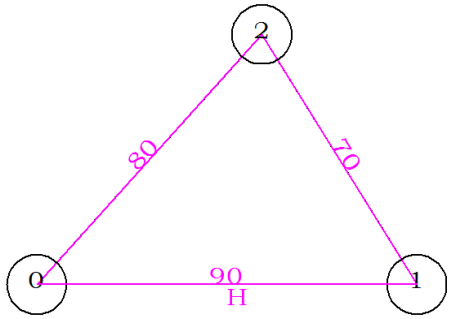
Hat man mehr Gleichungen als Variablen so ist das System überbestimmt und kann nicht gelöst werden.

Aber auch, wenn die die Anzahl der Variablen und die Anzahl der Gleichungen gleich sind, kann die Matrix singulär sein, wenn ein Teil der Matrix überbestimmt und ein Teil unterbestimmt ist.

Dies ist z.B. dann der Fall, wenn Bedingungen redundant sind, d.h. eine Bedingung genau aus der anderen folgt. Die Zeilen der Jacobi-Matrix sind dann linear abhängig.

Das Gleichungssystem kann direkt gelöst werden, wenn in der Jacobimatrix nach einer geeigneten Zeilen Permutation nur die obere oder untere Dreiecksmatrix besetzt ist.

Beispiel:



gegeben: x0, y0; d0=90, d1=70, d2=80
Bei noch 2 freien Punkten benötigt man 4 Gleichungen:

x1=x0+d0	Horizontaler Abstand 0 und 1
y1=y0	Horizontal 0 und 1
$(x1-x2)^2+(y1-y2)2=d1^2$	Abstand 1 und 2
$(x0-x2)^2+(y0-y2)2=d2^2$	Abstand 0 und 2

Bisweilen kann die Lösung des gesamten Gleichungssystems zurückgeführt werden auf die sequentielle Lösung von zwei Teilsystemen.

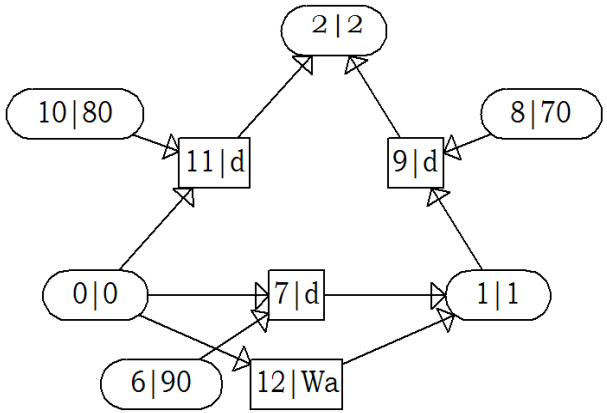
```
a1,1.....a1,i    0...........0
.............    0...........0
ai,1.....ai,i    0...........0
ai+1,1....ai+1,i ai+1,i+1.....ai+1,n
.................................
an,1.....an,i    an,i+1......an,n
```

Mit den ersten i Gleichungen können die Variablen x1 bis xi gelöst werden. Diese werden in die unteren Gleichungen eingesetzt und man bekommt ein Gleichungssystem für die Variablen xi+1 bis xn.

Ein System, das Gleichungsbasiert arbeitet ist **Solvespace**
Es kann herunter geladen werden unter http://solvespace.com

Graphenbasierte Methoden

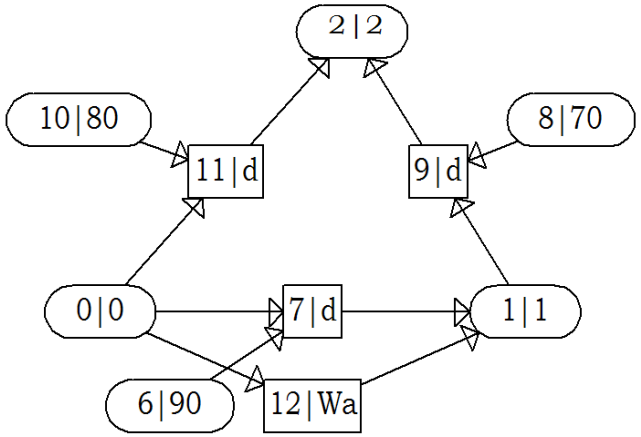
Bei allen Graphbasierten Methoden wird die Konstruktion durch einen (Hyper)graphen bzw. einen bipartiten Graphen modelliert.
Für das Dreieck oben:



Bei der Erstellung des Modells sind die Kanten zunächst ungerichtet.
Seit den 90er Jahren sind verschiedene Verfahren entwickelt worden, hieraus das Modell zu berechnen:

- C.M. Hoffman und andere zerlegen das Modell in Teile, deren relative Position berechnet werden kann.
  Diese Technik wird benutzt in dem System Ledas.
  Nach entsprechender Registrierung kann man bei der LEDAS LTD (Novosibirsk) das Programm LGS 2D runterladen.
  http://ledas.com/products/lgs2d/
- Verroust, F. Schonek, and D. Roller setzen das Modell nach bestimmten Regeln zusammen, etwa Konstruktion von Dreiecken aus Längen und Winkeln bzw. von Vierecken.
- Brüderlin löst das Problem über Ersetzungsregeln symbolisch.

1996 hat Roland Berling in seiner Dissertation "**Eine Constraint- basierte Modellierung für Geometrische Objekte**" hier das Problem gelöst, indem er entsprechend den Freiheitsgraden aus dem ungerichteten Graphen einen gerichteten erzeugt hat und danach die zu bestimmenden Variablen berechnet hat.



