Présentation du projet :

Correction des défauts géométriques d'une monture de télescope

Association VOYAGER 3 Astronomie
- Espace culturelle de la Garenne - 44530 Sévérac -

Association (Loi 1901) pour l'initiation et la pratique de l'astronomie,
Agrément Jeunesse & Education Populaire N°44-08-12

Commission technique de VOYAGER 3 Astronomie
Sébastien POIRIER

SOMMAIRE

I) Présentation et principes généraux

Une monture de télescope est en ensemble mécanique supportant et dirigeant l'axe optique de l'instrument.

Il existe différents types de monture que l'on peut regrouper en deux catégories. Les montures azimutales et les montures équatoriales. Ces deux catégories se subdivisent elles mêmes en diverses solutions techniques en fonction des choix géométriques (ou électronique pour les montures altazimutales)

Toutes c'est solutions ont un point commun: L'existence de deux axes de rotations théoriquement orthogonales l'un à l'autre. En effet, l'objectif est de diriger l'axe optique du télescope vers un point précis de la voûte céleste. La voûte céleste étant une surface sphérique, chacun de ses points est repérables par deux coordonnées.

Le document suivant est rédigé pour une monture équatoriale. On pourra facilement l'adapter pour une monture azimutale.

On se place dans le cas où le télescope est équipé d'un système d'encodeurs mesurant les positions angulaires des deux axes de la montures

Dans un premier temps, nous allons définir les liens entre les angles mesurés à l'aide des encodeurs de la monture et la directions réellement pointé par le télescope.

Nous verrons que les différences entre les coordonnées indiquées par les encodeurs et les coordonnées réelles des objets proviennent à la fois des défauts d'alignement avec l'axe polaire, des défauts d'orthogonalité des axes des rotations du télescope et finalement du défaut de coaxialité de l'axe mécanique et de l'axe optique du tube du télescope. Ces défauts introduiront au totale 5 paramètres permettant de passer des coordonnées indiquées par la monture aux coordonnées réellement pointées, et inversement. Il faut de plus ajouter une origine de valeur aux encodeurs ce qui introduit deux paramètres supplémentaires. Deux paramètres sont totalement liés (origine de l'encodeur de déclinaison et décalage en déclinaison de l'axe optique) ainsi nous montrons que la détermination de 6 paramètres est suffisante.

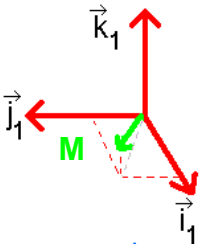
Dans un deuxième temps, nous présenterons le protocole puis l'algorithme permettant de déterminer ces 6 paramètres.

II) Mise en équation

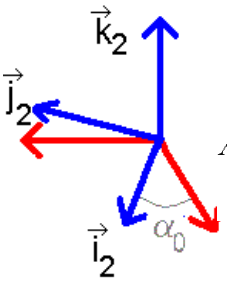
Dans cette partie, nous allons déterminer les relations entre les coordonnées indiquées par la monture du télescope **(α , δ)** et les coordonnées réelles de l'objet observé **(α_v , δ_v)**. Pour tenir compte du temps sidéral local (θ_{TLS}=**TLS** x π **/12** , **TSL** exprimé en heure décimale) il faudra substituer dans les expressions de ce chapitre, α et α_v respectivement par $\alpha+\theta_{TLS}$ et par $\alpha_v+\theta_{TLS}$.

- **Relation entre les systèmes de coordonnées**

Le vecteur M indique la direction de l'objet observé de coordonnées (α_v, δ_v). Le repère 1 correspond (exactement) à celui des coordonnées célestes. *(A droite, les coordonnée de M dans le repère 1).*

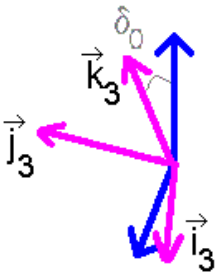


Repère 2 : Erreur d'alignement de l'axe polaire en « α ». Rotation par rapport au repère 1 avec un angle α_0 autour de l'axe $\mathbf{k_1}$ (pôle nord céleste). *(A droite, matrice **A** associée au passage du repère 1 au repère 2).*



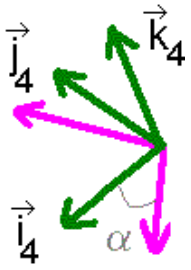
$$A = \begin{pmatrix} \cos\alpha_0 & -\sin\alpha_0 & 0 \\ \sin\alpha_0 & \cos\alpha_0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Repère 3 : Erreur d'alignement de l'axe polaire en « δ ». Rotation par rapport au repère 2 avec un angle δ_0 autour de l'axe $\mathbf{j_2}$. *(A droite, matrice **B** associée au passage du repère 2 au repère 3).*



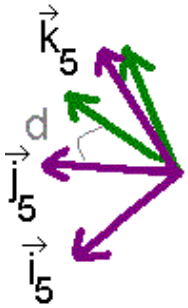
$$B = \begin{pmatrix} \cos\delta_0 & 0 & \sin\delta_0 \\ 0 & 1 & 0 \\ -\sin\delta_0 & 0 & \cos\delta_0 \end{pmatrix}$$

