รายงานวิจัยฉบับสมบูรณ์

โครงการ: การแก้ปัญหาย้อนกลับแบบผสมระหว่าง CG และ Occam สำหรับข้อมูล Magnetotelluric (A Hybrid CG/Occam Inversion for Magnetotelluric Data)

รศ. ดร. วีระชัย สิริพันธ์วราภรณ์

ภาควิชาฟิสิกส์ คณะวิทยาศาสตร์ มหาวิทยาลัยมหิดล

มิถุนายน 2553

สัญญาเลขที่ **RMU5080025**

รายงานวิจัยฉบับสมบูรณ์

โครงการ: การแก้ปัญหาย้อนกลับแบบผสมระหว่าง CG และ Occam สำหรับข้อมูล Magnetotelluric (A Hybrid CG/Occam Inversion for Magnetotelluric Data)

รศ. ดร. วีระชัย สิริพันธ์วราภรณ์

ภาควิชาฟิสิกส์ คณะวิทยาศาสตร์ มหาวิทยาลัยมหิดล

สนับสนุนโดยสำนักงานกองทุนสนับสนุนการวิจัย

(ความเห็นในรายงานนี้เป็นของผู้วิจัย สกว. ไม่จำเป็นต้องเห็นด้วยเสมอไป)

บทคัดย่อ

ในโครงการนี้ เราเสนอโปรแกรมใหม่ที่ใช้ในการแปลความหมายข้อมูล Magnetotelluric โปรแกรมนี้เรียกว่า WSMIX3DMT เป็นโปรแกรมที่มีพื้นฐานความคิดมาจากสองโปรแกรมเก่า คือ data space conjugate gradient (WSDCG3DMT) และ data space Occam's inversion (WSINV3DMT) โปรแกรม WSMIX3DMT นี้เป็นโปรแกรมที่ดัดแปลงทางคณิตศาสตร์มาจากโปรแกรม WSDCG3DMT ดังนั้นหน่วยความจำที่ต้องใช้จึงเล็กน้อยเมื่อเทียบกับ WSINV3DMT เหมือนกับของ WSDCG3DMT แต่แทนที่เราจะใช้ trade-off parameter ตัวเดิมตลอด inversion เราเปลี่ยนแปลงไปเรื่อยๆ เหมือนกับที่ทำใน WSINV3DMT แต่การเปลี่ยนแปลงนี้เป็นไปตาม run-time ไม่ได้เป็นไปตาม data misfit กระบวนการที่เราใช้นี้ทำให้โปรแกรม WSMIX3DMT รันได้เร็วกว่าทั้ง WSDCG3DMT และ WSINV3DMT และในขณะเดียวกันก็ใช้หน่วยความจำน้อยกว่า สิ่งนี้ทำให้ WSMIX3DMT เป็นโปรแกรม inversion ที่มีประสิทธิภาพสูงที่สุด โปรแกรมนี้ได้ถูกทดสอบและเปรียบเทียบกับโปรแกรมเก่าทั้งจากข้อมูลเทียมและข้อมูลจริง

Abstract

In this project, we create a new inversion scheme (WSMIX3DMT) based a mixed of the data space conjugate gradient (WSDCG3DMT) and the data space Occam's inversion (WSINV3DMT) methods. WSMIX3DMT is mathematically a slight modification of WSDCG3DMT, its memory requirement is therefore a fraction of WSINV3DMT as in WSDCG3DMT. Instead of fixing the trade-off parameter, it is varied similar to WSINV3DMT. However, the variation is according to the run-time, not based on the data misfit. This strategy makes WSMIX3DMT faster than both WSDCG3DMT and WSINV3DMT, and at the same time requires least memory. This makes WSMIX3DMT as the most efficient inversions. Computational performances and comparisons of all three methods are demonstrated with both synthetic and field datasets.

กิตติกรรมประกาศ

ข้าพเจ้าขอขอบคุณ สกว. ที่ให้โอกาสข้าพเจ้าได้ทำวิจัยในเรื่องที่ข้าพเจ้าถนัดต่อเนื่องมาตลอดตั้งแต่จบปริญญาเอก ทุนวิจัยสกว. นี้ทำให้ข้าพเจ้าสามารถใช้เวลาทำงานวิจัยได้อย่างเต็มที่และทำงานวิจัยที่มีคุณภาพเป็นที่ยอมรับในกลุ่มคนทำวิจัยเรื่องเดียวกัน ซึ่งวัดได้จากการอ้างอิงใน SCOPUS หรือ ISI database รวมทั้งการที่ข้าพเจ้าได้รับเชิญไปบรรยายในการประชุมต่างประเทศหลายๆ แห่ง ทุนวิจัยนี้ยังสามารถทำให้ข้าพเจ้าได้ใช้เวลาเต็มที่ในการเตรียมบุคลากรและนักศึกษาในการทำวิจัยในสาขาธรณีฟิสิกส์ในอนาคตเพื่อเป็นประโยชน์ต่อประเทศ ซึ่งผลงานที่เกิดกับนักศึกษาก็จะมีปรากฏออกมาเรื่อยๆ