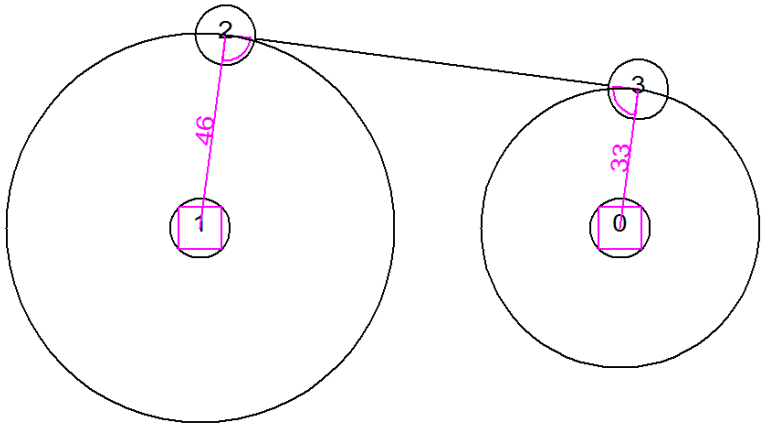
Im Beispiel sind der Punkt 0 und die Abstände gegeben. Deren Kanten müssen alle auslaufend sein.
Die vorkommenden Constraints konsumieren jeweils einen Freiheitsgrad, d.h. eine Kante muss auslaufend die anderen einlaufend sein.
Damit ergibt sich die Kantenrichtung für 7,12 und 11.
Der Punkt 1 hat nun 2 einlaufende Kanten, die restlichen müssen auslaufend sein. Damit ergibt sich die Kante 1-9
Dann hat 9 zwei einlaufende also muss die Kante nach 3 auslaufend sein.
11 hat ebenfalls 2 einlaufende Kanten, daher muss die Kante nach 2 auslaufend sein
Da der gerichtete Graph azyklisch ist, kann das Modell in topologischer Sortierung berechnet werden.
Jedes Constraint legt durch seine auslaufende Kante eine Ortslinie für den Zielknoten fest.
Durch 12|Waagrecht muss 1 auf einer horizontalen Geraden durch 0 liegen
Wegen 7|d Abstand muss 1 auf einem Kreis mit dem Radius 90 um Punkt 0 liegen.
1 liegt dann auf dem Schnittpunkt.
2 muss wegen 9|d auf einem Kreis mit Radius 70 um 1 liegen, wegen 10|d auf einem Kreis um Punkt 0 mit Radius 80.

Roland Berling benutzt bei seinem Verfahren nur Skalare und Punkte als Variablen.

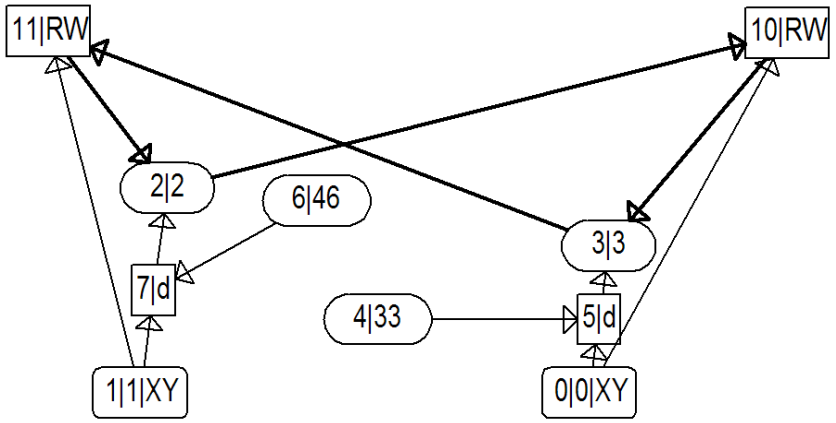
Als Constraint werden benutzt:

* Abstand von 2 Punkten
* Abstand eines Punktes von einer Geraden gegeben durch 2 Punkte
* Winkel gegeben durch 3 Punkte
* Winkel zwischen 2 Geraden gegeben durch jeweils 2 Punkte
* Arithmetische Gleichungen

Mit diesen Variablen und Constraints lassen sich die anderen geometrischen Beziehungen ausdrücken.
Beispiel: Tangente an 2 Kreise:



Der Graph hierzu:



Der Graph lässt sich nicht zykelfrei orientieren. Damit kann die Konstruktion nicht sequentiell berechnet werden.

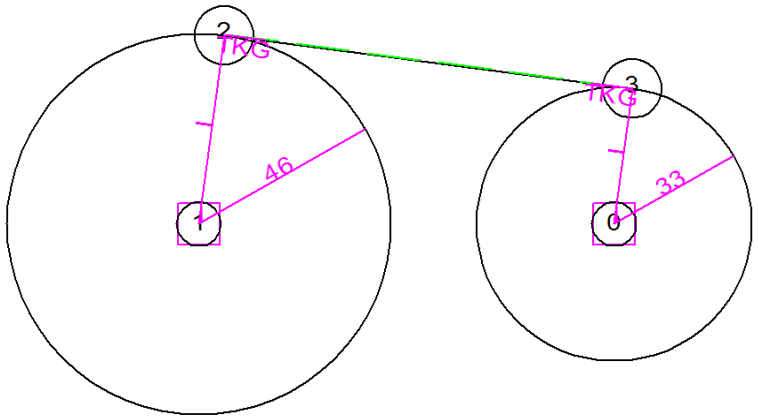
Die Tangente an 2 Kreise kann jedoch durch einen Algorithmus berechnet werden.
Um dieses Problem zu lösen muss man weitere Variablen und Constraints einfügen.
Neue Variablentypen:

