

Arbeitsblatt zur Ermittlung von Orbitalelementen aus Zeitserien von Radialgeschwindigkeitsmessungen

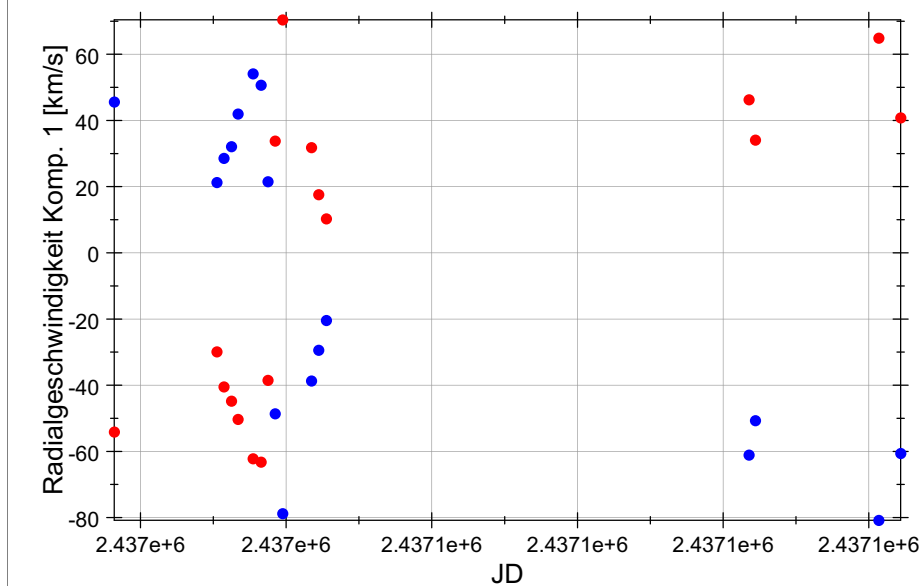
Ermittlung von Orbitalelementen von Doppelsternen aus Radialgeschwindigkeitmessungen am Beispiel veröffentlichter Messwerte von Fehrenbach (1961) über Mizar A (zet UMa)

Die Messwerte werden eingelesen:

```
[ data1:=import::readdata("C:\\Dokumente und Einstellungen\\Lothar Schanne  
[[2437016.413, 45.6], [2437030.48, 21.3], [2437031.451, 28.6], [2437032.49, 32.1], [2437033.376, 42.0], [2437035.351, 52.0],  
data2:=import::readdata("C:\\Dokumente und Einstellungen\\Lothar Schanne\\Eigene  
[[2437016.413, -54.1], [2437030.48, -29.9], [2437031.451, -40.5], [2437032.49, -44.8], [2437033.376, -50.3], [2437035.351, -60.0]
```

und gegen die Zeit geplottet (Komponente 1 blau, Komponente 2 rot):

```
[ plot(plot::Listplot(data1, LinesVisible=FALSE, PointColor=RGB::Blue), plot::Listplot
```



Die mit Peranso unabhängig ermittelte Periode P [d] wird im Sinne einer Voraussetzung eingeführt:

```
[ P:=20.5184  
20.5184
```

Die Messdaten werden auf eine Periode reduziert. Dazu ist ein beliebiger Bezugszeitpunkt T [JD] festzulegen. Hier wird einfach der erste Beobachtungstag benutzt. T kann durch eine additive Konstante so gewählt werden, dass im nachfolgenden Phasendiagramm die Nullstellen innerhalb des Phasenbereichs liegen.

```
[ JDA:=op(data1, [1, 1])  
2437016.413  
[ JDE:=op(data1, [nops(data1), 1])  
2437124.355  
[ T:=JDA-11.5  
2437004.913
```

Damit werden die Messdaten reduziert auf eine Periode:
Komponente 1:

```
[ aa:=map(data1, op, 1):  
[ ab:=map(aa, aa->(aa-T)/P):  
[ ac:=map(ab, ab->ab-map(ab, trunc)):  
[ data1rv:=map(data1, op, 2):
```