นอกจากนี้ ข้าพเจ้าต้องขอขอบคุณ Prof. Dr. Gary Egbert จาก Oregon State Universiy, Associate Professor Dr. Makoto Uyeshima และ Professor Dr. Hisashi Utada จาก Earthquake Research Institute (ERI), University of Tokyo และ Professor Dr. Yasuo Ogawa จาก Tokyo Institute of Technology ที่สนับสนุนงานวิจัยข้าพเจ้ามาโดยตลอด และทั้งนี้ก็ต้องไม่ลืมที่จะขอบคุณ ผศ. ดร. ศรีสุดา วรามิตร หัวหน้าภาควิชาฟิสิกส์และผู้ร่วมงานคนอื่นๆ ในภาควิชาฟิสิกส์ที่ช่วยสนับสนุนข้าพเจ้ามาโดยตลอด รวมทั้งนักศึกษาในกลุ่มวิจัยธรณีฟิสิกส์ มหาวิทยาลัยมหิดล ที่ร่วมกันฝ่าฟันอุปสรรคต่างๆ เพื่อความเป็นเลิศในงานวิจัยด้านนี้ นอกจากนี้แล้วยังมีเพื่อนๆ จากภาควิชาอื่นๆ อีกด้วย รวมทั้งท่านคณบดีและทีมงานคณะวิทยาศาสตร์ มหาวิทยาลัยมหิดล ที่คอยกระตุ้นและสนับสนุนงานวิจัยมาตลอด

อีกกลุ่มหนึ่งที่ข้าพเจ้าจะไม่ขอบคุณไม่ได้ คือ บุคลากรของสกว. ฝ่ายวิชาการที่เป็นมิตรอันดีและคอยช่วยเหลือในเรื่องต่างๆ เป็นอย่างดี

สุดท้ายนี้ ข้าพเจ้าต้องขอขอบคุณบุคคลที่คอยเป็นกำลังใจ เข้าใจในทุกสิ่งทุกอย่าง บุคคลเหล่านี้คือคุณแม่ที่เพิ่งล่วงลับไปและครอบครัวพี่น้องของข้าพเจ้า ขอบคุณมากครับ

รศ. ดร. วีระชัย สิริพันธ์วราภรณ์

เนื้อหางานวิจัย

บทนำ

Magnetotelluric เป็นเทคนิคหนึ่งทางธรณีฟิสิกส์ เทคนิคนี้เริ่มต้นจากการวัดสนามแม่เหล็กและสนามไฟฟ้าที่บริเวณพื้นผิวของโลก อัตราส่วนของสนามแม่เหล็กและสนามไฟฟ้าสามารถนำมาใช้เป็นตัวบ่งบอกถึงสภาพความต้านทานไฟฟ้า (electrical resistivity) หรือ ความสามารถในการนำไฟฟ้า (electrical conductivity) ภายใต้พื้นโลกที่ความลึกต่างๆ ได้ เราสามารถนำข้อมูล electrical resistivity นี้ไปใช้ในการอธิบายโครงสร้างของโลกเพื่ออธิบายการเกิดแผ่นดินไหว (Siripunvaraporn et al., 1998; Unsworth et al., 2000; Boonchaisuk et al., 2010) การศึกษาเทคโทนิคของพื้นที่ (Jones, 1992) หรือใช้ในการสำรวจหาทรัพยากรธรรมชาติ (Tuncer et al., 2006; Orange, 1989; Vozoff, 1972) และอื่นๆ

ข้อมูลคลื่นแม่เหล็กไฟฟ้าที่วัดได้มาจากการสำรวจแต่ละพื้นที่จะเป็นข้อมูลดิบที่ต้องนำมาผ่าน data processing เพื่อให้ได้ข้อมูลที่เรียกว่า apparent resistivity และ phase หรือ impedance tensor ซึ่งเป็นฟังก์ชันของความถี่หรือว่าคาบ เพื่อนำไปใช้ในการตีความหมายต่อไป การตีความหมายจากข้อมูลโดยตรงนั้นเป็นไปได้ยาก เนื่องจากข้อมูลที่ได้มาไม่ได้เป็นฟังก์ชันของความลึก ดังนั้น inversion หรือการแก้ปัญหาย้อนกลับ คือกระบวนการที่นำเอาค่า apparent resistivity และ phase ที่เป็นฟังก์ชันของความถี่หรือคาบ ไปแปลงให้เป็นค่า electrical resistivity กับความลึก โดยผ่านกระบวนการทางคณิตศาสตร์ที่สลับซับซ้อน ทั้งนี้ผลลัพธ์สุดท้ายที่ได้คือแบบจำลอง (model) สภาพความต้านทานไฟฟ้าแบบสามมิติ (3-D)

การพัฒนาโปรแกรม inversion สำหรับข้อมูล MT มีมาต่อเนื่องโดยผู้วิจัย เริ่มต้นจากการพัฒนา 2-D inversion (Siripunvaraporn and Egbert, 2000) โดยโปรแกรม 2-D นี้ที่มีชื่อว่า REBOCC มีนักวิจัยจากทั่วโลกนำไปใช้ในการแปลความหมายข้อมูลจริง (**SCOPUS: อ้างอิง 97 ครั้ง as of 24 June 2010**) เมื่อได้รับการสนับสนุนจากสกว. ผู้วิจัยก็ได้พัฒนาเป็นโปรแกรม 3-D (Siripunvaraporn et al., 2005) โดยมีชื่อโปรแกรมว่า WSINV3DMT ซึ่งสามารถทำงานได้แม้บนเครื่อง PC ธรรมดา ซึ่งถือว่าเป็นโปรแกรมแรกของโลกที่มีการ release สู่สาธารณะและได้มีการนำไปใช้จริง (**SCOPUS: อ้างอิง 33 ครั้ง as of 24 June 2010**) เทคนิคของ WSINV3DMT ได้ถูกนำไปประยุกต์ใช้กับข้อมูลประเภทอื่นๆ ด้วย เช่นข้อมูล Network-MT data (Siripunvaraporn et al., 2004) ข้อมูล 2-D DC Resistivity (Boonchaisuk et al., 2008) และข้อมูล Phase Tensor (Patro et al., 2010) เป็นต้น