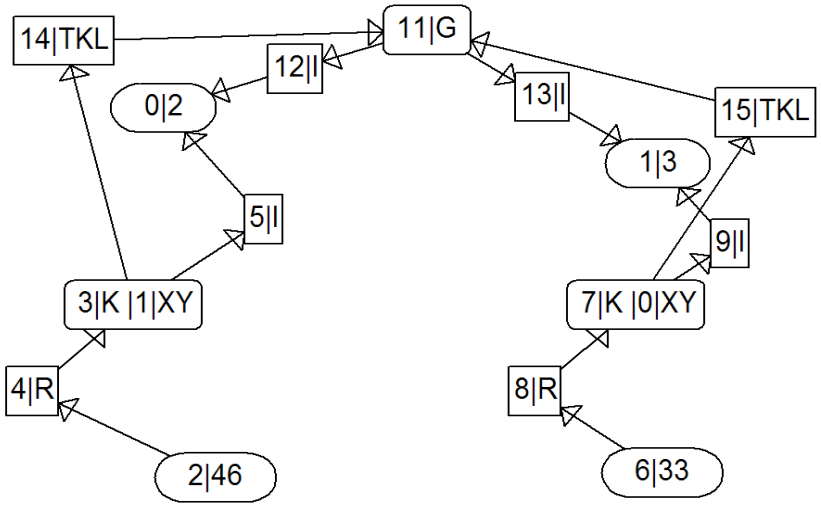
* Kreis (3 Freiheitsgrade)
* Gerade (2 Freiheitsgrade)

Neue Constraints:

* Radius: (Skalar, Kreis)
* Inzident (Punkt, Kreis oder Gerade)
* Tangential (Kreis, Kreis oder Gerade)
* Distanz Punkt Gerade (Skalar, Gerade, Punkt)
* Richtung Gerade (Skalar, Gerade)
* Parallel (Gerade, Gerade)

Man erhält dann für die Tangente folgendes Modell:

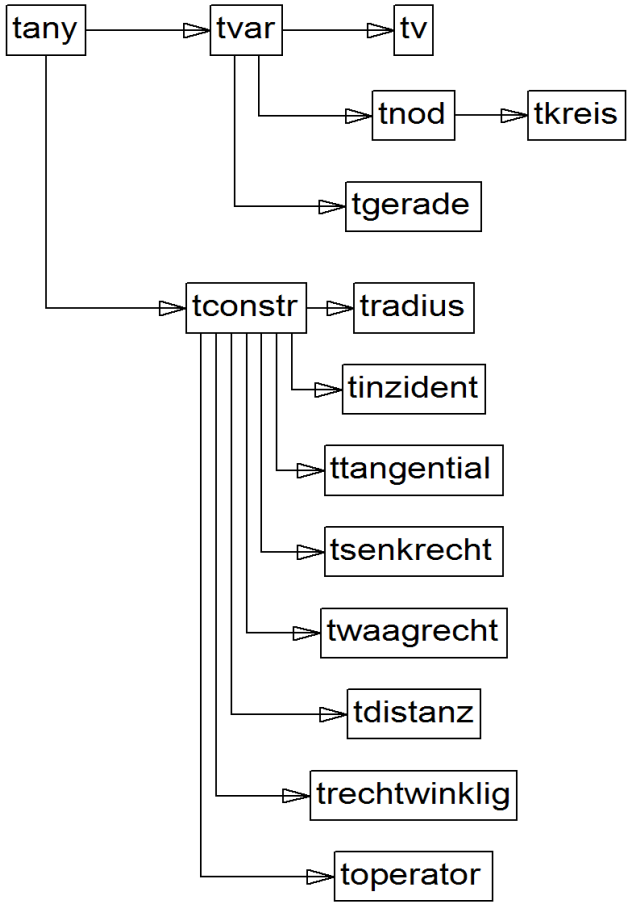




Aus den beiden Kreisen wird die tangentiale Gerade berechnet, die Punkte müssen dann auf der Geraden und einem Kreis liegen.

Implementation:

Die Knoten des Graphen liegen in der Klasse **`tany`**. Davon abgeleitet:



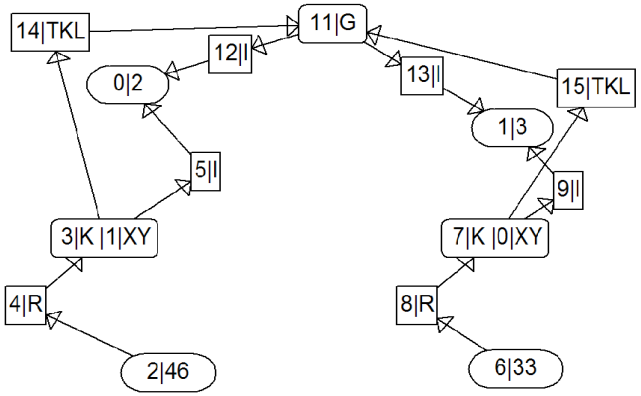
Für jede Klasse abgeleitet von tany existiert eine Methode: compute

Bei den Constraints wird aus den Variablen der einlaufenden Kanten eine Ortslinie bzw. Wert für die Variable der auslaufenden Kante berechnet.

Diese Ortslinien bzw. Werte sind:

* Skalare (tv) : Skalarwert
* Punkte (tnod): X Wert, Y Wert oder Gerade bzw. Kreis auf denen der Punkt liegt.
* Kreis (tkreis) zusätzlich: Radius, Gerade oder Kreis an denen der Kreis tangential liegt, Punkt der auf dem Kreis liegt.
* Gerade (tgerade): Richtung, tangentialer Kreis, Punkt auf der Geraden

Für jede Kombination dieser Ortslinien ist ein Algorithmus implementiert, der die Variable berechnet.
Bei einem Kreis gibt es 4 Ortslinien für den Mittelpunkt und 3 zusätzliche Ortslinien und den Radius.
Den Radius kann man getrennt berücksichtigen, es bleiben noch 7 Möglichkeiten
Bei 3 gegebenen Ortslinien hat man 7 über 3 also (7*6*5)/(1*2*3)=35 Kombinationen.
Bei 2 gegeben Ortslinien hat man 7 über 2 also (7*6)/(1*2)= 21 Kombinationen
Bei einer Ortslinie noch 7 Möglichkeiten also insgesamt 35+21+7=63 Kombinationen



Beispiel: Gerade 11: Ortslinien: Tangente an Kreis 0, Tangente an Kreis 1
Hier wird die Funktion: Tangente an 2 Kreise aufgerufen.
Man hat 4 Lösungen. Genommen wird die, die der bisherigen Geraden am nächsten ist.