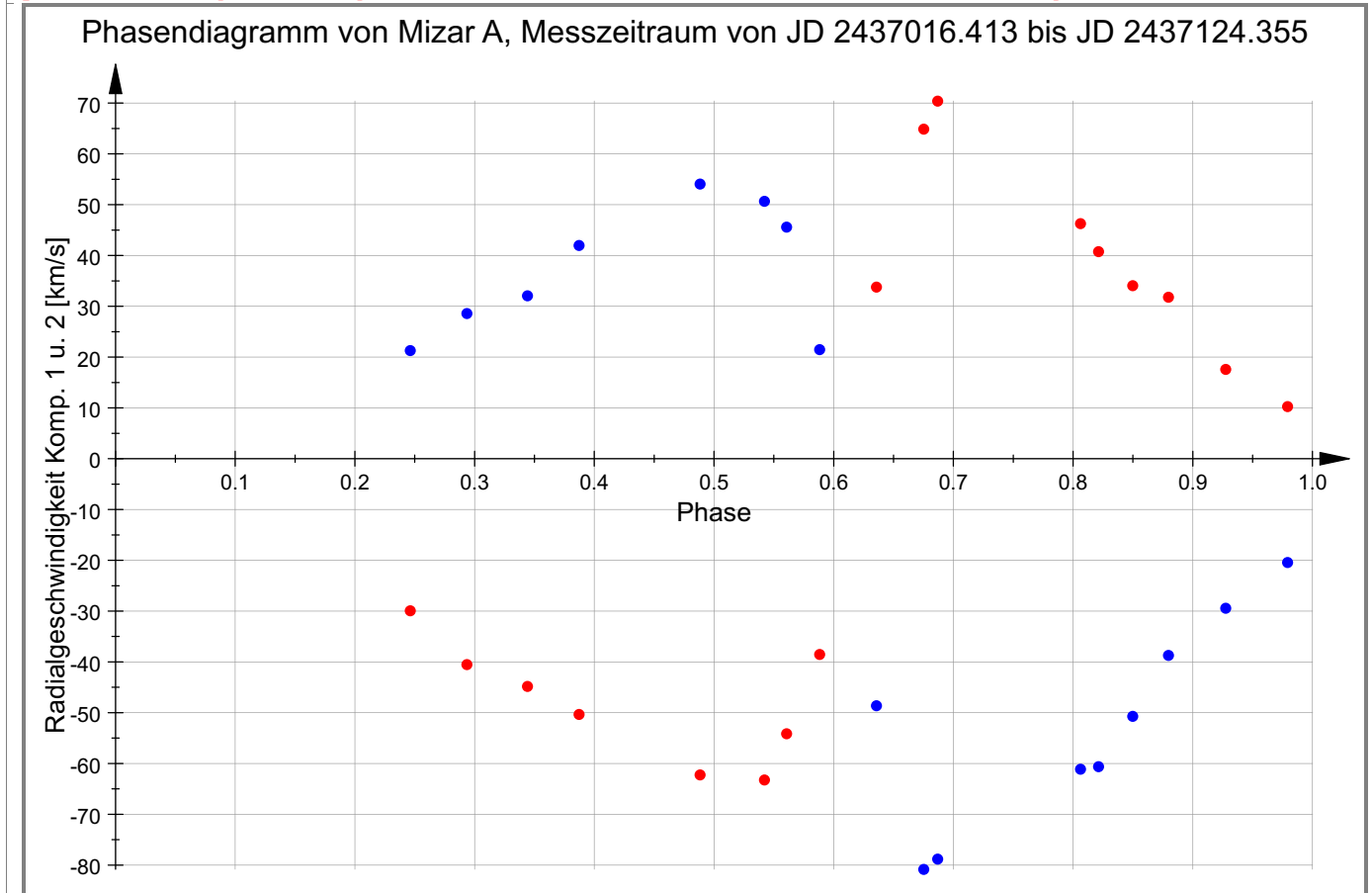
```
[ data1rv:=map(data1,op,2):
[ data1Phasenwerte:=zip(ac,data1rv,(ac,data1rv)->[ac,data1rv]):
```

Komponente 2:

```
[ ba:=map(data2,op,1):
[ bb:=map(ba,ba->(ba-T)/P):
[ bc:=map(bb,bb->bb-map(bb,trunc)):
[ data2rv:=map(data2,op,2):
[ data2Phasenwerte:=zip(bc,data2rv,(bc,data2rv)->[bc,data2rv]):
```

Graphik:

```
[ Ueberschrift:="Phasendiagramm von Mizar A, Messzeitraum von JD ".expr2te.
[ plot(ViewingBoxXRange=0..1,XAxisTitle="Phase",XAxisTitleAlignment=Center
```