Repère 4 : Coordonnée α selon la monture du télescope. Rotation par rapport au repère 3 avec un angle α autour de l'axe $\mathbf{k_3}$. *(A droite, matrice **C** associée au passage du repère 3 au repère 4).*



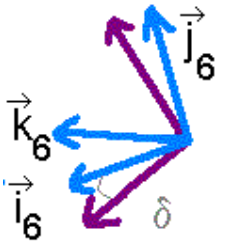
$$C = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Repère 5 : Erreur de perpendicularité de l'axe α par rapport à l'axe δ . Rotation par rapport au repère 4 avec un angle **d** autour de l'axe $\mathbf{i_4}$. *(A droite, matrice **D** associée au passage du repère 4 au repère 5).*



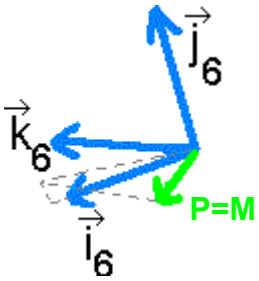
$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos d & -\sin d \\ 0 & \sin d & \cos d \end{pmatrix}$$

Repère 6: Coordonnée δ selon la monture du télescope. Rotation par rapport au repère 5 avec un angle **δ** autour de l'axe $\mathbf{j_5}$. Afin de simplifier les expression s (obtention des termes $\delta+\delta_2$), on transforme $\mathbf{j_5}$ en $\mathbf{j_6}$. La base obtenu est orthornormale indirecte. *(A droite, matrice **E** associée au passage du repère 5 au repère 6).*



$$E = \begin{pmatrix} \cos\delta & -\sin\delta & 0 \\ 0 & 0 & 1 \\ \sin\delta & \cos\delta & 0 \end{pmatrix}$$

Finalement, le vecteur P indique la direction de l'objet observé dans le repère 6.Le vecteur P correspond à l'axe optique. δ_2 et γ expriment l'erreur d'alignement de l'axe optique et mécanique.



$$P = \begin{pmatrix} \cos\delta_2 \cos\gamma \\ \sin\delta_2 \cos\gamma \\ \sin\gamma \end{pmatrix}$$

Ainsi,

M = ABCDEP

On pose,

X = CDEP donc **M=ABX**

Avec :

$$X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

D'où

$$\begin{aligned} x &= \cos(\delta+\delta_2)\cos\gamma\cos\alpha - \Big(\sin\gamma\cos d - \cos\gamma\sin d\sin(\delta+\delta_2)\Big)\sin\alpha \\ y &= \cos(\delta+\delta_2)\cos\gamma\sin\alpha + \Big(\sin\gamma\cos d - \cos\gamma\sin d\sin(\delta+\delta_2)\Big)\cos\alpha \\ z &= \sin(\delta+\delta_2)\cos\gamma\cos d + \sin\gamma\sin d \end{aligned}$$

Et finalement,

$$\begin{aligned} \cos\alpha_v\cos\delta_v &= x\cos\delta_0\cos\alpha_0 - y\sin\alpha_0 + z\sin\delta_0\cos\alpha_0 \\ \sin\alpha_v\cos\delta_v &= x\cos\delta_0\sin\alpha_0 + y\cos\alpha_0 + z\sin\delta_0\sin\alpha_0 \\ \sin\delta_v &= -x\sin\delta_0 + z\cos\delta_0 \end{aligned}$$

Remarquons que l'introduction du paramètre δ_2 revient à un changement d'origine pour l'axe des déclinaisons. Ainsi, cette valeur est confondue avec le décalage d'origine de l'encodage en déclinaison. Pour la suite, nous substituons donc $\delta+\delta_2$ par **δ** (ce qui revient à prendre δ_2 **= 0**).

- **Expression de (α_v , δ_v) en fonction de(α , δ) :**

On pose,

$$M = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \quad \text{donc} \quad \begin{aligned} a &= x\cos\delta_0\cos\alpha_0 - y\sin\alpha_0 + z\sin\delta_0\cos\alpha_0 \\ b &= x\cos\delta_0\sin\alpha_0 + y\cos\alpha_0 + z\sin\delta_0\sin\alpha_0 \\ c &= -x\sin\delta_0 + z\cos\delta_0 \end{aligned}$$

Or

$$M = \begin{pmatrix} \cos\alpha_v\cos\delta_v \\ \cos\alpha_v\sin\delta_v \\ \sin\delta_v \end{pmatrix}$$