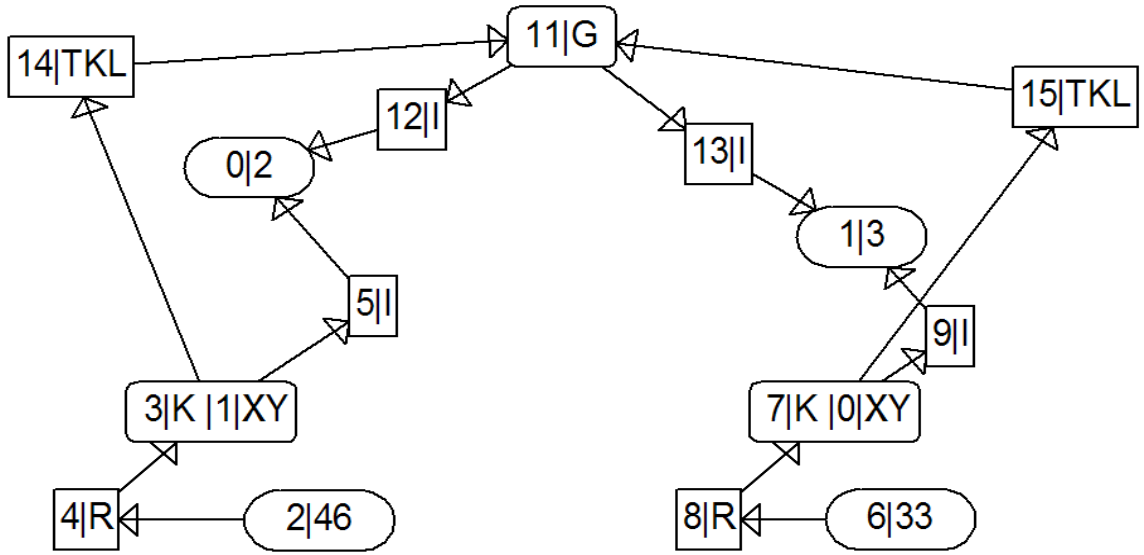
Orientierung des Constraint- Graphen

Zunächst werden bei fixierten Variablen alle Kanten auslaufend gemacht.

Dann wird bei allen Constraints, die eine ungerichtete Kante und den Rest einlaufend haben, die ungerichtete Kante auslaufend gemacht.

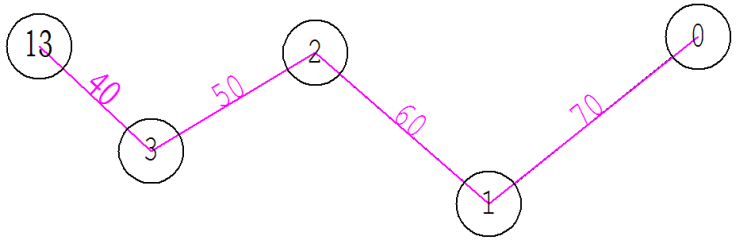
Alle Variablen, bei denen alle restlichen ungerichteten Kanten einlaufend gemacht werden können, kommen in eine Schlange.

Aus der Schlange wird eine Variable V entnommen
  Die restlichen Kanten von V werden einlaufend
  Für alle Constraints C von V mit auslaufender Kante nach V werden die restlichen Kanten einlaufend gemacht
  alle adjazenten Variablen W von C, bei denen alle restlichen ungerichteten Kanten einlaufend gemacht werden können, kommen in die Schlange
solange bis die Schlange leer ist.
Der Algorithmus findet zu jedem Graphen eine Orientierung, sofern dies ohne Zyklen möglich ist.

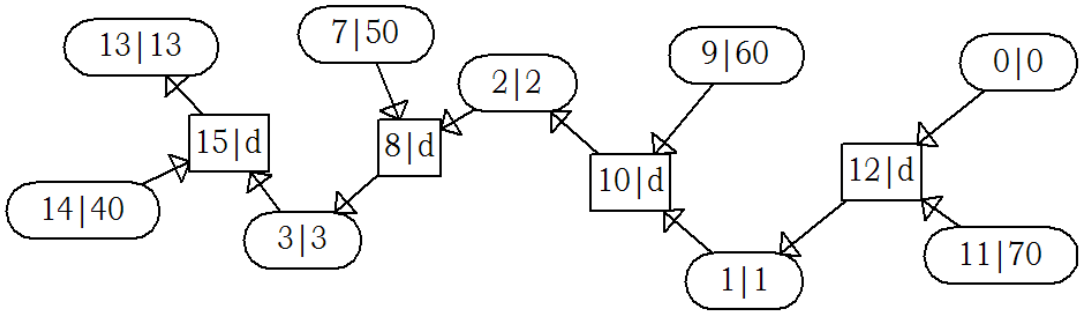


Für die Praxis wichtig ist jedoch die Reihenfolge.

Beispiel:(Vschlange)



Wenn Knoten 0 verschoben wird, sollen die anderen nachgezogen werden



0 12 11 1 10 9 2 8 7 3 15 14 13

Dazu muss die Verschiebung von Knoten 0 bis Knoten 13 propagiert werden.
Dies wird erreicht durch Breitensuche im Graphen von zu verschiebenden Knoten 0 aus.
Entsprechend der umgekehrten Breitensuche werden die Kanten der Variablen einlaufend gemacht.
Ansonsten würden nur Knoten 0 und 1 verschoben

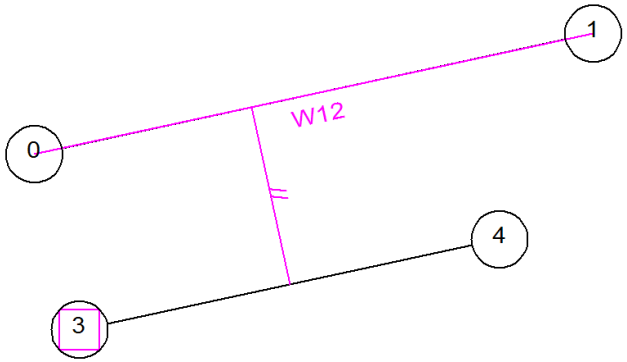
Für die Orientierung müssen die noch vorhandenen Freiheitsgrade der Variablen berechnet werden.
Dabei reicht es nicht nur die einlaufenden Kanten zu zählen.
Ein Punkt kann nicht 2 einlaufende Kanten von einem Horizontal Constraint haben, da dies jeweils nur durch die Y Koordinate erfüllt werden kann.
Den Constraints wird ein Attribut Fix zugeordnet, dass bestimmt durch welche Attribute der Variablen das Constraint erfüllt werden kann.