Aus diesem Diagramm wird visuell die maximale und die minimale Geschwindigkeit der Komponenten 1 (blau) und 2 (rot) abgeschätzt und nachfolgend in das Berechnungsblatt eingetragen:

```
[ V1max:=54: V1min:=-80.5: V2max:=70.5: V2min:=-63:
```

Die mittleren Geschwindigkeiten V1m und V2m [km/s] betragen dann

```
[ V1m:=1/2*(V1max+V1min): V2m:=1/2*(V2max+V2min)
```

-13.25

3.75

Dann berechnen sich die Konstanten **K1** und **K2** [km/s] zu

```
[ K1:=V1max-V1m;K2:=V2max-V2m
```

67.25

2

66.75

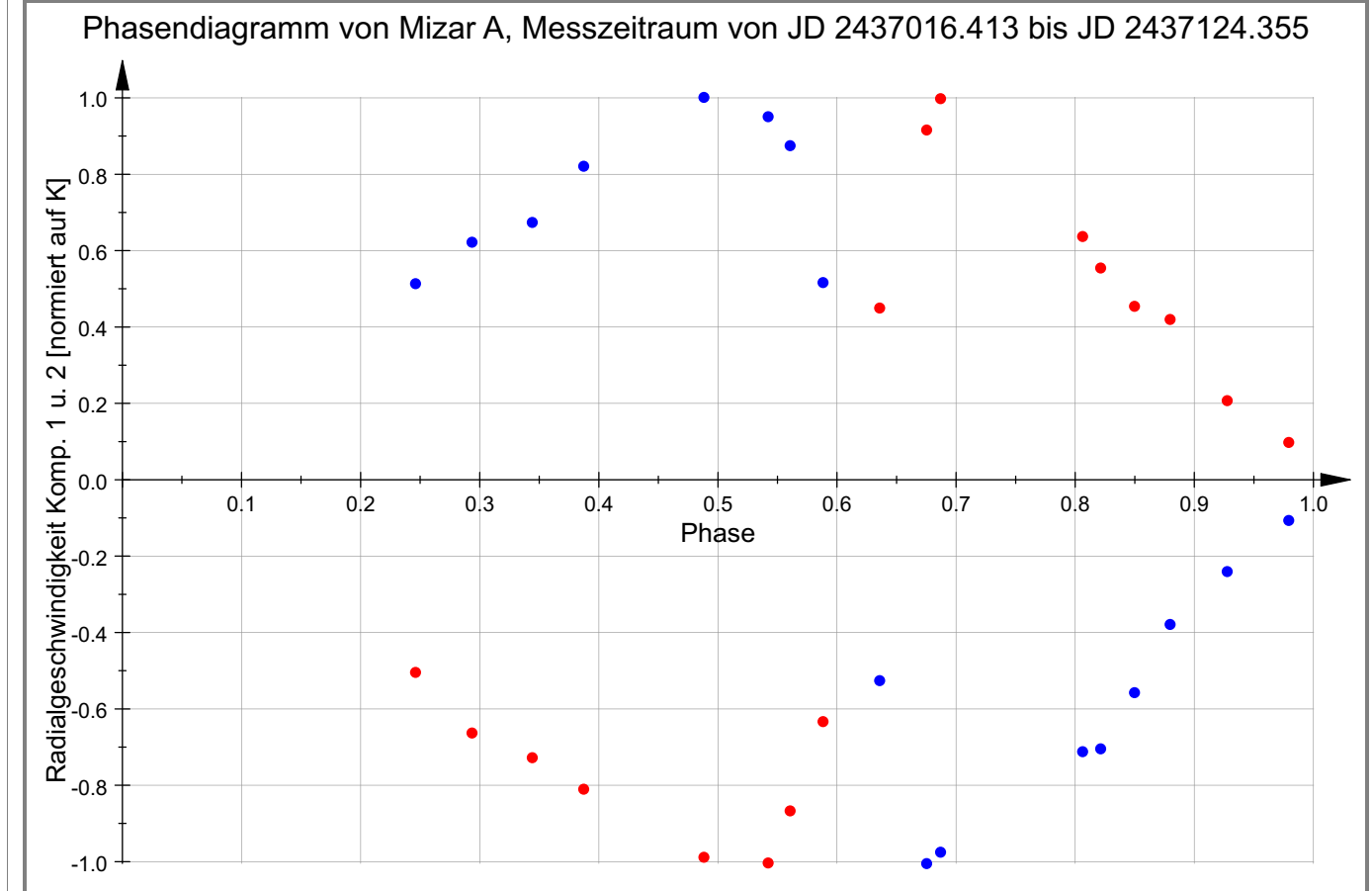
Alle RV's der Komponenten werden jetzt auf die mittlere Geschwindigkeit V1m bzw. V2m bezogen

