TÜBİTAK

TÜRKİYE BİLİMSEL VE
TEKNİK ARAŞTIRMA KURUMU

THE SCIENTIFIC AND TECHNICAL
RESEARCH COUNCIL OF TURKEY

ANİZOTROPİ PARAMETRELERİNİN BELİRLENMESİ

Yer Deniz ve Atmosfer Bilimleri Araştırma Grubu

Earth Marine and Atmospherical Sciences
Researches Grant Group

TRANSVERSELY İZOTROPİK ORTAMDA DERİNLİK MİGRASYONU UYGULANMIŞ CRP GRUPLARININ TOMOGRAFİSİ İLE HIZ, DERİNLİK VE ANİZOTROPİ PARAMETRELERİNİN BELİRLENMESİ

PROJE NO: YDABÇAG-100Y105

YRD. DOÇ. DR. SELMA KADIOĞLU

kadioglu@eng.ankara.edu.tr

ANKARA-2003

1. GİRİŞ

Sismik yansıma yönteminde anizotropi hızın yöne bağlı olarak değişmesidir. Anizotropi yeraltı hız tayinini, sismik görüntülemeyi, zaman derinlik dönüşümünü, sismik genliği ve AVO analizini etkileyen önemli bir unsur olarak görülmektedir. Sismik anizotropinin ana sebebi bir çizgi halinde dizilmiş mineral taneleri, küçük çatlaklar, dizilmiş kristaller, periyodik ince tabakalanma (Thomsen, 1986; Helbig, 1994) olarak sayılabilir. Periyodik tabakalanma, genellikle şeyl, aratabakalanmış şeyller ve kumtaşlarında görülür. Sediman tabanların çoğunun büyük ölçüde şeyl içerdiği, hatta çoğu hidrokarbon reserv yerlerinde bulunduğu bilinmektedir (Schoenberg, ve Sayers, 1995). Düşey sismik hızlar arasındaki farklar anizotropi derecesini artıran önemli bir unsurdur (Schoenberg ve diğ., 1996) ve bu durum bir kaç kilometre derinlikteki hedefler için birkaçyüz metre daha rezidüel derinlik oluşmasına neden olur. Yine eğimli tabakaların hız değişiminin anizotropik olduğu Alkhalifah ve Tsvankin (1995), Gaiser (1990) tarafından vurgulanmıştır.

Anizotropi tek kelimeyle ifade edilmekle birlikte simetri eksenlerine göre çeşitlilik göstermektedir. En basit realistik simetri, tek bir simetri eksenine sahip kutupsal (polar) simetridir. Kutupsal simetri eksenine sahip anizotropi '**kutupsal anizotropi**' veya daha çok '**enine (transverse) izotropi**' veya '**TI**' olarak adlandırılır. Simetri ekseninin düşey eksen olması durumunda bu tür anizotropi durumu '**Düşey simetri eksenli enine izotropi (vertical transverse isotropy) veya VTI**' olarak adlandırılır. Yani buradaki izotropi yatay düzlem ile sınırlandırılmış olunur (Thomsen, 2002).

Düşey simetri eksenli enine izotropi yansımalı sismikte en yaygın tiptir (Thomsen 1986). Ofset açılımı hedef derinlikten daha büyük olduğu zaman anizotropinin etkisinin görünür olduğu ortaya atılmıştır (Tsvankin ve Thomsen, 1994; Alkhalifah ve Tsvankin, 1995). Genellikle bu kadar büyük açılımlar geleneksel sismik yansıma araştırmalarında uygun değildir. Bu engeli ortadan kaldırmak için Tsvankin (1995), Alkhalifah ve Tsvankin (1995) farklı eğimleri kullanan bir hız analizini önermişlerdir. Bu durum eğime bağlı moveout hız (yığma hızı) değişiminin anizotropiye duyarlı olduğunu göstermiştir. Ancak onların yöntemleri aynı bölgede oldukça farklı

eğimli ve açıkça görülebilir yansımların olmasını gerektirmektedir (örneğin bir tabaka içinde oldukça dik eğimli faylar). Bu tür olayların her zaman mümkün olmayacağı açıktır.

Projenin konusu; derinlik migrasyonu uygulanmış CRP açılımlarının tomografisine dayanan iki boyutlu düşey simetri eksenli enine izotropik ortam için yeraltı hızının, derinliğinin ve δ anizotropi parametresinin tayini için bir yöntem geliştirmektir. Tomografi işlemi tabaka hızlarını, derinliklerini ve Thomsen anizotropi parametrelerini gerçeğine en yakın olarak bulan bir ters çözüm işlemidir. Yöntem anizotropik ortam için tomografi prensiplerine dayanmaktadır. Tomografi prensiplerine göre denklemleri oluşturmak için bir CRP ışın çifti boyunca seyahat zamanlarından yararlanarak arayüzey parametrelerindeki ve arayüzey derinliklerindeki küçük farklılıklar kullanılır. Tomografi denklemleri ise elde edilen denklemlerin özel ayrıklaştırılması ile elde edilir. Parametreleri bulmak için bu denklemler sönümlü en küçük kareler yöntemi ile çözülür.

Projenin amacı; iki boyutlu düşey simetri eksenli enine izotropik (VTI) ortam için jeolojik yapısal bir modele dayalı yığma öncesi derinlik migrasyonu uygulanmış ortak yansıma gruplarının (CRP gathers) tomografisi ile tabaka hızlarının, arayüzey derinliklerinin ve δ anizotropi parametresinin bulunmasıdır.

Hızlar belirlenen jeolojik tabakaların (horizonların) yapısal çerçevesini temsil edilebilirliğini sağlamalıdır. Bu şartı sağlayan yöntemler tabaka hızlarını başka bir deyişle ara hızları kullanarak hız modeli oluşturur ve yapısal kesiti oluştururlar (Fagin, 1998). Zaman boyutunda çalışan yöntemler ise zaman kesitleri boyunca yığma hızı (NMO hızı) değerlerini kullanırlar. Yığma hızı değerlerini kullanan yöntemlerde, yoruma dayalı yapısal bir bilgi gerektirmediği için, doğrudan model oluşturma ve update etme geleneksel tomografi akış düzeneğinde devam ettirilebilir ve sonuca ulaşabilir. Sonuçta derinlik bilgisi elde edilemez. Jeolojik yapısal modele dayalı ve tabakalara ait ara hızları kullanan yöntemler modele dayalı bir giriş ve aynı zamanda görüntüleme bilgisi gerektirir (Fagin, 1998, sh. 27-28). Bu yöntemler hız değişimini kontrol eden faktörlere dayanır. Hız değişimi jeolojik yaşa, litolojiye, derinliğe ve basınca bağlıdır. Hız ve derinlik tespiti ile ilgili bu faktörler yapısal geometriyi resimlemektedirler. Bu nedenle projede jeolojik yapısal

modele dayalı tabaka hızlarını kullanan yığma öncesi derinlik migrasyonu CRP grupları ile çalışılmaktadır.

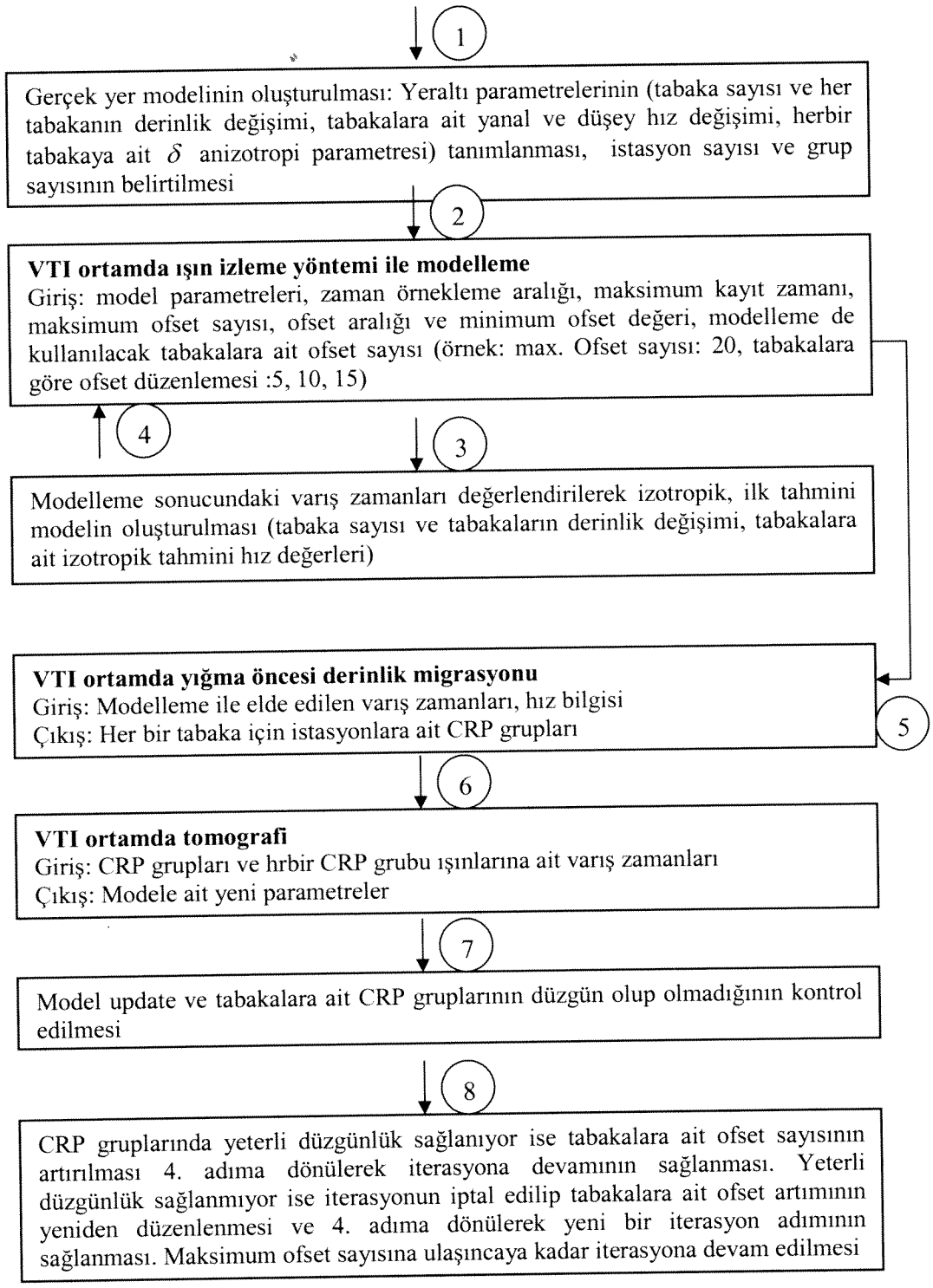
Derinlik migrasyonu yığma hızı yerine doğrudan ara hızların kullanımına dayanmaktadır. Ara hız değerlerini taşıyan hız bilgisi düşey P dalgası hızıdır. Düşey P dalgası hızını δ anizotropi parametresi ağırlıklı olarak etkilerken, ε anizotropi parametresi ise yatay P dalgası hızı değişimininde etkin olmaktadır (Thomsen, 1986; 18a ve 27a denklemi). Bu nedenle ε terimi düşeye yakın yayılımlar için ihmal edilebilir. (Thomsen, 1986). Proje önerimiz doğrultusunda yaptığımız çalışmanın derinlik ortamında olması ve düşey P dalgası hızını belirleme amacında olmamız nedeniyle ε parametresi ihmal edilmiştir.

Geliştirilmesi hedeflenen yönteme göre yatay ve düşey yönde değişebilen hız modeli, arayüzeyler ile ayrılmış tabakalar ile tanımlanır. VTI ortam için elastik cisim parametreleri ve ışın denklemleri (Thomsen, 1986; Farra ve Madariaga, 1988; Cerveny, 1972) ve anizotropik dalga yayılımı (Payton, 1983) kullanılarak sıfır ofset ışın izleme yöntemi ve değişken ofsetli CRP ışın çiftleri ile modelleme yapılır. Sıfır ofset ışın izleme, tomografi işleminde ilk parametreleri belirlemek amacı ile gerçek model kullanılarak yapılmaktadır. Değişken ofset CRP ışın izleme ise belirlenen parametrelere göre bir CRP noktasına ait farklı ofsetli varış zamanlarını hesaplamak amacıyla kullanılır.

CRP noktalarına ait farklı ofsetli varış zamanları kullanılarak modele dayalı görüntüleme (model based imaging) yapılır. Modele dayalı görüntüleme hız analizi için düzenlenmiş VTI ortamda yığma öncesi derinlik göçü (migrasyon) işlemidir (Kosloff ve diğ., 1996; Fagin, 1998).

Gerçek yerlerine göç ettirilen CRP grupları kullanılarak, VTI ortam için bir CRP ışın çifti boyunca, ortam parametrelerindeki ve arayüzey derinliklerindeki saçılmalardan dolayı, seyahat zamanındaki değişiklikleri tanımlamayı saylayan tomografik prensipler kullanılır. Tomografik denklemler düzenlendikten sonra sönümlendirilmiş en küçük kareler yöntemi ile çözülür. Amaç sismik kesite ait yığma öncesi derinlik migrasyonunda düzgün CRP gruplarını elde etmektir.

Belirtilen üç temel adım daha ayrıntılı şekilde bir algoritma üzerinde şu şekilde tanımlanabilir.



VTI ortamda modellemeyi gerçekleştirebilmek amacıyla 'enine izotropik ortamda dalga yayılımı irdelenmiş çalışma sonuçları Tükçe Jeofizik dergisinde yayınlanmıştır (**EK 1.**). Yine VTI ortamda ışın izleme yöntemi ile modelleme yapmak amacı ile P dalgası için seyahat zamanı hesaplanması çalışmaları yapılmış, izotropik ortamda seyahat zamanı hesaplaması Türkçe jeofizik dergisinde yayınlanmıştır (**EK 2.**). VTI ortamda ışın izleme yöntemi ile modelleme '14th International petroleum and naturel gas congress and exhibition of Turkey' de bildiri olarak sunulmuştur (**EK 3.**). VTI ortamda yığma öncesi derinlik migrasyonu tamamlanmış, migrasyon sonucu CRP grupları elde edilmiştir. En yoğun ve önemli adımı olan sönümlü en küçük kareler yöntemi ile istasyonlara ait CRP gruplarının tomografisi sonucunda doğruya yakın hız, derinlik ve δ anizotropi parametreleri elde edilmesi gerçekleştirilmiştir. Bütün bu aşamalar teorik olarak üretilen üç model üzerinde denenmiştir. Sonuçlar, teorik çalışmanın doğruluğunu ortaya koymaktadır. Modelleme, yığma öncesi derinlik migrasyonu ve tomografi programı birleştirilmiş olarak **EK 4.**'de verilmiştir.

İzleyen bölümlerde VTI ortamda ışın izleme yöntemi ile modelleme, yığma öncesi migrasyon ve tomografi konularının teorileri sunulmuş ve uygulamalarla yöntemin çalışabilirliği tartışılmıştır.

2. DÜŞEY SİMETRİ EKSENLİ ENİNE İZOTROP ORTAM (VERTICAL TRANSVERSELY ISOTROPIC MEDIUM; VTI MEDIUM)

2.1. ELASTİSİTİ TENSÖR VE SİMETRİ SINIFLARI

Elastik dalga denklemi gerilme-yamulma ilişkisi ve yamulma yerdeğiştirme ilişkilerinin birleşimi ile momentum korunum denklemlerinin birleştirilmesi sonucu elde edilir (Lay ve Wallace, 1995). Momentum korunumu

$$\frac{\partial \sigma_{ji}}{\partial x_j} = \rho f_i + \rho \frac{\partial^2 u_i}{\partial t^2}, \qquad i,j = 1 \cdots 3 \tag{2.1}$$

ile tanımlanır. Burada σ_{ij} gerilme bileşenlerini ρ yoğunluğu, f_i cisim kuvvetlerini ve u_i yerdeğiştirme bileşenlerini temsil etmektedir. Yamulma-yerdeğiştirme ilişkisi

$$e_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \qquad i,j = 1 \cdots 3 \tag{2.2}$$

dir. Burada e_{ij} yamulma bileşenleridir. Gerilme-yamulma ilişkisi de

$$\sigma_{ij} = \sum_{k=1}^{3}\sum_{l=1}^{3} C_{ijkl} e_{kl}, \qquad i,j = 1 \cdots 3 \tag{2.3}$$

dir (Thomsen, 1986; Lay and Wallace, 1995). Burada C_{ijkl} elastik sabitlerdir. Yamulma enerjisinin varlığı ve hem σ_{ij} nin hem de e_{ij} nin simetrikliğinden dolayı $C_{ijkl} = C_{jikl} = C_{ijlk} = C_{klij}$ (Thomsen, 1986; Lay ve Wallace, 1995) dir. Böylece maksimum elastik sabit sayısı 21 dir. Daha ileri bir simetriklik durumu bu katsayıların dahada azalmasını sağlamaktadır.

(2.3) ifadesi (2.1) hareket denkleminde yerine konulduğunda ve f_i ihmal edildiğinde

$$\frac{\partial^2 u_i}{\partial t^2} = \frac{C_{ijkl}}{\rho}\frac{\partial^2 u_k}{\partial x_j \partial x_l} \tag{2.4}$$

olur. (2.4)'ün düzlem dalga çözümü

$$u_i = A_i f(vt - \eta_j x_j) \tag{2.5}$$

dir. Burada A_i polarizayson vektör, η_i dalga cephesine dik birim vektör, v faz hızıdır. (2.4) deki türevler

$$\frac{\partial^2 u_i}{\partial t^2} = v^2 A_i f''(vt - \eta_j x_j) \tag{2.6}$$

$$C_{ijkl}\frac{\partial^2 u_k}{\partial x_j \partial x_l} = C_{ijkl} A_k \eta_j \eta_l f''(vt - \eta_j x_j) \tag{2.7}$$

olarak bulunur. Burada $f'(\tau) = \frac{df}{d\tau}$ dur. (2.6) ve (2.7) ifadeleri (2.4) de yerine konulduğunda

$$(C_{ijkl}\eta_j \eta_l - v^2 \delta_{ik})A_k = 0 \tag{2.8}$$

olur. Bu ifade bir özdeğer denklemini tanımlar. (2.8) de parantez içindeki ilk terim

$$\Gamma_{ik} = C_{ijkl}\eta_j \eta_l \tag{2.9}$$

ile tanımlandığında

$$\det(\Gamma - v^2 I) = 0 \tag{2.10}$$

sağlanır. Buradaki özdeğerler faz hızının karesidir ve her bir yöndeki η farklı hıza sahip olacaktır. Enine izotropide üç kapalı formülle tanımlanan özdeğerler vardır ve birbirlerine ortogonaldirler. Bu özdeğerler quasi-P dalgasını, quasi-SV dalgasını ve quasi-SH dalgasını tanımlamaktadırlar.

C elastisiti tensörün dört indisi vardır. İkisi gerilme diğer ikiside yamulma indisleridir. $3\times3\times3\times3$ lük C_{ijkl} elastisiti tensörünü daha kolay ifade edebilecek için 6×6 lık $C_{\alpha\beta}$ elastite matrise

$$(11)\rightarrow 1,(22)\rightarrow 2,(33)\rightarrow 3,(23)=(32)\rightarrow 4,(31)=(13)\rightarrow 5,(12)=(21)\rightarrow 6, \qquad (2.11)$$

şeklinde indirgenebilir (Thomsen, 1986). Bu indislemeyi daha kolay hatırlamak için

$$\begin{matrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{matrix} \Rightarrow \begin{matrix} 1 & 6 & 5 \\ 6 & 2 & 4 \\ 5 & 4 & 3 \end{matrix}$$

haritalamasından yararlanılabilir (Thomsen, 2002).

Öncelikle elastisite matrisinin izotropik (parametreler yönden bağımsız) bir ortamdaki durumunu ele alalım. İzotropik ortamda $C_{\alpha\beta}$ matrisinin çoğu elemanlarının değeri sıfırdır. Sadece iki tane farklı elemana sahiptir. Bunlar ***Lame parametreleri*** olarak bilinen μ ve λ dır. μ makaslama modülü olarak da tanımlanmaktadır. Bu iki parametre P-dalgası hızını control etmektedirler. P-dalgası hızı $V_p=\alpha=\sqrt{(\lambda+2\mu)/\rho}=\sqrt{(K+4\mu/3)/\rho}$ dir. Burada K ***Bulk modülü*** veya ***sıkışmazlık*** olarak tanımlanır.

Dalga yayılımını anlamada en kolay simetri tipi olduğu için ***izotropik simetri*** ve belli başlı simetri tiplerini tanıtmanın faydalı olacağı düşünülmektedir. Elastisite matrisi için izotropik simetri en basit simetri tipidir ve

$$\begin{bmatrix} p & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & p & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & p & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{2.12}$$

olarak tanımlanır. İkinci kolay simetri tipi ***kübik simetri***dir. Kübik simetri izotropik simetri ile aynı formdadır. Ancak kübik simetriye sahip jeofiziksel bir olay olmadığı için pek kullanılmayan bir simetri tipidir ve

$$\begin{bmatrix} C_{33} & C_{12} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{33} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{12} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{44} \end{bmatrix} \tag{2.13}$$

ile verilir (Thomsen, 1986; 2002).

En basit gerçek simetri tipi ***kutupsal (polar) simetri***dir. Çünkü polar simetrinin rotasyonal simetri eksenli bir kutbu vardır. Diğer iki eksen, rotasyonal simetri ekseninden farklıdır ancak birbirine eşittir. Bu simetri tipi en çok yatay ince yataklanmalarda, veya yatay masiv şeyl tabakalarında görülür (Thomsen, 1986; Helbig, 1994). Periyodik tabakalanma genellikle şeyl, ara tabakalanmış şeyller ve kumtaşlarında görülür. Sediman tabanların çoğunun büyük ölçüde şeyl içerdiği, hatta çoğu hidrokarbon reserv yerlerinde bulunduğu bilinmektedir (Schoenberg, 1994; Sayers, 1994).

Kutupsal anizotropi daha çok ***enine (transverse) izotropi*** veya **TI** olarak adlandırılır. En çok görüleni rotasyonel simetri ekseninin düşey eksen olma durumudur. Kristalografide bu durum hegzagonal simetri olarak tanımlanır. Buna ***düşey enine izotropi (vertical transverse isotropy)*** veya ***VTI*** denmektedir. Buradaki izotropi yatay düzlem (transverse plane) ile sınırlandırılmıştır. Bu nedenle enine izotropi kelimeleri ile tanımlanmıştır. **Bu çalışmada kutupsal anizotropi yani düşey enine izotropi (vertical transverse isotropy, VTI) kullanılmaktadır.**

VTI ortamda elastisite matrisi

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{13} & 0 & 0 & 0 \\ C_{13} & C_{13} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{bmatrix} \qquad (2.14)$$

olarak tanımlanır (Thomsen, 1986). $C_{66} = (C_{11} - C_{12})/2$ dir. Buradada görüldüğü gibi kutupsal simetrinin beş farklı elemanı vardır ve 1 ve 2 eksenleri eşittir.

Farklı simetri olarak ortorombik simetri ve triklinik simetri de litaratürlerde tanımlanmıştır. Ortorombik simetrinin 9, triklinik simetrinin 21 farklı elemanı vardır. Bu nedenle triklinik simetri en kötü simetri olarak bilinir (Thomsen, 2002). Burada bu simetriler üzerinde durulmayacaktır.

2.2. THOMSEN ANİZOTROPİ PARAMETRELERİ

Elastik parametreler daha çok Thomsen parametreleri ile tanımlanırlar. Bunlar;

$$\alpha = \sqrt{C_{33}}$$

$$\beta = \sqrt{C_{44}}$$

$$\varepsilon = \frac{C_{11} - C_{33}}{2C_{33}} \qquad (2.15)$$

$$\gamma = \frac{C_{66} - C_{44}}{2C_{44}}$$

$$\delta = \frac{(C_{13} + C_{44})^2 - (C_{33} - C_{44})^2}{2C_{33}(C_{33} - C_{44})}$$

dir. Burada $\alpha = \alpha_0$ düşey yönde P dalgası hızı, $\beta = \beta_0$ düşey S dalgası hızı, ε yatay ve düşey P dalgası hızlarının kareleri farklarının düşey P dalgası karesinin iki katına oranı, γ ise S dalgası için ε ye eşittir. δ ek bir parametredir.

Burada beş parametrenin anizotropik dalga yayılımını kontrol ettiği düşünülse de, düşey yönde ara hızların etkin olduğu derinlik ortamı için, gerçekte bir tanesi oldukça etkili rol oynamaktadır.

Tsvankin (1995)'e göre S dalga hızı P dalga yayılımını çok az etkilemektedir. Dolayısı ile P dalgasının S dalgasına oranı fix edilebilir. Bu durumda β ve γ parametreleri ihmal edilebilir. Yine Thomsen (1986) ve Tsvankin (1995)' e göre düşey yönde değişen hıza ε parametresinin etkisi ihmal edilebilecek düzeydedir.

2.3. VTI ORTAMDA DALGA YAYILIMI

Bu konu Türkçe makale olarak (Kadıoğlu, 2002) yayınlanmıştır. Burada **EK 1** olarak sunulmuştur.

2.4. VTI ORTAMDA CİSİM DALGALARI VE DALGA HIZLARI

(2.15) ile verilen Thomsen parametrelerinden yararlanılarak quasi-P, quasi-SV and SH dalgası sırasıyla

$$V_P^2(\theta) = \alpha_0^2\left[1 + \varepsilon \sin^2\theta + D^*(\theta)\right], \tag{2.17}$$

$$V_{SV}^2(\theta) = \beta_0^2\left[1 + \frac{\alpha_0^2}{\beta_0^2}\varepsilon \sin^2\theta - \frac{\alpha_0^2}{\beta_0^2}D^*(\theta)\right], \tag{2.18}$$

$$V_{SH}^2(\theta) = \beta_0^2\left[1 + 2\gamma \sin^2\theta\right], \tag{2.19}$$

denklemleri ile tanımlanır. Burada

$$D^*(\theta) \equiv \frac{1}{2}\left(1 - \frac{\beta_0^2}{\alpha_0^2}\right)\left\{\left[1 + \frac{4\delta^*}{\left(1-\beta_0^2/\alpha_0^2\right)^2}\sin^2\theta\cos^2\theta + \frac{4\left(1-\beta_0^2/\alpha_0^2+\varepsilon\right)\varepsilon}{\left(1-\beta_0^2/\alpha_0^2\right)^2}\sin^4\theta\right]^{1/2} - 1\right\} \tag{2.20}$$

ve

$$\delta^* \equiv \frac{1}{2C_{33}^2}\left[2(C_{13}+C_{44})^2-(C_{33}-C_{44})(C_{11}+C_{33}-2C_{44})\right] \tag{2.21}$$

dir (Thomsen, 1986). θ **faz açısını** tanımlamaktadır. Enerjinin yayılımı boyunca ışının düşey ile yaptığı açı ϕ, faz açısı θ dan farklıdır (Sekil 2.1). Dalga cephesi lokal olarak yayılma vektörü k ya diktir. Bu nedenle faz artışının maksimum yönünü k belirler. **Faz hızı** $v(\theta)$ aynı zamanda **dalga cephesi hızı** olarak da tanımlanır. Çünkü faz hızı $k(\theta)$ boyunca dalga hızını vermektedir. Dalga cephesi küresel olmadığı için θ ya dalga cephesine **normal açı** adı da verilir. Dalga vektörü

$$k = k_x\vec{x} + k_z\vec{z} \tag{2.22}$$

dir. k vektör bileşenleri

$$k_x = k(\theta)\sin\theta, \tag{2.23a}$$

$$k_z = k(\theta)\cos\theta, \tag{2.23b}$$

ve

$$k_y = 0 \tag{2.23c}$$

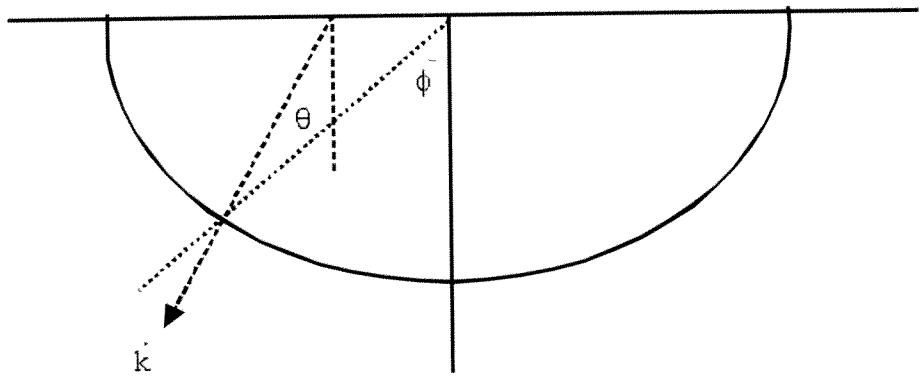
olarak tanımlanır. $k(\theta)$ skaler büyüklüğü ise

$$k(\theta) = \sqrt{k_x^2 + k_z^2} = \frac{w}{v(\theta)} \tag{2.24}$$

dir. Burada w açısal frekanstır. Buna göre ışın hızı

$$V = \frac{\partial(kv)}{\partial k_x}\vec{x} + \frac{\partial(kv)}{\partial k_z}\vec{z} \tag{2.25}$$

ifadesi ile tanımlanır (Thomsen, 1986). Burada V **grup hızı** ϕ de **grup açısı** olarak bilinir. (2.25) aynı zamanda grup ve faz hızı arasındaki ilişkiyi vermektedir. ϕ açısı θ açısına bağlı olarak



Şekil 2.1. Faz (dalga cephesi) açısı θ ve grup (ışın) açısı ϕ arasındaki ilişki (Thomsen 1986' dan uyarlanmıştır).

$$\tan(\phi(\theta)) = \frac{\partial kv}{\partial k_x} \Big/ \frac{\partial kv}{\partial k_z} \tag{2.26}$$

bağıntısı ile tanımlanmaktadır. Buna göre faz ile grup hızı arasındaki skaler bağıntı

$$V^2(\phi(\theta)) = v^2(\theta) + \left(\frac{dv}{d\theta}\right) \tag{2.27}$$

ile verilir. $\theta = 0$ ve $\theta = 90$ olduğunda ikinci terim kaldırılır. Bu durumda grup hızı faz hızına eşittir.

2.4.1. Zayıf Anizotropide Dalga Hızları

Kayaçlardaki anizotropinin daha çok zayıf anizotropi olduğu gözlenmiştir (Thomsen, 1986; 2002). Bu nedenle (2.17), (2.18) ve (2.19) ile verilen hız bağıntıları zayıf anizotrop ortam için

$$V_P(\theta) = \alpha_0 \left[1 + \delta \sin^2\theta \cos^2\theta + \varepsilon \sin^4\theta\right], \tag{2.28}$$

$$V_{SV}(\theta) = \beta_0 \left[1 + \frac{\alpha_0^2}{\beta_0^2}(\varepsilon - \delta)\sin^2\theta \cos^2\theta\right], \tag{2.29}$$

$$V_{SH}(\theta) = \beta_0 \left[1 + \gamma \sin^2\theta\right], \tag{2.30}$$

dir. Burada

$$\begin{aligned} \delta &\equiv \frac{1}{2}\left[\varepsilon + \frac{\delta^*}{\left(1 - \beta_0^2/\alpha_0^2\right)}\right] \\ &= \frac{(C_{13} + C_{44})^2 - (C_{33} - C_{44})^2}{2C_{33}(C_{33} - C_{44})} \end{aligned} \tag{2.31}$$

Thomsen'nin beşinci parametresi olarak literatüre geçmiştir (Thomsen, 1986).