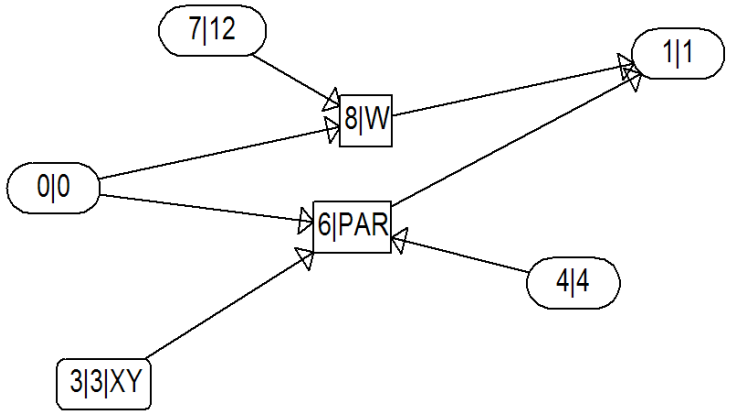
* Distanz: isxy
* Horizontal: isy
* Vertikal: isx
* Radius: isr
* Tangential an Kreis: isxyr
* Inzident: isxyr

Die Klasse Tvariable definiert die abstrakte Methode **`sdf(fix:tfix):`**
Für die abgeleiteten Klassen wird entsprechend **`fix`** der verbleibende Freiheitsgrad berechnet.

Dies löst aber noch nicht das Problem, dass die Constraints Ortslinien bestimmen, die keinen Schnittpunkt haben, z.B. parallele Linien oder Kreise mit gleichem Mittelpunkt. Ein Problem tritt ferner auf, wenn der Schnittpunkt auf einen anderen Punkt im Modell fällt.

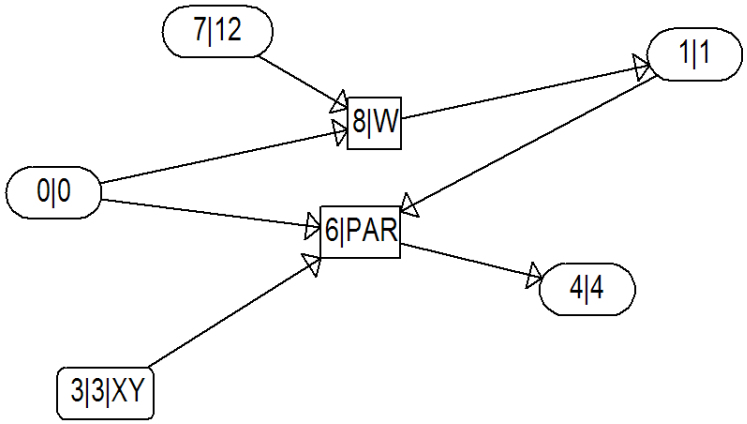
Beispiel:





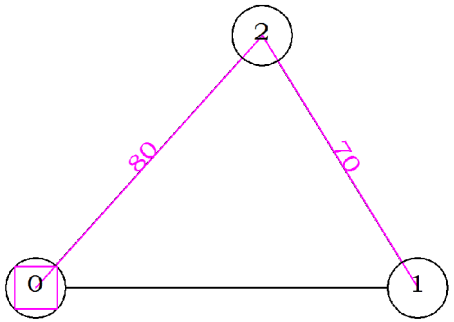
Zwischen Punkt 0 und 1 muss der Winkel 12 Grad bestehen, Die Linien 0-1 und 3-4 sind parallel und Punkt 3 ist fixiert. Punkt 4 kann nun nur noch in X Richtung verschoben werden. Eine formale Orientierung ist jedoch auch möglich, wenn Punkt 4 in alle Richtungen verschoben wird. Für den Punkt 1 sind die beiden Ortslinien von 8 und 6 entweder parallel, wenn 4 noch nicht verschoben ist, oder sie schneiden sich im Punkt 0. Das 1 mit 0 zusammenfällt ist natürlich immer eine Lösung aber nicht die gesuchte.

Verschiebt man den Punkt 4 nur in X Richtung so ist folgende Orientierung möglich:



Dies kann man aus der obigen Orientierung erhalten, wenn das Problem beim Punkt 1 erkannt wird, und dann eine der einlaufenden Kanten, hier 6 -> 1 umgedreht wird. Dies ist möglich, da 4 noch einen Freiheitsgrad hat, und daher 6->4 als auslaufende Kante von 6 genommen wird. 4 muss dann auf der Parallelen zu 0-1 durch den Punkt 3 und auf der vorgegebenen X Koordinate liegen.