Ainsi,

$$\delta_v = \arcsin c$$

Si $\dfrac{a}{\cos(\delta_v)} > 0$ alors $\alpha_v = \arcsin\left(\dfrac{b}{\cos\delta_v}\right)$

sinon $\alpha_v = \pi - \arcsin\left(\dfrac{b}{\cos\delta_v}\right)$

- **Expression de (α , δ) en fonction de (α_v , δ_v) :**

Nous avons vu que **M = ABX**. Ainsi, **$X = B^{-1}A^{-1}M$**

D'après l'expression des matrices **A** et **B**, nous obtenons :

$$A^{-1} = \begin{pmatrix} \cos\alpha_0 & \sin\alpha_0 & 0 \\ -\sin\alpha_0 & \cos\alpha_0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad B^{-1} = \begin{pmatrix} \cos\delta_0 & 0 & -\sin\delta_0 \\ 0 & 1 & 0 \\ \sin\delta_0 & 0 & \cos\delta_0 \end{pmatrix}$$

Ainsi,

$$B^{-1}A^{-1} = \begin{pmatrix} \cos\alpha_0\cos\delta_0 & \sin\alpha_0\cos\delta_0 & -\sin\delta_0 \\ -\sin\alpha_0 & \cos\alpha_0 & 0 \\ \cos\alpha_0\sin\delta_0 & \sin\alpha_0\sin\delta_0 & \cos\delta_0 \end{pmatrix}$$

On pose

$$B^{-1}A^{-1}M = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

D'où, après simplification :

$$\begin{aligned} a &= \cos\delta_0 \cos\delta_v \cos(\alpha_v - \alpha_0) - \sin\delta_0 \sin\delta_v \\ b &= \cos\delta_v \sin(\alpha_v - \alpha_0) \\ c &= \sin\delta_0 \cos\delta_v \cos(\alpha_v - \alpha_0) + \cos\delta_0 \sin\delta_v \end{aligned}$$

La coordonnée **« z »** du vecteur **X** nous donne :

$$\sin\delta = \frac{c - \sin d \sin\gamma}{\cos\gamma \cos d}$$

Si $\frac{c - \sin d \sin\gamma}{\cos\gamma \cos d} > 1$, il n'y pas de solution (zone du ciel ne pouvant être pointée).

Sinon

$$\delta = \arcsin\left(\frac{c - \sin d \sin\gamma}{\cos\gamma \cos d}\right)$$

On pose

$$\begin{aligned} a_1 &= \cos\gamma \cos\delta \\ b_1 &= \cos\gamma \sin d \sin\delta - \sin\gamma \cos d \end{aligned}$$

Les coordonnées **« x »** et **« y »** du vecteur **X** nous donne :

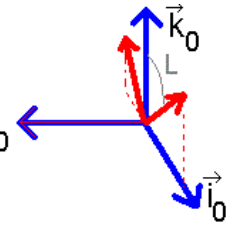
Si $(aa_1 - bb_1 \geq 0)$ alors $\alpha = \arcsin\left(\frac{ba_1 + ab_1}{a_1^2 + b_1^2}\right)$

sinon $\alpha = \pi - \arcsin\left(\frac{ba_1 + ab_1}{a_1^2 + b_1^2}\right)$

- **Conversion des paramètres (α_0 , δ_0) en paramètre exprimant les erreurs de latitude et d'azimut (Δl , Δa).**

L est la latitude du lieu d'observation.
j_0 est orienté vers l'Ouest.
k_0 est orienté vers le pôle Nord céleste.
(**i_0**, **j_0**, **k_0**) repère orthonormal.

$$F = \begin{pmatrix} \sin L & 0 & -\cos L \\ 0 & 1 & 0 \\ \cos L & 0 & \sin L \end{pmatrix}$$



$$M_0 = F\,A\,B \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Ainsi,

cos (Δa) cos (Δl +L) = cos α_0 sin δ_0 sin L - cos δ_0 cos L
sin (Δa) cos (Δl +L) = sin α_0 sin δ_0
sin (Δl +L) = cos α_0 sin δ_0 cos L + cos δ_0 sin L

D'où

$$\Delta l = \arcsin(\cos \delta_0 \sin L + \cos \alpha_0 \sin \delta_0 \cos L) - L$$

Et, en supposant $|\Delta a| < \pi/2$,

$$\Delta a = \arcsin \frac{\sin \alpha_0 \sin \delta_0}{\cos(\Delta l + L)}$$

Si $\Delta a > 0$ l'axe horaire doit être déplacer plus vers l'est. Si $\Delta l > 0$ l'axe horaire doit être déplacer plus vers le zénith.

- **Programmation C**

- Définition des paramètres de correction de la monture :

alpha_0	: erreur d'alignement de l'axe polaire en ascension droite
delta_0	: erreur d'alignement de l'axe polaire en déclinaison
d	: erreur de perpendicularité des axes d'ascension droite et de déclinaison
gamma	: erreur de coaxialité de l'axe optique et de l'axe mécanique du télescope.
alpha_co	: décalage de l'origine de l'encodeur d'ascension droite
delta_co	: décalage de l'origine de l'encodeur de déclinaison

De plus on associe au paramètre du modèle la valeur de la longitude, utilise notamment pour le calcul du temps sidéral local, ainsi que la latitude du lieu. Le paramètre chi2 quantifie la pertinence du modèle:

```
typedef struct S_parameterMount{

  double alpha_0;
  double delta_0;
  double d;
  double gamma;
  double alpha_co;
  double delta_co;

  double longitude;
  double latitude;
  double chi2;

} ParameterMount;
```