Enine izotropinin (TI) özel bir durumu **eliptik anizotropi**dir. Eliptik anizotropik ortam eliptik dalga cephesi ile karakterize edilir ve Thomsen parametreleri ile eliptik anizotrop olma şartı

$$\delta = \varepsilon \tag{2.32}$$

ile verilir (Daley ve Hron, 1977). Thomsen (1986)' ya göre δ ve ε tam ilişkili değildir. Genellikle birbirlerine göre ters işaretlidirler. Bu kabulün ciddi hatalara yol açabileceği düşünülmektedir.

2.4.2. Grup Açısı ve Faz Açısı ile İlişkisi

(2.26) ile verilen grup açısı ve faz açısı arasındaki bağıntı lineer bir yaklaşım

$$\tan\phi = \tan\theta\left(1+\frac{1}{\sin\theta\cos\theta}\frac{1}{v(\theta)}\frac{dv}{d\theta}\right) \tag{2.32}$$

ile verilmektedir. P, SV ve SH dalgaları için tam lineerleştirilmiş bir yaklaşım kullanıldığında sırasıyla

$$\tan\phi_P = \tan\theta_P\left(1+2\delta+4[\varepsilon-\delta]\sin^2\theta_P\right), \tag{2.33}$$

$$\tan\phi_{SV} = \tan\theta_{SV}\left(1+2\frac{\alpha_0^2}{\beta_0^2}[\varepsilon-\delta]\left(1-\sin^2\theta_{SV}\right)\right) \tag{2.34}$$

ve

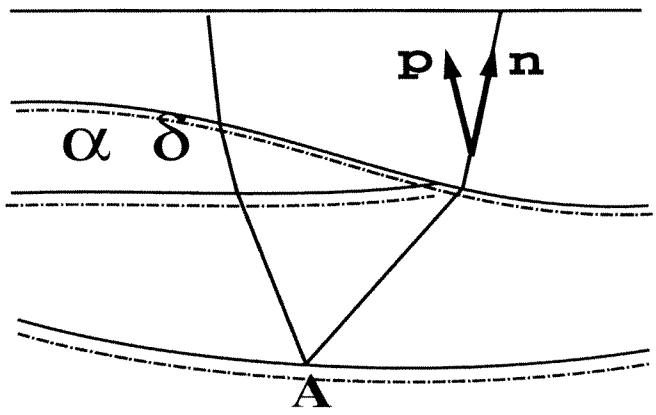
$$\tan\phi_{SH} = \tan\theta_{SH}\left(1+2\gamma\right) \tag{2.35}$$

olmaktadır (Thomsen, 1986). Buna göre verilen faz (dalga cephesine normal) açısına karşılık grup açısı bu ifadeler kullanılarak hesaplanabilir.

VTI ortamda hız eğrileri ve dalga cephesi eğrileri, proje yürütücüsü (Kadıoğlu, 2002) tarafından 'Enine Yön Bağımsız Ortamda Dalga Yayılımı' adlı makalede ayrıntılı olarak (**EK 1.**) anlatılmıştır.

3. VTI ORTAMDA IŞIN İZLEME YÖNTEMİ İLE MODELLEME

3.1. BİR IŞIN ÇİFTİ BOYUNCA SEYAHAT ZAMANI



Şekil 3.1. VTI ortamda bir ışın çiftinin ilerleyişi

Genel olarak ışın teorisi ayrıntılı bir şekilde Cerveny (1985) ve Ergin (1995) tarafından sunulmuştur. Burada VTI ortamda ışının ilerlemesi dikkate alınacaktır. Elastik parametrelerin sürekli olarak değiştiği, arayüzeylerle birbirinden ayrılan, tabakalı ve bu tabakalar içinde parametrelerin sürekli değiştiği bir VTI ortam düşünüldüğünde bir CRP ışın çifti boyunca seyahat zamanı

$$t = \int_{ray} p_i \eta_i dl \tag{3.1}$$

ile verilir (Fagin, 1998). Burada p_i yavaşlık vektörü $Sl = 1/\alpha$ nın bileşenlerini, η_i ışına tanjant birim vektörü, integral ise özel boyutlardaki ışının toplamını göstermektedir.

Quasi P dalgası yayılımı için hız vektör bileşenleri

$$G = 1 - K + L = 0 \ , \tag{3.2}$$

ile tanımlanan bir ilişkiyi sağlamaktadır. Burada

$$K = (C_{44} + C_{11})P_1^2 + (C_{33} + C_{44})P_3^2 \tag{3.3}$$

ve

$$L = \left((C_{11}p_1^2 + C_{44}p_3^2)(C_{44}p_1^2 + C_{33}p_3^2) - (C_{13} + C_{44})^2 p_1^2 p_3^2\right) \tag{3.4}$$

(Thomsen, 1986; Farra and Madariaga, 1988; Kadıoğlu, 2003). C_{11}, C_{33}, C_{44} ve C_{13} elastik sabitlerdir. $\beta = \sqrt{(C_{44}/\rho)}$ düşey S dalgası hızı P dalga yayılımını pek fazla etkilemediği için P ve S dalgasını etkileyen katsayılar arasında

$$C_{44} = fC_{33} \tag{3.5}$$

gibi bir bağıntı kullanılabilir. Yine C_{33} yerine düşey yavaşlık vektörü

$$S_1 = 1/\sqrt{C_{33}} \tag{3.6}$$

kullanılabilir. Bu durumda P dalgası yavaşlık vektör bileşenleri

$$G = S_1^4 - K + L = 0 \ , \tag{3.7}$$

ile tanımlanan bir ilişkiyi sağlamaktadır. Burada

$$K = (1 + f)S_1^2\left(P_x^2 + P_z^2\right) + 2\varepsilon S_1^2 P_x^2 \tag{3.8}$$

ve

$$L = \left(f\left(P_x^2 + P_z^2\right)^2 - 2\delta(1-f)P_x^2P_z^2 + 2\varepsilon P_x^2\left(fP_x^2 + P_z^2\right)\right) \tag{3.9}$$

dir.

Işın çiftinin çözümü ise birbirine bağlı diferansiyel denklem takımı

$$\frac{dx}{d\sigma} = -G_{,p_x}, \qquad \frac{dz}{d\sigma} = -G_{,p_z}, \qquad \frac{dP_x}{d\sigma} = G_{,x}, \qquad \frac{dP_z}{d\sigma} = G_{,z}, \tag{3.10}$$

ile cözülür (Cerveny, 1972; Payton, 1983; Farra ve Madariaga, 1988). σ ışın boyunca sürekli değişen değişkeni tanımlamaktadır.

Işın çiftinin çözümü için ilk koşullar

$$x(0) = x_0,\ z(0) = z_0,\ P_x(0) = P_x^0,\ P_z(0) = P_z^0 \tag{3.11}$$

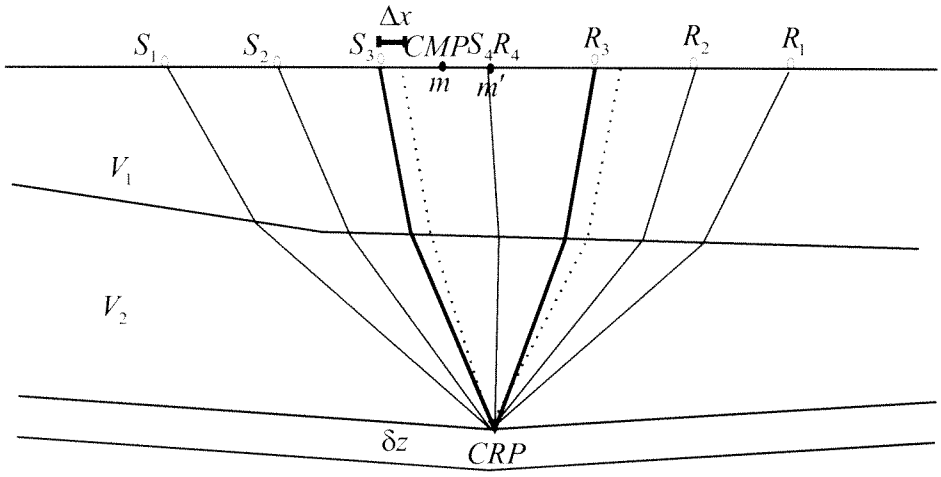
şeklinde tanımlanır.

Çalışmanın ilk adımı olarak izotropik ortamda ($\delta = 0$) sıfır ofset ışın izleme modellemesi sonucunda varış zamanları değerlendirilerek izotropik ilk parametreler (tabaka sayısı ve tabakaların değişimi, tabakalara ait izotropik tahmini hız değerleri) belirlenir. Bu değerler tomografi işlemi için giriş verileri olarak kullanılır. Daha sonra maksimum ofset sayısı, ofset aralığı ve minimum ofset değeri, modelleme de kullanılacak tabakalara ait ofset sayısı (örnek: max. Ofset sayısı: 20, tabakalara göre ofset düzenlemesi :5, 10, 15) verilerek VTI ortamda ortak yansıma noktası (CRP;common reflecting point) ışın izleme yöntemi (bir ışın çiftinin arayüzey üzerindeki yansıma noktasından yüzeye daha önce tanımlanmış ofsete ulaşması) ile modelleme yapılır. Modelleme için ışının seyahat zamanı hesabı beşinci derece Cash-Karp Runge-Kutta yöntemi ile hesaplanmıştır (Kadıoğlu ve diğ., 2002)(**EK 2.**).

4. YIĞMA ÖNCESİ DERİNLİK MİGRASYONU

4.1. MODELE DAYALI GÖRÜNTÜLEME

Modele dayalı görüntüleme hız analizi için düzenlenmiş yığma öncesi derinlik migrasyonudur (Kosloff ve diğ., 1996). Migrasyon çıkışı ortak yansıma noktası (CRP) gruplarıdır. Snell kanunu arayüzeyden yansıyan bir ışın çiftinin yüzeydeki orta noktası (CMP) ile CRP noktası arasındaki ilişkiyi sağlar. Doğru yığma öncesi derinlik migrasyonunda bütün atış alıcı çiftleri verilen ofsetleri ile CRP noktasının görüntülenmesine katkı sunarlar. Yansıma olayları için orta noktaya en yakın olan atış alıcı çifti en büyük genlik katkısını sunar. Tek bir izin gidiş geliş aralığı ortak orta nokta (CMP) düzeninden CRP düzenine geçişi sağlayacaktır. Yani m' deki yansıma zamanı, m CMP noktasındaki yansıma zamanından yararlanılarak bulunur (Şekil 4.1.). Bu işlem ışın izleme yöntemi ile gerçekleştirilir. Bu düzenleme ile tek bir CMP noktası farklı tabakalardaki farklı CRP noktalarına katkı sunabilmektedir.



Şekil 4.1. Yığma öncesi derinlik migrasyonu için CRP ışın yolları ve δz derinlik sapması. S noktaları atış, R noktaları alıcı pozisyonlarını göstermektedir.

$$t_{m'} \approx t_m + (x_{m'} - x_m)(P_1 + P_2) \tag{4.1}$$

bağıntısı ile m CMP noktasından arayüzey üzerindeki CRP noktasının yüzey izdüşümü olan m' noktasına zaman değeri aktarılır (Kosloff ve diğ., 1996). Burada P_1 ve P_2 değerleri CRP ışınlarının yüzeydeki ışın parametreleridirler. Bir arayüzey üzerindeki herbir CRP ışın çiftinin kaynak ve alıcı noktalarındaki ışın parametrelerinin hesaplanması ve seyahat zamanı hesaplanması beşinci dereceden Cash-Carp Runge-Kutta yöntemi ile yapılmıştır.

5. YIĞMA ÖNCESİ DERİNLİK MİGRASYONU İLE ELDE EDİLEN CRP GRUPLARININ TOMOGRAFİSİ VE PARAMETRELERİN ELDE EDİLMESİ

5.1. TOMOGRAFİK PRENSİPLER

Yeraltı parametrelerinde δS_l, $\Delta\delta$ and δz gibi bir saçılma olduğu düşünüldüğünde ışın yolu durağan olduğu için seyahat zamanında lineer bir yaklaşım ile değişiklik olacaktır. Buna göre

$$\delta t = \int_{ray} \delta p_i \eta_i dl + \sum_{k=1}^{2N_L-1} \Delta P_z^k \delta z_k \ . \tag{5.1}$$

dir. Burada δp_i, δS_l ve $\Delta\delta$ 'deki saçılmalardan dolayı meydana gelen değişikliği göstermektedir. N_L CRP noktaları üzerindeki arayüzey sayısını, ΔP_z^k ise arayüzeyler ile ışının k. arayüzeydeki düşey yavaşlık vektöründeki değişikliği tanımlamaktadır. İntegral saçılmamış orjinal ışın yolu boyunca alınır.

Sağdaki ilk terim parametrelerdeki değişikliğin sonucu olarak seyahat zamanındaki değişikliği, ikinci terim ise Farra ve Madariaga (1988) ve Kosloff ve diğ. (1996)'e göre arayüzeyin hareketinin bir sonucu olarak tariflenmiştir.

Parametrelerdeki saçılmalardan dolayı (3.7) denklemi

$$G\left(p_i + \delta p_i, S_l + \delta S_l, \delta + \Delta\delta\right) = 0 \tag{5.2}$$

olmalıdır. Böylece

$$\delta G = \frac{\partial G}{\partial S_l}\delta S_l + \frac{\partial G}{\partial \delta}\Delta\delta + \frac{\partial G}{\partial p_i}\delta p_i = 0 \tag{5.3}$$

olur. Işın denklemlerinden

$$\eta_i = \frac{dx_i}{dl} = -G_{,p_i} / \left(G_{,p_i} G_{,p_i}\right)^{1/2} \tag{5.4}$$

bulunur. (5.3) ve (5.4) denklemleri birleştirildiğinde

$$\eta_i \delta p_i = \left(\frac{\partial G}{\partial S_l} \delta S_l + \frac{\partial G}{\partial \delta} \Delta\delta \right) \Big/ \left(G_{,p_i} G_{,p_i}\right)^{1/2} \tag{5.5}$$

olur. Bu sonuç (5.1) denkleminde yerine konulduğunda

$$\delta t = \int_{ray} \left(\frac{\partial G}{\partial S_l} \delta S_l + \frac{\partial G}{\partial \delta} \Delta\delta \right) \Big/ \left(G_{,p_i} G_{,p_i}\right)^{1/2} dl + \sum_{k=1}^{2N_L-1} \Delta P_z^k \delta z_k \tag{5.6}$$

tomografik prensipleri tanımlayan denklem tamamlanmış olur. δt değerleri yığma öncesi derinlik migrasyonundaki değişken ofsetler üzerindeki derinlik hatalarından yararlanılarak düşey zaman değerleri olarak hesaplanır.

5.2. DERİNLİK HATA DEĞERİNİ DÜŞEY ZAMANDA HATA DEĞERİNE DÖNÜŞTÜRME

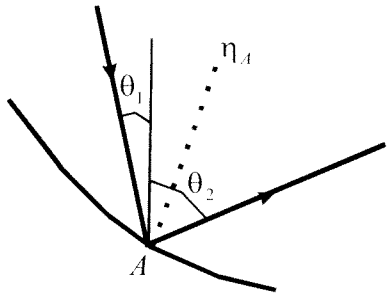
Tomografide sonuçlar zaman değerleri cinsinden kullanılırlar. Bu nedenle migrasyon sonucu elde edilen CRP grupları üzerindeki derinlik hata değerleri düşey zaman hata değerlerine dönüştürülür. Bunun için bir CRP noktasından gelen bir ışın çifti ele alındığında (Şekil 5.1) bu ışınların CRP noktasında Snell kanununu sağlamalıdır. CRP noktasındaki δz kadarlık bir düşey değişim (5.1) ifadesi kullanıldığında

$$\delta t = \Delta P_z \delta z \tag{5.7}$$

zamanda değişikliğe denk gelmektedir. Burada

$$\Delta P_z = (\cos\theta_1 + \cos\theta_2)/\alpha \tag{5.8}$$

dır. θ_1 ve θ_2 açıları CRP noktasındaki ışın çiftinin düşey eksen z ile yaptıkları açıdır. α düşey P dalgası hızıdır.



Şekil 5.1. A yansıma noktasındaki düşey yavaşlık açıları (Kosloff ve diğ., 1996'dan uyarlanmıştır).

Derinlik değişimi δz değeri $\delta\tau$ birim zamana göre ölçeklenirse

$$\delta z = \alpha \frac{\delta\tau}{2} \tag{5.9}$$

ve buradan

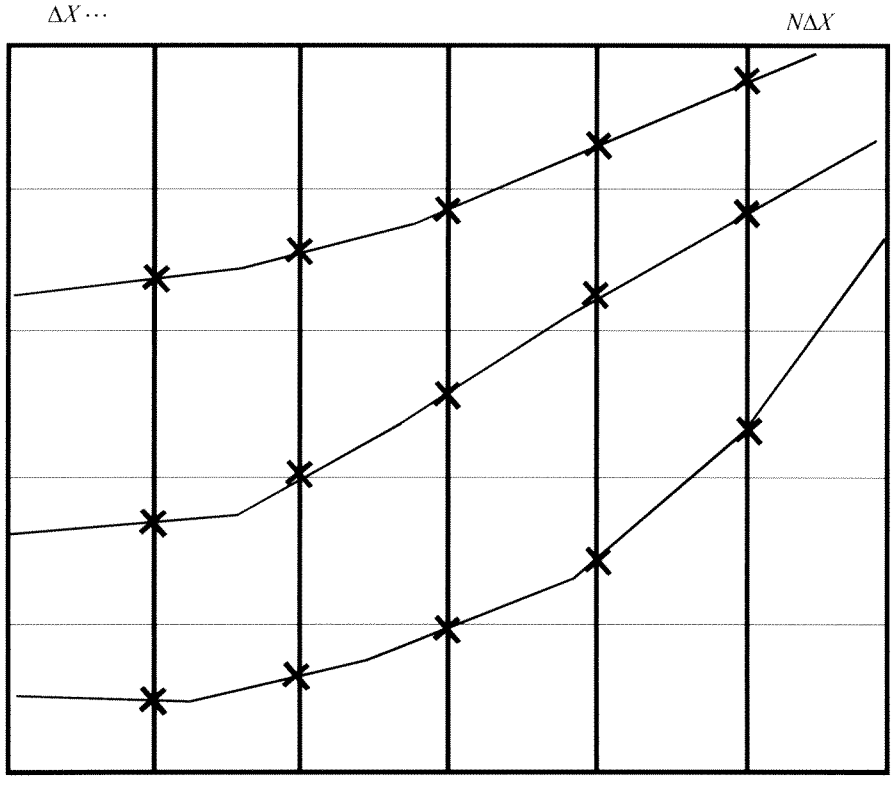
$$\delta t = \frac{\cos\theta_1 + \cos\theta_2}{2}\delta\tau \tag{5.10}$$

elde edilir. Bu ifade birim zaman $\delta\tau$ ile δt arasındaki ilişkiyi verir. Cosinüslü terim 'germe faktörü' olarak adlandırılır (Kosloff ve diğ., 1996). Buna göre δz verilen referans derinlik ile tabakanın imaj derinliği arasındaki farkı tanımlar. Buna uygun olarak δt değeri referans derinlikteki CRP ışın çifti boyunca seyahat zamanı ile tabaka imajından gelen seyahat zamanı

arasındaki farkı tanımlamaktadır. Sonuç olarak derinlik migrasyonu üzerindeki tüm CRP noktalarına ait δt değerleri tomografi için giriş değerlerini oluşturur.

5.3. TOMOGRAFİ DENKLEMLERİ

Tomografi denklemleri özel ayrıklaştırma işleminden sonra tomografik prensiplerden yararlanaran elde edilir. Burada δS_l, $\Delta\delta$ ve δz düşey yönde sabit bir değer, yanal yönde değişken kabul edilmektedirler ve eşit olarak belirlenmiş node noktalarından yararlanılarak Hermite interpolasyon ile hesaplanırlar (Raltson ve Rabinovitch, 1978; Kosloff ve diğ., 1996) (Şekil 5.2).



Şekil 5.2. Yeraltı modeli için belirlenmiş sabit node noktaları.

N_l tabaka sayısı, N_p herbir tabakadaki interpolasyon nokta sayısı olarak tanımlandığında bilinmeyen parametre sayısı $M = 3N_l N_p$ dır. Veri sayısı N_d ise tabaka sayısına, CRP sayısına ve herbir CRP deki ofset sayısına bağlıdır.

Sonuç olarak tomografi denklemi

$$\boldsymbol{A}\delta\boldsymbol{m} = \delta\boldsymbol{t} \tag{5.11}$$

dir. Burada

$$\begin{aligned} \delta \mathrm{m} &= \left(\delta S_l^0 \Delta\delta^0 \delta z^0 \delta S_l^1 \Delta\delta^1 \delta z^1 \cdots\cdots\right) \\ \delta t &= \left(\delta t^0 \delta t^1 \delta t^2 \cdots\cdots\right) \end{aligned} \tag{5.12}$$

şeklinde tanımlanır. δm S_l yavaşlık vektörü, δ anizotropi parametresi ve z tabaka derinlik değişimlerini içermektedir. δt tabakalardaki bütün CRP noktalarındaki bütün ofsetler için seyahat zamanı hata miktarını tanımlamaktadır. A Jacobian matrisini ifade etmektedir.

(5.11) matris denklemi aşırı tanımlı bir problem olup ağırlıklandırılmış sönümlü en küçük kareler yöntemi ile çözülmektedir. Buna göre

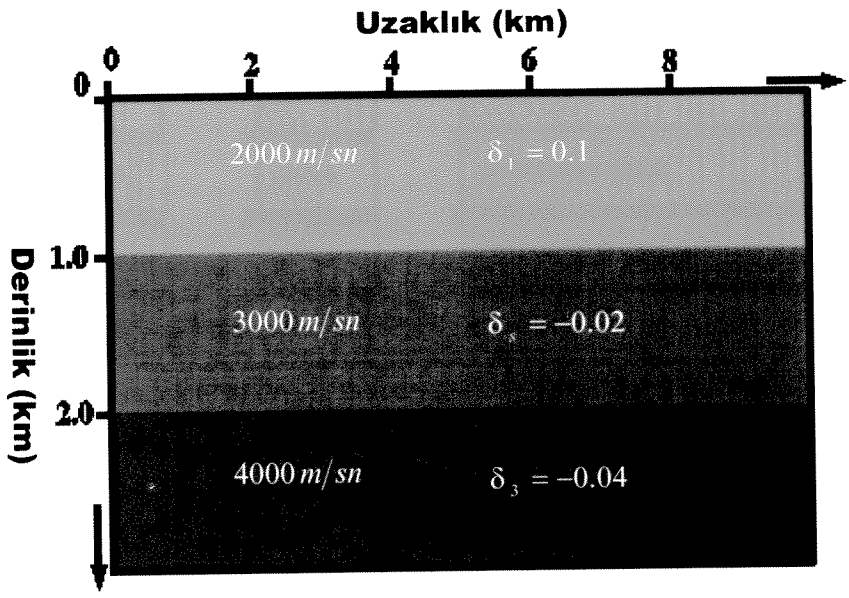
$$\delta m = \left(A^T C_d^{-1} A + C_m^{-1}\right)^{-1} A^T C_d^{-1} \delta t \tag{5.13}$$

dir. Burada C_d ağırlıklandırma fonksiyonu verinin diagonali üzerindeki varyansı içeren kovaryans matrisidir, C_m sönümleme faktörü parametre kovaryans matrisi olarak alınmıştır (Menke, 1989 sh. 58 ve 83). Parametre kovaryans matrisi ters çözümün duyarlılığını kontrol etmektedir.

6. UYGULAMALAR

6.1. YATAY TABAKALI YER MODELİ

VTI ortamda tomografinin ilk modeli üç düzlem tabakadan oluşmaktadır. Tabakaların hızları sırasıyla 2000, 3000, ve 4000 m/sn dir. Tabaka derinlikleri ise yine sırasıyla 1000, 2000 ve 3000m dir. δ parametresinin tabakalar için değerleri sırasıyla 0.1, -0.02, -0.04. Şekil 6.1. modeli tanımlamaktadır.

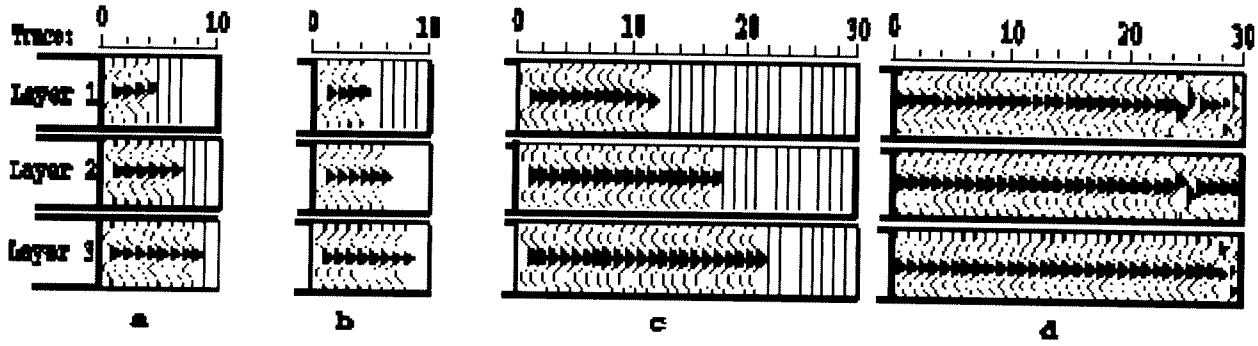


Şekil 6.1. Yatay tabakalı yer modeli için doğru derinlik-hız ve Thomsen δ anizotropi parametresi değerleri.

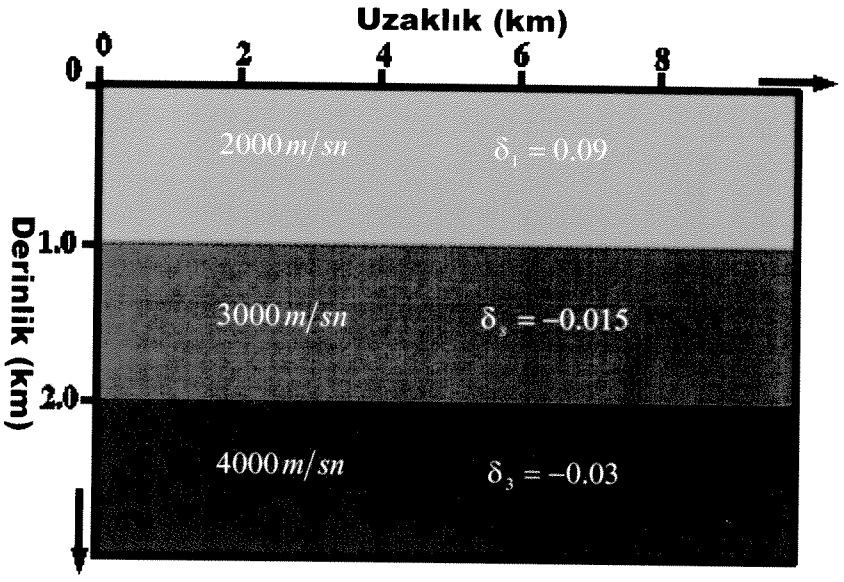
Yatay tabakalı gerçek yer modeli parametre değerlerinin elde edilebilmesi testi için oluşturulan giriş modeli izotropik bir model olup gerçek modelin sıfır ofset varış zamanları korundu. İzotropik tabaka hızları sırasıyla 2200, 3300 ve 4400 m/sn olarak alındı. Tabaka derinlikleri 100m daha derin alındı. δ parametresi izotropik ilk model için sıfır alındı.

Yığma öncesi modelleme, derinlik migrasyonu ve tomografi için 20 m aralıklarla 500 CMP noktası alındı. Her bir CMP noktası için minimum ofset değeri 0 m olup 200 m artışlarla 31 ofset kullanıldı. Tomografi için altı iterasyon yapıldı. İterasyon adımlarında herbir tabaka için kullanılan ofset sayısı aynı olmayıp tabaka konumlarına, iterasyon sayısına, iterasyon sonucu

elde edilen CRP gruplarının düzgünlüğüne göre ayarlandı. İlk iterasyonda yatay üç tabaka için ofset sayısı sırasıyla 4-6-8 düzenindedir. İkinci iterasyonda 8-12-16, üçüncü iterasyonda 12-18-22, dördüncü iterasyonda 20-24-28, beşinci iterasyonda 25-28-30 ve son iterasyonda 31-31-31 şeklinde tüm ofsetler kullanıldı. 300. istasyon (CMP noktası) için iterasyon öncesi ve bazı iterasyon adımları sonucu elde edilen CRP grupları Şekil 6.2. de verildi. Tomografi düzgün CRP gruplarını sağlayan anizotropik derinlik hız modelini hesapladı. Tomografi sonucu elde edilen yer modeli ise Şekil 6.3. ile verildi. CRP gruplarının düzgünlüğü iterasyonun başarısını göstermektedir.



Şekil 6.2. 300. istasyondaki **a)** iterasyon öncesi, **b)** birinci iterasyon sonrası, **c)** üçüncü iterasyon sonrası, **d)** altıncı iterasyon sonrası, CRP grupları.



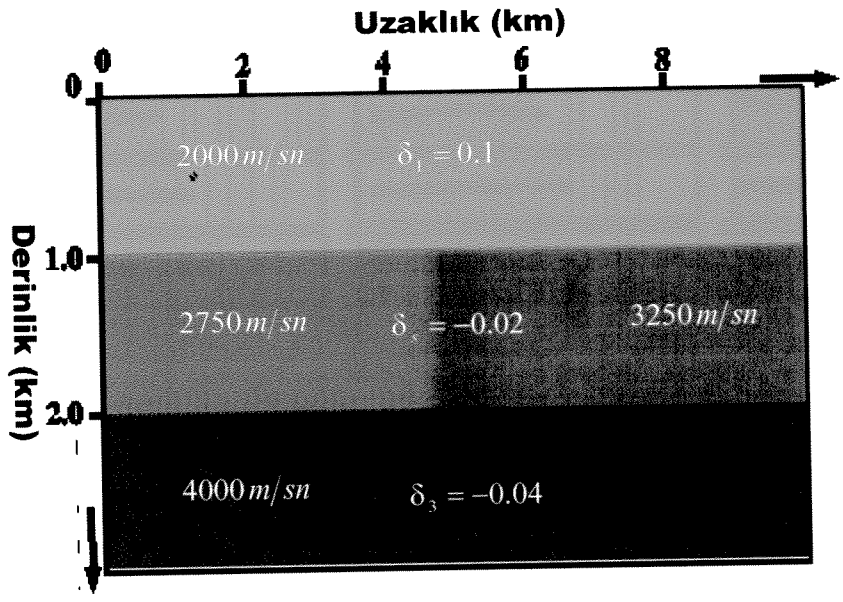
Şekil 6.3. Tomografide altı iterasyon sonucu elde edilen derinlik-hız değerleri ve δ değerleri.