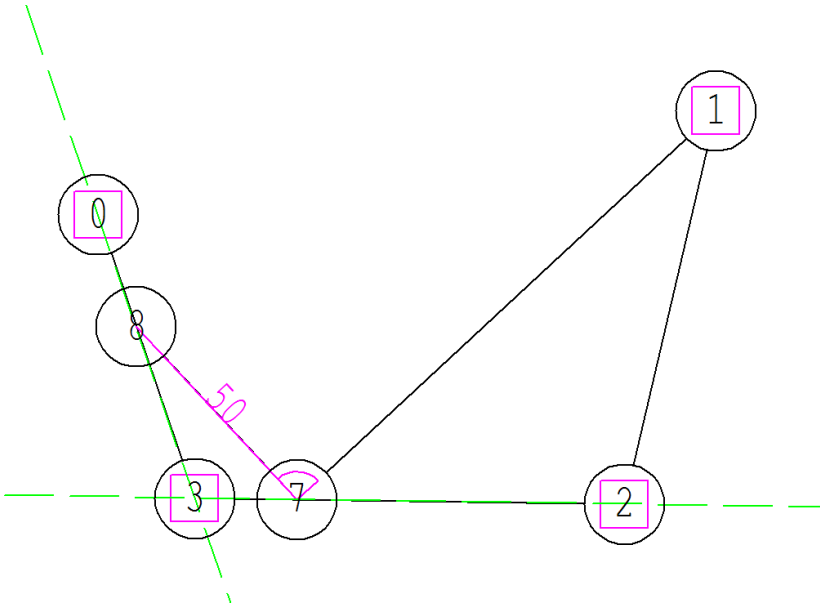
Durch Verschieben eines Punktes oder Ändern eines Wertes kann der Fall eintreten, dass 2 Kreise bzw. Kreis und Linie keinen Schnittpunkt mehr haben. Dann wird nicht weiter verschoben. Beispiel: Vdreieck

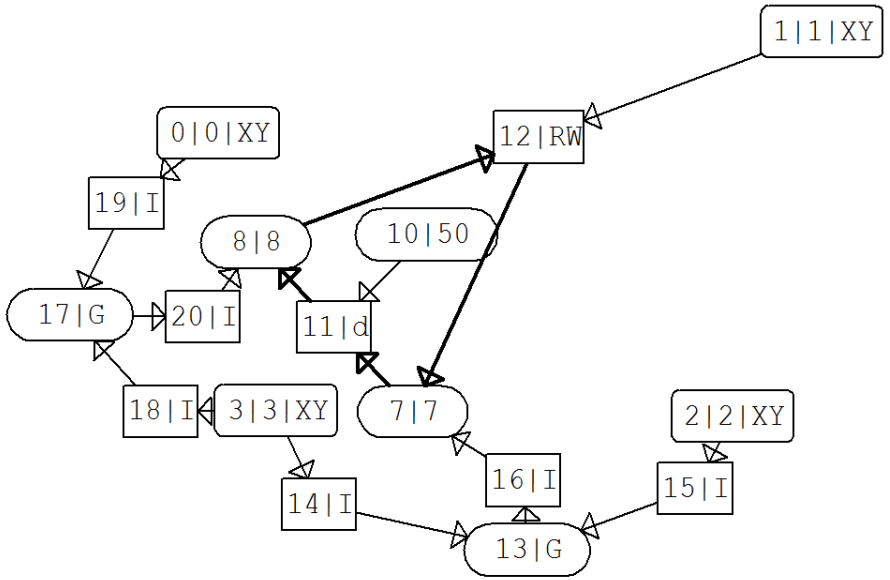


Der Abstand 0-1 kann nicht größer als 80+70 werden

Nicht sequentiell berechenbare Modelle

Manche Modelle sind prinzipiell nicht sequentiell berechenbar, damit auch nicht lösbar mit Zirkel und Lineal (Vwanne,Vwanneklotz)





Die Punkte 7 und 8 müssen so auf den Geraden 0--3 bzw. 3--2 platziert werden, dass der Abstand 7-8 50 ist und der Winkel bei 7 ein rechter. Die Punkte 0,1,2,3 sind gegeben.

Für dieses Modell existiert keine zyklenfreie Orientierung.
Roland Berling benutzt zur Orientierung ein Verfahren von Edmonds. Dabei werden die verbliebenen Kanten zunächst alle so orientiert, dass sie bei den Constraints einlaufend sind. Hat eine Constraint nun mehr einlaufende Kanten, als erlaubt, so wird ein Weg zu einer Variablen mit noch einem Freiheitsgrad gesucht und auf diesen Weg werden die Kanten alle umgedreht.

Das Verfahren findet zwar eine Lösung aber nicht immer die optimale, d.h. die mit den wenigsten Zyklen. Hier wird daher ein Brute-Force Verfahren angewendet. Im Beispiel wird eine Lösung mit dem Zykel 7,11,8,12 gefunden.

Der Zykel 7,11,8,12 muss gemeinsam gelöst werden. Dies geschieht durch Newton Iteration
Im Zykel sind 2 Punkte und damit 4 Variablen.
Die Gleichungen werden durch alle Constraints im Zykel und alle einlaufenden Constraints gebildet.
Das sind die Constraints: 11,12,16 und 20. Damit hat man 4 Gleichungen
Der Funktionswert ist jeweils der Fehler in den Gleichungen
Das System ist prinzipiell lösbar, da die Anzahl der Variablen und Gleichungen gleich ist.
Hat man mehr Variablen als Gleichungen muss man die überzähligen Variablen streichen.
Hat man mehr Gleichungen als Variablen ist das System überbestimmt und daher nicht lösbar

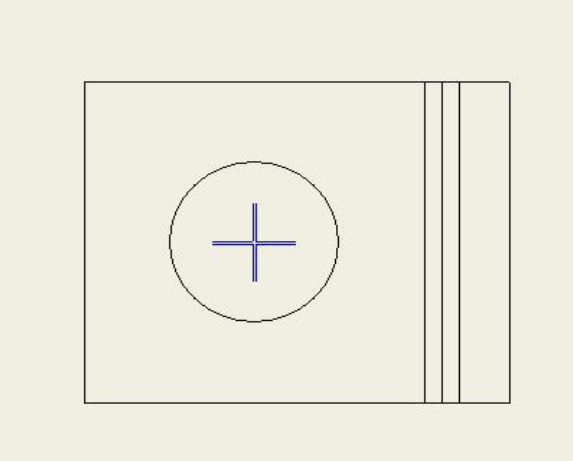
Die partielle Ableitung von Funktion F nach Variable xi ist:

```
(F(x1,..,xi+delta,...,xn)-F(x1,...,xi,...,xn))/delta
```

Wenn die Determinante der Jacobi-Matrix<>0 ist und man von einer Lösung ausgeht und dann einen Punkt verschiebt, ist die Chance dass die Iteration konvergiert groß.