Chaque point de la voûte céleste est repéré par la donnée de ses deux coordonnées (alpha, delta). On ajoute de plus un paramètre de gestion des erreurs.

```
typedef struct S_coord{

  double alpha;
  double delta;
  int    erreur;
} coord
```

Procédure de conversion des angle réels en les valeurs mesurées par les encodeurs :

```
Coord OutToIn_Mount(Coord cv, ParameterMount p)
{
	Coord co;
	double delta,alpha,a,b;
	double cg,cd,cdv,cd0,caa0;
	double sg,sd,sdv,sd0,saa0,sco;
	double a1,b1;

	cdv  = cos( cv.delta  );
	sdv  = sin( cv.delta  );
	caa0 = cos( cv.alpha - p.alpha_0 );
	saa0 = sin( cv.alpha - p.alpha_0 );

	cg   = cos( p.gamma   );
	sg   = sin( p.gamma   );
	cd   = cos( p.d       );
	sd   = sin( p.d       );
	cd0  = cos( p.delta_0 );
	sd0  = sin( p.delta_0 );


	sco   = (sd0*cdv*caa0 + cd0*sdv - sd*sg)/(cg*cd);

	if (fabs(sco)<=1)
	{
		delta = asin( sco );

		if (fabs(delta)>0.999997*PI/2)
		{
			if (delta>0) delta= PI/2;
		   else		delta=-PI/2;
			alpha=0.;
		}
		else
		{
			a = cd0*cdv*caa0 - sd0*sdv;
			b =     cdv*saa0         ;

			a1=  cg*cos(delta);
			b1=  cg*sd*sco - sg*cd;

			alpha = asin( (cdv*saa0*a1+a*b1)/(a1*a1+b1*b1) );

			if (a*a1-b*b1<0) alpha = PI-alpha;

		}

		co.alpha = modulo( alpha - p.alpha_co, DeuxPI);
		co.delta = modulo( delta - p.delta_co, DeuxPI);
		co.erreur = 0;
	}
	else co.erreur = 1;

	return(co);
}
```

Procédure de correction géométrique des valeurs données par les encodeurs :

```
Coord InToOut_Mount(Coord co, ParameterMount p)
{
	Coord  cv;
	double delta,alpha,a,b,c,x,y,z;
	double cdd,ca,cg,cd,cd0,ca0;
	double sdd,sa,sg,sd,sd0,sa0;

	cdd = cos( co.delta + p.delta_co );
	sdd = sin( co.delta + p.delta_co );
	ca  = cos( co.alpha + p.alpha_co );
	sa  = sin( co.alpha + p.alpha_co );

	cg  = cos(p.gamma);
	sg  = sin(p.gamma);
	cd  = cos(p.d);
	sd  = sin(p.d);
	cd0 = cos(p.delta_0);
	sd0 = sin(p.delta_0);
	ca0 = cos(p.alpha_0);
	sa0 = sin(p.alpha_0);

	x=cdd*ca*cg - sa*sg*cd + sa*cg*sd*sdd;
	y=cdd*sa*cg + ca*sg*cd - ca*cg*sd*sdd;
	z=sdd*cg*cd +   sg*sd;

	a=  x *cd0*ca0 - y *sa0 + z *sd0*ca0;
	b=  x *cd0*sa0 + y *ca0 + z *sd0*sa0;
	c= -x *sd0             +z *cd0;

	if (fabs(c)<=1)
	{
		delta=asin(c);

		if (fabs(delta)>0.999997*PI/2)
		{
			if (delta>0) delta= PI/2;
		   else		delta=-PI/2;
			alpha=0.;
		}
		else
		{
			alpha=asin(b/cos(delta));
			if (a/cos(delta)<0) alpha= PI-alpha;
		}

		cv.alpha = modulo( alpha, DeuxPI);
		cv.delta = modulo( delta, DeuxPI);
		cv.erreur= 0;
	}
	else cv.erreur=1;

	return(cv);
}
```

III) Détermination des paramètres géométriques

Sur la voûte céleste, une étoile est repère pas 2 coordonnées (ascension droite et déclinaison). Dans le cas parfait où il n'y aurait aucune imprécision sur les mesures réaliser par les encodeurs, il suffirait de considérer la position de 3 étoiles et comparer les valeurs des coordonnées données par l'encodeur par rapport aux coordonnées réelles de ces étoiles. Nous obtiendrions ainsi 3x2=6 équations à 6 inconnues, système dont on peut trouver une solution.

Dans la pratique, il existe toujours une incertitude sur les mesures. Ainsi, l'observation de 3 étoiles pour la détermination des paramètres de corrections est un strict minimum, minimum qui s'avère parfois insuffisant. Les testes réalisés sur la méthode décrite dans la suite de ce document montre que l'observation de 6 étoiles conduit à la détermination avec une précision suffisante des 6 paramètres modèle. Bien sur, rien n'interdit d'utiliser plus d'étoiles. Plus on utilise d'étoiles, meilleurs est la détermination de la valeur numérique des paramètres. Toutefois la durée de calcule pour la détermination de ces paramètres peut croître de façon importante à mesure que l'on considère plus d'étoile.