6.2. ARA TABAKASI DEĞİŞKEN HIZA SAHİP YATAY TABAKALI MODEL

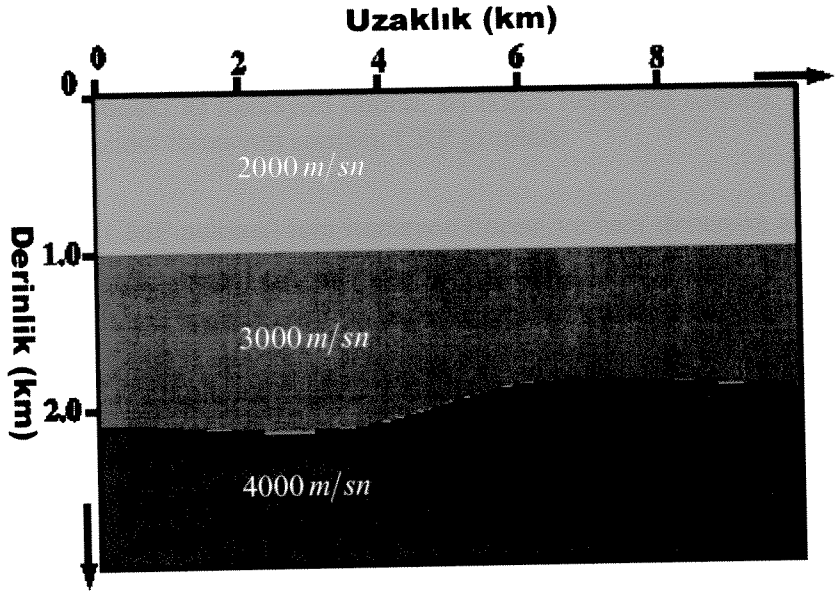
Bu model yine yatay üç tabakadan oluşmakta ancak ara tabakada hız yanal olarak değişmekte, tabaka ortasında hız için bir geçiş zonu bulunmaktadır. Tabaka kalınlıkları birinci modeldeki gibi 1000m olarak alınmıştır. Birinci ve son tabaka sabit hızlara sahip olup sırasıyla 2000 m/sn ve 4000 m/sn olarak alınmışlardır. İkinci tabakanın sol tarafında hız 2750, sağ tarafında 3250 ve ortada iki hız arasında bir hız geçiş zonu bulunmaktadır. Tabakalara ait δ anizotropi parametresi değerleri sırasıyla 0.01, -0.02 ve –0.04 dir (Şekil 6.4).

Giriş modeli izotropik bir model olup gerçek modelin sıfır ofset varış zamanları korundu. İzotropik tabaka hızları sırasıyla 2000, 3000 ve 4000 m/sn olarak alındı. δ parametresi izotropik ilk model için sıfır alındı. Şekil 6.5. de Giriş parametrelerine göre elde edilen model sunuldu.

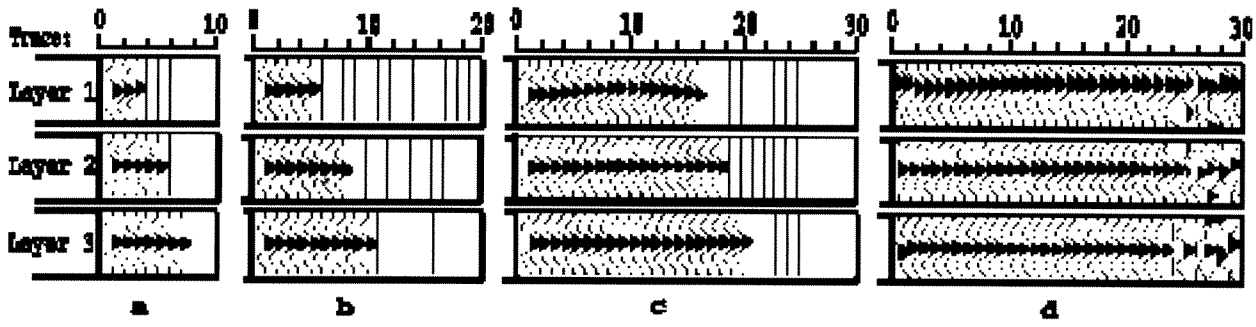
Yığma öncesi modelleme, derinlik migrasyonu ve tomografi için 20 m aralıklarla 500 CMP noktası alındı. Her bir CMP noktası için minimum ofset değeri 0 m olup 200 m artışlarla 31 ofset kullanıldı. Tomografi için sekiz iterasyon yapıldı. İterasyon adımlarında herbir tabaka için kullanılan ofset sayısı aynı olmayıp tabaka konumlarına, iterasyon sayısına, iterasyon sonucu elde edilen CRP gruplarının düzgünlüğüne göre ayarlandı. Herbir iterasyon için tabaka sıralarına göre kullanılan ofset sayısı şu şekildedir. 3-5-7, 5-8-10, 8-10-12, 12-14-16, 14-18-20, 20-22-24, 24-26-28, 31-31-31.



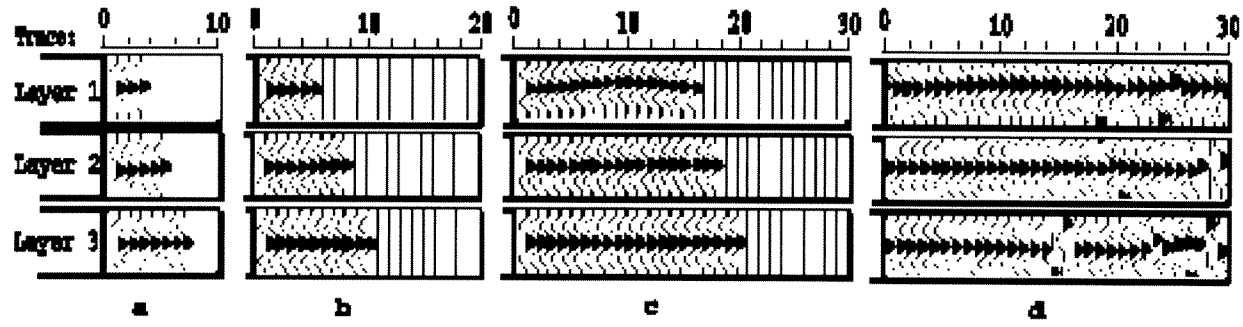
Şekil 6.4. Ara tabakası değişken hızlı yatay tabakalı yer modeli için doğru derinlik-hız ve δ anizotropi parametresi değerleri.



Şekil 6.5. Tomografiye giriş olarak verilen izotropik derinlik-hız modeli.



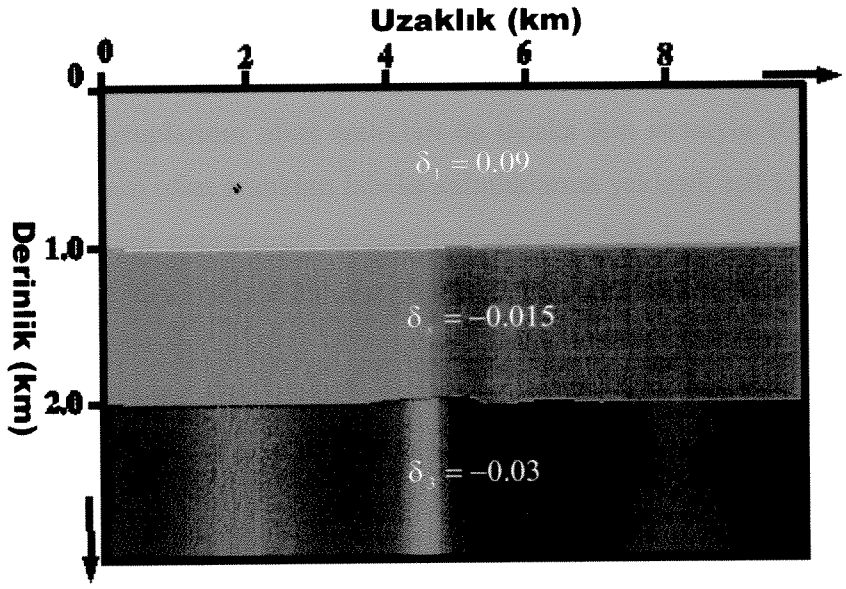
Şekil 6.6. 100. istasyondaki **a)** iterasyon öncesi, **b)** ikinci iterasyon sonrası, **c)** beşinci iterasyon sonrası, **d)** sekizinci iterasyon sonrası, CRP grupları.



Şekil 6.7. 450. istasyondaki **a)** iterasyon öncesi, **b)** ikinci iterasyon sonrası, **c)** beşinci iterasyon sonrası, **d)** sekizinci iterasyon sonrası, CRP grupları.

100. ve 450. istasyon (CMP noktası) için iterasyon öncesi ve bazı iterasyon adımları sonucu elde edilen CRP grupları sırasıyla Şekil 6.6. ve Şekil 6.7. de verildi.

Tomografi düzgün CRP gruplarını sağlayan anizotropik derinlik hız modelini hesapladı. Şekil 6.8. Tomografi sonucu elde edilen yer modelini göstermektedir. Şekil 6.4, Şekil 6.6, Şekil6.7. ve Şekil 6.8. gerçek model ile tomografi sonucu elde edilen modellerin yakınlığını göstermektedirler. Orta tabaka derinlikte çok azda derinlik saçılması olsada tomografi sonucu elde edilen parametreler, gerçek parametrelere çok yakındır.



Şekil 6.8. Tomografide altı iterasyon sonucu elde edilen anizotropik derinlik-hız değerleri ve δ değerleri.

6.3. SENKLİNAL MODEL

Senklinal model tabaka hızları sırasıyla 2000 m/sn, 3000m/sn ve 4000m/sn olan üç tabakadan oluşmaktadır. İkinci tabakanın kalınlığı 800metredir. Anizotropi parametresi δ 'nın tabakalara göre değeri sırasıyla 0.1, -0.050 ve –0.010 dir.
Bu model 20 metre aralıklı 500 CMP noktası ile üretildi. Maksimum ofset sayısı diğer modellerdeki gibi 31 ofset olarak alındı. ilk ofset aralığı 0 olup ofsetler arası artış 200 m alındı.

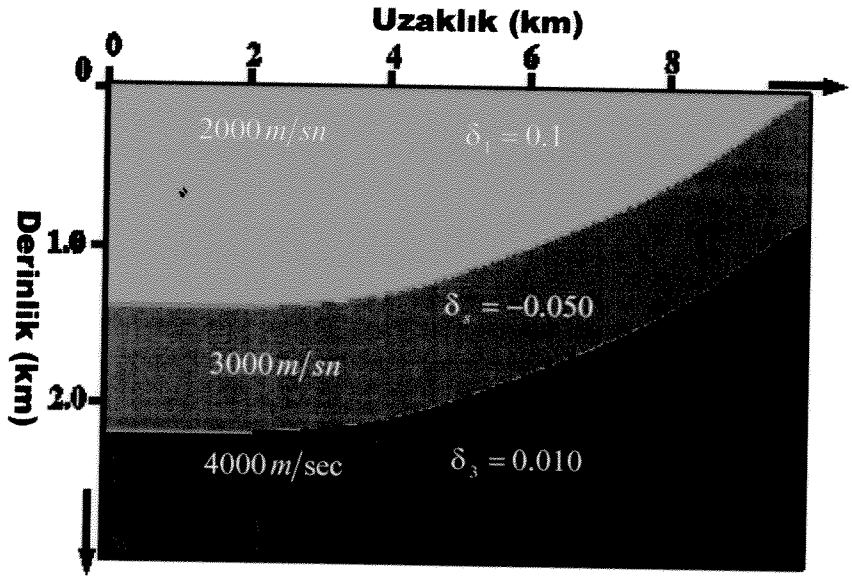
Giriş modeli izotropik ortamda oluşturuldu. İzotropik tabaka hızları sırasıyla 2200, 3700 ve 4000 m/sn alındı. δ parametresi ise izotropik ilk model için sıfır alındı. İlk tabakanın derinliği ise gerçek modelden yaklaşık 100m daha derin alındı. Giriş parametrelerine göre elde edilen model Şekil 6.10. da sunuldu.

Yığma öncesi modelleme, derinlik migrasyonu ve tomografi için yine 20 m aralıklarla 500 CMP noktası alındı. Her bir CMP noktası için minimum ofset değeri 0 m olup 200 m artışlarla 31 ofset kullanıldı.

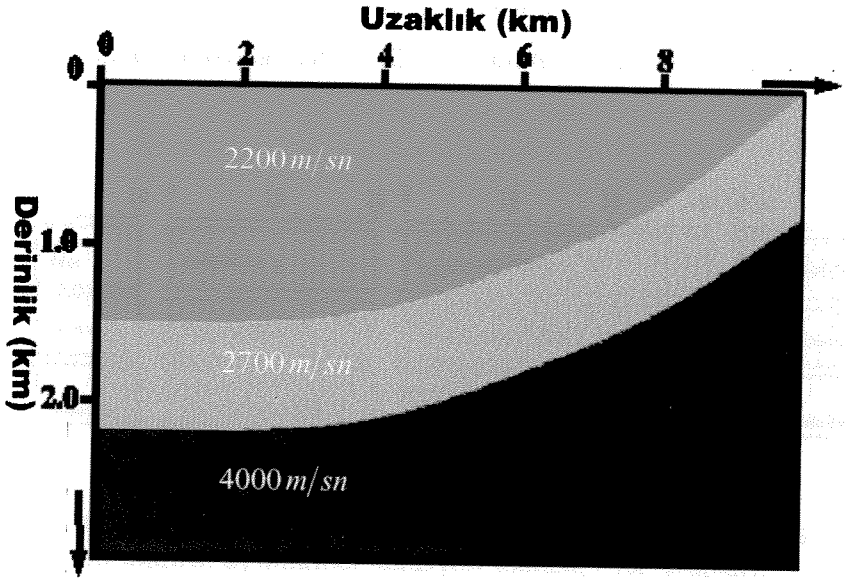
Öncelikle Alkhalifah ve Tsvankin (1995) tarafından verilen tomografinin anizotropiyi iyi çözebilmesi için çok geniş bir ofset aralığı gerektiği sonucu test edildi ve yığma öncesi göç ettirilmiş CRP gruplarının ofset aralıkları üç tabaka için sırasıyla 1000 m, 2000 m ve 3000 m olarak alındı. Yani ofset aralıkları tabaka kalınlıklarına yaklaşık eşit alındı. Tomografi sonucunda bulunan anizotropi parametresi değerlerinin gerçek değerlerle uyuşmadığı görüldü. Hatta hız değerlerinin geliştirilmesi durumuda zayıftı. Daha sonra tüm tabakalar için ofset aralığı 3000m olarak alındı. Sonucun gerçek modele çok yakın olduğu görüldü. Sonuç tomografinin doğru parametre değerlerine ulaşabilmesi için çok geniş bir ofset aralığına ihtiyaç olduğunu gösterdi. Sonuçlar Alkhalifah ve Tsvankin (1995) tarafından varılan sonucu destekledi.

Tomografi için oniki iterasyon yapıldı. İterasyon adımlarında herbir tabaka için kullanılan ofset sayısı iterasyon sonucu elde edilen CRP gruplarının düzgünlüğüne göre ayarlandı. Herbir iterasyon için tabaka sıralarına göre kullanılan ofset sayısı şu şekildedir: 3-5-7, 5-8-10, 8-10-12, 10-12-14, 12-14-16, 14-16-18, 16-18-120, 18-20-22, 20-22-24, 22-24-26, 26-28-30, 31-31-31.

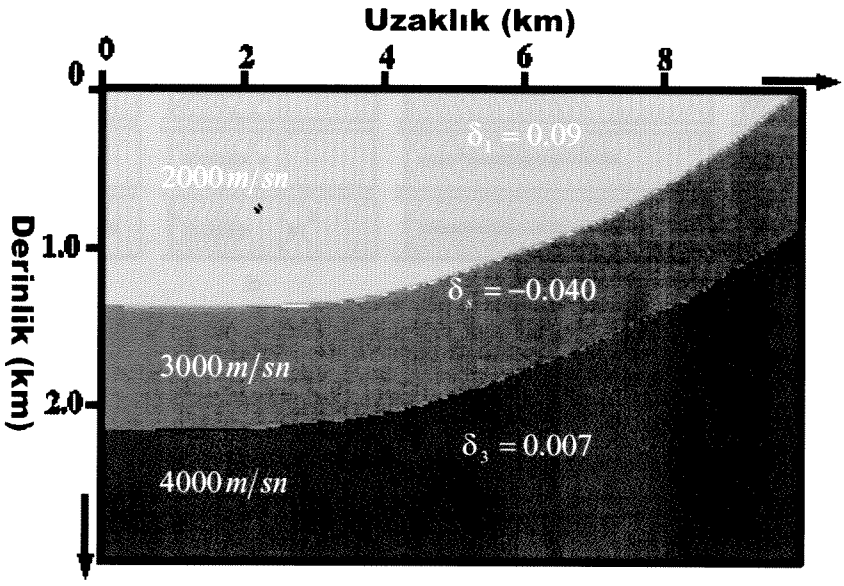
Tomografi düzgün CRP gruplarını sağlayan anizotropik derinlik hız modelini hesapladı. Tomografi sonucu elde edilen anizotropik model Şekil 6.11. de verilmektedir. Şekil 6.9. ve Şekil 6.11. gerçek model ile tomografi sonucu elde edilen modellerin yakınlığını göstermektedirler. Orta tabaka derinlikte çok azda derinlik saçılması olsada tomografi sonucu elde edilen derinlik, anizortopik hız değerleri, gerçek modelin derinlik, hız değerleriyle oldukça örtüşmektedir. δ değerleri gerçeğine çok yakındır.



Şekil 6.9. Senklinal yapıdaki yer modeli için gerçek derinlik-hız ve δ anizotropi parametresi değerleri.

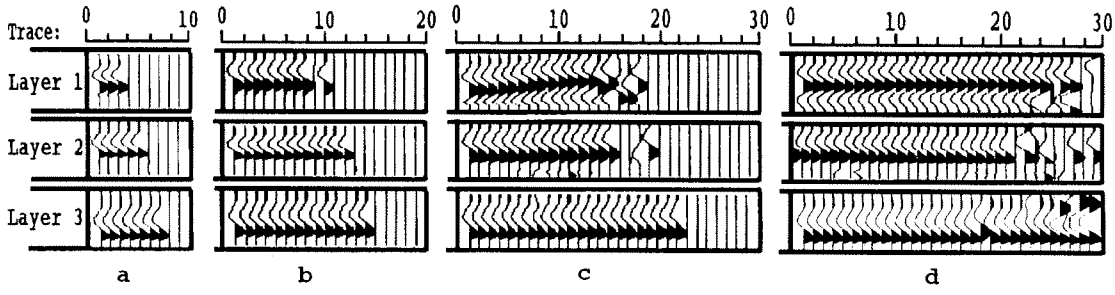


Şekil 6.10. Senklinal model için tomografiye giriş olarak verilen izotropik derinlik-hız modeli.

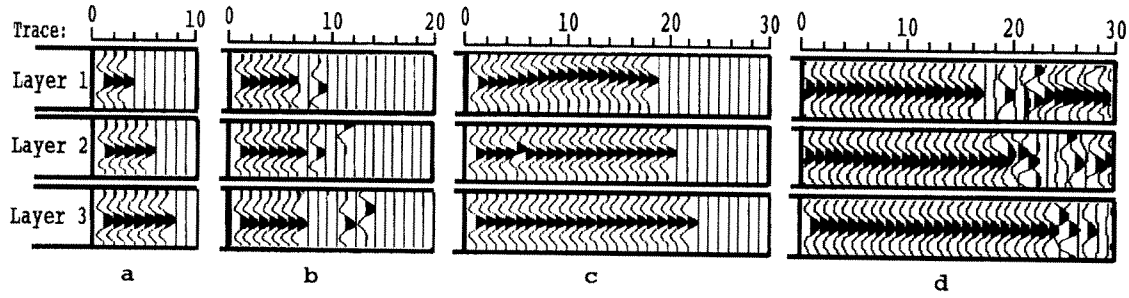


Şekil 6.11. Tomografide oniki iterasyon sonucu elde edilen derinlik-hız ve δ değerleri.

100. ve 356. istasyon (CMP noktası) için iterasyon öncesi ve bazı iterasyon adımları sonucu elde edilen CRP grupları sırasıyla Şekil 6.12. ve Şekil 6.13. de verilmiştir.



Şekil 6.12. 100. istasyondaki **a)** iterasyon öncesi, **b)** dördüncü iterasyon sonrası, **c)** sekizinci iterasyon sonrası, **d)** onikinci iterasyon sonrası, CRP grupları.



Şekil 6.13. 356. istasyondaki **a)** iterasyon öncesi, **b)** dördüncü iterasyon sonrası, **c)** sekizinci iterasyon sonrası, **d)** onikinci iterasyon sonrası, CRP grupları.

7. SONUÇLAR

Proje kapsamında VTI ortamda ışın izleme yöntemi ile modelleme, yığma öncesi derinlik migrasyonu ve tomografi işlemleri teoriksel olarak araştırıldı ve geliştirildi. C programlama dili kullanılarak her bir çalışmanın programı yazıldı ve birleştirildi. Yazılan programın doğruluğunu test etmek amacıyla yatay düzlemin düşey simetri ekseni etrafında rotasyonu durumunda özelliklerinin değişmediği VTI ortam şartını sağlayabilecek, yatay arayüzeyli tabakalar ve senklinal tipi eğimli arayüzeylerden oluşan tabakalardan oluşan, modeller oluşturuldu. Oluşturulan modellerin tabaka derinlikleri, anizotropik hızları ve Thomsen δ anizotropi parametresi değerleri gerçek yani doğru parametre değerleri olarak alındı. Doğru modellerin ışın izleme yöntemi ile sıfır ofset modellemesi yapılıp tomografi işlemi için parametre giriş modeli oluşturuldu. Giriş modellerinde izotropik hız değerleri kullanıldı ve tabakalara ait Thomsen δ anizotropi parametresi değerleri sıfır olarak alındı. Giriş modellerine ait tabakalar için başlangıç ofset sayıları belirlendi ve ışın izleme yöntemi kullanılarak modellemeleri yapıldı. Elde edilen CRP gruplarına derinlik migrasyonu uygulandı. Parametrelerdeki saçılmaları baz alarak oluşturulan tomografi denklemleri sönümlü en küçük kareler yöntemi ile çözümlenip eşit aralıklı node noktalarına ait parametrelerdeki sapma miktarları belirlendi. Lineer Hermite interpolasyon ile tüm noktalara ait parametrelerdeki sapma değerleri atandı. Bu sapmalar ilk parametre değerlerine eklenip yeniden modellemeleri yapıldı ve tabakalara ait CRP gruplarının düzgünlüklerinin korunup korunmadığı kontrol edildi. Düzgünlüklerinin korunması durumunda iterasyon için tabakalara ait ofset sayıları bir öncekine göre artırılıp işlemler aynı sıra ile her bir iterasyonda tekrarlandı. CRP gruplarının düzgünlüğünün korunmaması durumunda son iterasyon iptal edilip tabakalar için ofset sayısı daha küçük adımlarla artırılarak iterasyon işlemi devam ettirildi. Herbir iterasyonda tabakalar için verilen ofset sayıları, CRP gruplarının düzgünlüğüne bağlı olarak büyük veya küçük adımlar olarak belirlendi. Yani tomografi başlangıcında iterasyon sayısı sabit bir değer olarak verilmeyip CRP gruplarının düzgünlüğünü koruyacak şekilde iterasyon sayısı kontrol edildi. İterasyonlarda tabakalara ait ofset sayısı eşit alınmadı. Tabakanın derinliğine göre ofset sayısı artırıldı.

Elde edilen düzgün CRP grupları tomografinin doğruluğunu denetleyen önemli bir unsurdur. Yine tomografi sonucu elde edilen düşey yöne sabit yanal yöne değişen tabaka derinlikleri ve anizotropik tabaka hızları, Tabakalara ait Thomsen δ anizotropi parametresi değerleri ile bu parametrelerin gerçek değerleri arasındaki uyum tomografinin doğruluğunu göstermektedir. Elde edilen sonuçlar tomografi programının doğruluğunu ispatlamaktadır.

Teori kısıtlamalarından dolayı diğer Thomsen parametreleri tomografiye dahil edilememiştir. Teorik çalışmaların sonuçları, Alkhalifah ve Tsvankin (1995) çalışmalarının sonuçlarına benzer olarak, tüm parametrelerin doğru çözülebilmesi için çok geniş bir ofset dizilimini gerektirmektedir. Hatta ofset dizilimi yeterli olmazsa anizotropik tomografi hız tayinini geliştirememektedir. Bu nedenle tabakaların derinliği ile orantılı olarak ofset sayıları artırılmalıdır. Tomografinin başarıya ulaşması için hedeflenen derinlikten daha büyük bir profil açılımının yapılması gerekmektedir.

VTI ortam tomografi ile modele dayalı görüntüleme daha doğru hız, gerçeğine daha yakın derinlik bilgisi ve ortamın anizotropi bilgisi itibari ile sismik yansıma yönteminde daha doğru yeraltı bilgisi sunmaktadır.

8. REFERANSLAR

Alkhalifah, T., and Tsvankin, L., Velocity analysis for transversely anisotropic media. Geophysics 60, 1550-1666, (1995) .

Cerveney, V., Seismic rays and ray intensities in inhomogeneous anisotropic media. Geophys. J.R. astr. Soc., 29, 1-13, (1972).

Cerveney, V., The application of ray tracing to the numerical modelling of seismic ray fields in complex structures. Seismic Shear waves, Part A., Geophysical Press, 1-24, London, (1985).

Daley P.F., and Hron, F., Reflection and transmission coefficients for transversely isotropic media. Bull., Seis. Soc. Am., 67, 661-675, (1977)

Ergin, K., Advanced seismology (rays and waves). Tübitak Marmara Research Center, Department of Earth Sciences, (1995)

Farra, V., Madariaga, R., Non linear reflection tomography. Geophys. J. Internat., 95, 135-147, (1988).

Fagin, Model-based depth imaging. SEG, Course notes Serie No.10, Tulsa,Oklahoma USA, (1998).

Gaiser, J.E., Transversely isotropic phase velocity analysis from slownes estimates. J. Geophys. Res., 95, No. B7, 241-254, **(1990)**

Helbig, K., Foundations of anisotropy for exploration seismics. Pergamon, (1994).

Kadıoğlu, S., Enine Yönbağımsız Ortamda Dalga Yayılımı, Jeofizik, Vol. 15, No. 2, S. 83-94, (2002).

Kadıoğlu, S., Gürpınar, S. and Güreli, O., Beşinci Derece Cash-Karp Runge-Kutta, Yöntemi ile Dalga Cephesi Oluşturarak Seyahat Zamanı Hesaplaması, Jeofizik Vol. 15, No.2, S. 95-104, (2002).

Kadıoğlu, S., Modeling by Ray Tracing Method in Polar Anisotropic Medium, 14th International Petroleum Congress and Natural Gas Congress and Exhibition of Turkey, May, 12-14, Ankara-TURKEY, Proceedings, p.539-540, (2003).

Kosloff, D., Sherwood, J., Koren, Z., Machet, E., Falkovitz, Y., Velocity and interface depth determination by tomography of depth migrated gathers. Geophysics, 61, 1511-1523, (1996).

Lay, T. and Wallace, T.C., Modern global seismology, International Geophysical Series, 58, (1995).

Menke, W., Geophysical data analysis: Discrete inverse theory. Revised edition, Academic Press, Inc., New York, London, (1989).

Payton, R.G., Elastic wave propagation in transversely anisotropic media. Martinus Nijhoff Publishers, (1983).

Raltson, A., and Robinowitz, P., A first course in numerical analysis, McGraw-Hill Book, (1978).

Schoenberg, M., Sayers, C.M., Seismic anisotropy of fractured rock. Geophysics, 60, No.1., 204-211, (1995)

Schoenberg, M., Muir, F., and Sayers, C., Introducing ANNIE: A simple three-parameter anisotropic velocity model for shales. Journal of Seismic Exploration, **5**, 35-49, (1996).

Thomsen, L., Weak elastic anisotropy. Geophysics, 51, 1954-1966, (1986).

Thomsen, L., Understanding Seismic Anisotropy in Exploration and Exploitation. SEG, EAGE, Distinguished Instructor Series, No.5, Tulsa, OK USA, (2002).

Tsvankin, I., Model-based depth imaging. SEG, Tusla, Oklahoma,USA, 131-142, (1998).

Tsvankin, I., Normal moveout from dipping reflectors in anisotropic media. Geophysics, 60, 268-284, (1995).

Tsvankin, I., and Thomsen, L., Nonhyperbolic reflection moveout in anisotropic media. Geophysics, **59**, No. 8, 1290-1304, (1994).

JEOFİZİK 15, 83-94, 2001

ENİNE YÖN BAĞIMSIZ ORTAMDA DALGA YAYILIMI

Wave Propagation in Transversely Isotropic Media

Selma KADIOĞLU[1]

ÖZET

Bu çalışmada, iki boyutlu enine yön bağımsız bir ortamda (EY ortam; transversely isotropic media; TI medium) dalga yayılımı irdelenmiştir. Önce EY ortam tanımlanmış sonra iki boyutlu ortam için hareket denklemleri çıkartılmıştır. Dalga yayılımının hesaplanması için sayısal hesaplama yöntemi geliştirilmiştir. Hız ve dalga cephesi eğrileri için temel yaklaşımlar incelenmiştir. Son olarak geliştirilen hesaplama yöntemi ile EY ortamda dalga yayılımı hesaplanmasının yapıldığı iki uygulama sunulmuştur.

ABSTRACT

In this study, wave propagation has been considered in two-dimensional transversely isotropic media (TI media). Firstly, transversely isotropic media has been defined. Then equations of motion for two-dimensional case have been derived. A numerical algorithm has been developed for computation of the wave propagation. Some basic concepts about velocity and wave front curves have been reviewed. Finally two applications of the numerical computation of the wave propagation in transversely isotropic media have been presented.

GİRİŞ

Yön bağımlı (anisotropic) ortam, hızın yayılma yönüyle değişmesi olarak tanımlanır. Yön bağımlı ortamlar birkaç çeşit simetriye sahip olabilirler. Simetriklik özelliklerine göre yön bağımlı ortamların herbiri farklı dalga yayılımı karakterine sahiptirler (Fagin, 1998; Helbig, 1994). Bunlardan en yaygın olanı enine yön bağımsız ortamdır (EY ortam; transversely isotropic media; TI media). EY ortam bir simetri eksenine sahiptir. Bu simetriklik bir eksen etrafında koordinat ekseninin rotasyonu durumunda, elastik katsayıların değişmemesi anlamındadır (Helbig, 1994).

Bir EY ortamın tanımlanması için c_{11}, c_{12}, c_{13}, c_{33} ve c_{44} olmak üzere beş adet elastisite katsayısının tanımlanması gerekmektedir (Payton, 1983). Yine bu standart gösterime alternatif bir gösterim Thomsen (1986) tarafından geliştirilmiş ve "Thomsen yön bağım (anisotropi) parametreleri" olarak adlandırılmıştır. Bunlar sırasıyla düşey P ve S dalgalarının hızları α, β, SH dalga hızını belirleyen γ ve anizotropi parametreleri ε, δ (Thomsen, 1986) dır.

[1] Ankara Üniversitesi, Mühendislik Fakültesi, Jeofizik Mühendisliği Bölümü, 06100 Tandoğan / ANKARA
Fax: 0 312 215 04 87 e-mail: kadioglu@eng.ankara.edu.tr

Bu çalışmada EY ortamda dalga yayılımı irdelenmiştir. Bunun için sayısal hesaplama yöntemi geliştirilmiştir. Uygulama olarak Thomsen (1986)' dan elastisite katsayıları alınan iki örnek için hız ve dalga cephesi eğrileri ve yerdeğiştirme bileşenleri hesaplanıp bir t zamanındaki durumu görüntülenmiştir.