อย่างไรก็ตาม แม้ว่า WSINV3DMT จะทำงานได้บนเครื่อง PC ทั่วไป แต่ก็ยังมีปัญหาอยู่ โดยปัญหาหลัก คือ ยังคงต้องใช้หน่วยความจำของเครื่องคอมพิวเตอร์ในปริมาณมากเมื่อใช้กับข้อมูลที่มีขนาดใหญ่ สิ่งนี้คือข้อจำกัดของตัวโปรแกรม วิธีแก้ไขก็คือการเพิ่มหน่วยความจำของคอมพิวเตอร์ให้มากที่สุดเท่าที่เครื่องจะรับได้ ซึ่งก็จะทำให้ต้นทุนการทำงานสูงหรือแพงมากขึ้น

ต่อมาผู้วิจัยก็ได้รับการสนับสนุนจากสกว. เพื่อแก้ไขข้อบกพร่องนี้โดยพัฒนาเป็น algorithm ใหม่ขึ้นมาเพื่อลดปริมาณหน่วยความจำนี้ วิธีหนึ่งที่เราใช้ก็คือ การแก้ระบบสมการด้วยวิธี conjugate gradient (CG) แทนที่จะแก้แบบโดยตรง คือใช้ Cholesky decomposition เหมือนที่ทำใน WSINV3DMT การใช้วิธี CG ทำให้เราไม่ต้องเก็บ sensitivity matrix (J) ซึ่งมีขนาดใหญ่ในหน่วยความจำของคอมพิวเตอร์ซึ่งทำให้เราลดปริมาณการใช้หน่วยความจำได้เป็นอย่างมาก โปรแกรมใหม่นี้เราเรียกว่า data space conjugate gradient method (DCG) หรือ WSDCG3DMT จากการทดลองของ Siripunvaraporn and Egbert (2007) และ Siripunvaraporn and Sarakorn (2010) สำหรับข้อมูล 2-D และ 3-D พบว่า ข้อดีของเทคนิคนี้ก็คือ ไม่ต้องใช้หน่วยความจำในปริมาณมากเหมือน WSINV3DMT แต่ทว่าข้อเสียของเทคนิคใหม่นี้อยู่ที่เวลาที่ใช้รันโปรแกรมนั้นมากกว่า ดังนั้นมันจึงเป็น trade-off ซึ่งกันและกันระหว่างเวลากับหน่วยความจำ

ในข้อเสนอโครงการนี้ เราเสนอที่จะผสมโปรแกรมทั้งสองอันไว้ด้วยกัน เพื่อคงไว้ในข้อดี นั่นคือใช้หน่วยความจำน้อย ในขณะเดียวกันก็ใช้เวลาในการรันน้อยด้วย ในรายงานฉบับนี้เราจะเริ่มต้นจากการบรรยายโปรแกรม WSINV3DMT แล้วตามด้วย WSDCG3DMT จากนั้นก็เสนอแนะเทคนิคใหม่ที่เรียกว่า WSMIX3DMT รวมทั้งผลที่ได้จากการรันโปรแกรม

Inversion : Overview

การทำ inversion คือการหาแบบจำลอง (**m**) ที่สามารถให้ค่า model responses **F[m]** ที่ fit ข้อมูล **d** ที่มีทั้งหมด *N* ค่าได้สมเหตุสมผล ซึ่งสามารถเขียนเป็นสมการคณิตศาสตร์ได้ดังนี้

$$U(\mathbf{m}, \lambda) = (\mathbf{m} - \mathbf{m_0})^T \mathbf{C_m}^{-1} (\mathbf{m} - \mathbf{m_0}) + \lambda^{-1}\{(\mathbf{d} - \mathbf{F}[\mathbf{m}])^T \mathbf{C_d}^{-1} (\mathbf{d} - \mathbf{F}[\mathbf{m}]) - X^2_*\} \qquad (1)$$

เมื่อ $\mathbf{C_d}$ คือ data covariance และ ^{T} คือ transpose of matrix, **m** คือ model ที่มีทั้งหมด *M* ค่า ส่วน $\mathbf{m_o}$ คือ base model และ $\mathbf{C_m}$ คือ model covariance และ λ^{-1} คือ Lagrange multiplier

สมการที่ (1) นี้มีความหมายว่าเรากำลังทำการ search หาแบบจำลอง (model) ที่มีลักษณะแบบ minimum structure โดยมีข้อแม้ว่าแบบจำลองที่ได้จะต้อง fit ข้อมูลได้เป็นอย่างดีซึ่งถูกกำหนดโดยค่า X^2_* การกำหนดในลักษณะนี้ทำให้ inversion นั้น stable มากขึ้น

การ minimize สมการนี้ คือ การคำนวณหา stationary point ของสมการที่ (1) นี้เมื่อเทียบกับ λ และ **m** ซึ่งคำนวณได้ยาก วิธีหนึ่งคือการแก้สมการ penalty functional แทน ซึ่งมีลักษณะดังนี้

$$\Phi^m_\lambda = (\mathbf{d} - \mathbf{F}[\mathbf{m}])^T \mathbf{C_d}^{-1} (\mathbf{d} - \mathbf{F}[\mathbf{m}]) + \lambda (\mathbf{m} - \mathbf{m_0})^T \mathbf{C_m}^{-1} (\mathbf{m} - \mathbf{m_0}), \qquad (2)$$