Ainsi, le choix de six étoiles parait être un bon compris.

- **Protocole (description pour l'observation de 6 étoiles)**

  DEBUT

  1.1) Pointer le télescope en direction d'une étoile. Placer cette étoile au centre du champ visuel.
  1.2) Mémoriser les coordonnées indiquées par les encodeurs du télescope
  1.3) Mémoriser les coordonnées réelles de cette étoile (utilisations d'un catalogue)

  2) Itérer les étapes 1 pour les 6 étoiles de référence (choisir ces étoiles de tel façon que soit équitablement répartie sur l'ensemble de la voûte céleste accessible par le télescope).

  3) Lancer le programme de détermination des 6 paramètres

  FIN

Le programme de détermination des 6 paramètres va mémoriser les valeurs dans un fichier qui sera par la suite utiliser par le logiciel de pointage. Apres cette manipulation, le logiciel de pilotage du télescope convertira automatiquement les coordonnées données par les encodeurs en coordonnées réellement observée. Les défauts géométriques n'apparaîtront donc plus au yeux de l'utilisateur.

IV) Algorithme de recherche des paramètres géométriques

Introduction

La méthode de détermination des paramètres repose sur une méthode de minimisation des erreurs du type méthode des moindres carrée. Nous allons minimiser la distance angulaire entre les coordonnées réelles de chaque étoile et les coordonnées estimer par le modèle en fonction des données des encodeurs. L'algorithme de convergence, après une procédure d'initialisation, est itératif.

DEBUT

1) Phase d'initialisation

2) Cartographie bi-dimentionnelle d'une zone de solution
2.1) A partir d'un modèle donné, on modifie 2 paramètres.
2.2) Détermination du meilleur modèle selon la méthode de Levenberg-Marquardt en fixant les 2 paramètres précédent choisi.
Itération de la cartographie afin de couvrir une zone de valeurs des 2 paramètres

3) Choix d'une nouvelle zone de recherche en considérant le meilleur modèle de la zone cartographiée. Diminution de la prochaine zone de recherche en fonction de la pertinence du nouveau modèle.
Si la zone de recherche est suffisamment petite on passe a l'étape 4 sinon on retourne a l'étape 2

4) Si la zone de recherche n'a pas diminué, on l'augmente jusqu'à une zone de recherche de taille maximale. Si cette zone de recherche maximale est atteinte c'est terminé sinon on retourne à l'étape 2.

FIN

L'étape 4 est utile pour limiter le problème de minima relatifs, problème difficilement géré par la méthode de Levenberg-Marquardt.

1) Description de la phase d'initialisation

Pour l'initialisation, on suppose dans un premier temps qu'il n'y aucun défaut géométrique ou d'alignement sur l'axe polaire. En d'autre terme les 6 paramètres ont une valeur nul.

Le décalage le plus grossier est sur l'origine de l'encodeur. C'est un simple décalage à l'origine (offset) qui n'affecte pas les mesures relatives mais uniquement les mesure absolues. On estime le décalage de l'encodeur en ascension droite en effectuant la moyenne de la différence entre la mesure de l'encodeur et la valeur réelle donnée par le catalogue d'étoiles. On fait de même pour l'encodeur de déclinaison.

On détermine le carrée de la différence angulaire moyenne quadratique entre les coordonnées données par le modèle et les coordonnées réelles des étoiles de références. Cette valeur servira à évaluer la pertinence du modèle. En effet plus cette valeur sera petite plus les valeurs des 6 paramètres seront proches de la réalité. Cette valeur correspond au χ^2 pour notre méthode des moindres carrées.

2) Description de la phase de cartographie

Lors des testes, nous avons remarqué que la méthode de Lenvart-Marquart convergeaient très régulièrement vers un meilleur modèle qui différait fortement du vrai modèle. Il est apparu qu'en fixant la valeur de deux paramètres (désalignement en ascension droite de l'axe polaire alpha_0 et décalage de l'encodeur en ascension droite alpha_co) le problème disparaissaient et que la méthode de Levenberg-Marquardt, moyennant un nombre raisonnable d'étoiles de références, convergeaient vers la solution attendue. Cette constatation nous a mener a considérer une méthode combinant cartographie (mapping) et méthode Lenvart-Marquart.