ENİNE YÖN BAĞIMSIZ (EY) ORTAMIN TANIMLANMASI

Tekdüze olmayan (heterogen), yön bağımlı (anisotropic) bir ortam genelleştirilmiş Hook kanunu ile tanımlanır:

$$\sigma_{ij} = c_{ijkl} e_{kl} \qquad i,j,k,l = 1,2,3. \tag{1}$$

Burada $\sigma_{ij}(\mathbf{x},t)$ ve $e_{kl}(\mathbf{x},t)$ sırasıyla gerilme (stress) ve yamulma (strain) tansörleridir, $c_{ijkl}(\mathbf{x})$ dördüncü düzen tansör bileşenleri, ortamın elastisiteleri olarak tanımlanır. $\mathbf{x}$ konum vektörünü, t ise zamanı tanımlar.

EY ortamda gerilme-yamulma ilişkisini elde etmek için burada literatürde ortak kullanılan kısaltılmış matris gösterimi kullanılmaktadır. Buna göre elastisiteyi tanımlayan alt indis çifti uygun bir düzenekle

$$\begin{aligned}&(11)\rightarrow 1,(22)\rightarrow 2,(33)\rightarrow 3,(23)=\\&(32)\rightarrow 4,(31)=(13)\rightarrow 5,(12)=(21)\rightarrow 6,\end{aligned} \tag{2}$$

şeklinde tek bir sayı ile tanımlanır. Bu gösterimi kullanarak simetri ekseni z ekseni olan EY katıda gerilme-yamulma ilişkisi

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{23}\\ \sigma_{31}\\ \sigma_{12}\end{bmatrix}=\begin{bmatrix}c_{11} & c_{12} & c_{13} & 0 & 0 & 0\\ c_{12} & c_{11} & c_{13} & 0 & 0 & 0\\ c_{13} & c_{13} & c_{33} & 0 & 0 & 0\\ 0 & 0 & 0 & c_{44} & 0 & 0\\ 0 & 0 & 0 & 0 & c_{44} & 0\\ 0 & 0 & 0 & 0 & 0 & c_{66}\end{bmatrix}\begin{bmatrix}e_{11}\\ e_{22}\\ e_{33}\\ 2e_{23}\\ 2e_{31}\\ 2e_{12}\end{bmatrix}, \tag{3}$$

olarak verilir (Thomsen, 1986; Payton, 1983). Burada $c_{66} = (c_{11} - c_{12})/2$ dir. Bu bağıntı (3) denklemindeki beş bağımsız elastisite katsayısının uzaya bağımlı olabileceğini göstermektedir.

İKİ BOYUTLU EY ORTAMDA HAREKET DENKLEMLERİ

Yatay eksen x ve düşey eksen z olarak tanımlandığında iki boyutlu gerilme-yamulma ilişkisi

$$\begin{bmatrix}\sigma_{xx}\\ \sigma_{zz}\\ \sigma_{xz}\end{bmatrix}=\begin{bmatrix}c_{11} & c_{13} & 0\\ c_{13} & c_{33} & 0\\ 0 & 0 & c_{44}\end{bmatrix}\begin{bmatrix}e_{xx}\\ e_{zz}\\ 2e_{xz}\end{bmatrix}, \tag{4}$$

olarak bulunur. Burada iki boyutlu EY ortam için bağımsız elastisite katsayı sayısının dörde indiği görülmektedir.

Dalga yayılımı, gerilme-yamulma ilişkisine bağlı olan momentum korunumuna dayanmaktadır. Sürekli bir ortamda doğrusallaştırılmış hareket denklemi

$$\rho\ddot{\mathbf{u}} = \nabla.\boldsymbol{\sigma} + \mathbf{f}, \tag{5}$$

ile tanımlanır. Burada $\ddot{\mathbf{u}}(\mathbf{x},t)$ yerdeğiştirme alanının zamana göre ikinci türevini, $\boldsymbol{\sigma}(\mathbf{x},t)$ gerilme tansörünü, $\mathbf{f}(\mathbf{x},t)$ cisim kuvvetlerini, $\rho(\mathbf{x})$ yoğunluğu ve $\mathbf{x}$ konum vektörünü tanımlamaktadır. İki boyutlu durumda (5) denklemi

$$\rho\ddot{u}_x = \frac{\partial\sigma_{xx}}{\partial x} + \frac{\partial\sigma_{xz}}{\partial z} + f_x, \tag{6.a}$$

$$\rho\ddot{u}_z = \frac{\partial\sigma_{xz}}{\partial x} + \frac{\partial\sigma_{zz}}{\partial z} + f_z, \tag{6.b}$$

olur. Burada $\ddot{u}_x$ ve $\ddot{u}_z$ yer değiştirme alanı bileşenlerinin zamana göre ikinci türevleri, f_x ve f_z cisim kuvveti bileşenleridir. Yine iki boyutlu ortamda yamulma-yer değiştirme ilişkini (Lay ve Wallace, 1995) veren

$$e_{ij} = (u_{i,j} + u_{j,i})/2, \quad (7)$$

ifadesi kullanılarak sırasıyla

$$\rho\ddot{u}_x = \frac{\partial}{\partial x}\left(c_{11}\frac{\partial u_x}{\partial x} + c_{13}\frac{\partial u_z}{\partial z}\right) + \frac{\partial}{\partial z}\left(c_{44}\left(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x}\right)\right) + f_x, \quad (8.a)$$

$$\rho\ddot{u}_z = \frac{\partial}{\partial z}\left(c_{13}\frac{\partial u_x}{\partial x} + c_{33}\frac{\partial u_z}{\partial z}\right) + \frac{\partial}{\partial x}\left(c_{44}\left(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x}\right)\right) + f_z, \quad (8.b)$$

elde edilir. (8) denklemleri yön bağımsız (isotrop) ortam için tanımlandığında $c_{11} = c_{33} = \lambda + 2\mu$, $c_{13} = \lambda$ ve $c_{44} = \mu$ olur (Thomsen, 1986). Burada λ ve μ lame sabitleridir.

Sayısal çözüm için, bu denklemler

$$\dot{U} = MU + F, \quad (9)$$

şeklinde birinci düzen denklem çiftine dönüştürülebilir. Burada

$$M = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ M_{31} & M_{32} & 0 & 0 \\ M_{41} & M_{42} & 0 & 0 \end{bmatrix}, \quad (10)$$

ile tanımlanan özel işleç (operator) matrisidir. M bileşenleri ise

$$M_{31} = \frac{1}{\rho}\left(\frac{\partial}{\partial x}c_{11}\frac{\partial}{\partial x} + \frac{\partial}{\partial z}c_{44}\frac{\partial}{\partial z}\right), \quad (11.a)$$

$$M_{32} = \frac{1}{\rho}\left(\frac{\partial}{\partial x}c_{13}\frac{\partial}{\partial z} + \frac{\partial}{\partial z}c_{44}\frac{\partial}{\partial x}\right), \quad (11.b)$$

$$M_{41} = \frac{1}{\rho}\left(\frac{\partial}{\partial x}c_{44}\frac{\partial}{\partial z} + \frac{\partial}{\partial z}c_{13}\frac{\partial}{\partial x}\right), \quad (11.c)$$

$$M_{42} = \frac{1}{\rho}\left(\frac{\partial}{\partial x}c_{44}\frac{\partial}{\partial x} + \frac{\partial}{\partial z}c_{33}\frac{\partial}{\partial z}\right), \quad (11.d)$$

dir. Yerdeğiştirme ve cisim kuvvet dizileri ise

$$U^T = [u_x, u_z, \dot{u}_x, \dot{u}_z], \quad (12)$$

$$F^T = [0, 0, f_x/\rho, f_z/\rho], \quad (13)$$

ile tanımlanabilir. Ek A' da işleç matrisinin tekil değerlerinin bulunuşu verilmektedir.

HIZ EĞRİSİ

EY ortam için normal eğri bileşenleri

$$p = R\pm(\theta)\cos\theta \text{ ve } q = R\pm(\theta)\sin\theta, \quad (14)$$

ile tanımlanmaktadır (Payton, 1983). Burada

$$R\pm(\theta) = \left(\frac{B(\theta) \pm B^2(\theta) - 4A(\theta)^{1/2}}{2A(\theta)}\right)^{1/2}, \quad (15)$$

ve

$$A(\theta) = \alpha\cos\theta + \gamma\cos^2\theta\sin^2\theta + \beta\sin^4\theta, \quad (16.a)$$

$$B(\theta) = (\alpha+1)\cos^2\theta + (\beta+1)\sin^2\theta, \quad (16.b)$$

dir (Payton, 1983). Burada θ ışının yayılma açısıdır. (16) denklemlerindeki α, β ve γ sırasıyla

$$\alpha = c_{33}/c_{44}, \quad (17.a)$$

$$\beta = c_{11}/c_{44}, \quad (17.b)$$

ve

$$\gamma = 1 + \alpha\beta - (c_{13}/c_{44} + 1)^2, \qquad (17.c)$$

dir. Hız eğrisi, normal eğrinin tersi olarak tanımlanır ve bileşenleri

$$V_x \pm = V \pm (\theta)\sin\theta, \qquad (18.a)$$

$$V_z \pm = V \pm (\theta)\cos\theta, \qquad (18.b)$$

denklemleri ile tanımlanmaktadır (Payton, 1983). Burada

$$V \pm (\theta) = \frac{V_s}{R \pm (\theta)}, \qquad (19)$$

ve

$$V_s = (c_{44}/\rho)^{1/2}, \qquad (20)$$

dir. V_s ,düşey ve yatay tam (pure) enine (transverse) dalga hızına uygun, EY ortamın referans dalga hızıdır (Postma, 1955).

Hız eğrileri (18) denklemlerinden de anlaşılacağı gibi iki bölümden oluşmaktadır ve

$$V_-(\theta) < V_+(\theta), \qquad (21)$$

dir. (Payton, 1983). Eksi işaret ile tanımlanmış hız eğrisi quasi-boyuna kipi (quasi-longitudinal mod), Artı işaretlisi ise quasi-enine kipi (quasi-transverse mod) tanımlamaktadır. Bu kiplere uygun yerdeğiştirme genlikleri her zaman ortogonaldir. Ancak yönbağımsız katıdaki gibi dalga cephesi ile eşleşmiş veya dik değildir. Sismolojide bu kipler P ve SV dalga cephelerini temsil etmektedirler. Fakat gerçekte dalgalar sadece enine yön bağımsız katının simetri ekseni $V_p = (c_{33}/\rho)^{1/2}$ ile birleştiğinde veya yayılma yönü $V_p = (c_{11}/\rho)^{1/2}$' ye dik olduğunda tam makaslama (pure shear) veya tam basınç (pure compressional) dalgalarıdır (Postma, 1955).

DALGA CEPHESİ EĞRİSİ

Dalga cephesi eğrisi, verilen bir t zamanında düzlem dalgaların zarflanması ile oluşturulur ve

$$z\pm = \frac{V \pm t\cos\theta}{2A}\left(2\alpha\cos^2\theta + \gamma\sin^2\theta \pm \frac{\sin^2\theta\left(k_1\cos^2\theta - k_2\sin^2\theta\right)}{\left(B^2 - 4A\right)^{1/2}}\right), \qquad (22.a)$$

$$x\pm = \frac{V \pm t\sin\theta}{2A}\left(2\beta\sin^2\theta + \gamma\cos^2\theta + \frac{\cos^2\theta\left(k_1\cos^2\theta - k_2\sin^2\theta\right)}{\left(B^2 - 4A\right)^{1/2}}\right), \qquad (22.b)$$

şeklinde iki bölümden oluşur (Payton, 1983). Burada

$k_1 = 2\alpha(\beta + 1) + \gamma(\alpha + 1)$ ve

$k_2 = 2\beta(\alpha + 1) - \gamma(\beta + 1)$

Enine yön bağımsız materyallerde normal ve dalga cephesi eğrileri büküm noktalarının yerine göre veya eğrilerin teğetlerinin birleştiği yere göre sınıflandırılırlar (Payton, 1983).

Hız ve dalga cepheleri arasında basit bir uyum yoktur. Burada $V\pm$ hızı, $k(\theta)$dalga cephesi eğrisine normal olan dalga sayısı vektörü boyunca dalga cephesinin ilerleme hızını tespit etmektedir. Bu eğri üzerinde verilen bir $P(x_0, z_0)$ nokta için hız eğrisini kesen ve k vektörü yönüne paralel orjinden geçen bir hat boyunca uygun $Q(v_{x_0}, v_{z_0})$ noktası bulunur (Postma, 1955).

UYGULAMALAR

Burada Thomsen (1986)' in farklı çalışmalardan alıp derlediği çeşitli yerlerdeki sedimanter kayaçlara ait ölçülmüş yön bağımlılık (anisotropic) özellikleri tablosundan seçilen Mesaverde killi şeyl ve Mesaverde kalkerli kumtaşı kullanılmıştır. Uygulamalarda bu iki materyale ait hız eğrileri, dalga cephesi eğrileri ve belli bir t zamanı için sismik dalga yayılımı hesaplanmış ve sonuçlar görüntülenmiştir.

Hesaplamalar için ortam 165x165 lik bir alanda ve grid aralığı $DX = DZ = 20m$ olarak tanımlanmıştır. Hareket z yönünde, tanımlanan alanın merkezinde, bir nokta kaynak tarafından oluşturulmuştur. Kaynak fonksiyonu

$$S(t) = e^{-0.5 f_0^2 (t-t_0)^2} \cos \pi f_0 (t - t_0), \qquad (23)$$

olarak tanımlanmıştır. Burada $f_0 = 50Hz$ ve kaynak gecikme zamanı $t_0 = 60msn$ dir.

Elastisite katsayıları; basınç ve düşey makaslama hızları sırasıyla α_0, β_0, yoğunluk ρ, yön bağımlılık değiştirgen (parameter) değerleri ε ve δ Thomsen (1986) tarafından yayınlanmış verilerden yararlanılarak hesaplanmıştır. Buna göre Mesaverde killi şeyl için $\alpha_0 = 3794 m/s$, $\beta_0 = 2074 m/s$, $\varepsilon = 0.189$, $\delta = 0.204$ ve $\rho = 2.560 g/cm^3$ dir. Yine Mesaverde kalkerli kumtaşı için $\alpha_0 = 5460 m/s$, $\beta_0 = 3219 m/s$, $\varepsilon = 0.000$, $\delta = -0.264$ ve $\rho = 2.690 g/cm^3$ dir.

Elastisite katsayıları

$$c_{33} = \alpha_0^2 \rho, \qquad (24.a)$$

$$c_{44} = \beta_0^2 \rho, \qquad (24.b)$$

$$c_{11} = c_{33}(2\varepsilon + 1), \qquad (24.c)$$

ve

$$c_{13} = \left(2\delta c_{33}(c_{33} - c_{44}) + (c_{33} - c_{44})^2\right)^{1/2} - c_{44}, \qquad (24.d)$$

bağıntıları yardımıyla hesaplanır. Mesaverde killi şeyl için

$c_{11} = 66.6GPa$, $c_{13} = 39.4GPa$, $c_{33} = 39.9GPa$ ve $c_{44} = 10.9GPa$

olarak bulunur. Mesaverde kalkerli kumtaşı için ise

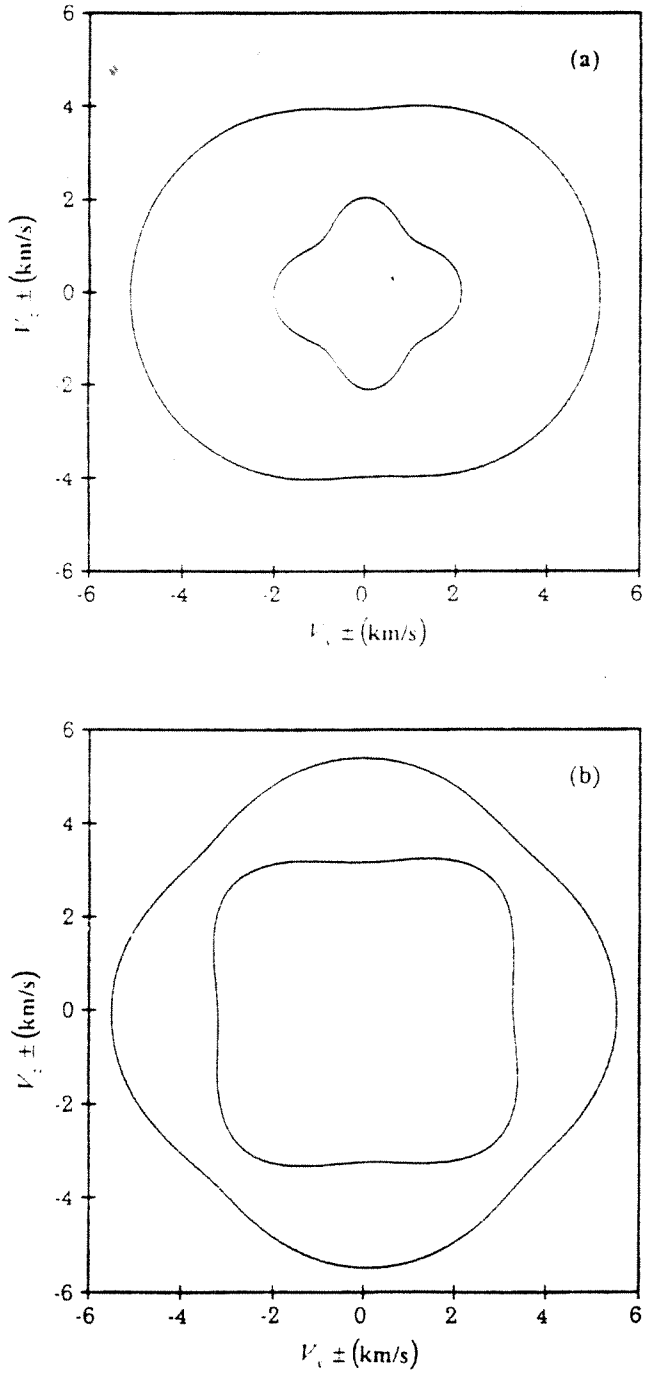
$c_{11} = 80.2GPa$, $c_{13} = -5.0GPa$, $c_{33} = 80.2GPa$ ve $c_{44} = 27.9GPa$

olarak bulunur.

Şekil 1. sırasıyla Mesaverde killi şeyl'e ve Mesaverde kalkerli kumtaşı' na ait hız eğrilerini göstermektedir.

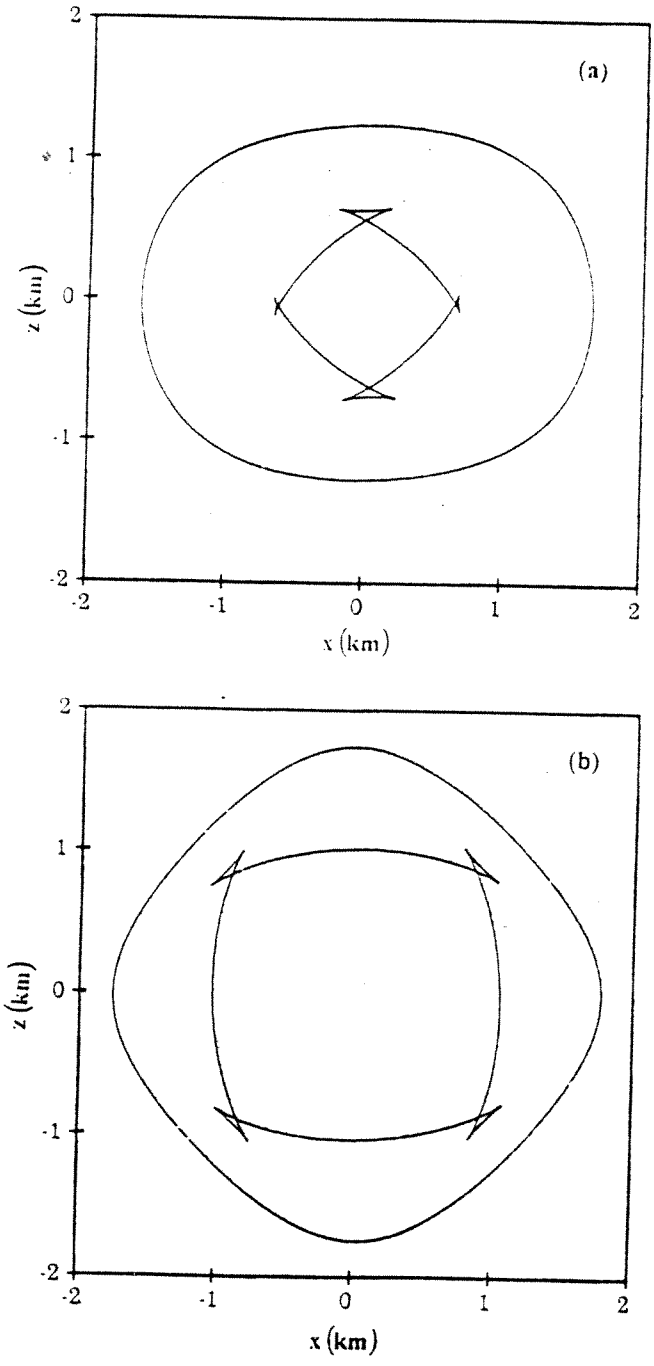
Şekil 2. sırasıyla Mesaverde killi şeyl'e ve Mesaverde kalkerli kumtaşı' na ait dalga cephesi eğrilerini göstermektedir.

Şekil 3. Mesaverde killi şeyl ve Mesaverde kalkerli kumtaşı için düşey bir kuvvetten dolayı u_x yatay yerdeğiştirme bileşeninin $t = 0.32s$'deki durumunu görüntülemektedir. Yine Şekil 4. Mesaverde killi şeyl ve Mesaverde kalkerli kumtaşı için düşey bir kuvvetten dolayı u_z düşey yerdeğiştirme bileşeninin $t = 0.32s$'deki durumunu görüntülemektedir.



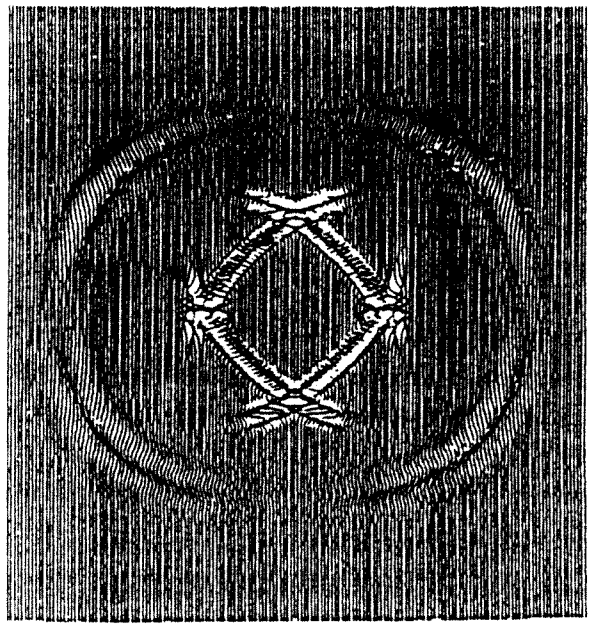
Şekil 1. Hız eğrileri ($t = 0.32s$ için). a) Mesaverde killi şeyl. b) Mesaverde kalkerli kumtaşı. İçteki eğri yaklaşık enine kipi, dıştaki eğri yaklaşık boyuna kipi göstermektedir.

Figure 1. Velocity curves (for $t = 0.32s$). a) Shale with clay in Mesaver b) Sandstone with calcerous in Mesaver. Inside of the curve displays approximate transverse modal, outside of the curve displays approximate steadily modal.

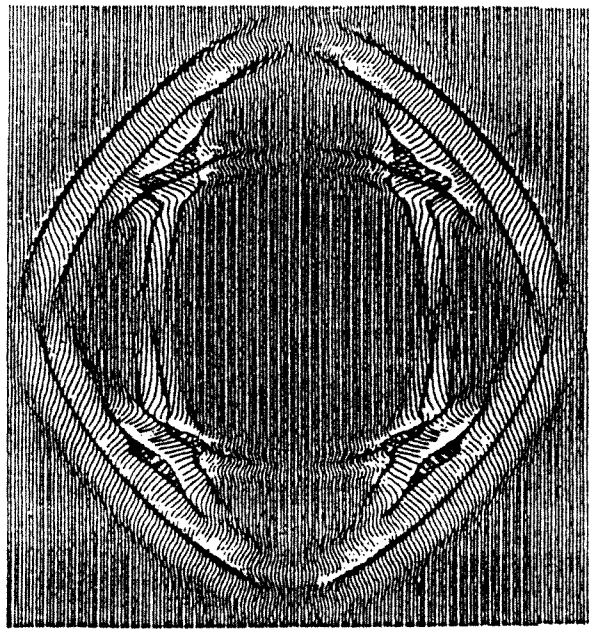


Şekil 2. Dalga cephesi eğrileri ($t = 0.32s$ için). a) Mesaverde killi şeyl. b) Mesaverde kalkerli kumtaşı. İçteki eğri yaklaşık enine kipi, dıştaki eğri yaklaşık boyuna kipi göstermektedir.

Figure 2. Wavefront curves (for $t = 0.32s$).·a) Shale with clay in Mesaver b) Sandstone with calcerous in Mesaver. Inside of the curve displays approximate transverse modal, outside of the curve displays approximate steadily modal.



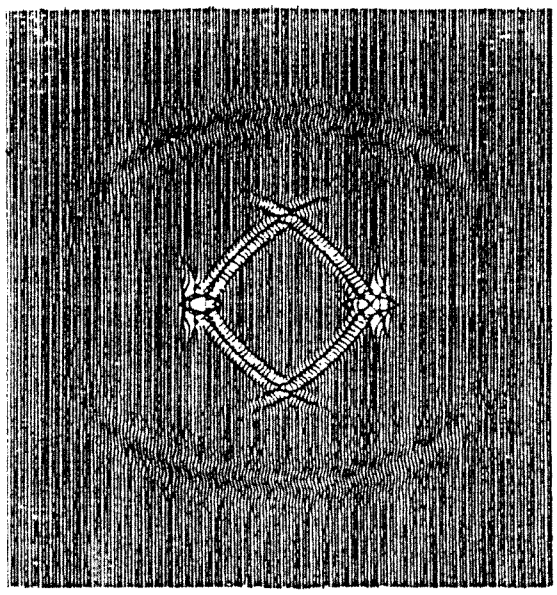
(a)



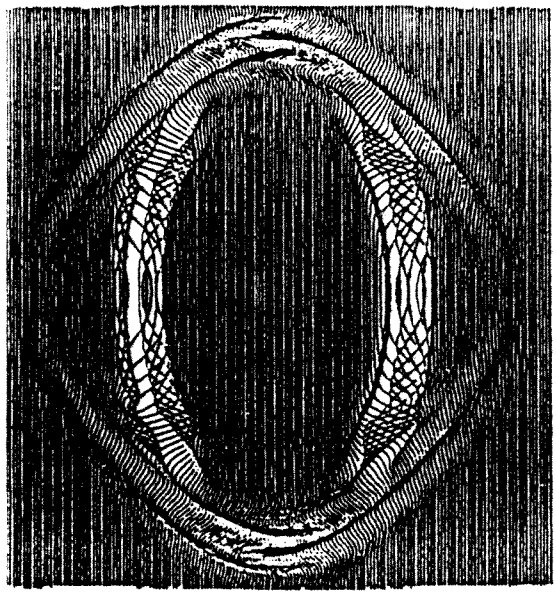
(b)

Şekil 3. Düşey kuvvetin meydana getirdiği yerdeğiştirme alanının $t = 0.32s$' de u_x yatay bileşeni a) Mesaverde killi şeyl. b) Mesaverde kalkerli kumtaşı.

Figure 3. The vertical force that composes the horizontal component of U_x in $t = 0.32s$ a) Shale with clay in Mesaver b) Sandstone with calcerous in Mesaver.



(a)



(b)

Şekil 4. Düşey kuvvetin meydana getirdiği yerdeğiştirme alanının $t = 0.32s$' de u_z düşey bileşeni a) Mesaverde killi şeyl. b) Mesaverde kalkerli kumtaşı.

Figure 4. The vertical force that composes the vertical component of u_z in t=0.32 s a) Shale with clay in Mesaver b) Sandstone with calcerous in Mesaver.

SONUÇLAR

Bu çalışmada iki boyutlu enine yön bağımsız bir ortamda dalga yayılımı hesaplamaları üzerinde duruldu. Hesaplanan dalga cepheleri göreceli dalga cepheleri tarafından üretilmiş karakteristikleri gösterdi. Burada kullanılan sayısal hesaplama analitik çözümler çok karmaşık olduğu veya bilinmediği zaman elastodinamik çözümleri kararlaştırmada çok önemli olabilir. Bu sayısal hesaplama yöntemi farklı simetriye sahip materyaller için dalga yayılımı hesaplamalarına kolaylıkla dönüştürülebilir.

TEŞEKKÜR

Çalışmanın bir kısmı Ankara Üniversitesi Yurtdışı Araştırma-Eğitim Bursu ile gerçekleştirilmiştir. Ayrıca, Türkiye Bilimsel ve Teknik Araştırma Kurumu (TÜBİTAK) tarafından YDABÇAG-100Y105 Nolu Proje kapsamında desteklenmektedir. Katkılarından dolayı Ankara Üniversitesi'ne ve TÜBİTAK'a çok teşekkür ederim.

Bilimsel yardımlarından dolayı sayın Prof. Dan KOSLOFF'a (Tel-Aviv Universitesi/İSRAİL) çok teşekkür ederim.

KAYNAKLAR

Fagin, S., 1998. Model-Based Imaging. Course Notes Series, No. 10, Society of Exploration Geophysicists, Tusla, Oklahoma USA.

Helbig, K., 1994. Foundations of Anisotropy for Exploration Seismics. Handbook of Geophysical Exploration Sections. Vol. 22.

Lay, T. and Wallace, T. C., 1995. Modern Global Seismology. International Geophysical series, 58.

Payton, R. G., 1983. Elastic Wave Propagation in Transversely Isotropic Media., Martinus Nijhoff Publishers.

Postma, G. W., 1955. Wave Propagation in Stratified Medium. Geophysics, 20, 780-806.

Thomsen, L., 1986. Weak Elastic Anisotropy, Geophysics, 51, 1954-1966.

EK A

ENİNE YÖN BAĞIMSIZ BİR KATI İÇİN İŞLEÇ MATRİSİNİN TEKİL DEĞERLERİNİN VE R 'NİN BULUNMASI

(9) denkleminin düzlem dalga çözümü

$$U = U_0 e^{i(wt - k.x)}, \tag{A.1}$$

olduğu kabul edilir (Payton, 1983). Burada w açısal frekansı, k dalga sayısını ve $\mathbf{x}$ konum vektörünü tanımlamaktadır. Sabit elastisite katsayıları ve yoğunluk değerleri olduğu kabul edildiğinde ve kaynak terimi sıfır alındığında, (A.1) ifadesi (9)'da yerine konulduğunda

$$iwU = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \tilde{M}_{31} & \tilde{M}_{32} & 0 & 0 \\ \tilde{M}_{41} & \tilde{M}_{42} & 0 & 0 \end{bmatrix} U, \tag{A.2}$$

elde edilir. Burada M bileşenleri

$$\tilde{M}_{31} = -\frac{c_{11}k_x^2 + c_{44}k_z^2}{\rho}, \tag{A.3a}$$

$$\tilde{M}_{32} = \tilde{M}_{41} = -\frac{(c_{13}+c_{44})k_x k_z}{\rho}, \qquad \text{(A.3b)}$$

$$\tilde{M}_{42} = -\frac{c_{44}k_x^2 + c_{33}k_z^2}{\rho}, \qquad \text{(A.3c)}$$

ile verilir. (A.2) denklemi $\lambda_i = iw$, $i = 1,2,3,4$. tekildeğerleri için tekil değer denklemini düzenler. Buna göre (A.2) denkleminin karakteristiği

$$\lambda^4 - \left(\tilde{M}_{31} + \tilde{M}_{42}\right)\lambda_i^2 + \left(\tilde{M}_{31}\tilde{M}_{42} - \tilde{M}_{41}\tilde{M}_{32}\right) = 0, \qquad \text{(A.4)}$$

ile verilir.