เนื่องจากเมื่อ λ นั้นคงที่ หรือ fixed ไว้ เราจะได้ว่า $\partial U/\partial \mathbf{m} = \partial \Phi_\lambda/\partial \mathbf{m}$ ดังนั้นเราสามารถแก้สมการ (2) แทนที่สมการที่ (1) ได้แต่ต้อง vary ค่า λ ไปเรื่อยๆเพื่อให้ได้ค่า misfit ที่น้อยที่สุดหรือตามที่ตั้งเอาไว้

สมการที่ (2) เป็นสมการใน model space ซึ่ง Siripunvaraporn et al. (2005) and Siripunvaraporn and Egbert (2000) แสดงให้เห็นว่าการแก้ปัญหาใน model space นั้นมีข้อเสียคือใช้เวลานานมากๆ และใช้หน่วยความจำสูงมากๆ Siripunvaraporn and Egbert (2000) and Siripunvaraporn et al. (2005) จึงเสนอให้แก้ปัญหาใน data space แทน

ดังนั้นขั้นตอนแรกคือการแปลงสมการที่ (2) จาก model space ให้อยู่ใน data space ซึ่งสามารถทำได้ดังนี้ โดยการเขียน แบบจำลอง **m** ให้เป็นฟังก์ชันของ sensitivity matrix ดังนี้ $\mathbf{m} - \mathbf{m_0} = \mathbf{C_m J}^T \boldsymbol{\beta}$ เมื่อ $\boldsymbol{\beta}$ คือ unknown expansion coefficient vector ดังนั้นสมการที่ (2) จะกลายเป็น

$$\Phi^d_\lambda = \lambda^{-1} (\mathbf{đ} - \mathbf{J C_m}^T \mathbf{J}^T \boldsymbol{\beta})^T \mathbf{C_d}^{-1} (\mathbf{đ} - \mathbf{J C_m}^T \mathbf{J}^T \boldsymbol{\beta}) + (\boldsymbol{\beta}^T \mathbf{J C_m}^T \mathbf{J}^T \boldsymbol{\beta}), \qquad (3)$$

เมื่อ $\mathbf{J} = [\partial \mathbf{F}/\partial \mathbf{m}]$ คือ *N* x *M* sensitivity matrix ซึ่งเป็นตัวอธิบายการเปลี่ยนแปลงของข้อมูลเนื่องจากการเปลี่ยนแปลงของ model และ $\mathbf{đ} = \mathbf{d} - \mathbf{F[m]} + \mathbf{J}(\mathbf{m} - \mathbf{m_0})$

เนื่องจาก **F[m]** นั้นเป็น non-linear problem ดังนั้น iterative solutions จึงจำเป็น (Constable et al., 1987) model response F[m] จึงจำเป็นต้องถูก linearized ก่อนโดยใช้ first order Taylor's series expansion,

$$\mathbf{F}[\mathbf{m}_{k+1}] = \mathbf{F}[\mathbf{m}_k] + \mathbf{J}_k(\mathbf{m}_{k+1} - \mathbf{m}_k), \qquad (4)$$

เมื่อ k คือ iteration number ในการหา stationary points ของ (3) เราทำได้โดยการ differentiate (3) with respect to $\boldsymbol{\beta}$ เราได้ว่าในแต่ละ iteration จะมี solution ดังนี้

$$\mathbf{m}_{k+1} - \mathbf{m}_0 = \mathbf{C_m J}_k^{\mathbf{T}} \mathbf{C_d}^{-½} [\lambda \mathbf{I} + \mathbf{C_d}^{-½} \mathbf{J}_k \mathbf{C_m J}_k^{\mathbf{T}} \mathbf{C_d}^{-½}]^{-1} \mathbf{C_d}^{-½} \mathbf{đ}_k, \qquad (5)$$

ข้อดีของการแก้สมการ (1) ใน data space ก็คือ matrix ที่ต้องทำการ invert มีขนาดเพียง *N x N* เท่านั้น ไม่ใช้ *M x M* เหมือนในกรณีของ model space เมื่อ *N* คือจำนวนข้อมูลและ *M* คือขนาดของแบบจำลอง สำหรับข้อมูลเพิ่มเติมศึกษาใน Siripunvaraporn and Egbert (2000) and Siripunvaraporn et al. (2005).