Segmente

Für die Parametrisierung braucht man nicht immer alle Elemente einer Zeichnung sondern nur 1-3 Referenzpunkte.
Beispiel ein Kolben:



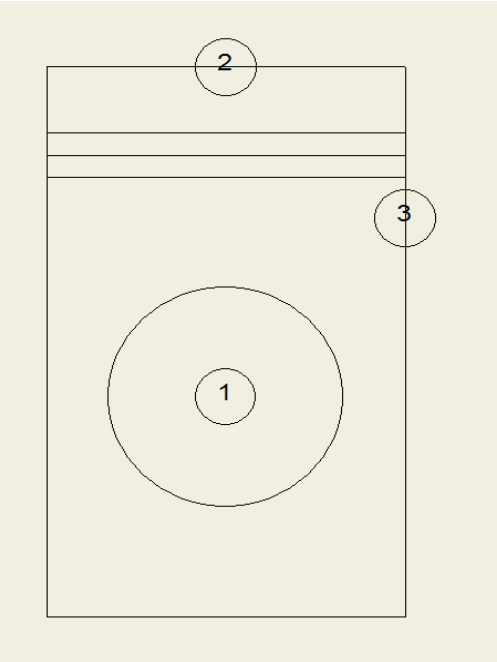
Soll der Kolben Form und Ausrichtung beibehalten, genügt ein Referenzpunkt. Der erste Referenzpunkt legt die Position fest.
Über 2 Referenzpunkte wird Größe und Richtung bestimmt. Der 1. Referenzpunkt bestimmt die Position und der 2. Ausrichtung und Größe.
Beispiel:

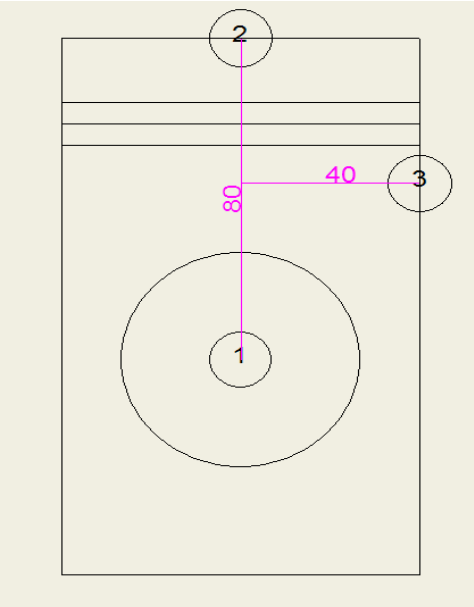
Als Bezugspunkt für die Position wird die Mitte des Kreises genommen.
Als 2. Referenzpunkt für die Größe die Mitte auf dem Kolbenboden rechts.

Mit 3 Referenzpunkten kann die Richtung und die Größe in X und Y Richtung bestimmt werden.
Der 2. Referenzpunkt bestimmt zusätzlich die Größe in X Richtung
Der Abstand des 3. Referenzpunktes zur Linie zwischen dem 1. und dem 2 bestimmt die Größe in Y Richtung.
Als 3 Referenzpunkt wird hier ein Punkt auf der oberen oder der unteren Linie genommen

Im obigen Kolben ändert sich bei 2 Referenzpunkten die Kolbenhöhe und der Kolbendurchmesser jeweils gleich, entsprechend dem Abstand des 2. Punktes.
Durch Angabe eines 3. Referenzpunktes, auf dem Rand des Kolbens, kann die Höhe unabhängig vom Durchmesser geändert werden.

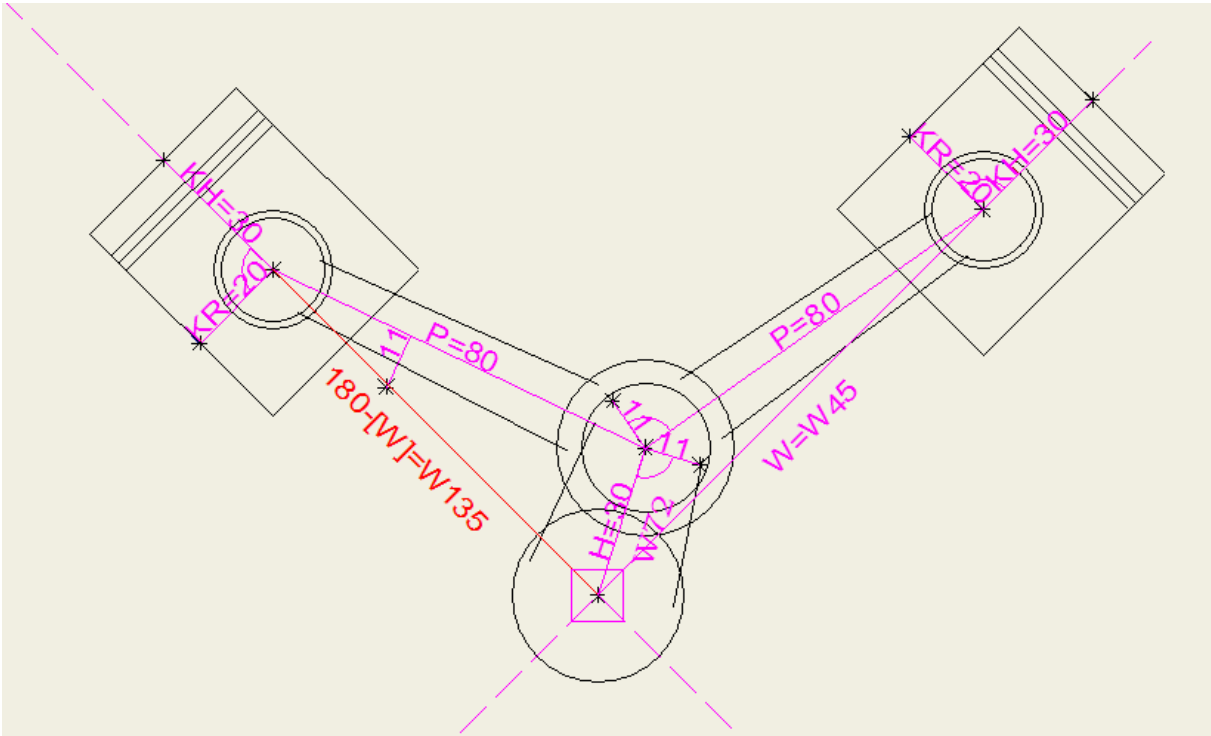
Beispiel:
Es soll eine Instanz des oben definierten Kolbens erzeugt werden.