EY bir katı için $c_{11}, c_{33}, c_{44} > 0$ ve $c_{13} + c_{44} < \left((c_{11}c_{33})^{1/2} + c_{44}\right)$ dir (Payton, 1983). Bu $B \equiv -\left(\tilde{M}_{31} + \tilde{M}_{42}\right) > 0$.

$C \equiv \left(\tilde{M}_{31}\tilde{M}_{42} - \tilde{M}_{41}\tilde{M}_{32}\right) > 0$ ve $B^2 - 4C > 0$ olduğunu göstermektedir. Bu şartlarda (A.4) denkleminin kökleri, yani tekil değerleri

$$\lambda_1 = \frac{i}{\sqrt{2}}\left[\left|\tilde{M}_{31} + \tilde{M}_{42}\right| + \left[\left(\tilde{M}_{31} - \tilde{M}_{42}\right)^2 + 4\tilde{M}_{41}\tilde{M}_{32}\right]^{1/2}\right]^{1/2}, \qquad \text{(A.5a)}$$

$$\lambda_2 = -\lambda_1, \qquad \text{(A.5b)}$$

$$\lambda_3 = \frac{i}{\sqrt{2}}\left[\left|\tilde{M}_{31} + \tilde{M}_{42}\right| - \left[\left(\tilde{M}_{31} - \tilde{M}_{42}\right)^2 + 4\tilde{M}_{41}\tilde{M}_{32}\right]^{1/2}\right]^{1/2}, \qquad \text{(A.5c)}$$

$$\lambda_4 = -\lambda_3, \qquad \text{(A.5d)}$$

olarak bulunur. Yön bağımsız bir ortam için (A.3) denklemleri

$$\tilde{M}_{31} = -\frac{(\lambda+2\mu)k_x^2 + \mu k_z^2}{\rho}, \qquad \text{(A.6a)}$$

$$\tilde{M}_{32} = \tilde{M}_{41} = -\frac{(\lambda+\mu)k_x k_z}{\rho}, \qquad \text{(A.6b)}$$

$$\tilde{M}_{42} = -\frac{\mu k_x^2 + (\lambda+2\mu)k_z^2}{\rho}, \qquad \text{(A.6c)}$$

olur. Bu denklemlerin (A.5) denklemlerinde yerine konulması ile

$$\lambda_1' = i\left(\frac{\lambda+2\mu}{\rho}\right)\left(k_x^2 + k_z^2\right)^{1/2}, \quad \lambda_2' = -\lambda_1', \qquad \text{(A.7a)}$$

$$\lambda_3' = i\left(\frac{\mu}{\rho}\right)\left(k_x^2 + k_z^2\right)^{1/2}, \quad \lambda_4' = -\lambda_3', \qquad \text{(A.7b)}$$

elde edilir. Burada birinci ve ikinci tekil değer değerleri boyuna (longitudinal) dalga yayılımına, üçüncü ve dördüncü tekil değer değerleri makaslama (shear) dalga yayılımına uygundur. Bu değerler dağınım (dispertion) ilişkisini tanımlamaktadırlar. Buna göre

$$w = v_p\left(k_x^2 + k_z^2\right)^{1/2}, \qquad \text{(A.8a)}$$

ve

$$w = v_s\left(k_x^2 + k_z^2\right)^{1/2}, \qquad \text{(A.8b)}$$

dir. Boyuna ve makaslama dalgaları için sırasıyla $v_s = \sqrt{\mu/\rho}$ ve $v_s = \sqrt{\mu/\rho}$ dur.

Benzer şekilde EY katı için $\lambda_1 = iw$ ve $\lambda_3 = iw$ dağınım ilişkisini tanımlamaktadır. Dalga sayısı ifadesinin değiştirgen hali

$$k_x = k(\theta)\sin\theta, \qquad k_z = k(\theta)\cos\theta, \tag{A.9}$$

dir. Dağınım ilişkisi

$$k_{\pm}(\theta) = w/V_{\pm}(\theta), \tag{A.10}$$

bağıntısını sağlar. Burada $V_{\pm}(\theta)$, (19) ile verilen faz hızıdır. Dağınım ilişkisinin sayısal elde edilmesi $|\lambda_1(\theta)| = V_-(\theta)$ yaklaşık-boyuna kip (quasi-longitudinal mod) ve $|\lambda_3(\theta)| = V_+(\theta)$ yaklaşık enine kip (quasi-transverse mod) ilişkisini ortaya çıkarır.

Yayılma matrisi R' nin teklil değerlerinin aralığı

$$k_x^N = \pi/DX, \qquad k_z^N = \pi/DZ, \tag{A.11}$$

değerlerinin (A.5a) ve (A.5b)'de yerine konulması ile bulunur. Nyquist dalga sayısı en yüksek tekil değerleri verir. Yön bağımsız katı için $v_p > v_s$ kullanılarak

$$R' = \pi v_p \left(\frac{1}{DX^2} + \frac{1}{DZ^2} \right)^{1/2} \tag{A.12}$$

bulunur. Enine yön bağımsız katı için

$$R = |\lambda_1(k_x^N, k_z^N)|$$

olarak bulunur.

SEMBOLLER DİZİNİ

α	basınç dalgası hızı
β	düşey makaslama dalgası hızı
c_{ijkl}	ortamın elastisiteleri veya elastisite katsayıları
DX	yatay örnekleme aralığı
DZ	düşey örnekleme aralığı
θ	ışının yayılma açısı
e_{ij}	yamulma tansörü
ε	yön bağımlılık değiştirgeni
f	cisim kuvvetleri
F	cisim kuvvetleri dizisi
γ	yön bağımlılık değiştirgeni
i, j, k, l	dördüncü düzen tansör bileşenleri
k	dalga sayısı
k_x^N	Nyquist dalga sayısının x bileşeni
k_z^N	Nyquist dalga sayısının z bileşeni
λ ve μ	Lame sabitleri
λ_i	operatör matrisinin tekil değerleri
M	işleç (operator) matrisi
R	yayılma matrisi
ρ	yoğunluk
S	kaynak fonksiyonu
δ	yön bağımlılık değiştirgeni
t	zaman
$u(\mathbf{x}, t)$	yer değiştirme alanı
U	yerdeğiştirme dizisi
$V_-(\theta)$	quasi-boyuna dalga hızı
$V_+(\theta)$	quasi-enine dalga hızı
V_s	kaynak dalga hızı
v_p	boyuna (longitudinal) dalga hızı
v_s	makaslama (shear) dalga hızı
w	açısal frekans
$\mathbf{x}$	konum vektörü

JEOFİZİK15, 95-104, 2001

BEŞİNCİ DERECE CASH-KARP RUNGE-KUTTA YÖNTEMİ İLE DALGA CEPHESİ OLUŞTURARAK SEYAHAT ZAMANI HESAPLANMASI

Travel Time Computation Using Wavefront Construction Via Fifth Order Cash-Karp Runge-Kutta Method

Selma KADIOĞLU[1], Selda GÜRPINAR[2] ve Orhan GÜRELİ[2]

ÖZET

Homojen olmayan, izotrop bir ortamda ilerleyen ışınların herhangi bir noktaya varış zamanları, dalga cephelerinin oluşturulması yöntemi ile bulunabilmektedir. Dalga cephesi oluşturma yöntemi ile ışınların ilerleme zamanlarının başka bir deyişle seyahat zamanlarının hesaplanması eikonal denklem karakteristiklerinin çözümüne dayanmaktadır. Literatürde bu karakteristiklerin çözümü için Taylor açılımı, dördüncü derece Runge-Kutta ve sonlu farklar yöntemleri kullanılmıştır. Burada kullanılan beşinci derece (Cash-Karp) Runge-Kutta yöntemidir. Bu yöntem ile her işlem adımındaki hata kontrolü daha doğru bir şekilde yapılabilmektedir.

İlk dalga cephesi değerleri $t = 0$ anında kaynak noktasındaki yeterli sayıda ışın değerleri ile tanımlanmaktadır. Bu değerler ışınların vektör

ABSTRACT

Travel times of the rays in a heterogen, isotrop medium can be determined by wave front construction method. This method is based on the solution of eikonal equation characteristics of the ray. Fouth order Taylor expansion, Runge-Kutta and finite difference methods have been used for the solution of these characteristics in literarture. The method employed here is fifth order Cash-Karp Runge-Kutta method. Control of the error at every step can be made more accurately with this method.

Initial values of the wave front are defined by a source point with a sufficient number of the rays at $t = 0$. These values are components of position vector and components of ray parameter of the rays. The new positions of the wave front are determined by calculating the eikonal equations at every Δt time increment.

Ankara Üniversitesi, Mühendislik Fakültesi, Jeofizik Mühendisliği Bölümü, Tandoğan, 06100 ANKARA
TPAO, Mustafa Kemal Mah. 2. Cad. No:86 06520, ANKARA
Fax: 0 312 215 04 87, e-mail: kadioglu@eng.ankara.edu.tr

bileşenleri ile ışın parametresi bileşenleridir. Daha sonra her zaman artımında dalga cephesinin yeni konumu eikonal denklemlerin çözümü ile bulunur.

Uygulamalarda hızı yatay ve düşey eksende lineer olarak artan modele, tabakalı ve faylı modellere ait dalga cepheleri hesaplanmıştır. Buna göre; hız modeline ait türev değerleri sürekli olmalıdır. Modeldeki hız değişimleri yumuşak olmalıdır. Ayrıca daha duyarlı çözümler için küçük uzay ve zaman adımları önerilir.

In applications, the wave fronts belonging to the model where velocity increases linearly in the horizontal and vertical axes, layered and failed models were computed. Applications showed that the derivatives of the velocity model have to be continuous and the model has to be smooth in order to obtain successful results. In addition, it is suggested small steps be used both in time and space domains for more sensitive computations.

GİRİŞ

Sismik dalgaların ilerleme zamanının başka bir deyişle seyahat zamanının bulunması göç (migrasyon), tomografi ve modellemede gerekli olan işlemlerden biridir. Geleneksel olarak ilerleme zamanının bulunması için atış ve alıcı noktaları olmak üzere iki noktayı birbirine bağlayan ışın bulunmaya çalışılır. Işın izleme yöntemleri olarak paraxial uzanım (Cerveny 1985), sonlu farklar yöntemi (Reshef ve Kosloff 1986, Vidale 1988, Podvin ve Lecomte 1991) ve dalga cephesi yöntemi (Vinje ve diğ., 1993) kullanılabilir.

Bu çalışma Vinje ve diğ. (1993) tarafından eikonal ışın denklemlerinin Taylor açılımı kullanılarak çözümü ile dalga cephesi oluşturma yöntemine paralel bir çalışmadır. Vinje eikonal denklem çözümünü Taylor açılımı ile yapmıştır. Burada ise hata kontrolünün daha duyarlı olarak yapılabildiği, gizli Runge-Kutta yöntemi olan, beşinci derece (Cash-Karp) Runge-Kutta yöntemi (Cash and Karp 1990) kullanılmaktadır. Yöntem ile yanal ve düşey hız değişimin olduğu bir ortamda bir kaynaktan çıkan ışınların ilerleme zamanları, yeni konumları ve ışın parametreleri bulunmaktadır. Ayrıca bu yöntem ile istenirse genlik değişimi de hesaplanabilir.

Geleneksel ışın izleme geniş geometrik yayılımlı bölgelerde varış zamanını doğru bulamayabilir. Yüksek frekans ışın yaklaşımı heryerde geçerli olsa bile kaynaktan yayılan ışınların tam modeli örtmesi zor olabilir. Dalga cephesi yöntemi bu sorunu çözer.

Aşağıdaki bölümlerde izotropik, homojen olmayan ortamlarda eikonal denklemlerin tanımı, dalga cephesi oluşturma ve beşinci derece (Cash-Karp) Runge-Kutta yöntemi ile dalga cephesi interpolasyon tekniği tanımlanmaktadır. Son olarak da bir kaç modele ait uygulamalar sunulmaktadır.

EIKONAL DENKLEM KARAKTERİSTİKLERİ

Işın izleme yöntemi bir çok bilim adamı tarafından irdelenmiş bir yöntemdir (Cerveny ve diğ., 1977; Cerveny ve Hron, 1980; Ergintav ve Canıtez, 1992). İki boyutlu izotropik ortamda eikonal denklem karakteristikleri:

$$\frac{dx}{d\sigma} = P_x$$

$$\frac{dz}{d\sigma} = P_z$$

$$\frac{dP_x}{d\sigma} = \frac{1}{2}\frac{\partial}{\partial x}\left(\frac{1}{V^2}\right) \qquad (1)$$

$$\frac{dP_z}{d\sigma} = \frac{1}{2}\frac{\partial}{\partial z}\left(\frac{1}{V^2}\right)$$

ile tanımlanır (Cerveny ve diğ., 1977). Işın yolu bu denklem sisteminin çözümü ile bulunmaktadır. Burada σ ışın yolu boyunca düzenli olarak artan bir parametredir. $V(x,z)$ izotropik ortamın hızını, x ve z kartezyen koordinat sisteminde ışının pozisyon koordinatlarını tanımlar. $P_x = \cos\theta/V$ ışın parametresinin yatay bileşeni $P_z = \sin\theta/V$ ise düşey bileşenidir. Ayrıca θ ışının yatay eksene göre çıkış açısıdır. Eikonal denklemler zaman değişkeni τ' ya göre

$$\frac{dx}{d\tau} = V^2 P_x$$

$$\frac{dz}{d\tau} = V^2 P_z$$

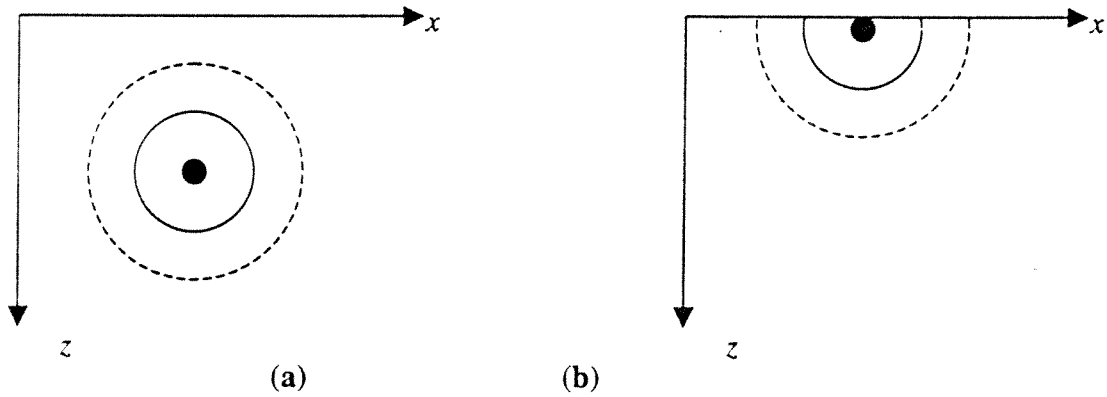
$$\frac{dP_x}{d\tau} = -\frac{1}{V}\frac{\partial V}{\partial x} \qquad (2)$$

$$\frac{dP_z}{d\tau} = -\frac{1}{V}\frac{\partial V}{\partial z}$$

olarak verilir. Bu denklem sisteminin çözümü için değişik yaklaşımlar vardır. Vinje ve diğ. (1993) çözüm için Taylor açılımını kullanmışlar ve ilk terimi çözüm olarak almışlardır. Bunun dışında dördüncü derece Runge-Kutta yöntemi kullanılmıştır (Reshef ve Kosloff, 1986). Bu çalışmada ise beşinci derece Runge-Kutta yöntemi kullanılmıştır. Beşinci derece Runge-Kutta yönteminin tercih edilmesi nedeni gerek dördüncü derece Runge-Kutta yöntemine göre gerekse diğer yöntemlere göre daha sağlıklı hata kontrolünün yapılabilmesidir. Yöntem sadece ilk varış zamanlarının hesaplanmasını kapsamaktadır.

DALGA CEPHESİNİN İLK DEĞERLERİNİN TANIMLANMASI

Dalga cephesi hesaplama işlemine istenilen bir noktadan başlanabilir. Daha sonra bu noktadan yayılması istenen ışın sayısı belirlenmelidir. Işın sayısı için başlangıç noktasının konumu dikkate alınmalıdır. Mesela Şekil 1.b' deki gibi başlangıç noktası yüzeyde ise ışın sayısı Şekil 1.a' daki gibi herhangi bir başlangıç noktasına göre belirlenen ışın sayısının yarısı olarak alınabilir. Bu noktaya göre ışınların konumunu belirleyen ilk x, z, P_x ve P_z değerleri tanımlanır.



Şekil 1. Dalga cephesi hesabı için başlangıç noktası seçimi. Başlama noktası amaca uygun olarak seçilir.

Figure 1. Selection of orijin point for wavefront construction. Orijin point is selected in accordance with the aim.

EIKONAL DENKLEM KARAKTERİSTİKLERİNİN RUNGE - KUTTA YÖNTEMİNE ADAPTE EDİLMESİ

Genel olarak Runge-Kutta yöntemi ve çeşitli versiyonları Ek A' da ayrıntılı olarak tanımlanmaktadır. Burada beşinci derece Cash-Karp Runge-Kutta yöntemini tanımlayan (A-4), (A-5) ve (A-6) denklemlerinden yararlanarak dalga cepheleri hesaplanır. Çözüm için kullanılan y fonksiyonu

$$\underline{y} = \begin{pmatrix} x \\ z \\ P_x \\ P_z \end{pmatrix} \tag{3}$$

şeklinde bir vektör fonksiyon olarak tanımlanır. Ikinci olarak gerekli olan bu fonksiyonun türevini tanımlayan f fonksiyonu da (2) ve (A-1) ifadelerinden yararlanılarak

$$\underline{f}(y,t) = \frac{d\underline{y}}{dt} = \begin{pmatrix} V^2 P_x \\ V^2 P_z \\ -\frac{1}{V}\frac{\partial V}{\partial x} \\ -\frac{1}{V}\frac{\partial V}{\partial z} \end{pmatrix} \tag{4}$$

olarak bulunur. (4) bağıntısına göre hız modelinin x ve z'e göre türevlerinin sürekli olması gerekmektedir. Diğer bir önemli konu da adım ölçüsünün belirlenmesidir. Adım ölçüsü kriteri için (A-7) bağıntısı kullanılır. Her adım sonrasında bulunan yeni x ve z değerleri çoğu zaman tam grid noktaları üzerine denk gelmezler. Bu durumda (4) ifadesindeki hız ve hızın türevlerini alırken x ve z'in konumuna göre bir ağırlıklandırma fonksiyonu kullanılmalıdır. Bu çalışmada kullanılan fonksiyon dört elemanlı bir dizidir ve

$$w = \begin{bmatrix} (1-delx)(1-delz) \\ delx(1-delx) \\ delz(1-delz) \\ delxdelz \end{bmatrix} \tag{5}$$

olarak tanımlanır (Kosloff ve diğ., 1996). Burada bulunan x ve z değerlerine en yakın (i,j) grid noktası belirlendikten sonra aralarındaki uzaklık farkının grid aralıklarına oranı

$$delx = (x - idx)/dx \tag{6}$$

$$delz = (x - jdz)/dz \tag{7}$$

ile tanımlanır. Buna göre

$$V(x,z) = w(1)*V(i,j) + w(2)*V(i+1,j) + w(3)*V(i,j+1) + w(4)*V(i+1,j+1) \tag{8}$$

olarak bulunur. Benzer ağırlıklandırma türev değerleri için de yapılır.

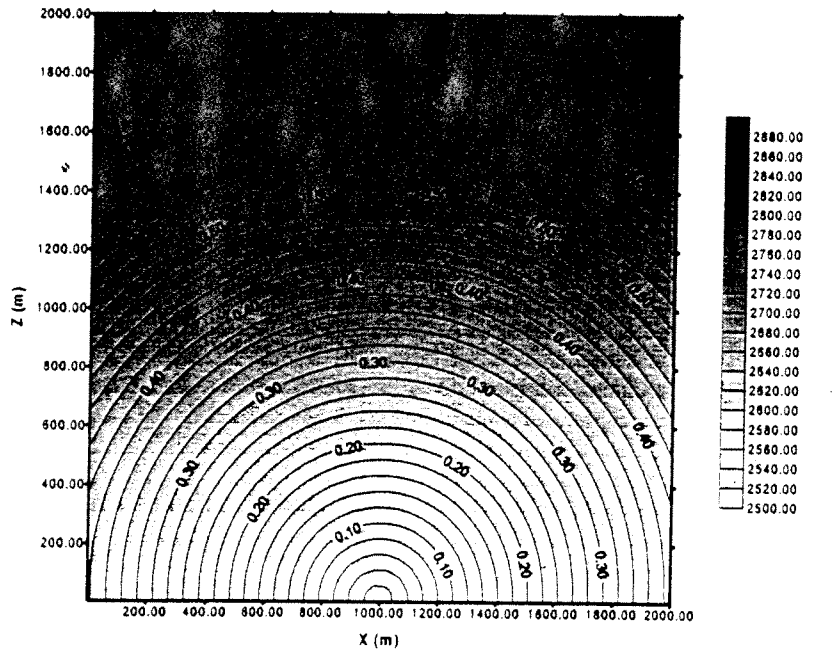
UYGULAMALAR

Model 1

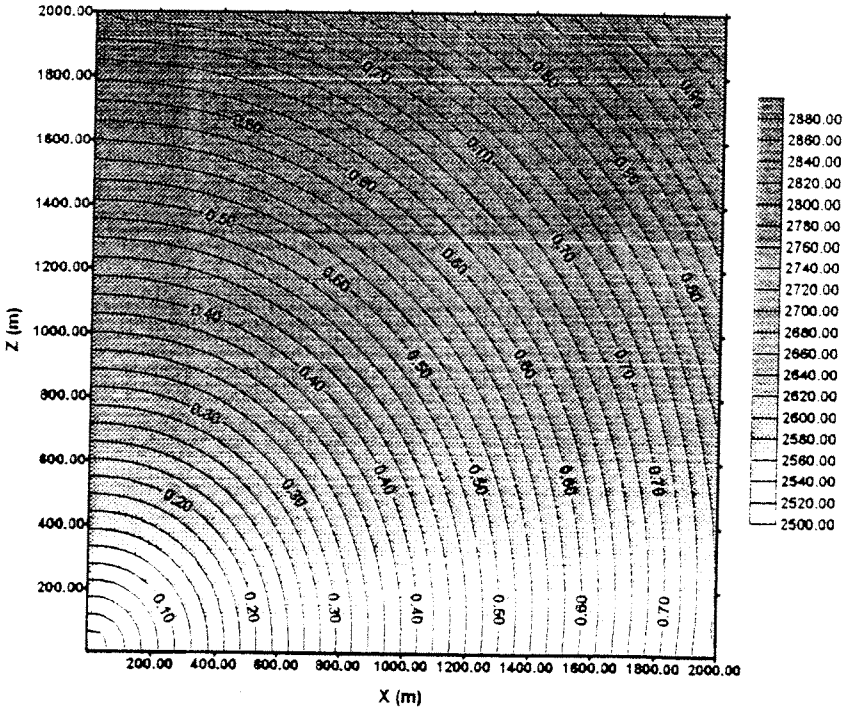
İlk uygulama olarak, yöntemin doğruluğunu test amacıyla, sabit yatay ve düşey değişime sahip

$$V(x,z) = 2500 + 0.0001x + 0.2z$$

ile tanımlanan bir hız modeli kullanıldı. Model $400*400$ grid noktası ile tanımlandı ve grid aralığı yatay ve düşey eksen için aynı olup 5m olarak alındı. Zaman hesaplama adımı $\Delta t = 0.01$ sn olarak alındı. Bu model için iki farklı kaynak noktası kullanılarak dalga cepheleri hesaplandı. İlkinde (Şekil 2.a) kaynak noktası olarak (200,4) noktası ikincisinde (Şekil 2.b) ise (4,4) noktası seçildi. Amaç hesaplama doğruluğunu farklı kaynak noktaları için göstermek idi.



(a)



(b)

Şekil 2. 0.0001 yatay ve 0.2 düşey hız değişimine sahip tek tabaka hız modeli ve modele ait dalga cephesi değişim grafiği. a) Kaynak (200,4) grid noktasında iken b) Kaynak (4,4) grid noktasında iken.

Figure 2. 0.0001 horizontal and 0.01 vertical variation of the velocity is possessed of velocity model of one layer and wavefront graph belonging to the model variation. b) Source (4,4) is on the grid point.

Model 2

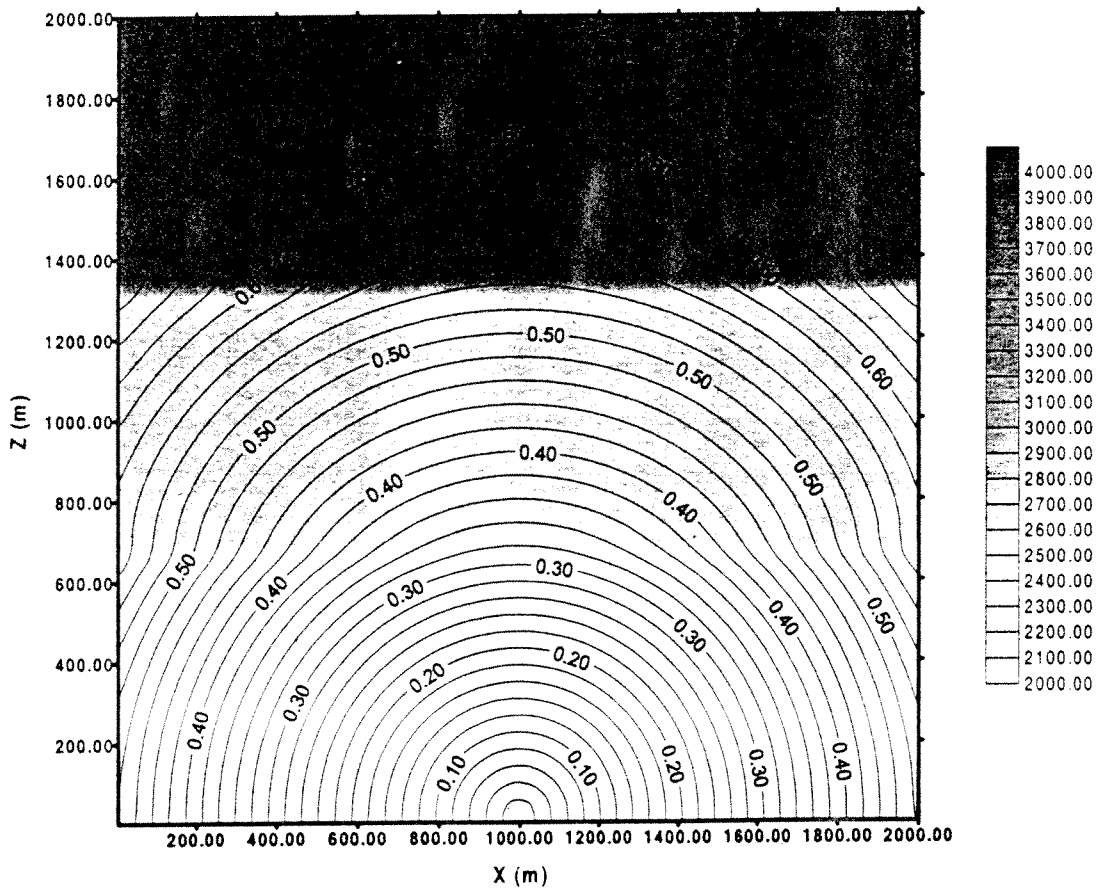
İkinci model yatay üç tabaka modelidir. Tabaka hızları sırasıyla 2000, 2500ve 4000 m/sn dir. Bu modelde de her tabaka içinde hız sabit olmayıp yatay yönde 0.0001 ve düşey yönde 0.01 değişime sahiptir. Yani

$$V(x,z) = V_{tab} + 0.0001x + 0.2z$$

ile tanımlıdır. V_{tab}, her tabaka için hızı tanımlamaktadır. Modeli tanımlayan alan, grid aralıkları ve zaman adımı model 1'deki ile aynıdır. Şekil 3.'de hız modeli ve modele ait dalga cephesi eğrilerinin değişimi verilmiştir.

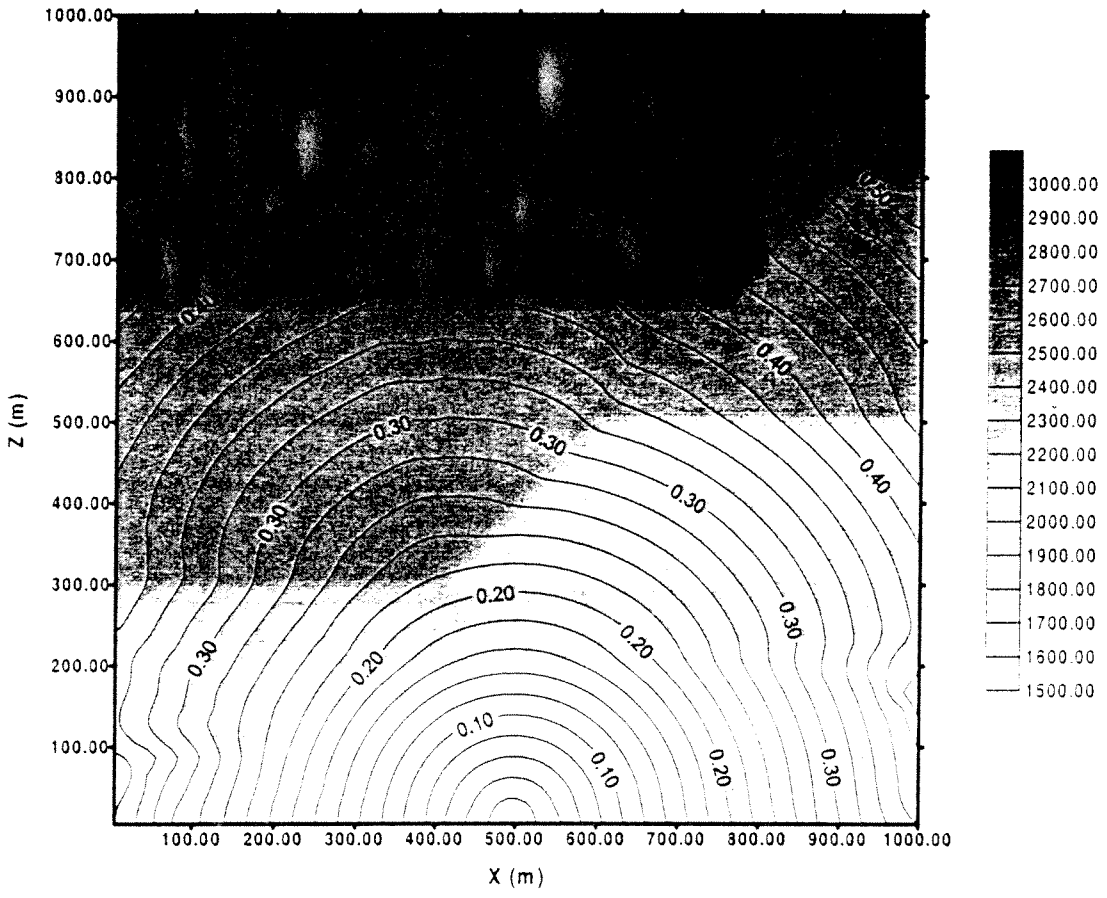
Model 3

Üçüncü model dört tabakalı yanal atımlı fay modelidir. Tabaka hızları sırasıyla 1500, 1900, 2400 ve 3000 m/sn dir. Her tabaka için yanal hız değişimi 0.000001 ve düşey hız değişimi 0.01 dir. Modeli tanımlayan alan, grid aralıkları ve zaman adımı model 1ve model 2'deki ile aynıdır. Şekil 4.'de hız modeli ve modele ait dalga cephesi eğrilerinin değişimi verilmiştir.



