Plánovanie, 3D modelovanie a antikolízne porovnávanie údajov usmernených vrtov

***Miroslav Kotula*[1] *a Vojtech Bálint*[2]**

Planning, 3D Modelling and Anti-Collision Data Comparison of Directional Wells

Planning, monitoring, modeling, and anti-collision data comparison of directional wells is observed by software in recent drilling companies. Software TRS 2.1 brings us a few upgrades by using mathematical modeling of well trajectory. There was also used a new mathematical algorithm for anti-collision data comparison of several wells that are situated close to each other in this software. These upgrades help to accelerate the trajectory calculation of the modeled well and to avoid a collision of two wells drilled from one base. Along with the monumental configuration tool, TRS 2.1 belongs to software for complete planning and monitoring of directional wells.

***Key words**: planning and monitoring wellbore, anti-collision software, 3D modelling, directional drilling*

Úvod

Za kolíziu dvoch vrtov považujeme miesto, kde sa k sebe porovnávané vrty priblížia na menej ako je vopred stanovená bezpečná vzdialenosť. Aby sme takéto kolízne úseky našli, potrebujeme vypočítať vzdialenosti týchto dvoch vrtov. Najjednoduchší algoritmus – na každom vrte zvolíme veľa bodov v malých vzdialenostiach a porovnávame dvojice bodov z jedného a druhého vrtu – je časovo náročný. Pokusy vylepšovať tento jednoduchý algoritmus pomocou sledovania narastania a klesania vzdialenosti medzi bodmi pravdepodobne narazia na problém nespojitosti takto definovanej funkcie (vzdialenosť rátame v diskrétnych bodoch) a tým pádom aj možného nenájdenia lokálneho minima. Preto je namieste použiť inú metódu.

Označenie:

Pred popísaním samotnej metódy niektoré označenia.

a, b, x, d malými písmenami označujeme skalárne veličiny (čísla)

u, v, w, ... malými tučnými písmenami označujeme vektory

A, B, X, S, ... veľkými písmenami označujeme body v priestore

u.v označujeme štandardný skalárny súčin vektorov v trojrozmernom euklidovskom priestore

|u| označuje veľkosť vektora u

d(A,B) vzdialenosť bodov A, B – to isté ako |B-A|

Algoritmus:

Z matematického hľadiska je naplánovaný vrt množina na seba naväzujúcich úsečiek a častí kružníc. Použitá metóda využíva nasledujúcich päť funkcií na výpočet vzdialenost

bodÚsečka	vypočíta vzdialenosť medzi úsečkou a bodom, na úsečke nájde bod ktorý je najbližšie k danému bodu
bodKružnica	vypočíta vzdialenosť medzi časťou kružnice a bodom, na kružnicovom oblúku nájde bod ktorý je najbližšie k danému bodu
úsečkaÚsečka	vypočíta vzdialenosť medzi dvoma úsečkami, na úsečkách nájde dvojicu bodov s najmenšou vzdialenosťou
úsečkaKružnica	vypočíta vzdialenosť medzi úsečkou a časťou kružnice, na útvaroch nájde dvojicu bodov s najmenšou vzdialenosťou
kružnicaKružnica	vypočíta vzdialenosť medzi dvoma kružnicovými oblúkmi, na častiach kružníc nájde dvojicu bodov s najmenšou vzdialenosťou

Metódy budú popísané neformálnym spôsobom podobným programovaciemu jazyku C s doplnením slovného popisu. Poznamenajme že za znakom // sa nachádza komentár.

Metóda bodÚsečka:

```
vstup:
    Bod X;
    Bod A, B;        // krajné body úsečky
```

[1] *Ing. Miroslav Kotula,* Drillsoft company, Fedákova 10, 841 02 Bratislava, Slovak Republic, tel.:+421-2-6544 4314, fax +421-2-6544 4315, kotula@drillsoft.sk, www.drillsoft.sk, www.drillsoft.com

[2] *Mgr. Vojtech Bálint,* Nám Ľ. Fullu 1666/15, 010 01 Žilina, Slovak Republic, tel.:+421-2-5022 2706, balint@drillsoft.sk

(Recenzovaná a revidovaná verzia dodaná 9. 9. 2004)

```
výstup:
    double d;          // hľadaná vzdialenosť
    Bod Q;  // bod na úsečke AB s minimálnou vzdialenosťou od bodu X

Vector u = B-A;
if (|u| = 0) {
    // degenerovana usecka A=B
    d = d(A, X);
    Q = A;
    return;
}

Vector v = X-A;
double t = u.v / u.u;

if (t >= 0) && (t <= 1) {
    // najblizsi bod je na usecke AB
    Q = A + tu;
} else {
    // najblizsi bod je mimo usecky AB
    if (d(X,A) < d(X,B)) {
            Q = A;
    } else {
            Q = B;
    }
}
d = d(X, Q);
```

Popis: Metóda hľadá bod Q = A + tu tak aby vector XQ bol kolmý na vektor u. Podmienka na kolmosť je vyjadrená rovnicou t = u.v / u.u.