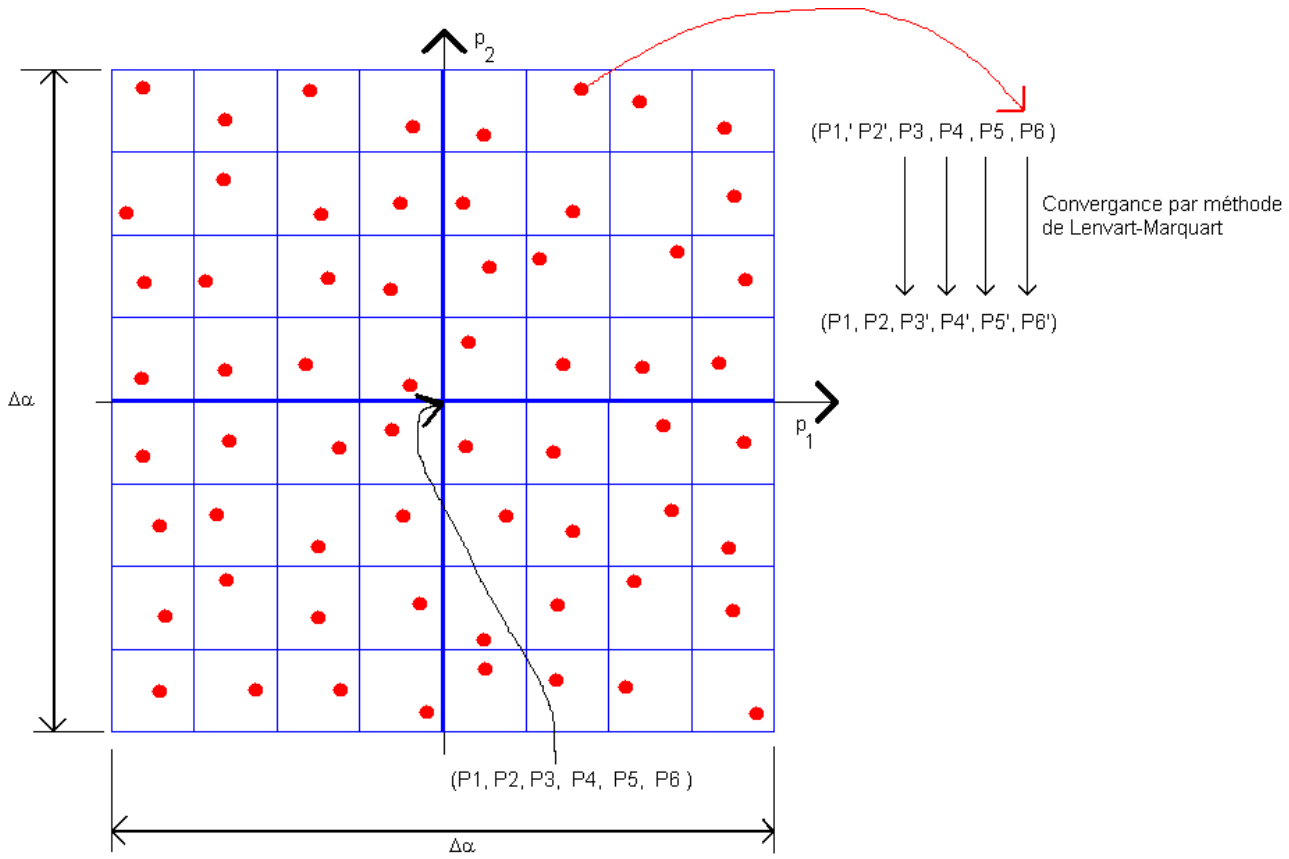
Le principe de la cartographie et de tester différence modèle en faisant varier la valeur de deux paramètres (alpha_co et alpha_0). Une fois fixé c'est deux paramètre, la méthode de Levenberg-Marquardt fait varié les quatre autre paramètre afin de trouver le meilleur modèle sur ces 6 paramètres dont 2 fixes (voir schéma ci-après).

Afin que l'algorithme converge, la zone cartographié (c'est à dire la zone de sélection de alpha_co et alpha_0) doit être de plus en plus petite. La taille de la zone cartographié doit diminué à mesure que le meilleur modèle est de plus en plus proche de la réalité. Pour des raisons de simplicité, la zone cartographié correspond à la selection d'un intervalle de valeur pour alpha_co et un intervalle de valeur pour aplha_0. Les longueurs respectives de ces deux intervalles vont être dépendant de notre indicateur de pertinence du modèle, le χ^2 précédemment introduit.

Si la valeur du χ^2 n'est pas nulle, ce modèle peut être améliorer. On suppose qu'un modèle plus pertinent se trouve au voisinage du meilleur modèle trouvé. L'intervalle de recherche est centrer sur les valeurs alpha_co_best et alpha_0_best du meilleur modèle trouvé. Les valeurs catographiées de alpha_co vont être comprises entre alpha_co_best - $\Delta\alpha/2$ et alpha_co_best + $\Delta\alpha/2$. De même, les valeurs catographiées de alpha_0 vont être comprise entre alpha_0_best - $\Delta\alpha/2$ et alpha_co_best + $\Delta\alpha/2$.

On détermine $\Delta\alpha$ de la façon suivante. $\Delta\alpha$ ne peut pas être plus petite qu'une valeurs minimale (typiquement un ordre de grandeur plus petit que la précision angulaire que l'on désire atteindre) et est proportionnelle à la somme des incertitudes normalisé du modèle (racine carrée de χ^2) multiplié par la valeur l'incertitude moyenne sur la mesures des étoiles. Afin d'assurer une bonne convergence, l'incertitude moyenne utiliser correspond à l'incertitude de mesures anguleurs des encodeurs plus 3 fois l'écart-type des ces incertitudes).