Şekil 3. 0.0001 yatay ve 0.01 düşey hız değişimine sahip yatay üç tabakalı hız modeli ve modele ait dalga cephesi değişim grafiği.

Figure 3. 0.0001 horizontal and 0.01 vertical variation of the velocity is possessed of velocity model of three layer and wavefront graph belonging to the model variation.



Şekil 4. 0.000001 yatay ve 0.01 düşey hız değişimine sahip dört tabakalı yanal atımlı fay modeli ve modele ait dalga cephesi değişim grafiği.

Figure 4. 0.000001 horizontal and 0.01 vertical variation of the velocity is possessed of velocity model of four layer strike-slip fault model and wavefront graph belonging to the model variation.

SONUÇLAR

Dalga cephesi oluşturarak çıkış açısı bilinen bir ışının hız modeline göre ilerleme yolu, seyahat zamanı ve istenirse genlik bilgisi hesaplanabilir. Bu değerler özellikle sismik yığma öncesi migrasyon, tomografi ve modellemede kullanılmaktadır.

Dalga cephesi yöntemi özellikle komplex ve düşük hız kontrastlı modeller için idealdir. Yöntemin uygulanmasında kinematik ışın izleme için modele ait değerlerin birinci türevlerinin sürekli olması gerekir. Ayrıca dinamik ışın izleme için ikinci türevlerin de sürekli olması gerekmektedir. Yöntemin uygulanabilirliğinde uzay ve zaman sınırı yoktur. İstenildiği kadar geniş bir alan için hesaplama yapılabilir. Yöntem diğer yöntemlere göre oldukça güvenilir sonuçlar vermektedir. Yöntemin duyarlılığını bozmamak için küçük zaman ve uzay adımları kullanılması önerilir. Yöntemlerin karşılaştırılması Leidenfrost ve diğ. (1999) tarafından yapılmış olduğundan burada yeniden ele alınmamıştır.

TEŞEKKÜR

Çalışma Türkiye Bilimsel ve Teknik Araştırma Kurumu (TÜBİTAK) tarafından YDABÇAG-100Y105 Nolu Proje kapsamında desteklenmektedir. Katkılarından dolayı TÜBİTAK'a çok teşekkür ederiz.

KAYNAKLAR

Cash, J. R., and Karp, A. H., 1990, ACM Transactions on Mathematical Software, vol.16, 201-222.,

Cerveny, V., 1985. The Application of Ray Tracing to the Numerical Modeling of Seismic Wavefields in Complex Structures. Hanbook of Geophys. Expl, 15A, 1-24, Geophysical Press.

Cerveny, V., and Hron, F., 1980. The Ray Series Method and Dynamic Ray-Tracing System for Three-Dimensional Inhomogeneous Media. Bull. Seis. Soc. Am., 70, 47-77.

Cerveny, V., Molotkov, I. A., and Psencik, I., 1977. Ray Method in Seismology. Univerzita Karlova, Praha.

Ergintav, S., Canıtez, N., 1992. Yapay Sismogram Üretimi.Jeofizikte Modelleme Kollokyumu.TMMOB Jeofizik Mühendisleri Odası İstanbul Şubesi,127-156.

Kosloff, D., Sherwood, J., Koren, Z., Machet, E., falkovitz, Y., 1996. Velocity and Interface Depth Determination by Tomography of Depth Migrated Gathers. Geophysics, 61, 1511-1523.

Leidenfrost, A., Ettrich, N., Gajewski, D., and Kosloff, D., 1999. Comparison of Six Different Methods for Calculating Traveltimes. Geophysical Prospecting, 47, 269-297.

Podvin, P., and Lacomte, I., 1991. Finite Difference Computation of Traveltimes in Very Contrusted Velocity Models: A Massively Parallel Approach and its Associated Tools. Geophys. J. Int. 105, 271-284.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery B. P., 1992. Numerical Recipes in Fortran. Second Edition. Cambridge University Press.

Reshef, M., and Kosloff D. 1986. Migration of Common Shot Gathers. Geophysics, 51, 324,331.

Vidale, J., 1988. Finite Difference Calculation of Travel Times. Bull. Seis. Soc. Am., 78, 2062-2076.

Vinje, V., Iversen, E., and Gjøystdal, H., 1993. Traveltime and Amplitude Estimation Using Wavefront Construction. Geophysics, 58, 1157-1166.

EK A

RUNGE-KUTTA YÖNTEMİ

Runge-Kutta yöntemi bir veya birden fazla bağımsız değişkene bağlı bir fonksiyonun bir adım sonraki değerinin hesaplanmasını gerçekleştiren bir yöntemdir. Klasik Runge-Kutta yönteminde t bağımsız değişkenine bağlı bir y fonksiyonunun n. değerinden h gibi bir adım sonraki yeni y_{n+1}. değerini bulma işlemi için öncelikle y 'nin türevi

$$f(t_n, y_n) = \frac{dy_n}{dt} \qquad \text{(A-1)}$$

ile tanımlanır. En basit ikinci derece Runge-Kutta'ya göre h adım sonrası fonksiyonun değeri

$$k_1 = hf(t_n, y_n)$$

$$k_2 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \qquad \text{(A-2)}$$

$$y_{n+1} = y_n + k_2 + O(h^3)$$

ile verilir. Buna orta nokta yöntemi adı da verilir (Press ve diğ., 1992). Burada $O(h^3)$ hata terimi olarak tanımlanır. Daha sık kullanılan dördüncü derece Runge-Kutta formülüzasyonu ise

$$k_1 = hf(t_n, y_n)$$

$$k_2 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$$

$$k_3 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \qquad \text{(A-3)}$$

$$k_4 = hf(t_n + h, y_n + k_3)$$

$$y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} O(h^5)$$

ile verilir (Press ve diğ., 1992). Bir değerden yararlanarak bir sonraki değeri bulmak için kullanılan adım sayısı derece sayısını belirler. Derece sayısı arttıkça hata terimini tanımlayan katsayıda değişir. Derece sayısının artması her zaman daha doğru sonuç anlamında değildir. Bu sorunu kaldırmak için ayarlanabilir adım ölçüsü kontrollü Runge-Kutta yöntemi geliştirilmiştir. Uygulanabilmesi için adımlama algoritması gerekmektedir. Bunun için de önemli olan kesme hatası hesaplanmasıdır. Kesme hatası hesaplaması için yöntem her adımda iki kere uygulanır. İlkinde tam adım, ikincisinde yarım adım kullanılır. Daha sonra her iki sonuç arasındaki fark alınarak, bu farka göre istenen doğruluk derecesi belirlenebilir.

Adım ölçüsü kontrollü yönteme alternatif olarak gizli Runge-Kutta yöntemi geliştirilmiştir. Buna göre altı fonksiyonlu beşinci derece yöntemin yine farklı altı fonksiyonlu dördüncü derece yöntemi verdiği bulunmuştur. Her iki derece sonucunda bulunan değerlerin farkları adım ölçüsünü belirlemek için kullanılır. Bu mantıkla birden fazla gizli Runge-Kutta yöntemi bulunmuştur. Bunlardan biri de beşinci derece Cash-Karp Runge-Kutta yöntemidir. Bu çalışmada tam beşinci derece Runge-Kutta ve Gizli dördüncü derece Runge-Kutta yöntemleri kullanılmıştır. Tam beşinci derece Runge-Kutta formülüzasyonu

$$k_1 = hf(t_n, y_n)$$

$$k_2 = hf(t_n + a_2 h, y_n + b_{21} k_1)$$

$$k_3 = hf(t_n + a_3 h, y_n + b_{31} k_1 + b_{32} k_2)$$

$$\vdots \qquad \text{(A-4)}$$

$$k_3 = hf\left(t_n + a_6 h, y_n + b_{61} k_1 + b_{62} k_2 + b_{63} k_3 + \cdots + b_{65} k_5\right)$$

$$y_{n+1} = y_n + c_1 k_1 + c_2 k_2 + c_3 k_3 + c_4 k_4 + c_5 k_5 + c_6 k_6 + O(h^6)$$

ile verilir (Press ve diğ., 1992). Gizli dördüncü derece Runge-Kutta formülüzasyonu ise

$$y^*_{n+1} = y_n + c^*_1 k_1 + c^*_2 k_2 + c^*_3 k_3 + c^*_4 k_4 + c^*_5 k_5 + c^*_6 k_6 + O(h^5) \qquad \text{(A-5)}$$

dir. Buna göre hata tahmini

$$\Delta = y_{n+1} - y^*_{n+1} = \sum_{i=1}^{6} (c_i - c^*_i) k_i \qquad \text{(A-6)}$$

şeklinde bulunur. Press ve diğ. (1992), Cash ve Karp (1990) tarafından verilen ve Felberg'in orjinal değerlerinden daha etkin olan sabitlerin değerlerini tablo halinde vermişler ve yöntem ile ilgili 'rkck' adlı bir fortran alt programı sunmuşlardır. Sonuç olarak bu yöntem için h_1 adımı kullanıldığında üretilen hata Δ_1 ise h_0 adımı kullanıldığında üretilen Δ_0 hatası

$$h_0 = \begin{cases} Sh_1 \left|\dfrac{\Delta_0}{\Delta_1}\right|^{0.2} & \Delta_0 \geq \Delta_1 \\ Sh_1 \left|\dfrac{\Delta_0}{\Delta_1}\right|^{0.25} & \Delta_0 < \Delta_1 \end{cases} \qquad \text{(A-7)}$$

bağıntısından bulunur. Burada S birim değerden biraz daha küçük olan 'safety faktörü' dür (Press ve diğ., 1992). Buna göre istenen doğruluk Δ_0 ile bulunabilir. Her bir adım için, Δ_1 değeri Δ_0 değerinden küçük olduğunda bu ifadeden yararlanarak adım ölçüsünün ne kadar azaltılması gerektiği veya tersi olduğunda ne kadar artırılması gerektiği hesaplanabilir.

SEMBOLLER DİZİNİ

a, b, c, c^*	Beşinci düzen Cash-Karp Runge-Kutta yönteminde kullanılan ve değerleri bilinen katsayılar.
delx	Yatay eksende ağırlıklandırma parametresi.
delz	Düşey eksende ağırlıklandırma parametresi.
Δ	Hata miktarı.
$\underline{f}$	$\underline{y}$ vektör fonksiyonunun türevini tanımlayan vector fonksiyonu.
h	Runge-Kutta yönteminde hesaplama adımı miktarı.
$k_1, k_2, k_3, k_4, k_5, k_6$	Runge-Kutta yönteminde düzen fonksiyonları.
$O(h^n)$	Runge-Kutta yönteminde düzen sayısına göre hata fonksiyonu.
P_x	Işın parametresi yatay bileşeni.
P_z	Işın parametresi düşey bileşeni.
S	Safety faktör olarak bilinen değeri birden biraz daha küçük olan bir katsayı.
σ	Işın yolu boyunca düzenli olarak artan parametre.
τ	Zaman değişkeni
x	yüzeyde yatay uzaklık değişkeni.
$V(x,z)$	İzotrop, homojen olmayan ortamın hızı.
w	Ağırlıklandırma fonksiyonu.
z	derinlik değişkeni.

MODELING BY RAY TRACING METHOD IN POLAR ANISOTROPIC MEDIUM

KUTUPSAL İZOTROP OLMAYAN BİR ORTAMDA IŞIN İZLEME YÖNTEMİ İLE MODELLEME

Selma KADIOGLU*

kadioglu@eng.ankara.edu.tr
*Ankara Universitesi, Muhendislik Fakultesi, Jeofizik Muhendisligi Bolumu, 06100 Tandogan-Ankara, TURKEY

ABSTRACT

Seismic anisotropy is defined as dependence of seismic velocity upon angle. Main causes of seismic anisotropy are due to aligned mineral grains, aligned crystals, aligned cracks and fractures and periodic thin layering. Moreover if the seismic wavelengths are large compared to the scale of ordered layers of the subsurface, they obey the laws of anisotropic wave propagation. The anisotropy types are called according to their symmetry properties.

In this study, modeling by ray tracing method was done in polar anisotropic medium that is often called vertical transverse isotropic (VTI) medium because polar anisotropy has vertical symmetry axis and the isotropy is limited to the horizontal (the transverse) plane. Firstly, ray equations and eikonal equation were obtained by using the equations of momentum conservation and the matrix of elastic constants in polar anisotropic medium. The plane wave solution was acquired via closed formulas for quasi P wave. Secondly, the characteristics of eikonal equation were constructed. Thirdly, subsurface model was constructed by discrete number of layers separated by interfaces across which the velocity can change discontinuously. Finally, initial conditions were defined and travel times were computed by fifth order Cash-Carp Runge-Kutta method for given subsurface model. Modeling was tested against three synthetic examples. The tests demonstrated the ability of the modeling of the ray tracing method in polar anisotropic medium.

This research was supported by the Scientific and Technical Research Council of Turkey (TUBITAK) under grant no. YDABCAG-100Y105.

ÖZ

Sismik anizotropi (izotrop olmayan), sismik hızın açıya bağımlılığı olarak tanımlanır. Sismik anizotropinin başlıca nedenleri, sıralı yataklanmış mineral damarları, sıralı kristaller, arka arkaya gelen çatlak ve kırıklar ve periyodik ince tabakalanmalardır. Hatta sismik dalga boyunun yeraltındaki düzenli sıralanmış tabakaların ölçeğinden daha büyük olması durumunda, anizotropik dalga yayılımı geçerlidir. Anizotropi tipleri onların simetri özelliklerine göre adlandırılır.

Bu çalışmada kutupsal anizotropik bir ortamda ışın izleme yöntemi ile modelleme yapıldı. Kutupsal anizotropik ortamın düşey simetri eksenine sahip olması ve izotropinin yatay düzlem ile sınırlı olmasından dolayı düşey enine izotrop ortam (vertical transverse isotropy = VTI) olarak da tanımlanır. Öncelikle, kutupsal anizotropik ortamda momentum korunumu denklemlerini ve elastik sabitler matrisini kullanarak ışın denklemleri ve eikonal denklem elde edildi. Quasi P dalgaları (partikül hareketi büyük çoğunlukla boyunadır ve küçük enine bileşene sahiptir) için kapalı formüller yardımı ile düzlem dalga çözümü elde edildi. İkinci olarak eikonal denklem karakteristikleri kuruldu. Üçüncü olarak, içinde hızın düzensiz olarak değişebileceği ve arayüzeyler ile ayrılmış tabakalardan oluşan yer modeli oluşturuldu. Son olarak, giriş şartları tanımlandı ve verilen model için Beşinci düzen Cash-Carp Runge-Kutta yöntemi ile seyahat zamanı hesaplandı. Modelleme üç sentetik örnek ile test edildi. Testler kutupsal anizotropik ortamda ışın izleme yöntemi ile modellemenin doğru çalıştığını gösterdi.