Metóda bodKružnica

Pre jednoduchosť predpokladajme, že kružnica leží v rovine xy a má stred v bode (0, 0, 0). Toto sa dá dosiahnuť jednoduchým afinným zobrazením, matica ktorého je vypočítaná už pri plánovaní vrtu (všetky výpočty sa pre jednoduchosť robia v tejto rovine).

```
vstup:
    Bod X;
    double φ1, φ2;    // počiatočný a konečný uhol kružnicového oblúka φ1 < φ2
    double R;                  // polomer
výstup:
    double d;          // hľadaná vzdialenosť
    Bod Q;  // bod na kružnicovom oblúku s minimálnou vzdialenosťou od bodu X

Bod X0 = kolmy priemet X do roviny xy;
double φ = uhol bodu X0;
if (φ1 < φ < φ2) {
    // bod Q je na kružnici
    Q = bod na kružnici s uhlom φ;
} else {
    Q = krajný bod kružnicového oblúka bližšie k X
}
d = d(X, Q);
```

Popis: Metóda využíva jednoduchý fakt, že ak je bod Q najbližšie k bodu X, tak je najbližšie aj ku kolmému priemetu bodu X do roviny kružnice. Vyriešiť túto úlohu v rovine je triviálne.

Metóda úsečkaÚsečka

```
vstup:
    Bod A, B;          // krajné body prvej úsečky
    Bod C, D;          // krajné body druhej úsečky
```

```
výstup:
    double d;          // hľadaná vzdialenosť
    Bod P, Q;          // body na úsečkách AB, CD s minimálnou vzdialenosťou

Vector u = B-A;
Vector v = D-C;
Vector w = A-C;

double d = (u.u)(v.v) – (u.v)(u.v);
double dt = (u.v)(v.w) – (u.w)(v.v);
double ds = (u.u)(v.w) – (u.v)(u.w);

if (d = 0) {
    d = d(A,C);
    P = A;
    Q = C;
    return;
}
double t = dt / d;
double s = ds / d;

if (0 <= t <= 1) && (0 <= s <= 1) {
    P = A + tu;
    Q = C + sv;
    d = d(P,Q);
    return;
}
using methods
    bodUsecka(A, (C,D));
    bodUsecka(B, (C,D));
    bodUsecka(C, (A,B));
    bodUsecka(D, (A,B));
find the shortest distance and points P,Q.
```

Popis: Metóda hľadá body P = A + tu a Q = C + sv tak aby vektor PQ bol kolmý na obidve úsečky. Podmienka na kolmosť je vyjadrená sústavou rovníc

u(-tu – w + sv) = 0
v(-tu – w + sv) = 0

ktorú riešime pomocou determinantov.

Metódy úsečkaKružnica a kružnicaKružnica

Obe tieto metódy iteratívnym spôsobom hľadajú najbližšie body. Postup budeme demonštrovať na metóde úsečkaKružnica. Predpokladajme, že A, B sú krajné body úsečky a C,D, S sú krajné body a stred kružnicového oblúku.

```
bodUsecka(C, (A,B)); //  oznacme X bod usecky AB najblizsie
                              k bodu C najdeny metodou bodUsecka
bodKruznica(X, (C,D,S)); // oznacme Y bod kruznicoveho obluka
                              najblizsie k bodu X najdeny metodou
                              bodKruznica
bodUsecka(Y, (A,B)); //  oznacme X bod usecky AB najblizsie
                              k bodu Y najdeny metodou bodUsecka
bodKruznica(X, (C,D,S)); // oznacme Y bod kruznicoveho obluka
                              najblizsie k bodu X najdeny metodou
                              bodKruznica
```

Popis: Postupne hľadáme bod X na úsečke AB, a bod Y na kružnicovom oblúku, tak aby boli k sebe čo najbližšie. Tieto dve metódy na rozdiel od predchádzajúcich metód nenájdu presne tieto dva body, ale na rozdiel od jednoduchého algoritmu (porovnávanie bodov rozmiestnených na vrte), táto metóda sa v každom kroku snaží nájsť optimálne riešenie a vďaka tomu je výrazne rýchlejšia.

Hľadanie kolízií

Ako bolo uvedené vyššie, z matematického hľadiska je naplánovaný vrt množina na seba naväzujúcich úsečiek a častí kružníc.

1. Použitím metód bodÚsečka, bodKružnica, úsečkaÚsečka, úsečkaKružnica, kružnicaKružnica zistíme vzdialenosti medzi jednotlivými úsekmi. Poznamenajme, že v tomto prípade, podobne ako u spomínaného triviálneho algoritmu, porovnávame každý úsek s každým. Zásadný rozdiel je však v tom, že v jednom počet úsekov pravdepodobne neprekročí desiatku (nezáleží na dĺžke vrtu, iba na počte úsekov tzn. tvare vrtu), zatiaľ čo počet porovnávaných bodov musí dosahovať rádovo tisícky ak chceme dostať dostatočne presné výsledky.
2. Porovnaním s vopred danou bezpečnou vzdialenosťou nájdeme dvojice úsekov ktoré kde dochádza ku kolízii. Keďže porovnávame všetky dvojice úsekov, väčšina dvojíc pravdepodobne nebude spôsobovať kolízie.
3. Vypíšeme najbližšie body pre dvojice kolíznych úsekov spolu s ich blízkym okolím. Veľkosť okolia môžeme meniť, môže sa stať, že niektorý úsek je v kolízii s viacerými úsekmi druhého vrtu. V tomto prípade program nájde „najtesnejšiu“ kolíziu, vypíše ju a ostatné uvedie iba v rámci okolia tejto kolízie

Literatúra - References

[1] G., Birkhoff, S., M. ‚Lane: Prehľad modernej algebry*, Alfa, Bratislava, 1974*

[2] S., M., Lane, G., Birkhoff: Algebra*, Alfa, Bratislava, 1974*

[3] M., Hejný, V., Zaťko, P., Kršňák: Geometria *1, Slovenské pedagogické nakladateľstvo, 1985*