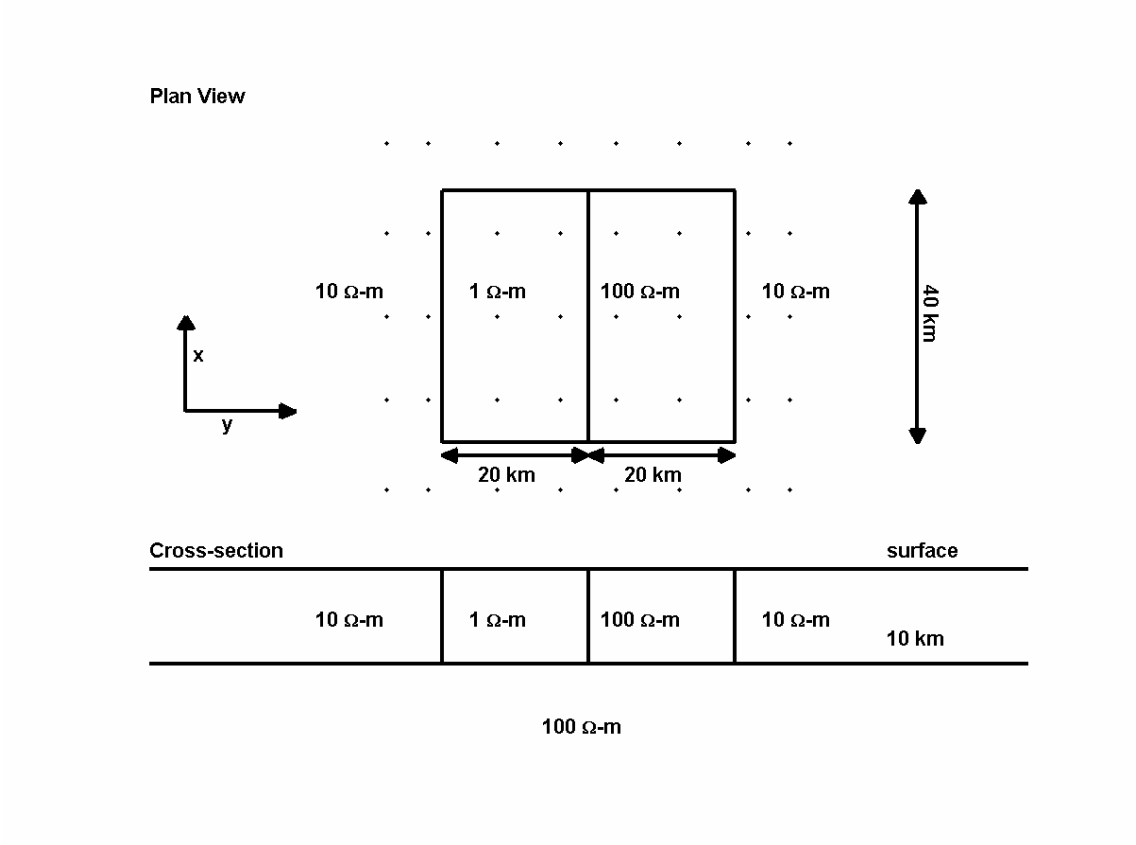
การแก้สมการที่ (5) สามารถทำได้สองวิธี วิธีแรกใช้ใน WSINV3DMT ส่วนวิธีที่สองถูกนำไปใช้ใน WSDCG3DMT

WSINV3DMT : Data Space Occam's Inversion

วิธีแรกคือสร้าง matrix **J** และ $\mathbf{R} = [\lambda \mathbf{I} + \mathbf{C_d}^{-½} \mathbf{J}_k \mathbf{C_m J}_k^{\mathbf{T}} \mathbf{C_d}^{-½}]$ และเก็บเมตริซ์เหล่านี้ไว้ในหน่วยความจำ จากนั้นก็ใช้วิธี Cholesky decomposition ในการแก้สมการที่ (5) วิธีนี้เป็นวิธีที่ใช้ใน WSINV3DMT (Siripunvaraporn et al., 2005; Siripunvaraporn and Egbert, 2009) และ DASOCC (Siripunvaraporn and Egbert, 2000) วิธีนี้จะเปลืองหน่วยความจำเนื่องจากต้องเก็บเมตริกซ์ **J** และ **R** ซึ่งมีขนาด *N* x *M* และ *N* x *N* ซึ่งอาจมีค่าสูงมากก็ได้ถ้าจำนวนข้อมูลมาก

WSDCG3DMT : Data Space Conjugate Gradient Algorithm

อีกวิธีหนึ่งที่ใช้แก้สมการที่ (5) คือการใช้วิธี conjugate gradient วิธีนี้ทำให้ไม่ต้องสร้างเมตริกซ์ **J** และ **R** ที่ต้องเก็บไว้ในหน่วยความจำอีกต่อไป เทคนิคนี้จึงประหยัดหน่วยความจำไปได้มาก ด้วยเทคนิคนี้เราไม่ได้สร้างเมตริกซ์ **J** โดยตรงแต่เราคำนวณผลคูณของเมตริกซ์ **J** กับเวกเตอร์ใดๆ เช่น **Jx** หรือ $\mathbf{J}^T\mathbf{y}$ เทคนิคนี้เป็นเทคนิคที่ใช้ใน WSDCG3DMT ข้อเสียของโปรแกรมนี้คือใช้เวลารันนานกว่า WSINV3DMT ซึ่งจะแสดงให้เห็นในตอนถัดไป