Bu çalışma, Türkiye Bilimsel ve Teknik Araştırma Kurumu (TÜBİTAK) tarafından YDABÇAG-100Y105 nolu proje ile desteklendi.

```
#include <stdio.h>
#include <math.h>
#include <fcntl.h>
#include "tomo.h"

/*dtomag(nx_d,skip_cmp,stat1,stat2,nOffs,Noffs,nSurf,
                    nGateT,nGate,itrmax,
                dx,offs1,doffs,dtt,zv,wb,flag_delay)
int nx_d,skip_cmp,stat1,stat2,nOffs,*Noffs,nSurf,nGateT,nGate,itrmax;
float *dx,*offs1,*doffs,*dtt;
Widget wb;
int flag_delay;   */
main()
{
  DTMG *Tm= (DTMG *) malloc(sizeof(DTMG));
  float group,dmin,doffset,efhz,dt;
  int np,ntot,nof,ns,nl,ne;
  int i;
  int itt;
  int itr,itrmax=4,itrr,limgat1,lm;
  int nll;
  int iunitjunk;
  FILE *fp;
  int jstat, kstat, nstat;


  float *pmig,*tout,*trace;
  float f1[30];
  int jf1[30];
  float  dmin_offset,doffst;
  int *icount,*id;
  char infile[80];
  int max=80;
  char Lb[200];
  void *alloc1(size_t n1, size_t size);
  void **alloc2(size_t n1,size_t n2, size_t size);
  void ***alloc3(size_t n1,size_t n2, size_t n3, size_t size);
  void ****alloc4(size_t n1,size_t n2, size_t n3,size_t n4, size_t size);

  extern void memopen3(),gnspline(),ata(),cg(),shiftcor(), zv_update();
  int icrp1;
  int iunit,iunit_zv,iunit_ray,iunit_cc;
  iunitjunk = creat("fort.88", PERMS);

  Tm->flag_delay_hyp = 0;    /* no hyperbolic delays   */
  Tm->flag_use_pan = 1;      /* use panels  */

  printf("enter file name for zv model\n");
  scanf("%s",infile);
  iunit_zv = open(infile,O_RDONLY,0);
  read(iunit_zv,jf1,2 * sizeof(int));
  nstat = jf1[1];
  Tm->nstat = nstat;
  Tm->nSurf = jf1[0];
  read(iunit_zv,f1, sizeof(float));
  Tm->dx = group = f1[0];

  printf("enter file name for cross correlations\n");
  scanf("%s",infile);
  Tm->flag_delay_hyp = 0;
  iunit_cc = open(infile,O_RDONLY,0);
```

```
   read(iunit_cc,jf1,4 * sizeof(int));
   Tm->edge = jstat = jf1[0];        /* edge    */
   kstat = jstat + jf1[2] - jf1[1] ;
   Tm->nCrp = kstat - jstat +1;
   icrp1 = jf1[1];
   Tm->nOffs = jf1[3];
   Tm->Offs = (float *)calloc(Tm->nOffs , sizeof(float));
   read(iunit_cc, Tm->Offs, Tm->nOffs * sizeof(float));
   read(iunit_cc, &Tm->nGateW, sizeof(int));
   Tm->nGate = 5;
   read(iunit_cc, &Tm->dt, sizeof(float));
   Tm->Var_Sl =  (float*)calloc( Tm->nSurf , sizeof(float));
   Tm->Var_Tv =  (float*)calloc( Tm->nSurf , sizeof(float));
   Tm->Var_Da =  (float *)calloc( Tm->nSurf,  sizeof(float));
   Tm->Var_Del =  (float *)calloc( Tm->nSurf,  sizeof(float));

   Tm->Delta = (float *)calloc( Tm->nSurf,  sizeof(float));

   Tm->Noffs = (int *)calloc( Tm->nSurf ,  sizeof(int));
   Tm->Flag_Update = (int**) alloc2(2, Tm->nSurf, sizeof(int));
   Tm->Flag_Use_Surf = (int*) calloc(Tm->nSurf, sizeof(int));
   Tm->Flag_Use_Surf_Crp = (int**) alloc2(Tm->nCrp, Tm->nSurf, sizeof(int));
   Tm->Ioffs_Surf_Crp = (int***) alloc3(Tm->nOffs, Tm->nCrp, Tm->nSurf,
sizeof(int));
   Tm->Noffs_Surf_Crp = (int**) alloc2(Tm->nstat, Tm->nSurf, sizeof(int));
   for(i=0; i < Tm->nCrp * Tm->nSurf; ++i)
     for(nof=0; nof < Tm->nOffs; ++nof)Tm->Ioffs_Surf_Crp[0][i][nof]= nof;
   memset(Tm->Flag_Use_Surf, 1, Tm->nSurf * sizeof(int));

   Tm->Shifts = (float*)calloc(Tm->nOffs, sizeof(float));



   for(nl=0; nl < Tm->nSurf * 3; ++nl) Tm->Flag_Update[0][nl] = 1; /* no
static updates   */
   printf("enter file name for ray paths \n");
   scanf("%s",infile);
   iunit_ray = open(infile,O_RDONLY,0);


   Tm->incstat=1;
   Tm->ldx = 25;
   Tm->istr=0;
   Tm->npspline = (kstat - jstat) /Tm->ldx+1;


   trace=(float*)calloc(Tm->nGateW+1 , sizeof(float));

   Tm->Xcoords = (float *)calloc(nstat ,sizeof(float));
   Tm->Si = (float **)alloc2(nstat,  Tm->nSurf + 1, sizeof(float));
   Tm->Xcc = (float *)calloc( (2 * Tm->nSurf + 1) * Tm->nOffs ,
sizeof(float));
   Tm->Ycc = (float *)calloc( (2 * Tm->nSurf + 1) * Tm->nOffs ,
sizeof(float));
   Tm->Flag_Use_Surf_Crp = (int**)alloc2(nstat,  Tm->nSurf + 1,
sizeof(int));
   memset( Tm->Flag_Use_Surf_Crp[0], 1, nstat * Tm->nSurf  * sizeof(float));
   printf("\n \n \n");
   printf("number of stations ..............%d \n",nstat);
   printf("first station ...................%d \n",jstat);
   printf("last station  ...................%d \n",kstat);
```

```
  printf("distance between stations ........%f \n",group);

  printf(" \n \n number of offsets ....... %d \n",Tm->nOffs);
  printf(" offsets ");
  for(nof=0;nof < Tm->nOffs;++nof)printf("%f  ",Tm->Offs[nof]);
  printf(" \n \n Input gate size %d \n \n",Tm->nGateW);


/*    set storage for DTMG structure   */
  Tm->npar=Tm->nSurf * 3 * Tm->npspline;

  Tm->Dypstat= (float**)calloc(Tm->nSurf,sizeof(float*));
  for(nl=0; nl < Tm->nSurf; ++nl)Tm->Dypstat[nl] =
    (float*)calloc(nstat,sizeof(float));
  Tm->Xpstat=  (float*)calloc(Tm->npar , sizeof(float));
  Tm->R =  (float*)calloc(Tm->npar , sizeof(float));
  Tm->P =  (float*)calloc(Tm->npar , sizeof(float));
  Tm->Pp =  (float*)calloc(Tm->npar , sizeof(float));
  Tm->V =  (float*)calloc(Tm->npar , sizeof(float));
  Tm->Atav =  (float*)calloc(Tm->npar , sizeof(float));
  Tm->Sv =  (float*)calloc(Tm->npar , sizeof(float));
/*  Tm->iFlag_Update =  (int*)calloc(2* Tm->nSurf , sizeof(int)); */
  Tm->Sdvordz =  (int*)calloc( Tm->nSurf , sizeof(int));
  Tm->Z =  (float*)calloc( Tm->nSurf , sizeof(float));
  Tm->AtA=(float**) alloc2(Tm->npar, Tm->npar, sizeof(float*));

  Tm->Imin =  (int***)calloc( Tm->nstat , sizeof(int**));
  Tm->Imax =  (int***)calloc( Tm->nstat , sizeof(int**));
  for(ns=0; ns < Tm->nstat; ++ns){
    Tm->Imin[ns] = (int**)calloc(Tm->nSurf, sizeof(int*));
    Tm->Imax[ns] = (int**)calloc(Tm->nSurf, sizeof(int*));
    for(nl=0; nl < Tm->nSurf; ++nl){
      Tm->Imin[ns][nl] = (int*)calloc(nl+1, sizeof(int));
      Tm->Imax[ns][nl] = (int*)calloc(nl+1, sizeof(int));
    }
  }
  if(Tm->nGate > 4){
/* open storage for rp and rn      */
    Tm->Rp=(float**)calloc(Tm->npar,sizeof(float*));
    Tm->Rn=(float**)calloc(Tm->npar,sizeof(float*));
    for(np=0; np < Tm->npar; ++np){
      Tm->Rp[np]= (float*)calloc(Tm->nGate + 1,sizeof(float));
      Tm->Rn[np]= (float*)calloc(Tm->nGate + 1,sizeof(float));
    }
  }


  Tm->A_Jac = (float****) alloc4(3, Tm->npspline, Tm->nSurf, Tm->nOffs,
sizeof(float));



  Tm->Zi = (float**)alloc2( nstat, Tm->nSurf+1, sizeof(float));
  Tm->Vi = (float**)alloc2( nstat, Tm->nSurf+1, sizeof(float));



  for(nl=0; nl < Tm->nSurf; ++nl)Tm->Sdvordz[nl] = 0;


  for(nl=0; nl <= Tm->nSurf ; ++nl){
```

```
    for(ns=0; ns < nstat; ++ns){
      read(iunit_zv, &Tm->Zi[nl][ns], sizeof(float));
      read(iunit_zv, &Tm->Vi[nl][ns], sizeof(float));
    }
  }
  read(iunit_zv, Tm->Delta, Tm->nSurf * sizeof(float));
  for( nl = 0; nl < Tm->nSurf; ++nl) printf("nl= %d Delta= %f \n", nl, Tm-
>Delta[nl]);
  printf("enter file name for rewriting zv model\n");
  scanf("%s",infile);
  iunit=creat(infile,PERMS);
  f1[0] = nstat ;
  f1[1] = Tm->nSurf ;
/*  write(iunit, f1,8);*/
  for(ns=0; ns < nstat ; ++ns){
    for(nl=0; nl <= Tm->nSurf ; ++nl){
/*      if(ns > nstat-20)Tm->Zi[nl][ns][0] = Tm->Zi[nl][ns-20][0];
      if(ns > nstat-20)Tm->Zi[nl][ns][1] = Tm->Zi[nl][ns-20][1];
      write(iunit,Tm->Zi[nl][ns],8);  */
    }
  }
  close(iunit);


/*      open storage for tcor       */

  Tm->Tcor = (float ****) calloc(Tm->nSurf,sizeof(float***));

  for(nl=0; nl<Tm->nSurf; nl++){
    read(iunit_cc, jf1,  sizeof(int));
    read(iunit_cc, Tm->Var_Sl + nl ,  sizeof(float));
    read(iunit_cc, Tm->Var_Tv + nl ,  sizeof(float));
    read(iunit_cc, Tm->Var_Da + nl, sizeof(float));
    Tm->Var_Del[nl] = 0.f;
    read(iunit_cc, Tm->Noffs + nl, sizeof(int));
    printf("layer %d \n _________ \n Var_Sl %f Var_Tv %f \n",
         nl+1, Tm->Var_Sl[nl], Tm->Var_Tv[nl]);
    printf(" Var_Da= %f \n", Tm->Var_Da[nl]);
    printf("    number of offsets %d \n", Tm->Noffs[nl]);
    printf("\n");
    Tm->Tcor[nl] = (float ***)calloc(Tm->nstat, sizeof(float**));
    for(ns=0; ns < nstat ; ++ns){
      Tm->Tcor[nl][ns] = (float **)calloc(Tm->Noffs[nl] , sizeof(float*));
      for(nof=0; nof<Tm->Noffs[nl]; nof++){
      Tm->Tcor[nl][ns][nof] = (float * ) calloc(Tm->nGateW,sizeof(float));
      }
    }

    if(Tm->Noffs[nl] != 0)
      for(ns = 0; ns <Tm->nCrp;++ns){
      read(iunit_cc, f1, sizeof(float));
      read(iunit_cc, jf1, sizeof(int));
        for(nof=0 ; nof < Tm->Noffs[nl]; ++nof){
          if(nof >= jf1[0]){
            continue;
          } else {
            read(iunit_cc,trace,(Tm->nGateW) * 4);
            if(nl == 0) write(iunitjunk, trace, Tm->nGateW *
sizeof(float));
            for(lm = 0; lm < Tm->nGateW; ++lm)
            Tm->Tcor[nl][ns][nof][lm] = trace[lm];
```

```
            }
            }
        }
/*      free_cc();      */
  }
  for(ns=0; ns < Tm->nCrp; ++ns){
    for(nl=0; nl < Tm->nSurf; ++nl){
      Tm->Noffs_Surf_Crp[nl][ns] = Tm->Noffs[nl];
    }
  }
  Tm->Delays = (float***)calloc( Tm->nSurf,  sizeof(float**));
  for(nl=0; nl < Tm->nSurf; ++nl){
    Tm->Delays[nl] = (float**) calloc(Tm->nCrp, sizeof(float*));
    for(ns=0; ns < Tm->nCrp; ++ns) Tm->Delays[nl][ns] = (float*)calloc(Tm-
>Noffs[nl], sizeof(float));
  }


/*  OmUnmanageMessage ();  */


  gnspline(Tm);

/*  atar(t,flag_delay,wb);  */
  atar(Tm,iunit_ray);

  printf(" enter itrmax \n");
  scanf("%d",&itrmax);
  for(itr=0;itr < itrmax ; ++itr){
      itrr = itr + 1;
/*      sprintf(Lb,"Invert Matrix - iteration # %d " , itrr);
    OmManageMessage(Lb,"MESSAGE");  */
    printf("itr= %d \n", itr);
    cg(Tm);
    newr(Tm,&itr,&itrmax);    /*,flag_delay);  */
  }
  zv_update(Tm);
/*
  shiftcor(t,Tm->ngateW);
*/
/*  OmUnmanageMessage ();  */

/*  create_matrixZV(); */

  printf("enter file name for output zv model\n");
  scanf("%s",infile);
  iunit=creat(infile,PERMS);
  f1[0] = nstat ;
  f1[1] = Tm->nSurf ;
  write(iunit, f1,8);
/*  f1[0] = Tm->dx;
  write(iunit_zv, f1, sizeof(float));  */
  for(ns=0; ns < nstat ; ++ns){
    for(nl=0; nl <= Tm->nSurf ; ++nl){
      write(iunit,&Tm->Zi[nl][ns], sizeof(float));
      write(iunit,&Tm->Vi[nl][ns], sizeof(float));
    }
  }
  write(iunit, Tm->Delta, Tm->nSurf * sizeof(float));
/*  shiftcor(t,iunit_cc,Tm->ngateW);  */
  iunit=creat("fort.22",PERMS);
```

```
 }

   scanf("%s",infile);
   iunit=creat(infile,PERMS);
   f1[0] = nstat ;
   f1[1] = Tm->nSurf ;
   write(iunit, f1,8);
/*  f1[0] = Tm->dx;
   write(iunit_zv, f1, sizeof(float));  */
   for(ns=0; ns < nstat ; ++ns){
     for(nl=0; nl <= Tm->nSurf ; ++nl){
       write(iunit,&Tm->Zi[nl][ns], sizeof(float));
       write(iunit,&Tm->Vi[nl][ns],

#define PERMS 0660
#define ffabs(A) ( A > 0 ? A:-A)
#include <Mrm/MrmAppl.h>
#include "tomo.h"
#include <fcntl.h>
#include <stdio.h>
#include <math.h>

typedef float Flt2_t[2];


int atar(Tm, iunit_ray)
DTMG *Tm;
int iunit_ray;
{
    extern void residual_T();
    extern void residual_C();
    extern void autopick();
    extern void onesmth();
    int ix,lm,ipick=1,np,np1,j,jj,jjj,k,i,lenR,
        nGate,nGateW,igate1,noffs_surf_crp;
    int m,ity,nl,icrp,nof,n5,nee,ne,iErr = 0, icrpp,iof;
    int m1,m2,n,n1,ity1,n2,np2,ity2;
    float dt,f,amx,sdd,sum;
    float *Trace;
    int *Imin , *Imax;
    char Lb[200];
    int iunit;

    Imin   =  (int*)calloc( Tm->nSurf , sizeof(int));
    Imax   =  (int*)calloc( Tm->nSurf , sizeof(int));
    iunit=creat("fort.22",PERMS);
/*    if(logical_exist("tomo_print") ){
      iunit = creat("fort.3" , PERMS);
    }    */

    nGate  = Tm->nGate;
    nGateW = Tm->nGateW;
    igate1 = (nGateW - nGate) / 2;
    Trace = (float*)calloc(nGate+1, sizeof(float));
    dt = Tm->dt;

    if(!Tm->flag_delay_hyp)ipick = 1; /* autopick */
    else                   ipick = 0; /* use delays */

/*    create_matrixZV();  */
```

```
/*    weigh standard deviations according to layer thickness at each spline
node     */
    for(np=0; np < Tm->npspline ; ++np){

      j = Tm->edge + Tm->ldx * np;

      j = Tm->ldx * np;

      for(nl=0; nl < Tm->nSurf; ++nl){

          if(Tm->Var_Sl[nl] == 0)continue;
/* normalize velocity variance based on 500 ms traveltimes */
          f = Tm->Dypstat[nl][np] / ( Tm->del * Tm->Vi[nl+1][j] ) * 2.;

          if(Tm->Flag_Update[nl][0] == 1)
            f= Tm->Z[nl] / ( Tm->del * Tm->Vi[nl+1][j]) * 2.;
          m = (np+nl*Tm->npspline) * 3;

          Tm->Sv[m  ] = Tm->Var_Sl[nl] * f;
          Tm->Sv[m+1] = Tm->Var_Tv[nl];
          Tm->Sv[m + 2] = Tm->Var_Del[nl] * f;

      }
    }

    for ( nl=0; nl < Tm->nSurf  ; ++nl){
      if(!Tm->Flag_Use_Surf[nl])continue;

      lenR = 2 * nl + 3;
/*    start_ray(nl);   */

      for(icrp = 0;icrp < Tm->nCrp; icrp+=Tm->incstat){
      for(n=0; n <= nl; ++n){
        Tm->Imax[icrp][nl][n] = 0;
        Tm->Imin[icrp][nl][n] = Tm->npspline -1;          /* ONE    */
      }
      if(!Tm->Flag_Use_Surf_Crp[nl][icrp]){
        Tm->Noffs_Surf_Crp[nl][icrp] = 0;
        continue;
      }

      sdd = Tm->Var_Da[nl];

      icrpp = Tm->icrp1 + icrp;

      noffs_surf_crp = read_ray(Tm, icrpp, nl, iunit_ray);
      if(noffs_surf_crp == -1){
        Tm->nSurf = nl+1;
        return(0);
      }

      if(noffs_surf_crp < 1){
        Tm->Noffs_Surf_Crp[nl][icrp] = 0;
        continue;
      }

      if(noffs_surf_crp > Tm->Noffs_Surf_Crp[nl][icrp])
        noffs_surf_crp = Tm->Noffs_Surf_Crp[nl][icrp];

      Tm->Noffs_Surf_Crp[nl][icrp] = noffs_surf_crp;
```

```
        memset(Tm->A_Jac[0][0][0] , 0 , 3 * Tm->npspline *
                 Tm->nOffs * Tm->nSurf * sizeof(float));

        for( iof=0 ; iof <noffs_surf_crp; ++iof){

          nof = Tm->Ioffs_Surf_Crp[nl][icrp][iof];
          m = iof * lenR;

          a_mat(Tm,Tm->A_Jac[nof],Tm->Xcc+m,Tm->Ycc+m,nl);
        }
        for(np=0; np < Tm->npar ; ++np){
          for(sum = 0., iof=0 ; iof <noffs_surf_crp; ++iof){
            nof = Tm->Ioffs_Surf_Crp[nl][icrp][iof];
            sum += Tm->A_Jac[nof][0][0][np];
          }
          sum /= noffs_surf_crp;
          for(iof=0 ; iof <noffs_surf_crp; ++iof){
            nof = Tm->Ioffs_Surf_Crp[nl][icrp][iof];
            Tm->A_Jac[nof][0][0][np] -= sum;
          }
        }


        for( iof=0 ; iof <noffs_surf_crp; ++iof){

          nof = Tm->Ioffs_Surf_Crp[nl][icrp][iof];
          m = iof * lenR;

/*          if(logical_exist("tomo_print") ){
              write(iunit,&nl,4);
              write(iunit,&icrp,4);
              write(iunit,&nof,4);
              write(iunit,Tm->Xcc+m,lenR*4);
              write(iunit,Tm->Ycc+m,lenR*4);
            } */

          if(ipick){
            memcpy(Trace , Tm->Tcor[nl][icrp][nof] + igate1 ,
                 nGate * sizeof(float));
            residual_C(Tm, Tm->A_Jac[nof], Trace, &sdd, nGate);
          }else{

            residual_T(Tm,Tm->A_Jac[nof],&sdd,&Tm->Delays[nl][icrp][nof]);

          }


          for(amx = 0., n=0 ; n <= nl ; ++n){
            for(np=0 ; np < Tm->npspline ; ++np){
              for(ity=0 ; ity < 3 ; ++ity)
              if(amx < ffabs (Tm->A_Jac[nof][n][np][ity]))
                amx = ffabs (Tm->A_Jac[nof][n][np][ity]);
            }
          }

          for(n=0, Imin[n]=Tm->npspline-1, Imax[n]=0 ; n <= nl ; ++n){
            for(np=0 ; np < Tm->npspline ; ++np , m+=2){
              if(ffabs(Tm->A_Jac[nof][n][np][0])+ffabs(Tm-
>A_Jac[nof][n][np][1]) >
                 0.5e-3 * amx){
                Tm->Imin[icrp][nl][n] =
```

```
              np < Tm->Imin[icrp][nl][n]? np:Tm->Imin[icrp][nl][n];
/* TWO  */
            Tm->Imax[icrp][nl][n] =
              np > Tm->Imax[icrp][nl][n]? np:Tm->Imax[icrp][nl][n];
            Imax[n]=np;
            Imin[n]= np < Imin[n] ? np:Imin[n];
            }
          }
        }

        for(n1=0; n1 <= nl ; ++n1){
          for(np1=Imin[n1] ; np1 <= Imax[n1] ; ++ np1 ){
            for( ity1=0 ; ity1 < 3 ; ++ ity1){
            float sv_m1_sdd;
            m1 = ity1 + 3 * (np1 + n1 * Tm->npspline);
            sv_m1_sdd =  Tm->Sv[m1] * sdd;
            for(n2=0; n2 <= nl ; ++n2){
              int n2_nps = n2 * Tm->npspline;
              float jac1 = Tm->A_Jac[nof][n1][np1][ity1];

              for(np2=Imin[n2] ; np2 <= Imax[n2] ; ++ np2 ){
                int np2_n2_nps2 = (n2_nps + np2) * 3;
                float *Jac2;
                Jac2 = Tm->A_Jac[nof][n2][np2];
                for( ity2=0 ; ity2 < 3 ; ++ ity2){
                  m2 = ity2 + np2_n2_nps2;
                  Tm->AtA[m1][m2] +=
                  jac1 * *(Jac2+ity2) * sv_m1_sdd * Tm->Sv[m2];
                }
              }
            }
            }
          }
        }

      }                                          /*  offset loop    */

      for( iof=0 ; iof < Tm->Noffs_Surf_Crp[nl][icrp]; ++iof){
        nof = Tm->Ioffs_Surf_Crp[nl][icrp][iof];
        for(n=0; n <= nl; ++n){
          if(Tm->Imin[icrp][nl][n] <= Tm->Imax[icrp][nl][n])
            write(iunit, Tm->A_Jac[nof][n][Tm->Imin[icrp][nl][n]],
                3 * (Tm->Imax[icrp][nl][n]- Tm->Imin[icrp][nl][n] + 1) *
sizeof(float));
        }
      }


      }                                          /*  icrp loop   */
/*    close_ray();   */
    }                                            /*  layer loop    */
    close(iunit);
    if(ipick == 1){
/*  automatic picking          */
      autopick(Tm,nGate);
    }else {
/* dc constraint on Tm->r (with autopicking this is done in autopick)  */
      for(nl=0 , n=0 ; nl < Tm->nSurf ; ++nl){
          for(ity=0 ; ity < 3 ; ++ity, ++n){
            if( Tm->Flag_Update[nl][ity] == 1 ){
                for( np=0, sum=0. ; np < Tm->npspline ; ++np){
```

```
                m = ity + 3 * (np +nl*Tm->npspline);
                sum += Tm->R[m];
                Tm->R[m] = 0.;
              }
              Tm->R[ity+ 3 * nl*Tm->npspline]=sum;
            }
          }
        }
      }
/* add one on diagonal of AtA and add -1 2 -1 smoother      */
      onesmth(Tm);
/*    dc constraint on AtA     */
      for(nl=0 , m2=0 ; nl < Tm->nSurf ; ++ nl){
        for(ity=0 ; ity < 3 ; ++ ity, ++m2){
          if(Tm->Flag_Update[nl][ity] == 1){
            for(m1=0 ; m1 < Tm->npar ; ++m1){
              for(np=0,sum=0. ; np < Tm->npspline ; ++np){
                m = ity + (np + nl*Tm->npspline) * 3;
                sum += Tm->AtA[m1][m];
                Tm->AtA[m1][m]=0.;
              }
              Tm->AtA[m1][ity+ nl * 3 *Tm->npspline]=sum;
            }                                    /*        m1 loop     */
          }                                        /* Iwm if     */
        }                                        /* ity loop  */
      }                                          /* nl loop   */
      for(nl=0 , m2=0 ; nl < Tm->nSurf ; ++ nl){
        for(ity=0 ; ity < 3 ; ++ ity, ++m2){
          if(Tm->Flag_Update[nl][ity] == 1){
            for(m1=0 ; m1 < Tm->npar ; ++m1){
              for(sum=0., np=0 ; np < Tm->npspline ; ++np){
                m=ity + (np + nl*Tm->npspline) * 3;
                sum += Tm->AtA[m][m1];
                Tm->AtA[m][m1]=0.;
              }
              Tm->AtA[ity+nl * 3 * Tm->npspline][m1]=sum;
            }                                    /*        m1 loop     */
          }                                        /* Iwm if     */
        }                                        /* ity loop  */
      }                                          /* nl loop   */

      return(0);
}
```

```
#define PERMS 0660
#include "tomo.h"

void gnspline(Tm)
DTMG *Tm;
{
    float del,delt,delt1,dt;
    extern void hspline();
    int np,ns,nof,nl;
    int iunit;

/*    if(logical_exist("tomo_print") ){
      iunit = creat("fort.1" , PERMS);

      write(iunit,&Tm->nSurf,4);
      write(iunit,&Tm->nstat,4);
      write(iunit,&Tm->dx,4);

      for(nl=0 ; nl <= Tm->nSurf ; ++nl){
          for(ns=0 ; ns < Tm->nstat ; ++ns){
            write(iunit, Tm->Zi[nl][ns] , 4);
            write(iunit, Tm->Vi[nl][ns] , 4);
          }
      }
    }
*/
    dt = Tm->dt;

    for(ns=0; ns<Tm->nstat; ns++)
        Tm->Xcoords[ns] = (ns - Tm->edge);

    Tm->del = 1./Tm->dx;
    for(nof=0; nof < Tm->nOffs ; ++nof)  Tm->Offs[nof]   *= Tm->del;

    for( nl=0 ; nl < Tm->nSurf ; ++nl){
      for(Tm->Z[nl] = 0,ns=0 ; ns < Tm->nstat ; ++ns)
        Tm->Z[nl] += Tm->del *
        (Tm->Zi[nl+1][ns] - Tm->Zi[nl][ns])/ (float) Tm->nstat;
      Tm->Z[nl] = (Tm->Z[nl] > 0.1e-2*Tm->del) ? Tm->Z[nl]
                                               : 0.1e-2 * Tm->del;
    }

    for(np=0, ns=Tm->edge ; np < Tm->npspline ; ++np, ns+= Tm->ldx){
      Tm->Xpstat[np] = Tm->Xcoords[ns];
      for(nl = 0 ; nl < Tm->nSurf ; ++nl){
          Tm->Dypstat[nl][np] = (Tm->Zi[nl+1][ns] -
                                 Tm->Zi[nl  ][ns]) * Tm->del;
          if( Tm->Dypstat[nl][np] <
              0.1e-1 * (Tm->Zi[nl+1][ns] - Tm->Zi[0][ns]) * Tm->del)
            Tm->Dypstat[nl][np] = 0.;
      }
    }
    for(nl=1 ; nl <= Tm->nSurf ; ++nl){
      for(ns=0 ; ns < Tm->nstat ; ++ns){
          Tm->Si[nl][ns] = 1. / (Tm->Vi[nl][ns] * Tm->del * dt);
      }
    }
}
```

```
for(np=0, ns=Tm->edge ;
np < Tm
int isearch(xx,x,npspline,iflag)
/*-----------------------------------------------------------------
find to which interval the value of xx belongs to
  iflag=0     ok
  iflag=1     point is out of interval covered by spline
-----------------------------------------------------------------*/
float xx,*x;
int npspline,*iflag;
{
int j1,j2,jm;
  *iflag=0;
  if( xx < x[0] || xx > x[npspline-1]){
    *iflag=1;
    return 0.;
  }
  else {
    j1=0;
    j2=npspline-1;
    while(j1 < j2-1){
      jm=(j1+j2)/2;
      if(xx < x[jm])
      j2=jm;
      else
      j1=jm;
    }
    return j1;
  }
}
----------------------------------
find to which interval the value of xx belongs to
  iflag=0     ok
  iflag=1     point is out of interval covered by spline
-----------------------------------------------------------------*/
float xx,*x;
int npspline,*iflag;
{
int j1,j2,jm;
  *iflag=0;
  if( xx < x[0] || xx > x[npspline-1]){
    *iflag=1;
    return 0.;
  }
  else {
    j1=0;
    j2=npspline-1;
    while(j1 < j2-1){
      jm=(j1+j2)/2;
      if(xx <

#include <stdio.h>
#include <math.h>
#include <fcntl.h>
#include "tomo.h"
void cg(Tm)
DTMG *Tm;
{
    int npar=Tm->npar,np,npl,itr,itrmaxCg=100;
    float rs,rr0,vav,rr,rav,alpha,beta;
    int lm;
```

```
    rr0=0.;
    for(np = 0 ; np < npar ; ++np){
      rr0 += (Tm->R[np] * Tm->R[np]);
      Tm->V[np] = Tm->R[np];
      Tm->P[np] = 0.;
    }
    rr = rr0;

    if(!rr){
      printf("\n from cg: rr = 0 ");
      return;
    }

    for(itr=0 ; itr < itrmaxCg ; ++itr){
      vav=0.;
      for(np=0 ; np < npar ; ++np){
          Tm->Atav[np]=0.;
          for(np1=0 ; np1 < npar ; ++np1){
            Tm->Atav[np] += Tm->AtA[np][np1] * Tm->V[np1];
          }
          vav += (Tm->V[np] * Tm->Atav[np]);
      }

      if(!vav){
          printf("\n from cg: vav = 0 ");
          return;
        }

      alpha = rr/vav;

      rs=rr;
      rr=0.;
      rav=0.;
      for(np = 0 ; np < npar ; ++np){
          Tm->P[np] += alpha * Tm->V[np];
          Tm->R[np] -= alpha * Tm->Atav[np];

          rr += Tm->R[np]*Tm->R[np];
          rav += Tm->R[np]*Tm->Atav[np];
      }
      if(rr/rr0 < 0.1e-6)break;
      beta=rr/rs;
      for(np=0 ; np < npar ; ++np)
        Tm->V[np] = Tm->R[np] + beta * Tm->V[np];
    }
}


 * Tm->V[np1];
          }
          vav += (Tm->V[np] * Tm->Atav[np]);
      }

      if(!vav){
          printf("\n from cg: vav = 0 ");
          return;
        }

      alpha = rr/vav;
```

```
        rs=rr;
        rr=0.;
        rav=0.;
        for(np = 0 ; np < npar ; ++np){
            Tm->P[np] += alpha * Tm->V[np];
            Tm->R[np] -= alpha * Tm->Atav[np];

            rr += Tm->R[np]*Tm->R[np];
            rav += Tm->R[np]*Tm->Atav[np];
        }
        if(rr/rr0 < 0.1e-6)break;
        beta=rr/rs;
        for(np=0 ; np < npar ; ++np)
          Tm->V[np]



double hermite(f,xs,x,ityp)
float *f,*xs,x;
int ityp;
{
  float eps,dx;
  double retval;
  eps=(abs(f[1]+abs(f[0])))*0.1e-4;
  dx=xs[1]-xs[0];
  if(ityp == 3) {
/*     linear interpolation */
    retval=(f[0]*(xs[1]-x)+f[1]*(x-xs[0]))/dx;
    return retval;
  }
  else;
  {
    printf("exit hermite unimplemented");
    exit();
  }
}
/*       else if(ityp.eq.1) then
         dfm1=(f[1]-f[0])/dx
         dff=(f(2)-f[1])/(xs(2)-xs[1])
         w1=1./amax1(eps,abs(dfm1))
         w2=1./amax1(eps,abs(dff))
         df=(w1*dfm1+w2*dff)/(w1+w2)
        else if(ityp.eq.2) then
         df=(f[1]-f[0])/dx
         dffm1=(f[0]-f(-1))/
               (xs[0]-xs(-1))
         w1=1./amax1(eps,abs(dffm1))
         w2=1./amax1(eps,abs(df))
         dfm1=(w1*dffm1+w2*df)/(w1+w2)
        else
         dff=(f[1]-f[0])/dx
         dffm1=(f[0]-f(-1))/(xs[0]-xs(-1))
         dffp1=(f(2)-f[1])/(xs(2)-xs[1])
         w1=amax1(eps,abs(dffm1))
         w2=amax1(eps,abs(dff))
         w3=amax1(eps,abs(dffp1))
         dfm1=(w1*dffm1+w2*dff)/(w1+w2)
         df=(w2*dff+w3*dffp1)/(w2+w3)
        end if
        a=dx*(df+dfm1)-2.*(f[1]-f[0])
```

```
        b=3.*(f[1]-f[0])-dx*(2.*dfm1+df)
        c=dx*dfm1
        d=f[0]
        del=(x-xs[0])/dx
        hermite=d+del*(c+del*(b+del*a))
       return
       end
*/

1+w2)
        else
         dff=(f[1]-f[0])/dx
         dffm1=(f[0]-f(-1))/(xs[0]-xs(-1))
         dffp1=(f(2)-f[1])/(xs(2)-xs[1])
         w1=amax1(eps,abs(dffm1))
         w2=amax1(eps,abs(dff))
         w3=amax1(eps,abs(dffp1))
         dfm1=(w1*dffm1+w2*dff)/(w1+w2)




void residual_C(Tm,A_Jac,Tcor,Sdd,nGate)
DTMG *Tm;
float ***A_Jac;
float *Tcor,*Sdd;
int   nGate;
{
    float sdd;
    int lm,np,m,ity,n;

    sdd = *Sdd;

    for( m=0,n=0 ; n < Tm->nSurf ; ++n){
      for(np=0 ; np < Tm->npspline ; ++np){
          for(ity=0 ; ity < 3 ; ++ity,++m){
            if( A_Jac[n][np][ity] > 0) {
                Tm->Rp[m][nGate] += A_Jac[n][np][ity] * sdd;
                for( lm=0 ; lm < nGate ; ++lm)
                  Tm->Rp[m][lm] +=
                  A_Jac[n][np][ity]*Tcor[lm] * sdd;
            }else {
                Tm->Rn[m][nGate] += A_Jac[n][np][ity] * sdd;
                for( lm=0 ; lm < nGate ; ++lm)
                  Tm->Rn[m][lm] +=
                  A_Jac[n][np][ity]*Tcor[lm] * sdd;
            }
          }
      }
    }
}

m,ity,n;

    sdd = *Sdd;

    for( m=0,n=0 ; n < Tm->nSurf ; ++n){
      for(np=0 ; np < Tm->npspline ; ++np){
          for(ity=0 ; ity < 3 ; ++ity,++m){
            if( A_Jac[n][np][ity] > 0) {
```

```
                Tm->Rp[m][nGate] += A_Jac[n][np][ity] * sdd;
                for( lm=0 ; lm < nGate ; ++lm)
                  Tm->Rp[m][lm] +=
                  A_Jac[n][np][ity]*Tcor[lm] * sdd;
              }else {
                Tm->Rn[m][nGate] += A_Jac[n][np][ity] * sdd;
                for( lm=0 ; lm < nGate ; ++lm)
                  Tm
```

```
#define ffabs(A) ( A > 0 ? A:-A)
#include "tomo.h"
#include <fcntl.h>
#include <stdio.h>
#include <math.h>
void a_mat(Tm,A_Jac,Xcc,Ycc,nl)
DTMG *Tm;
float ***A_Jac;
int nl;
float *Xcc,*Ycc;
{
    extern void hermitw();
    int nn,n,i1,i2,k,kk,ityp,i,ip,iflag,nside;
    int isign,ithick,j1,n1,np,npsm1,npsm11;
    float Wb[4],We[4],Wm[4],x1,x2,y1,y2,dx,dy,dz,sint,ds;
    float f,sl1,a_j,x11,x22,xmed,xx1,xx2,dxx,sint_in;
    float xp,gama,gama1,dgama,cost,dx1,dy1,x3,y3,dh2, delta = 0.f;
    void fp_to_p(float, float,
             float, float, float*, float*, float*);
    nn= 2 * (nl + 1);
    npsm1 = Tm->npspline - 1;
    npsm11 = npsm1 - 1;

    dy = 2 * Ycc[nl+1] - Ycc[nl+2] - Ycc[nl];
    if(dy > 0.1){
/*     SLOWNESS JACOBIAN   */
      for(nside =0 ; nside < 2 ; ++nside){     /* left and right branches
of crp */
          for(n=0 ; n <= nl ; ++n){
            delta = Tm->Delta[n];
            if(nside == 0) {
                x1 = Xcc[n];
                x2 = Xcc[n+1];
                y1 = Ycc[n];
                y2 = Ycc[n+1];
            } else {
                x1 = Xcc[nn-n];
                x2 = Xcc[nn-n-1];
                y1 = Ycc[nn-n];
                y2 = Ycc[nn-n-1];
            }
            dx = x2 - x1;
            dy = y2 - y1;
            ds = dx * dx + dy * dy;
            if(ds > 0.1){    /* if skip zero thickness */
                sint = dx / sqrt(ds);
                if(ffabs(sint) < 0.1e-2){
                  /*     vertical ray     */
                  if(x1 < Tm->Xpstat[0])
                    A_Jac[n][0][0] += 1.;
                  else if( x1 >= Tm->Xpstat[npsm1])
```

```
                A_Jac[n][npsm1][0] += 1.;
              else {
                  i1 = index_spl(&x1,Tm->ldx,0,npsm11);
                  ityp=0;
                  if(i1== 0      ) ityp = 1;
                  if(i1 == npsm11) ityp = 2;
                  hermitw(&Tm->Xpstat[i1],&x1,Wb,ityp);
                  for(k=0 ; k < 4 ; ++k){
                    ip = i1 + k - 1;
                    if(ip < 0 ) ip = 0;
                    if(ip > npsm1 ) ip = npsm1;
                    A_Jac[n][ip][0] += Wb[k];
                  }
              }
            }else {
              sint_in = 1. / ( 6. * sint);
              if(x1 > x2) {
                  x11 = x1;
                  x1 = x2;
                  x2 = x11;
                  sint_in = - sint_in;
              }
              if(x1 < Tm->Xpstat[0]){
                  float Px, Py, Arg, F = 0.25f, K, Sl, Sl2;
                  float dGdSl, dGddel;
                  Sl = Tm->Si[n + 1][0];
                  Sl2 = Sl * Sl;
                  fp_to_p(Sl, delta, dx, dy, &Px, &Py, &Arg);
                  K = (1.f + F) * Sl2 * (Px * Px + Py * Py);
                  dGdSl = (4.f * Sl * Sl2 - 2.f * K / Sl) / Arg;
                  dGddel = -2.f * (1.f - F) * (Px * Px * Py * Py) /
Arg;

                  x22 = (x2 < Tm->Xpstat[0]) ? x2 : Tm->Xpstat[0];
                  dxx = x22 - x1;
                  f=1.;
                  if(Tm->Dypstat[n][0] > 0.1e-5)
                    f = Tm->Dypstat[n][0];
                  A_Jac[n][0][0] += 6.* dxx  * dGdSl * sint_in / f;
                  A_Jac[n][0][2] += 6.* dxx  * dGddel * sint_in / f;

                  x1 = x22;
              }
              else if(x2 > Tm->Xpstat[npsm1]){
                  float Px, Py, Arg, F = 0.25f, K, Sl, Sl2;
                  float dGdSl, dGddel;
                  Sl = Tm->Si[n + 1][Tm->nstat - 1];
                  Sl2 = Sl * Sl;
                  fp_to_p(Sl, delta, dx, dy, &Px, &Py, &Arg);
                  K = (1.f + F) * Sl2 * (Px * Px + Py * Py);
                  dGdSl = (4.f * Sl * Sl2 - 2.f * K / Sl) / Arg;
                  dGddel = -2.f * (1.f - F) * (Px * Px * Py * Py) /
Arg;

                  x11 = (x1 > Tm->Xpstat[npsm1]) ? x1 : Tm-
>Xpstat[npsm1];

                  dxx = x2 - x11;
                  f=1.;
                  if(Tm->Dypstat[n][npsm1] > 0.1)
                    f = Tm->Dypstat[n][npsm1];
                  A_Jac[n][npsm1][0] += 6. * dxx * dGdSl * sint_in / f;
                  A_Jac[n][npsm1][2] += 6. * dxx * dGddel * sint_in /
f;
```

```
                    x2 = x11;
                }
                if(x1 >= Tm->Xpstat[0] && x2 <= Tm->Xpstat[npsm1]){

                    i1 = index_spl(&x1,Tm->ldx,0,npsm11);
                    i2 = index_spl(&x2,Tm->ldx,0,npsm11);

                    for( i=i1 ; i <= i2 ; ++i){
                      float Px, Py, Arg, F = 0.25f, K, Sl, Sl2;
                      float dGdSl1, dGddel1;
                      float dGdSl2, dGddel2;
                      float dGdSlm, dGddelm;

                      xx1 = Tm->Xpstat[i];
                      if(i == i1) xx1 = x1;
                      xx2 = Tm->Xpstat[i+1];
                      if(i == i2) xx2 = x2;
                      xmed = 0.5 * (xx1 + xx2);
                      ityp=0;
                      if(i ==      0) ityp = 1;
                      if(i == npsm11) ityp = 2;
                      hermitw(&Tm->Xpstat[i],&xx1  ,Wb,ityp);
                      j1 = index_xcr(&xx1, Tm->edge, 0, Tm->nstat - 1);
                      Sl = Tm->Si[n + 1][j1];
                      Sl2 = Sl * Sl;
                      fp_to_p(Sl, delta, dx, dy, &Px, &Py, &Arg);
                      K = (1.f + F) * Sl2 * (Px * Px + Py * Py);
                      dGdSl1 = (4.f * Sl * Sl2 - 2.f * K / Sl) / Arg;
                      dGddel1 = -2.f * (1.f - F) * (Px * Px * Py * Py)
/Arg;

                      hermitw(&Tm->Xpstat[i],&xmed,Wm,ityp);
                      j1 = index_xcr(&xmed, Tm->edge, 0, Tm->nstat - 1);
                      Sl = Tm->Si[n + 1][j1];
                      Sl2 = Sl * Sl;
                      fp_to_p(Sl, delta, dx, dy, &Px, &Py, &Arg);
                      K = (1.f + F) * Sl2 * (Px * Px + Py * Py);
                      dGdSlm = (4.f * Sl * Sl2 - 2.f * K / Sl) / Arg;
                      dGddelm = -2.f * (1.f - F) * (Px * Px * Py * Py) /
Arg;

                      hermitw(&Tm->Xpstat[i],&xx2  ,We,ityp);

                      hermitw(&Tm->Xpstat[i],&xmed,Wm,ityp);
                      j1 = index_xcr(&xx2, Tm->edge, 0, Tm->nstat - 1);
                      Sl = Tm->Si[n + 1][j1];
                      Sl2 = Sl * Sl;
                      fp_to_p(Sl, delta, dx, dy, &Px, &Py, &Arg);
                      K = (1.f + F) * Sl2 * (Px * Px + Py * Py);
                      dGdSl2 = (4.f * Sl * Sl2 - 2.f * K / Sl) / Arg;
                      dGddel2 = -2.f * (1.f - F) * (Px * Px * Py * Py) /
Arg;

                      for(k=0 ; k < 4 ; ++k){
                          ip = i + k - 1;
                          if(ip < 0    ) ip = 0;
                          if(ip > npsm1) ip = npsm1;
                          f=1.;
                          if(Tm->Dypstat[n][ip] > 0.1)
                            f = Tm->Dypstat[n][ip];
```

```
                                A_Jac[n][ip][0] += sint_in*
                                  (Wb[k] * dGdSl1 + 4. * dGdSlm * Wm[k] + We[k]
* dGdSl2) *
                                  ( xx2 - xx1) / f;
                                A_Jac[n][ip][2] += sint_in*
                                  (Wb[k] * dGddel1 + 4. * dGddelm * Wm[k] +
We[k] * dGddel2) *
                                  ( xx2 - xx1) / f;

                            }                                              /* k=0 4  */
                        }                                                  /* i1 to i2
*/
                    }                                                    /* x1 <
Xpstat[0] etc */
                }                                                        /* if non vert
ray */
            }                                                          /* if thickness
!= 0 */
          }                                                            /* n=0 nl */
        }                                                            /* nside  */


        /*
           DEPTH JACOBIANS          */

        for(nside =0 ; nside < 2 ; ++nside){     /* left and right branches
of crp */
            gama = -1.;

            for(n=0 ; n <= nl ; ++n){
              if(nside == 0) {
                  x1 = Xcc[n];
                  x2 = Xcc[n+1];
                  y1 = Ycc[n];
                  y2 = Ycc[n+1];
              } else {
                  x1 = Xcc[nn-n];
                  x2 = Xcc[nn-n-1];
                  y1 = Ycc[nn-n];
                  y2 = Ycc[nn-n-1];
              }
              dx = x2 - x1;
              dy = y2 - y1;
              ds = dx * dx + dy * dy;
              j1 = index_xcr(&x2,Tm->edge,0,Tm->nstat-1);
              if(ds > 0.1){
          /* if skip zero thickness layer    */
                  if(gama < 0.){
                    cost = dy / sqrt(ds);
/*
                    j1 = index_xcr(&x2,Tm->edge,0,Tm->nstat-1);
*/
                    gama = cost * Tm->Si[n+1][j1];
                  }
                  dgama = gama;
                  ithick = 1;
              /* find next layer which is not of zero thickness */
                  if(n  != nl ){
                    ithick=0;
                    if(nside == 0 ) {
```

```
                    k = n + 2;
                    isign = 1;
                  }else {
                    k = nn - n - 2;
                    isign = -1;
                  }
                  for(dh2=0., kk = n + 2 ; kk <= nl + 1 && ithick ==0 ; k
+= isign , ++kk){
                    x3 = Xcc[k];
                    y3 = Ycc[k];
                    dx1 = x3 - x2;
                    dy1 = y3 - y2;
                    dh2 = dx1 * dx1 + dy1 * dy1;
                    if(dh2 > 0.1){
                      ithick=1;
                      cost = dy1 / sqrt(dh2);
                      gama1 = cost * Tm->Si[kk][j1];
    /* vertical slowness of next nonzero thickness layer */
                      dgama = gama - gama1;
                    }
                  }
                }
                if( ithick == 1){
                  gama = gama1;
                  if(x2 < Tm->Xpstat[0])
                    A_Jac[n][0][1] += dgama;
                  else if (x2 > Tm->Xpstat[npsm1])
                    A_Jac[n][npsm1][1] += dgama;
                  else   {
                    i1 = index_spl(&x2,Tm->ldx,0,npsm11);
                    ityp=0;
                    if(i1 == 0     ) ityp=1;
                    if(i1 == npsm11) ityp=2;
                    hermitw(&Tm->Xpstat[i1],&x2,Wb,ityp);
                    for(k=0; k < 4 ; ++k){
                      ip = i1 + k - 1;
                      if(ip < 0    ) ip = 0;
                      if(ip > npsm1) ip = npsm1;
                      A_Jac[n][ip][1] += dgama * Wb[k];
                    }                                    /* k =0..3 */
                  }                                     /* x2 < Xpstat[0]  */
                }                                       /* if ithick == 1 */
              }                                        /* dx*dx+dy*dy < y2 */
            }                                          /* n <= nl */
          }                                           /* nside loop */
          /*
             CONVERT TO THICKNESS JACOBIANS          */
          for(np=0 ; np < Tm->npspline ; ++np){
              xp = Tm->Xpstat[np];
              j1 = index_xcr(&xp,Tm->edge,0,Tm->nstat-1);

              for(n=0 ; n <= nl ; ++n){
                for(n1=0 ; n1 < n ; ++n1){
                    if(Tm->Dypstat[n1][np] > 0.1)
    /* avoid distribution to zero thickness layer */
      /* distribute A_Jac of layer n to all  previous layers  */
                    A_Jac[n1][np][1] +=
                    A_Jac[n][np][1];
                }                                       /* n1 loop  */
              }                                         /* n loop */
              /*
```

```
            NORMALIZE THICKNESS JACOBIAN TO TIME UNITS      */
          for( n=0 ; n <= nl ; ++n){
            sl1 = Tm->Si[n+1][j1];  /* slowness from closest point */
            A_Jac[n][np][1] /= sl1 ;
            A_Jac[n][np][2] /= (sl1 * 2.f );
          }
      }

      /* np loop */

      /*      CONVERT TO THICKNESS AND VERTICAL TIME OR SLOWNESS AND
            VERTICAL TIME       */

      for(n=0 ; n <= nl; ++n){
          for(np=0 ; np < Tm->npspline ; ++np){
            if(Tm->Sdvordz[n] == 1 ){
                /*        thickness - t0      */
                a_j = A_Jac[n][np][0];
                A_Jac[n][np][0] = A_Jac[n][np][1];
                A_Jac[n][np][1] = a_j;
            }
            A_Jac[n][np][0] -= A_Jac[n][np][1];
            A_Jac[n][np][1] = 0.5 * A_Jac[n][np][1];
            if(n > 0){
/*
   ignore zero thickness layers above layer n,
   find the first one which is not of zero thickness
*/
                n1=n-1;
                while(n1 > 0){
                  if(Tm->Dypstat[n1][np] > 0.1) break;
                  --n1;
                }                                              /* n1 loop */
                /* extra protection for the case where first layer is also
of zero
                   thickness     */
                if(Tm->Dypstat[n1][np] > 0.1)
                  A_Jac[n1][np][1] -= A_Jac[n][np][1];
            }                                                  /* n > 0  */
          }                                                    /* np loop */
      }                                                       /* n loop */
    }

}
```

```
                while(n1 > 0){
                  if(Tm->Dypstat[n1][np] > 0.1) break;
                  --n1;
                }                                              /* n1 loop */
                /* extra protection for the case where first layer is also
of zero
                   thickness     */
                if(Tm->Dypstat[n1][np] > 0.1)
                  A_Jac[n1][np][1] -= A_Jac[n][np][1];
```

```
            }                                                   /* n > 0


#define ffabs(A) ( A > 0 ? A:-A)
#include "tomo.h"
#include <fcntl.h>
#include <stdio.h>
#include <math.h>
void autopick(Tm,nGate)
DTMG *Tm;
int nGate;
{
    int nl,m,ity,lm,nGateH,imxp,imxn;
    int np,ll,m2;
    float amaxp,amaxn,taup,taun,denomp,denomn,anump,anumn;
    float sump,sumn;

/*                      apply dc constraint    */
    for(nl=0, m2=0 ; nl < Tm->nSurf ; ++nl){
      for(ity=0 ; ity < 3 ; ++ity, ++m2){
          if(Tm->Flag_Update[nl][ity] == 1){
            for(lm=0 ; lm < nGate+1 ; ++ lm){
                for(sump=0., sumn=0. , np=0 ; np < Tm->npspline ; ++np){
                  m = ity + 3 *(np + nl * Tm->npspline);
                  sump += Tm->Rp[m][lm];
                  sumn += Tm->Rn[m][lm];
                  Tm->Rp[m][lm]=0.;
                  Tm->Rn[m][lm]=0.;
                }
                Tm->Rp[ity + 3 * Tm->npspline * nl][lm]   = sump;
                Tm->Rn[ity + 3 * Tm->npspline * nl][lm]   = sumn;
            }                                          /* lm loop    */
          }                                            /* wm if   */
      }                                               /* ity loop */
    }                                                 /* nl loop */
    nGateH=nGate/2;
    for(np=0 ; np < Tm->npar ; ++np){
      amaxp=ffabs(Tm->Rp[np][0]);
      amaxn=ffabs(Tm->Rn[np][0]);
      imxp=0.;
      imxn=0.;
      for(ll=0, lm=-nGateH ; lm <= nGateH ; ++lm, ++ll){
          if(ffabs(Tm->Rp[np][ll]) > amaxp){
            amaxp = ffabs(Tm->Rp[np][ll]);
            imxp=ll;
          }
          if(ffabs(Tm->Rn[np][ll]) > amaxn){
            amaxn = ffabs(Tm->Rn[np][ll]);
            imxn=ll;
          }
      }
      if(amaxp < 0.1e-4 * ffabs( Tm->Rp[np][nGate])){
          taup = 0.;
      } else {
          if(imxp == 0) taup = -nGateH+0.5;
          else if(imxp == nGate-1) taup = nGateH-0.5;
          else {
            denomp = Tm->Rp[np][imxp+1] -2.*Tm->Rp[np][imxp] +
              Tm->Rp[np][imxp-1];
            anump = Tm->Rp[np][imxp+1]-Tm->Rp[np][imxp-1];
            taup =  imxp-nGateH - 0.5*anump/denomp;
```

```
            }
        }
        if(amaxn < 0.1e-4 * ffabs( Tm->Rn[np][nGate])){
            taun = 0.;
        } else {
            if(imxn == 0) taun = -nGateH+0.5;
            else if(imxn == nGate-1) taun = nGateH-0.5;
            else {
              denomn = Tm->Rn[np][imxn+1] -2.*Tm->Rn[np][imxn] +
                Tm*>Rn[np][imxn-1];
              anumn = Tm->Rn[np][imxn+1]-Tm->Rn[np][imxn-1];
              taun =  imxn-nGateH - 0.5*anumn/denomn;
            }
        }
        Tm->R[np] = Tm->Sv[np] * (taup * Tm->Rp[np][nGate] +
                              taun*Tm->Rn[np][nGate]);
    }
}


][imxp-1];
              taup =  imxp-nGateH - 0.5*anump/denomp;
            }
        }
        if(amaxn < 0.1e-4 * ffabs( Tm->Rn[np][nGate])){
            taun = 0.;
        } else {
            if(imxn == 0) taun = -nGateH+0.5;
            else if(imxn == nGate-1) taun = nGateH-0.5;
            else {
              denomn = Tm->Rn[np][imxn+1] -2.*Tm->Rn[np][imxn] +
                Tm->Rn[np][imxn-1];
              anumn = Tm->Rn[np][imxn+1]-Tm

void memopen3(a,n1,n2,n3)
float ***a;
int n1,n2,n3;
{
  int k1,k2,k3;
  for(k1=0; k1 < n1; ++k1){
    a[k1] = (float**) calloc(n2,sizeof(float*));
    for(k2=0; k2 < n2; ++k2)a[k1][k2] = (float*)calloc(n3,sizeof(float));
  }
}
amaxn < 0.1e-4 * ffabs( Tm->Rn[np][nGate])){
            taun = 0.;
        } else {
            if(imxn == 0) taun = -nGateH+0.5;
            else if(imxn == nGate-1) taun = nGateH-0.5;
            else {
              denomn = Tm->Rn[np][imxn+1] -2.*Tm->Rn[np][imxn] +
                Tm->Rn[np][imxn-1];
              anumn = Tm->Rn[np][imxn+1]-Tm


#include "tomo.h"

void zv_update(Tm)
DTMG *Tm;
{
```

```
    extern void hermitw();
    int np,nl,ip,i1,k,mm,j,npp,iflag,ityp, ns, j1;
    int nSurf,npspline,nstat,npar,*Sdvordz;
    float *Pp,*Xcoords,*Xpstat,f,tm1,dt;
    float dz,sl1,pp1,pp2,del,told,zold,t0,Wb[4];
/* --------------local variables -------*/

    dt = Tm->dt;
    del = Tm->del;
    Sdvordz = Tm->Sdvordz;
    npspline = Tm->npspline;
    nSurf = Tm-> nSurf ;
    nstat = Tm->nstat;
    npar = Tm->npar;
    Pp = Tm->Pp;
    Xcoords = Tm->Xcoords;
    Xpstat = Tm->Xpstat;

/* ------------------------------------*/

/*  multiply updates by variances and convert from sample units to actual
time units */

    for(np=0 ; np < npar ; ++np) Pp[np] *= Tm->Sv[np] * dt;
    for(np = 0; np < npspline; ++np){
      float xp = Tm->Xpstat[np];
      int n;
      j1 = index_xcr(&xp, Tm->edge, 0, Tm->nstat-1);
      for( n=0 ; n < nSurf ; ++n){
          float Sl1;
          Sl1 = Tm->Si[n+1][j1];
          Pp[n * 3 * npspline + np * 3 + 2] /= (dt *  Sl1 * 2.f * Tm-
>Dypstat[n][np]);
      }
    }
    for(nl = 0; nl < nSurf; ++nl)    Tm->Delta[nl] +=  Pp[nl * 3 * npspline
+ 2];

    for(nl = 0; nl < nSurf; ++nl)printf("nl= %d delta= %f \n", nl, Tm-
>Delta[nl]);
    for(np=0 ; np < npspline ; ++np){
      j = Tm->edge + Tm->Xpstat[np];
      for(nl=0 ; nl < nSurf; ++nl){
          mm = 3 * (np + npspline * nl ) ;
/*    Convert slowness (or thickness) updates from time units back
      into slowness (or distance) units     */
          if(Sdvordz[nl] ==0){ /* change slowness updates from time units
                        to inverse velocity units by
                        dividing by initial thickness*/
            if(Tm->Flag_Update[nl][0] == 0 && Tm->Dypstat[nl][np] > 0.1e-
2*del)
              f=Tm->Dypstat[nl][np];
            else f=Tm->Z[nl];    /* static constraint slowness normalized
                              by average thickness     */
            Pp[mm] *= del/f;
          }else {  /* change thickness updates from time units
                  to distance units by multiplying by
                  initial thickness*/
            Pp[mm] *=  Tm->Vi[nl+1][j];
          }
      }
```

```
    }
    /* In case of zero thickness layer set t0 update to that of the
previous layer
and zero slowness or thickness updates       */
    for(nl=1; nl < Tm->nSurf; ++nl){
      for(np=0; np < Tm->npspline; ++ np){
          if(Tm->Dypstat[nl][np] < 0.1e-2*del){
            Tm->Pp[np * 3 + 3 * nl*Tm->npspline] = 0.;
            Tm->Pp[1 + np * 3+ 3*nl*Tm->npspline] = Tm->Pp[1+np* 3 + 3
*(nl-1)*Tm->npspline];
          }
      }
    }
/* generate new Tm->ZV model    */
    for(ns=0; ns < nstat ; ++ns){

      if( Xcoords[ns] > Xpstat[0] && Xcoords[ns] < Xpstat[npspline-1]){
          i1 = Xcoords[ns]/Tm->ldx;
          ityp=0;
          if(i1 == 0 ) ityp=1;
          if(i1 == npspline-1) ityp=2;
          hermitw(Xpstat+i1,&Xcoords[ns],Wb,ityp);
      }
      zold = Tm->Zi[0][ns];
      for(nl=0, told=tm1=0. ; nl < nSurf ; ++nl){
          told += 2.*(Tm->Zi[nl+1][ns] - zold)/Tm->Vi[nl+1][ns];
          if( Xcoords[ns] <= Xpstat[0]) {

/* off beginning of line use updates of first spline point */
            npp=0;
            pp1 = Pp[nl * 3*npspline];
            pp2 = Pp[nl * 3*npspline+1];
/* off end of line use updates of last spline point   */
          } else if(Xcoords[ns] >= Xpstat[npspline-1]){
            npp = npspline -1;
            pp1 = Pp[3 *(nl*npspline + npp)];
            pp2 = Pp[3 *(nl*npspline + npp) + 1];
          } else {

            for(pp1=0., pp2=0., k=0 ; k<4; ++k){
                ip = i1 + k -1;
                if( ip < 0) ip=0;
                if(ip > npspline-1) ip=npspline-1;
                pp1 += Wb[k] * Pp[ 3 *(ip + nl * npspline)];
                pp2 += Wb[k] * Pp[ 3 *(ip + nl * npspline)+1];
            }
          }
          t0 = told + pp2;
          if(Sdvordz[nl] == 0) {                  /* slowness -t0   */
            sl1 = 1. / Tm->Vi[nl+1][ns] + pp1;
            Tm->Vi[nl+1][ns] = 1./sl1;
            zold = Tm->Zi[nl+1][ns];
            Tm->Zi[nl+1][ns] = Tm->Zi[nl][ns] + 0.5 * (t0-tm1) *Tm-
>Vi[nl+1][ns];
          } else {
/* thickness -t0   */
            dz = (Tm->Zi[nl+1][ns] - zold);
            zold = Tm->Zi[nl+1][ns];
            Tm->Zi[nl+1][ns] = Tm->Zi[nl][ns] + dz + pp1;
            if(t0 -tm1 > 0.1e-1 * t0 )
              Tm->Vi[nl+1][ns] = (Tm->Zi[nl+1][ns] - Tm->Zi[nl][ns]) * 2 /
```

```
                (t0-tm1);
            }
            tm1=t0;
        }
    }
}
 if(Sdvordz[nl] == 0) {                    /* slowness -t0    */
            sl1 = 1. / Tm->Vi[nl+1][ns] + pp1;
            Tm->Vi[nl+1][ns] = 1./sl1;


#define ffabs(A) ( A > 0 ? A:-A)
#include "tomo.h"
#include <fcntl.h>
#include <stdio.h>
#include <math.h>
void onesmth(Tm)
DTMG *Tm;
{
    int m1,m,np1,np2,np,ity,nl;
    float amps;

/*      add one on diagonal of AtA and add (-1 2 -1 ) smoother    */
    for(np1=0 ; np1 < Tm->npar  ; ++np1){
      if(Tm->Sv[np1] != 0.) Tm->AtA[np1][np1] += 1.;
      for(np2=0 ; np2 < np1 ; ++np2)
        Tm->AtA[np2][np1] = Tm->AtA[np1][np2];
    }
/* add smoother     */
    amps = 1.;    /* smoother amplitude    */
    for(m=0, nl=0; nl < Tm->nSurf ; ++nl){
      for(np=0 ; np < Tm->npspline ; ++np){
          for(ity=0 ; ity < 3 ; ++ity, ++m){
            if(Tm->Sv[m] != 0. && Tm->Flag_Update[nl][ity] == 0) {
                Tm->AtA[m][m] += 2.*amps;
                if(np > 0) {
                  if(Tm->Sv[m-3] != 0.){
                      Tm->AtA[m-3][m] -= amps;
                      Tm->AtA[m][m-3] -= amps;
                  }
                }
            }
          }
      }
    }
}
 += 1.;
      for(np2=0 ; np2 < np1 ; ++np2)
        Tm->AtA[np2][np1] = Tm->AtA[np1][np2];
    }
/* add smoother     */
    amps = 1.;    /* smoother amplitude    */
    for(m=0, nl=0hermitw.c

void hermitw(Xs,X,Wt,ityp)
float *Xs,*X,*Wt;
int ityp;
{
    float xx1sq,xx1,xx0sq, xx0,x2x0_inv,x2x0,x2x1_inv,x2x1,x1xm1_inv;
    float x1xm1,x1x0_invsq;
    float  x1x0_inv,x1x0,x0xm1_inv,x0xm1,wt2,wtm1;
```

```
    float x;
    float delta, delm1, delm12, delt2;
    x = *X;

    if(x < Xs[0]){
/* case when x is before first spline */
      Wt[0]=0.;
      Wt[1]=1.;
      Wt[2]=0.;
      Wt[3]=0.;
    } else if(x > Xs[1]){
/* case when x greater than third spline point */
      Wt[0] = 0.;
      Wt[1] = 0.;
      Wt[2] = 1.;
      Wt[3] = 0.;
    } else {
      delta = (x - Xs[0]) / (Xs[1] - Xs[0]);
      delt2 = delta * delta;
      delm1 = delta - 1.;
      delm12 = delm1 * delm1;
      Wt[0] = -0.5 * delta * delm12;
      Wt[1] = (1. + 2. * delta) * delm12 - 0.5 * delt2 * delm1;
      if(ityp ==1){
      Wt[1] += Wt[0];
      Wt[0] = 0.;
      }
      Wt[2] = (1. - 2. * delm1) * delt2 + 0.5 * delta * delm12;
      Wt[3] = 0.5 * delm1 * delt2;
      if(ityp == 2){
      Wt[2] += Wt[3];
      Wt[3] = 0.;
      }
    }
  }



      Wt[1] = 0.;
      Wt[2] = 1.;
      Wt[3] = 0.;
    }

#define ffabs(A) ( A > 0 ? A:-A)
#include "tomo.h"
#include <fcntl.h>
#include <stdio.h>
#include <math.h>
int newr(Tm)
/*  shift crosscorrelations according to time shifts resulting from last
    update and calculate new residuals                                    */
DTMG *Tm;
{
    extern void residual_T();
    extern void residual_C();
    extern void autopick();
    int nGate,nGateW,igate1,lm,lmm,np;
    int m,ity,sf,icrp,nof,iof;
    int m1,m2,n,n1,ity1,n2,ity2,mm;
    float sdd;
    float shift;
```

```
    int iErr , iCount , init_sf, iunit;
    int npspline,nSurf,npar;
    float *Trace,*Aux;

/*    open_matrixZV();    */
    iunit = open("fort.22",O_RDONLY, 0);

    nGate   = Tm->nGate;
    nGateW = Tm->nGateW;
    igate1 = (nGateW - nGate) / 2;

    Trace = (float*)calloc(nGateW+3 , sizeof(float));
    Aux   = (float*)calloc(nGateW+3 , sizeof(float));

    npar = Tm->npar;
    npspline = Tm->npspline;
    nSurf = Tm->nSurf;

/*  Tm->Atav used as temporary storage of size npar          */


    memset(Tm->R,0,Tm->npar*sizeof(float));
    if(Tm->flag_use_pan){
      for(np=0; np < Tm->npar ; ++np){
          memset(Tm->Rp[np],0,(nGate+1)*sizeof(float));
          memset(Tm->Rn[np],0,(nGate+1)*sizeof(float));
      }
    }

/*     redistribute after static constraint   */
    for(sf=0; sf < nSurf ; ++sf){
      for(ity=0; ity < 3 ; ++ity){
          if(Tm->Flag_Update[sf][ity] == 1){
            for(np=1; np < npspline; ++ np){
                mm = ity + 3 * (np + sf * npspline);
                Tm->P[mm] = Tm->P[ity + 3 * sf *npspline];
            }
          }
      }
    }

    for ( sf=0; sf < Tm->nSurf  ; ++sf){
      if(!Tm->Flag_Use_Surf[sf])continue;
      sdd = Tm->Var_Da[sf];

      iCount = 0;
      init_sf = 1;
      for(icrp=0;icrp < Tm->nCrp; icrp+=Tm->incstat){
          if(!Tm->Flag_Use_Surf_Crp[sf][icrp]){
            iCount ++;
            continue;
          }

          memset(Tm->A_Jac[0][0][0], 0, Tm->npar * Tm->nOffs *
sizeof(float));
          for( iof=0 ; iof < Tm->Noffs_Surf_Crp[sf][icrp]; ++iof){

            nof = Tm->Ioffs_Surf_Crp[sf][icrp][iof];

/*          read_matrixZV(Tm->Atav,Tm->npspline * 2 * (sf+1) );   */
            for(n=0; n <= sf; ++n){
```

```
                if(Tm->Imin[icrp][sf][n] <= Tm->Imax[icrp][sf][n])
read(iunit,
                                        Tm->A_Jac[nof][n][Tm-
>Imin[icrp][sf][n]],
                    3 * (Tm->Imax[icrp][sf][n]- Tm->Imin[icrp][sf][n] + 1) *
sizeof(float));
                }
                for(n=0, mm=0, shift=0. ; n <= sf; ++n){
                    for(np=0; np < Tm->npspline; ++np){
                     for(ity=0; ity < 3; ++ity, ++mm){
                          shift += Tm->A_Jac[nof][n][np][ity] *
                                   Tm->P[mm] * Tm->Sv[mm];
                     }
                    }
                }

                Tm->Delays[sf][icrp][nof] += shift;
                Tm->Shifts[nof] = shift;
              }

              if(!Tm->flag_use_pan)continue;

              if(Tm->flag_delay_hyp){
/*              iErr = read_tomo_pan_surf_crp(sf,icrp,init_sf);  */
                if(iErr == 1){
                    return(1);
                }
                init_sf = 0;
                if(iErr == 2){
                    continue;
                }
                iCount ++;
              }

              for( iof=0 ; iof < Tm->Noffs_Surf_Crp[sf][icrp]; ++iof){

                nof = Tm->Ioffs_Surf_Crp[sf][icrp][iof];
                memcpy(Trace , Tm->Tcor[sf][icrp][nof],
                       nGateW * sizeof(float));

                if(!Tm->flag_delay_hyp){
                    shift_trace(Trace, Aux, &Tm->Shifts[nof], nGateW);
                    memcpy(Tm->Tcor[sf][icrp][nof],Trace,
                         nGateW*sizeof(float));

                }

                residual_C(Tm , Tm->A_Jac[nof] ,Trace+igate1,&sdd,nGate);

              }
            }
          }

          if(Tm->flag_use_pan)autopick(Tm,nGate);

          Tm->flag_delay_hyp = 0;

          for(np=0; np < Tm->npar; ++np){
            Tm->Pp[np] += Tm->P[np];
            Tm->P[np] = 0.;
            Tm->V[np] = 0.;
```

```
    }

    return(0);

}


Trace , Tm->Tcor[sf][icrp][nof],
                    nGateW * sizeof(float));

            if(!Tm->flag_delay_hyp){
                shift_trace(Trace, Aux, &Tm


typedef  struct {
  float *xpstat;      /*coords of parameter points        */
  float **dypstat;    /* thicknesses at stations
                   size [itface][nstat]                  */
  float **ata;        /* ata [npar][npar]                 */
  float ***a_jac;     /*size [itface][npspline][2]        */
  float ****a_jac1;   /*size [nofs][itface][npspline][2] */
  float *r;           /*size npar                         */
  float *p;           /*size npar                         */
  float *pp;          /*size npar                         */
  float *v;           /*size npar                         */
  float *atav;        /*size npar                         */
  float **rp;         /* rp[npar][limgate]                */
  float **rn;         /* rn[npar][limgate]                */
  float *sv;          /* sv[npar]                         */
  float *svv;         /* size 2*itface                    */
  float ****tcor;     /* [nevent][nstat][limgate]         */

  int   *iwm;         /* size 2*itface                    */
  int *imin;          /* size itface                      */
  int *imax;          /* size itface                      */
  int *sdvordz;       /* size itface                      */
  float *z;           /* size itface                      */
  int npspline;
  int ldx;
  int limgate;
  int npar;
  int itface;
  int ievent;
  float del;          /* scale factor for normalized coordinates  */
  float *sl;          /*      slownesses at station locations     */
  float *xcc;
  float *ycc;
  float *xcyc;
  int nofs;           /*   number of offsets                   */
  int *nofsnl;        /*   number of offsets per layer         */
  float *offset;
  int nstat;
  int jstat;
  int kstat;
  float *xcoords;
  int nray;
  int incstat;
  int istr;
  float *sdd;
  } DTMG;
```

```
line;
  int ldx;
  int limgate;
  int npar;
  int itface;
  int ievent;
  float del;          /* scale factor for normalized coordinates  */
  float *sl;         /*      slownesses at station locations     */
  float *xcc;
  float *ycc;
  float *xcyc;
  int nofs;          /*   number of


typedef  struct {
    int lineName;
    float *SurfOrder;
    int c1,sc;

    float **Zi;
    float **Vi;
    float **Si;
    float *Xpstat;   /*coords of parameter points */
    float **Dypstat; /* thicknesses at parameter points
                    size [nSurf][npar]      */

    float *Var_Sl;
    float *Var_Tv;
    float *Var_Da;
    float *Var_Del;
    float *Delta;    /* anisotropic parameter  */

    int **Flag_Update; /* 0 - varying   1 - constant */
    int *Sdvordz;
    float *SembMin;

    float **AtA;     /* ata [npar][npar]           */
    float ****A_Jac;  /*size [nof][np][npspline][2]     */
    float *R;        /*size npar                   */
    float *P;        /*size npar                   */
    float *Pp;       /*size npar                   */
    float *V;        /*size npar                   */
    float *Atav;     /*size npar                   */
    float **Rp;      /* rp[npar][nGate]          */
    float **Rn;      /* rn[npar][nGate]          */
    float *Sv;       /* sv[npar]                   */

    float ****Tcor;  /* [surf][nstat][noffs][nGate]  X-Corr traces*/

    float ***Delays; /* [surf][nstat][noffs] */
    float ***Delays0; /* [surf][nstat][noffs] */
    float *Shifts;    /* [noffs] */

    float **Delays_hyp , **T0 , ref_off;

    int **Noffs_Surf_Crp;
    int ***Ioffs_Surf_Crp;
    int *Flag_Use_Surf;
    int **Flag_Use_Surf_Crp;
```

```
    float *Z;         /* size nSurf                */
    int npspline;
    int ldx;
    int nGateT,nGateW,nGate;    /* 15 11 5 */
    int npar;
    int nSurf;       /* no. of surfaces */
    int ***Imin;       /* size nstat,itface,nl          */
    int ***Imax;       /* size nstat,itface,nl          */

    float del;       /* scale factor for normalized coordinates  */
    float *Xcc;
    float *Ycc;
    int nOffs;         /*       number of offsets          */
    int *Noffs;        /*       number of offsets per layer */
    float *Offs;
    int nstat,nCrp;  /* nstat = 100 + nCrp + 100 */
    int icrp1,icrp2;
    int edge;
    float *Xcoords;
    int nray;
    int incstat;
    int istr;

    float dx,dt;
    int itrmax;

    int flag_Domain;
    int flag_delay_hyp;
    int flag_delay_nonhyp;
    int flag_use_pan;

} DTMG;
```