Alle RV's der Komponenten werden jetzt auf die mittlere Geschwindigkeit V1m bzw. V2m bezogen und auf K1 bzw. K2 normiert (das sind dann die Werte von $\cos(v+\omega)$):

```
ad:=map(data1,op,2): bd:=map(data2,op,2):
ae:=map(ad,ad->(ad-V1m)/K1): be:=map(bd,bd->(bd-V2m)/K2):
data1red:=zip(ac,ae,(ac,ae)->[ac,ae]): data2red:=zip(bc,be,(bc,be)->[bc,
[[0.5604725515, 0.8750929368], [0.2460523238, 0.5137546468], [0.2933757018, 0.6223048327], [0.3440131784, 0
```

Damit lässt sich ein neues Phasendiagramm plotten, aus dem die vorberechneten Werte nach Irwin entnommen werden können.

```
plot(ViewingBoxXRange=0..1,XAxisTitle="Phase",XAxisTitleAlignment=Center,YAxisTit
```



Nach Anwendung der graphischen Methode nach Irwin ergibt sich aus den im obigen Diagramm ablesbaren Größen für Komponente 1

$a = 0, b = 0.35, c = 0.48, d = 0.57, e = 0.61, f = 0.65, g = 0.67, h = 0.82, i = 1.02$

die **Schätzwerte für $\omega_1 = 110 \pm 20^\circ$ und $e_1 = 0.48 \pm 0.035$** (Details hier nicht darstellbar).

Damit lässt sich die **Systemgeschwindigkeit SV1** berechnen:

```
e1:=0.48:omega11:=110*PI/180:
SV1:=float(V1m-K1*e1*cos(omega11))
-2.209589773
```

Für die Komponente 2 ergibt die Irwin-Methode:

$a=0.61, b=0.66, c=0.685, d=0.78, e=1.02, f=0.31, g=0.54, h=0.58, i=0.62.$

$\omega_2 = 285 \pm 10^\circ, e_2 = 0.50 \pm 0.05.$

Damit lässt sich die **Systemgeschwindigkeit SV2** berechnen:

```
e2:=0.50:omega22:=285*PI/180:
SV2:=float(V2m-K2*e2*cos(omega22))
-4.88808563
```

Die ω -Werte [$^\circ$] werden sofort durch mitteln verbessert, weil sie naturgesetzlich um 180° unterschieden sind:

unterschieden sind:

```
[ omega1:=float(omega22+omega11-PI)/2; omega2:=float(omega1+PI):  
float(omega1*180/PI); float(omega2*180/PI);  
107.5  
287.5
```

Berechnung der zu den ermittelten Werten gehörenden Modellkurven

Die zeitliche Schrittweite S zur Berechnung der exzentrischen Anomalie EA wird in Tagesteilen festgelegt.

S = 1 bedeutet 1 Tag. Andere Werte können eingegeben werden. Z.B. S = 24 entspricht 1 h, S = 4 entspricht 24/4 = 4h.

```
[ S:=4:
```

Anfangs- und Enddatum werden so modifiziert, dass jeweils die komplette Periode in die Berechnung und grafische Darstellung aufgenommen wird (beide bezogen auf T). PA ist die Anfangs-Periode und PE die Endperiode des Beobachtungszeitraums nach T.

```
[ PA:=trunc((JDA-T)/P)-1; PE:=trunc((JDE-T)/P)+1:
```

Die Schrittzahl SA ist eine Größe zur Beurteilung der Auflösung der Modellkurven. Sie wird aus den bisher eingegebenen Daten berechnet.

```
[ SA:=round(S*(PE-PA)*P)
```

```
575
```

Berechnung der exzentrischen Anomalien EA [rad] als Werteliste in der oben definierten Schrittzahl SA:

```
[ EA:=float([i/SA*2*PI*(PE-PA) $ i = 0..SA]):
```