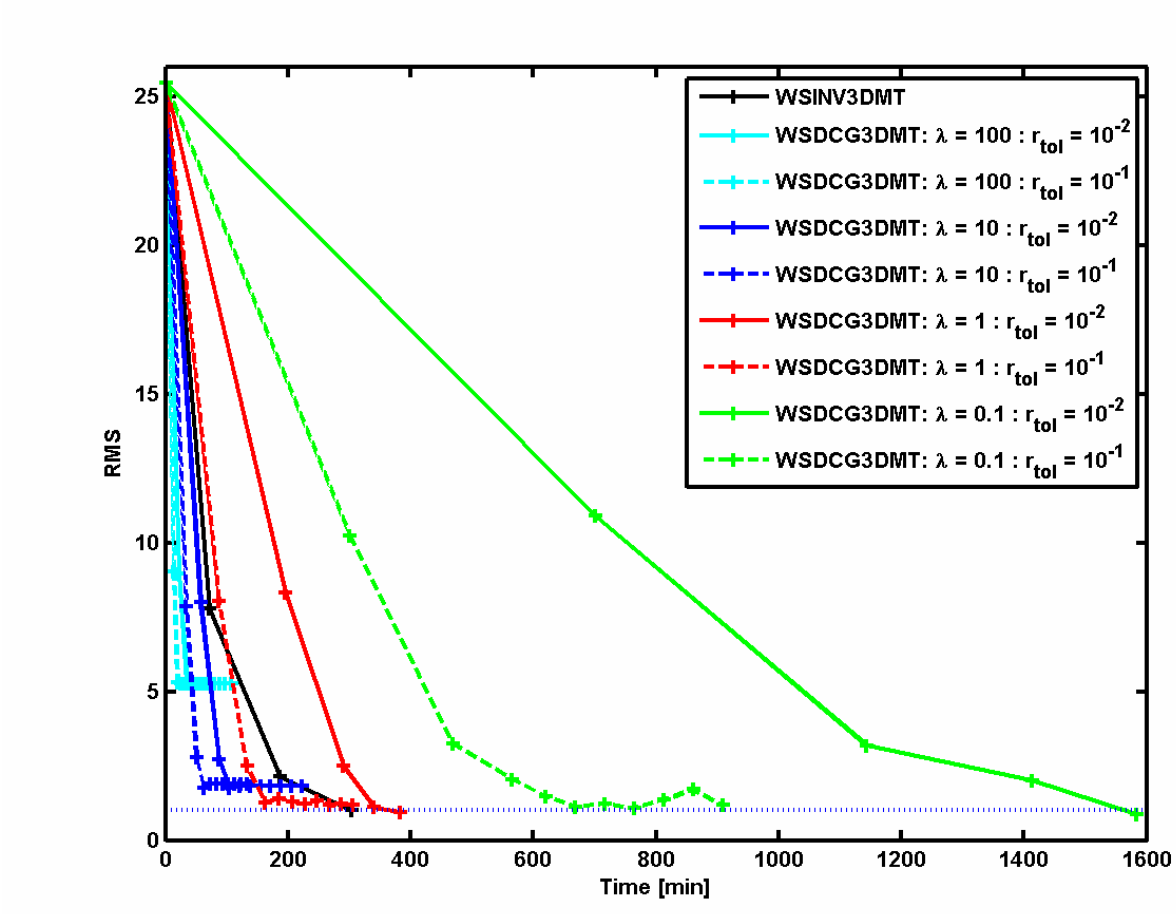
รูปที่ 1 รูปแสดงแบบจำลองเทียมที่ใช้สร้างข้อมูลเทียมเพื่อใช้ในการทดสอบโปรแกรม

การประเมินผลโปรแกรม WSINV3DMT และ WSDCG3DMT และผลการทดสอบ

การทดลองของเราเริ่มต้นจากทดสอบทั้งสองโปรแกรม WSINV3DMT และ WSDCG3DMT กับข้อมูลเทียม (synthetic data) โดยใช้แบบจำลองตามรูปที่ 1 ข้อมูลเทียมประกอบไปด้วย impedance tensor ทั้งสี่ components มีทั้งหมด 40 สถานีวัดและมีความถี่ทั้งหมด 16 ความถี่ ขนาดของแบบจำลองเท่ากับ 28 x 28 x 21 ดังนั้นข้อมูลนี้มี N = 40 x 16 x 8 = 5,120 และ M = 28 x 28 x 21 = 16,464 การทดลอง

ต่อไปนี้รันบนเครื่องเดียวกัน คือบนเครื่อง Intel Core Two Duo 6400, 2.13 GHz จากจำนวนข้อมูลนี้เราสามารถประมาณหน่วยความจำของทั้งสองโปรแกรมได้ว่า WSINV3DMT ต้องใช้หน่วยความจำถึง 1 GByte ในขณะที่ WSDCG3DMT จะใช้เพียง 0.4 Gbyte ซึ่งน้อยกว่าเกือบครึ่งหนึ่ง

การทดลองแรก เรารันโปรแกรม WSINV3DMT กับ WSDCG3DMT ที่หลายค่า λ = 100, 10, 1, 0.1, 0.01 ผลการทดลองแสดงในรูปที่ 2 ซึ่งแสดงให้เห็นสำหรับค่า λ ของ WSDCG3DMT ที่ converge สู่ 1 RMS นั้นจะใช้เวลาในการทำงานช้ากว่า WSINV3DMT ซึ่งใช้เวลาเพียง 300 นาที ในขณะที่ WSDCG3DMT สำหรับค่า λ = 1 และ 0.1 จะใช้เวลาถึง 400 นาทีและ 1600 นาที



รูปที่ 2 แสดงการลู่เข้าหาคำตอบของ WSINV3DMT (สีดำ) และ WSDCG3DMT ที่หลากหลายค่า λ ผลการทดลองแสดงให้เห็นว่า WSDCG3DMT ใช้เวลานานกว่า WSINV3DMT