PROJE ÖZET BİLGİ FORMU

Proje Kodu: YDABÇAG-100Y105

Proje Başlığı: TRANSVERSELY İZOTROPİK ORTAMDA DERİNLİK MİGRASYONU UYGULANMIŞ CRP GRUPLARININ TOMOGRAFİSİ İLE HIZ, DERİNLİK VE ANİZOTROPİ PARAMETRELERİNİN BELİRLENMESİ

Proje Yürütücüsü ve Yardımcı Araştırmacılar: Yrd. Doç. Dr. Selma KADIOĞLU

Projenin Yürütüldüğü Kuruluş ve Adresi: ANKARA ÜNİVERSİTESİ

Ankara Üniversitesi, Mühendislik Fakültesi, Jeofizik Mühendisliği Bölümü
06100 Tandoğan-ANKARA

Destekleyen Kuruluş(ların) Adı ve Adresi: TÜRKİYE BİLİMSEL TEKNİK VE ARAŞTIRMA KURUMU (TÜBİTAK) YER DENİZ VE ATMOSFER BİLİMLERİ ARAŞTIRMA GRUBU

Atatürk Bulvarı No:221 Kavaklıdere, 06100 ANKARA

Projenin Başlangıç ve Bitiş Tarihleri: 03.04.2001-03.04.2003

Öz (en çok 70 kelime)

Bu pojede tabaka hızlarının, arayüzey derinliklerinin ve arayüzeylere ait Thomsen's δ anizotropi parametresinin belirlenmesi için, iki boyutlu düşey simetri eksenli enine (transversely) izotropik ortamda (VTI ortam), yığma öncesi derinlik migrasyonu uygulanmış ortak yansıma noktası gruplarının (CRP gathers) tomografisine dayanan bir yöntem sunulmaktadır. Yöntem, VTI ortam için meteryal parametrelerindeki ve arayüzey derinliklerindeki saçılmalardan dolayı bir CRP ışın çifti boyunca seyahat zamanındaki değişiklikleri tanımlamayı sağlayan tomografik prensipleri kullanır. Özel ayrıklaştırma işleminden sonra tomografi denklemleri elde edilir. Yeni parametre değerleri için denklemler sönümlü enküçük kareler yöntemi ile çözülür.
Geliştirilen tomografi programı birkaç teorik model üzerinde test edilerek gerçek yeraltı parametrelerinin yeniden elde edilebilirliği gösterildi.

Anahtar Kelimeler: Anizotropi, Enine İzotropi, Derinlik Migrasyonu, CRP Grupları, Hız, Derinlik, Thomsen's δ anizotropi parametresi, Tomografi

Projeden Kaynaklanan Yayınlar:

KADIOĞLU, S. 2002, Enine Yönbağımsız Ortamda Dalga Yayılımı, Jeofizik, Vol. 15, No. 2, S. 83-94.

KADIOĞLU, S., GÜRPINAR, S. and GÜRELI, O. 2002, Beşinci Derece Cash-Karp Runge-Kutta, Yöntemi ile Dalga Cephesi Oluşturarak Seyahat Zamanı Hesaplaması, Jeofizik Vol. 15, No.2, S. 95-104.

KADIOĞLU, S. **2003**, Modeling by Ray Tracing Method in Polar Anisotropic Medium, 14th International Petroleum Congress and Natural Gas Congress and Exhibition of Turkey, May, 12-14, Ankara-TURKEY, Proceedings, p.539-540.

Bilim Dalı: JEOFİZİK MÜHENDİSLİĞİ

Doçentlik B. Dalı Kodu: 920