Chaque point cartographier est choisi dans une subdivision de la zone de cartographie afin d'avoir une répartition homogène de modèle dans cette zone et de façon aléatoire dans chaque subdivision afin d'éviter une quantification des valeurs des modèles. (voir figure ci-après)



Dans notre cas,
P1 = α_0
P2 = α_{CO}
P3 = d
P4 = γ
P5 = δ_0
P6 = δ_{CO}

3) Description du choix de l'augmentation de la zone de recherche

Une fois que le meilleur modèle suivant la méthode d'écrite ci-avant à été trouvé, la taille de la zone de recherche est élargie à un disque de recherche dans le plan des valeurs alpha_0 et alpha_co. Puis on opère une nouvelle phase de cartographie comme décrit précédemment. Cette étape est réalisée afin d'éviter le problème de minima locaux, provenant de l'hypothèse ou l'on suppose qu'un modèle plus pertinent se trouve dans le voisinage du meilleur modèle, voisinage définit par la valeur de Delta alpha. Dans la pratique cette étape n'est pas toujours utile mais évite le mauvaise surprise quant à l'obtention de valeurs biaisés des 6 paramètres de corrections.

4) Méthode de Levenberg-Marquardt

Algorithme de minimisation

La méthode de Levenberg–Marquardt (Levenberg 1944 , Marquardt 1963 , *Numerical Recipes* Press et al. 1992) est une méthode de type minimisation du χ^2 par le gradient minimum. Ainsi l'implémentation de cette méthode nécessite de déterminer les 6 fonctions associées aux dérivées partielles de nos 6 variables.

À partir d'un point initial dans l'espace des paramètres, cette méthode détermine le plus petit gradient local pour trouver un meilleur modèle M_n du profil. La relation de récurrence entre le n–ième modèle et le $n+1$–ième est donnée par :

$$M_{n+1} = M_n + (A + \lambda_n D_A)^{-1} \vec{\nabla}\chi^2(M_n)$$

Le vecteur M_n est défini par $M_n = (\ \alpha_0, \alpha_{CO}, d, \gamma, \delta_0, \delta_{CO}\ )$.
A est la matrice de covariance associée au χ^2.
D_A est la matrice, des valeurs diagonales de la matrice A.
λ_n est la valeur du paramètre de contrôle, à la n–ième itération.

Si le nouveau modèle conduit à une minimisation du χ^2, alors $\lambda_{n+1} = 0.1\lambda$, dans le cas contraire $\lambda_{n+1} = 10\lambda$.

Après plusieurs itérations, la méthode converge vers un minimun qui peut être local ou absolu. Le critère d'arrêt de convergence est un dépassement de la valeur minimale λ_{min} ou maximale λ_{max} par le paramètre de contrôle, ainsi $\lambda_{min} < \lambda_n < \lambda_{max}$. L'arrêt de convergence avec le critère de valeur minimale correspond à un critère de sécurité afin d'éviter un nombre d'itération trop importante. Cette valeur minimale est toutefois rarement atteinte. Ainsi le critère d'arrêt est principalement basé par un dépassement de la valeur maximale, ce qui permet de sortir de certains minima locaux et de trouver une meilleure estimation pour le modèle.

5) Calcul des dérivés partiels du modèle

La fonction que l'on désire minimiser est la distance angulaire entre les coordonnées du modèle et les coordonnées réelles de l'étoiles pointées. On note dy cette fonction. y est une fonction dépendant de 6 paramètre : α_0 , α_{CO}, d , γ, δ_0 , δ_{CO}

La méthode de Levenberg-Marquardt nécessite de connaître la valeur de dérivées partiels au point de coordonnées (α_0 , α_{CO}, d , γ, δ_0 , δ_{CO}).

Pour la correspondance à le programme en language C, on notera :

a[5]= α_0
a[1]= α_{CO}
a[3]= d
a[4]= γ
a[6]= δ_0
a[2]= δ_{CO}

La valeur de la dérivé partielle en (α_0 , α_{CO} , d , γ, δ_0 , δ_{CO}) de dy par rapport à la variable a[i] sera noté dyda[i]. Remarquons que pour les petites valeurs de dy une division par zéro risque de se produire dans le programme. Ainsipour les petites valeurs de y, on biaise légèrement la valeur de la dérivée. Ce procédé est uniquement mis en place pour la stabilité du programme et n'affecte pas la convergence vers les bon paramètre du modèle.