การทดลองนี้มีข้อสังเกตที่น่าสนใจมากๆ อันดับแรกก็คือ WSDCG3DMT ที่มีค่า λ สูงจะใช้เวลาไวมากในแต่ละ iteration ในขณะที่มีค่า λ ต่ำจะใช้เวลานานกว่ามาก เช่นที่ λ = 100 iteration แรกใช้เวลาไม่ถึง 20 นาที ในขณะที่ λ = 0.1 iteration แรกใช้เวลามากถึง 700 นาที อย่างที่สอง สำหรับค่า λ

สูงแม้ว่าจะใช้เวลาไวแต่ก็ไม่สามารถลู่เข้าหาคำตอบได้เลย เช่นที่ λ = 100 และ 10 แต่สำหรับค่า λ ต่ำจะสามารถลู่เข้าหาคำตอบได้ เช่นที่ λ = 1 และ 0.1 อย่างที่สามคือ iteration แรกจะใช้เวลาในการรันนานที่สุด iteration ถัดๆ ไปจะใช้เวลาในการรันน้อยลงไปเรื่อยๆ เช่นสำหรับ λ = 0.1, iteration ที่หนึ่ง สอง สามและสี่ จะใช้เวลารันประมาณ 700 นาที 500 นาที 400 นาที และ 200 นาที ตามลำดับ

จากผลการทดลองครั้งนี้ทำให้เราสามารถนำไปขยายผลเพื่อสร้าง Algorithm ใหม่ขึ้นมา

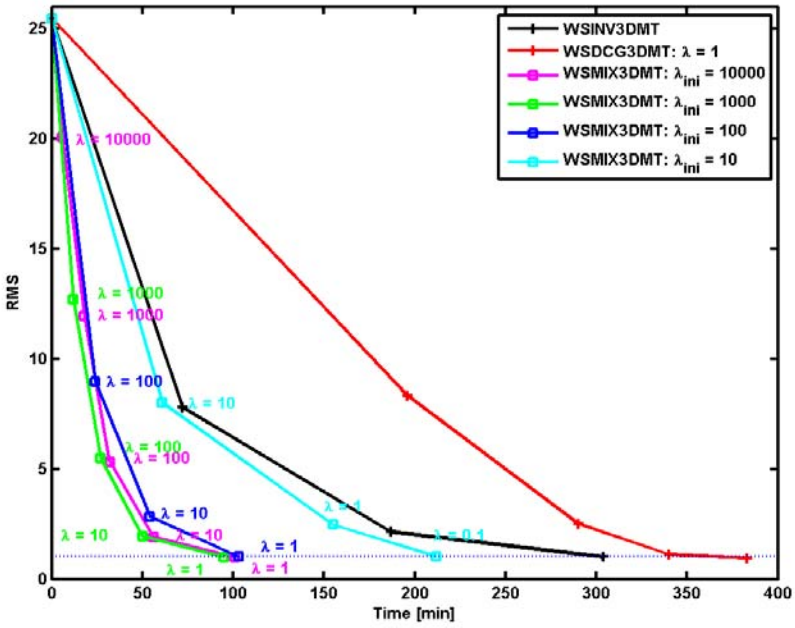
WSMIX3DMT : a Mixed Scheme of DCG and Occam's inversion

จากการทดลองเบื้องต้น เราพบว่าเราสามารถสร้างโปรแกรมขึ้นมาใหม่ โดยใช้หน่วยความจำเหมือน WSDCG3DMT แต่ทว่ามีความเร็วเร็วกว่าทั้ง WSDCG3DMT และ WSINV3DMT

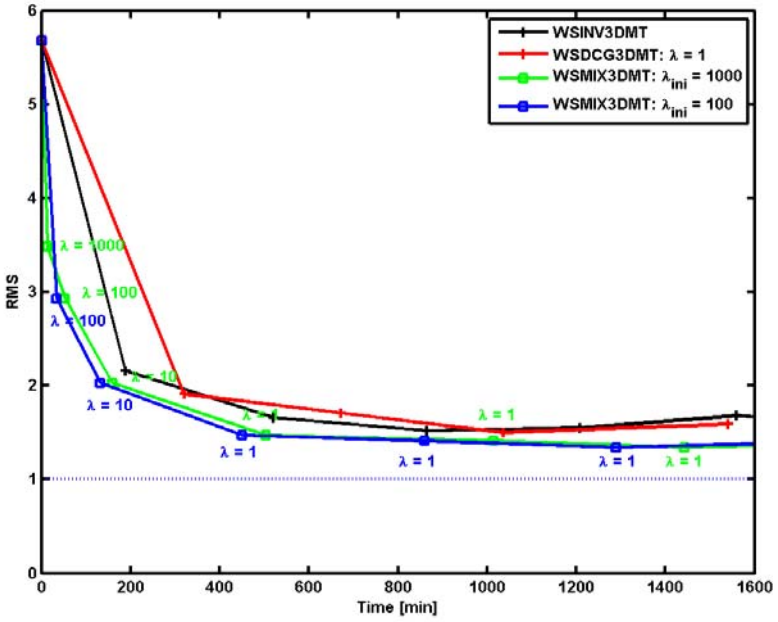
หลักการของโปรแกรมใหม่นี้มาจากการผสมกันของ WSDCG3DMT และ WSINV3DMT ตัวคณิตศาสตร์แทบจะเหมือนกับ WSDCG3DMT แต่หลักการของโปรแกรมใหม่นี้จะง่ายๆ คือการเปลี่ยนค่า λ ในแต่ละ iteration ที่คล้ายคลึงกับ WSINV3DMT แต่ทว่าในครั้งนี้เราจะเริ่ม iteration แรกด้วยค่า λ ที่มีค่ามาก เนื่องจากเมื่อ λ มาก เวลาทำงานก็จะสั้นลง โดยเฉพาะใช้กับ iteration แรกๆ ที่ต้องใช้เวลาในการทำงานสูง จากนั้นเราจะลดค่า λ ใน iteration ถัดไปเรื่อยๆ ตามแต่ที่กำหนด เช่นลดลง 10 เท่า เป็นต้น ตัวอย่างของการรันโปรแกรม เริ่มต้นจาก λ = 1000 ใน iteration แรก จากนั้นก็จะลดลงเป็น λ = 100 ใน iteration ที่สอง และเป็น λ = 10 ใน iteration ที่สาม ไปเรื่อยๆ จนถึงค่าน้อยที่สุดซึ่งในที่นี้เรากำหนดไว้ที่ λ = 0.1