Nach Auswahl der Definition werden die Punkte 1,2 und 3 festgelegt. Damit kann Position, Höhe und Durchmesser bestimmt werden. Die entsprechenden Parameter können auch noch bemaßt werden.

Durch Ändern der Maße können neue Varianten erzeugt werden.
Der Vorteil ist, dass im Constraint System nicht so viele Elemente definiert werden müssen, sondern die ausschmückenden Elemente nur in der VarioCAD Zeichnung vorhanden sind. So kann z.B. ein ganzer Motor beschrieben werden: (Vmotor)

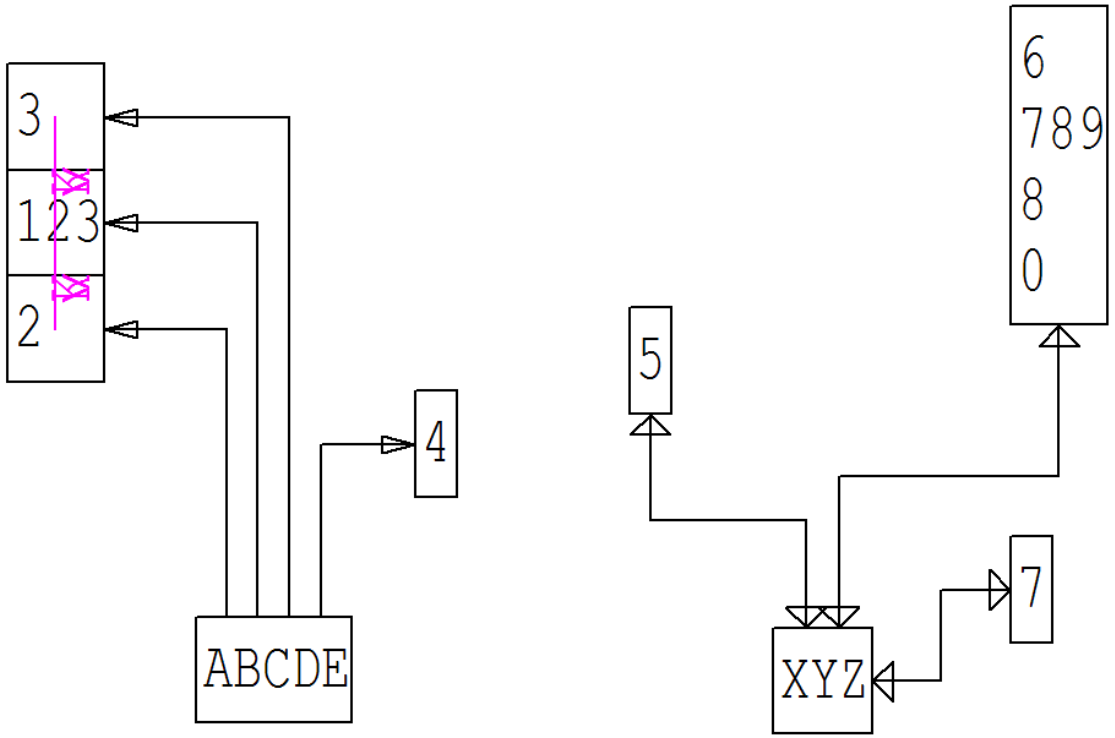


Er ist erzeugt mit den Segmenten Kurbel, Pleuel und Kolben. Durch Ändern des Winkels in der Kurbel (W272) kann der Motor gedreht werden. Z.B. mit der Funktion Bewegen auch endlos.
Durch Ändern des Winkels W in 60 wird aus dem 90° V Motor ein 60° V Motor. Für den linken Zylinder wird eine Formel benutzt.
Durch Ändern der anderen Maße können Kolben, Pleuel und Kurbelwelle geändert werden.

KOGGE5 als Diagramm Editor

Die Punkte im Modell können auch Knoten in Form von Rechtecken, Kreisen, Ovalen oder anderen Figuren sein, die mit Text gefüllt werden. Die Größe der Figur wird automatisch an den Text angepasst.
Zwischen diesen Knoten können verschiedene Verbindungen hergestellt werden.
Orthogonale Verbindungen

* Manhattan Kante waagrecht (waagrecht senkrecht)
* Manhattan Kante senkrecht (senkrecht waagrecht)
* Orthogonale Kanten (senkrecht, waagrecht, senkrecht oder waagrecht, senkrecht, waagrecht)



(Vdiagramm)

Hierfür werden 2 zusätzliche Constraints eingeführt

* Kette waagrecht
* Kette senkrecht
* Ausrichten Breite
* Ausrichten Höhe

Bei den Kette Constraints werden die Knoten in X bzw. Y Richtung so platziert, dass sie genau aufeinander folgen.
Diese Kette Constraints steuern die X bzw. Y Koordinaten der Knoten.
Die Kette Constraints werden im Allgemeinen benutzt verbunden mit einen Senkrecht bzw. Waagrecht Constraint.

Bei den Ausrichten Constraints werden mehrere Knoten in der Breite bzw. in der Höhe auf das Maximum ausgerichtet.
Diese Ausrichten Constraints beeinflussen nicht die Position und habe für das allgemeine Constraint System keine Bedeutung.

Vnurknoten
V9knoten
Vbohrer