Ci-dessous, exprimer en language C, on présente l'expression de la fonction y ainsi que celles de ses dérivées partielles :

```
void function(Star s, double a[], double *dy, double dyda[], int nvar)
{
        int    i;
        double v,x,y,z;
        double aa,bb,cc;
        double cdd,ca,cg,cd,cd0,ca0;
        double car,sar,cdr,sdr;
        double sdd,sa,sg,sd,sd0,sa0;

        ca  = cos( s.enco.alpha + a[ 1] );
        sa  = sin( s.enco.alpha + a[ 1] );
        cdd = cos( s.enco.delta + a[ 2] );
        sdd = sin( s.enco.delta + a[ 2] );

        cd  = cos(a[ 3]);
        sd  = sin(a[ 3]);
        cg  = cos(a[ 4]);
        sg  = sin(a[ 4]);
        ca0 = cos(a[ 5]);
        sa0 = sin(a[ 5]);
        cd0 = cos(a[ 6]);
        sd0 = sin(a[ 6]);

        x = cdd*ca*cg - sa*sg*cd + sa*cg*sd*sdd;
        y = cdd*sa*cg + ca*sg*cd - ca*cg*sd*sdd;
        z = sdd*cg*cd +   sg*sd;

        aa=  x *cd0*ca0 - y *sa0 + z *sd0*ca0;
        bb=  x *cd0*sa0 + y *ca0 + z *sd0*sa0;
        cc= -x *sd0     + z *cd0;

        car = cos( s.vrai.alpha);
        sar = sin( s.vrai.alpha);
        cdr = cos( s.vrai.delta);
        sdr = sin( s.vrai.delta);

        v=  cdr*(car*aa + sar*bb) + sdr*cc ;

        if (fabs(v)>1) v=1;
        else
        {
                dyda[1]=-( car * cdr * ( (-cdd*sa*cg - ca*sg*cd + ca*cg*sd*sdd ) *cd0*ca0
                                        -( cdd*ca*cg - sa*sg*cd + sa*cg*sd*sdd ) *sa0     )
                                + sar * cdr * ( (-cdd*sa*cg - ca*sg*cd + ca*cg*sd*sdd ) *cd0*sa0
                                        +( cdd*ca*cg - sa*sg*cd + sa*cg*sd*sdd ) *ca0     )
                        -       sdr * (  -cdd*sa*cg - ca*sg*cd + ca*cg*sd*sdd ) *sd0      );
```

```
                dyda[2]=-(car * cdr * ( (-sdd*ca*cg + sa*cg*sd*cdd ) *cd0*ca0
                                                    -(-sdd*sa*cg - ca*cg*sd*cdd ) *sa0
                                            +( cdd*cg*cd              ) *sd0*ca0 )
                                + sar * cdr * ( (-sdd*ca*cg + sa*cg*sd*cdd ) *cd0*sa0
                                                    +(-sdd*sa*cg - ca*cg*sd*cdd ) *ca0
                                                    +( cdd*cg*cd              ) *sd0*sa0 )
                                +       sdr * (-(-sdd*ca*cg + sa*cg*sd*cdd ) *sd0
                                                +( cdd*cg*cd              ) *cd0     ));

                dyda[3]=-(car  * cdr * ( (sa*sg*sd + sa*cg*cd*sdd)*cd0*ca0
                                        -(-ca*sg*sd -ca*cg*cd*sdd)*sa0
                                                                +(-sdd*cg*sd + sg*cd)*sd0*ca0 )
                                + sar  * cdr * ( (sa*sg*sd + sa*cg*cd*sdd)*cd0*sa0
                                                +(-ca*sg*sd -ca*cg*cd*sdd)*ca0
                                                                +( -sdd*cg*sd + sg*cd)*sd0*sa0 )
                                +       sdr * (-(sa*sg*sd + sa*cg*cd*sdd)*sd0
                                                +(   -sdd*cg*sd + sg*cd )*cd0     ));

                dyda[4]=-(car * cdr * (  (-cdd*ca*sg - sa*cg*cd - sa*sg*sd*sdd ) *cd0*ca0
                                                    -(-cdd*sa*sg + ca*cg*cd + ca*sg*sd*sdd ) *sa0
                                                    +(-sdd*sg*cd +    cg*sd             ) *sd0*ca0 )

                                + sar * cdr * (  (-cdd*ca*sg - sa*cg*cd - sa*sg*sd*sdd ) *cd0*sa0
                                                    +(-cdd*sa*sg + ca*cg*cd + ca*sg*sd*sdd ) *ca0
                                                    +(-sdd*sg*cd +    cg*sd             ) *sd0*sa0 )
                                +       sdr * ( -(-cdd*ca*sg - sa*cg*cd - sa*sg*sd*sdd ) *sd0
                                                    +(-sdd*sg*cd +    cg*sd             ) *cd0     ));

                dyda[5]=-(  car * cdr * ( -x *cd0*sa0  -y *ca0 -z *sd0*sa0 )
                                + sar * cdr * (  x *cd0*ca0  -y *sa0 +z *sd0*ca0 ));

                dyda[6]=-(  car * cdr * ( -x *sd0*ca0  +z *cd0*ca0 )
                                + sar * cdr * ( -x *sd0*sa0  +z *cd0*sa0 )
                                +       sdr * ( -x *cd0      -z *sd0     ));

                *dy=-acos(v);

                if (-*dy < 1e-10 ) for (i=1; i<7; i++ ) dyda[i] *=  1.e10;
                else               for (i=1; i<7; i++ ) dyda[i] *=  1 / sqrt(1-v*v);

        }
}
```