Den zu den diskreten EA-Werten zugeordneten wahren Anomalien v [rad] und Zeitpunkte t [JD] werden berechnet.

```
[ v:=float(map(EA, EA->2*arctan(tan(EA/2)*sqrt(1+e1)/sqrt(1-e1)))):
```

```
[ t:=float(map(EA, EA->T+P*PA+(P*(EA-e1*sin(EA))/(2*PI)))):
```

Und daraus die beobachtbaren Modell-Radialgeschwindigkeiten vr1 und vr2 beider Komponenten:

```
[ vr1:=float(map(v, v->SV1+ K1*(e1*cos(omega1)+cos(omega1+v))):
```

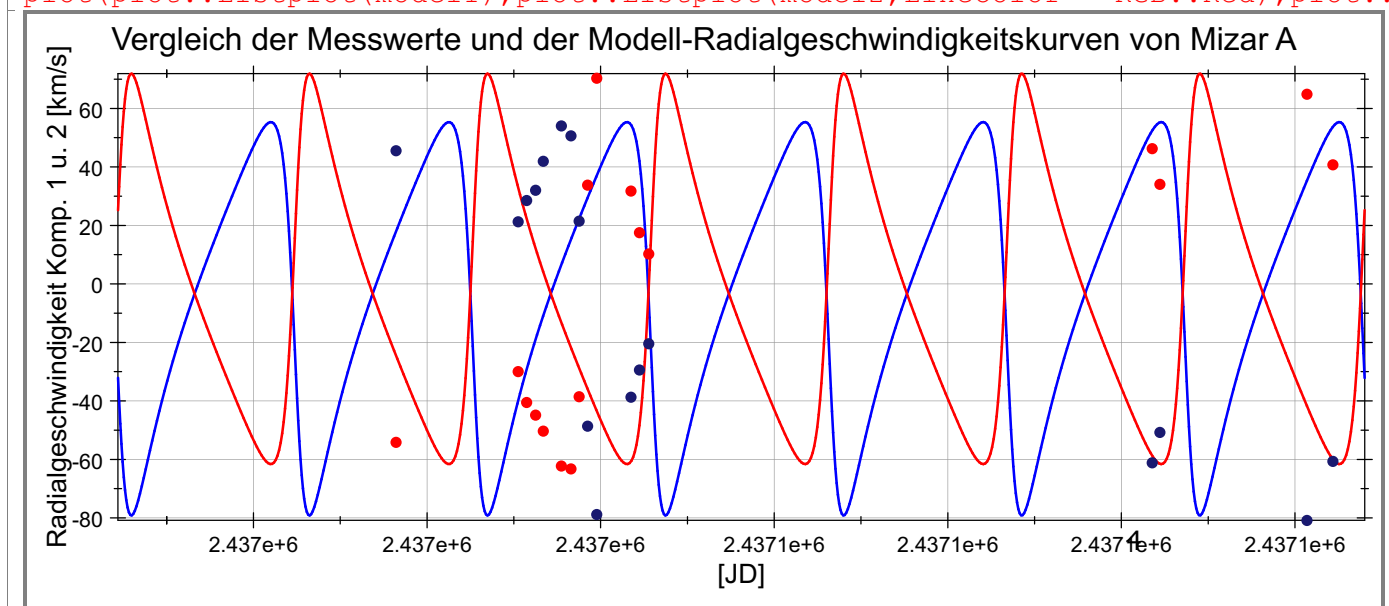
```
[ vr2:=float(map(v, v->SV2+ K2*(e2*cos(omega2)+cos(omega2+v))):
```

```
[ modell:=zip(t, vr1, (t, vr1)->[t, vr1]):
```

```
[ model2:=zip(t, vr2, (t, vr2)->[t, vr2]):
```

Zusammenschau der Messwerte und der Radialgeschwindigkeitskurven wie bisher bestimmt:

```
[ plot(plot::Listplot(modell), plot::Listplot(model2, LineColor = RGB::Red), plot::Poi
```



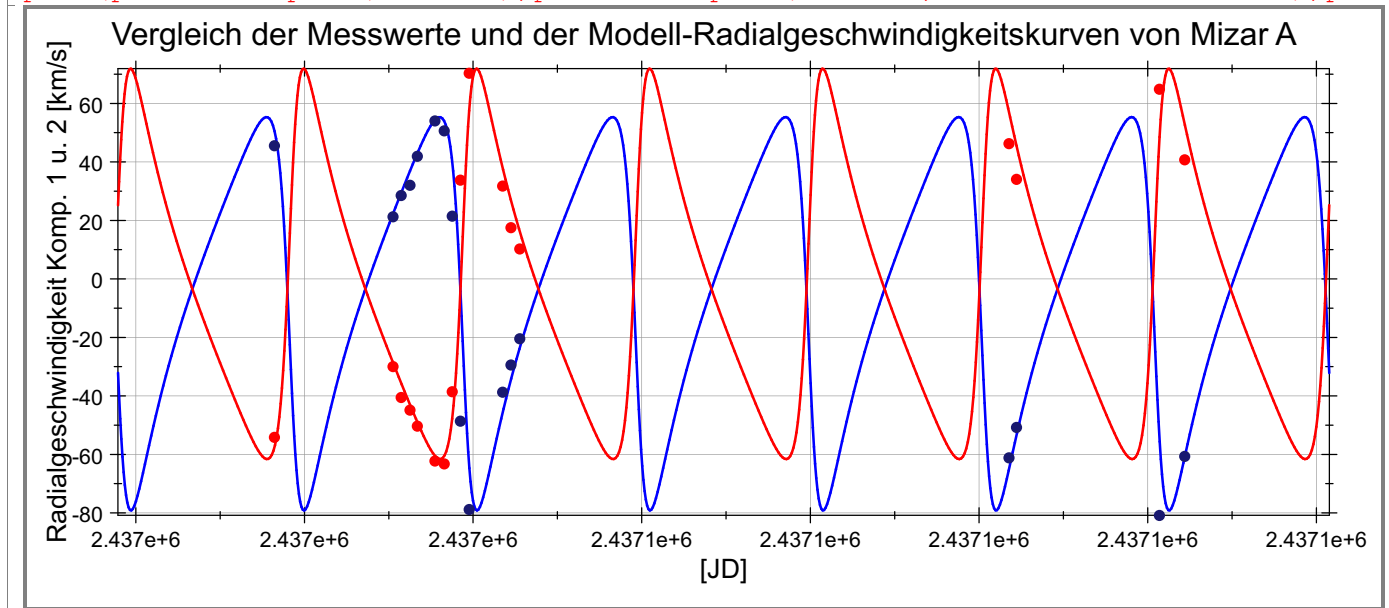
Man erkennt deutlich, dass die Modellkurven zu den Messwerten verschoben sind. Der Grund liegt in der willkürlichen Wahl vom Bezugszeitpunkt T. T soll jetzt angepasst werden, damit die

in der willkürlichen Wahl vom Bezugszeitpunkt T. T soll jetzt angepasst werden, damit die Übereinstimmung optimal ist. Die Anpassung erfolgt in der nächsten Zeile durch eine frei wählbare additive Konstante, welche den neuen Bezugszeitpunkt Tb (Periastrondurchgang) festlegt. Sie wird so lange variiert, bis die Deckung der Messwerte mit den berechneten Kurven in der nachfolgenden Grafik optimal ist.

```
Tb:=T+13.5
2437018.413
tb1:=float(map(EA,EA->Tb+P*PA+(P*(EA-e1*sin(EA))/(2*PI)))):
tb2:=float(map(EA,EA->Tb+P*PA+(P*(EA-e2*sin(EA))/(2*PI)))):
modell1b:=zip(tb1,vr1,(tb1,vr1)->[tb1,vr1]):
modell2b:=zip(tb2,vr2,(tb2,vr2)->[tb2,vr2]):
```

Damit wird die Konkruenz von Modell und Messwertekollektiv wesentlich besser:

```
plot(plot::Listplot(modell1b),plot::Listplot(modell2b,LineColor = RGB::Red),plot::P
```



Der bisher aus den Messdaten der beiden Komponenten ermittelte Satz der Bahnelemente steht für eine least squares Verbesserung (differentielle Korrektur) zur Verfügung:

Bezugszeitpunkt (Periastrondurchgang) T [JD]:

```
Tb
2437018.413
```

Periode P [d]

```
P
20.5184
```

K1 und K2 [km/s]

```
K1;K2
67.25
66.75
```

Exzentrizität e:

```
e1; e2
0.48
0.5
```

Länge des Periastrons omega1 und 2 [°]

```
float(omega1*180/PI); float(omega2*180/PI)
107.5
```

107.5

287.5

[