การประเมินผลโปรแกรม WSMIX3DMT และ WSINV3DMT และ WSDCG3DMT

เราทดสอบโปรแกรมใหม่ WSMIX3DMT กับข้อมูลเทียมเดิม และเทียบผลที่ได้กับสองโปรแกรมเก่า เนื่องจากคณิตศาสตร์ของโปรแกรมใหม่นั้นเหมือนกับ WSDCG3DMT ดังนั้นหน่วยความจำจึงเท่ากัน แต่เมื่อดูที่เวลาแล้วจะเห็นว่าโปรแกรมใหม่ WSMIX3DMT นั้นไวที่สุด คือใช้เวลาน้อยกว่า 100 นาทีไม่ว่าจะเริ่มต้นด้วยค่า λ ที่เท่าไรก็ตาม ส่วน WSDCG3DMT ใช้เวลา 400 นาที ส่วน WSINV3DMT ใช้เวลา 300 นาที ดังแสดงในรูปที่ 3



รูปที่ 3 แสดงการลู่เข้าหาคำตอบของ WSINV3DMT (สีดำ) และ WSDCG3DMT ที่ λ = 1 (สีแดง) และ WSMIX3DMT ที่เริ่มต้นจาก λ = 10000 (สีชมพู), 1000 (สีเขียวอ่อน), 100 (สีน้ำเงิน) และ 10 (สีฟ้าอ่อน) กับข้อมูลเทียมรูปที่ 1 ผลการทดลองแสดงให้เห็นว่า WSMIX3DMT ใช้เวลาไวกว่าทั้ง WSDCG3DMT และ WSINV3DMT



รูปที่ 4 แสดงการลู่เข้าหาคำตอบของ WSINV3DMT (สีดำ) และ WSDCG3DMT ที่ λ = 1 (สีแดง) และ WSMIX3DMT ที่เริ่มต้นจาก λ = 1000 (สีเขียวอ่อน) และ 100 (สีน้ำเงิน) กับข้อมูล EXTECH จริง (Tuncer et al, 2006) ผลการทดลองแสดงให้เห็นว่า WSMIX3DMT ใช้เวลาไวกว่าทั้ง WSDCG3DMT และ WSINV3DMT

นอกจากนี้ทั้งโปรแกรมใหม่และเก่าถูกนำไปทดสอบกับข้อมูลจริง EXTECH data (see Tuncer et al., 2006) พบว่าให้ผลที่เหมือนกัน คือ WSMIX3DMT (สีน้ำเงินและเขียว) ไวกว่าทั้ง WSINV3DMT (สีดำ) และ WSDCG3DMT (สีแดง) และยังใช้หน่วยความจำเท่ากับ WSDCG3DMT ซึ่งน้อยกว่า WSINV3DMT มากๆ ดังแสดงในรูปที่ 4

สรุปผล

เราได้พัฒนาโปรแกรมใหม่ขึ้นมา WSMIX3DMT โปรแกรมมีหลักการจากสองโปรแกรมเก่าคือ WSINV3DMT และ WSDCG3DMT คณิตศาสตร์ของโปรแกรมใหม่จะเหมือนกับ WSDCG3DMT แต่หลักการจะคล้ายกับ WSINV3DMT คือ vary λ ไปในแต่ละ iteration แต่กรณีนี้จะเลือกเริ่มต้นที่ λ มากก่อนที่จะค่อยๆ ลดลง ผลการทดลองทั้งกับข้อมูลเทียมและข้อมูลจริงพบว่า โปรแกรม WSMIX3DMT ใช้หน่วยความจำเท่ากับ WSDCG3DMT ซึ่งน้อยกว่า WSINV3DMT มากๆ แต่ในขณะเดียวกันก็ใช้เวลาในการรันน้อยลงกว่าโปรแกรมเก่าสองถึงสามเท่า

หนังสืออ้างอิง

- Boonchaisuk, S., Vachiratienchai, C., **Siripunvaraporn, W.**, 2008, Two-dimensional direct current (DC) resistivity inversion: Data space Occam's approach, *Physics of the Earth and Planetary Interiors* 168 (3-4), pp. 204-211
- Boonchaisuk, S., Satitpitakul, A., Vachiratienchai, C., Nualkhow, P., Amatyakul, P., Unhapipat, S., Rung-Arunwan, T., Sarakorn, W., Siripunvaraporn, W, and Ogawa, Y., 2010, Three-dimensional crustal resistivity structure beneath Kanchanaburi province, Wester part of Thailand., SPC 2010, Kanchanaburi, Thailand, 25-27 March.
- Constable, C. S., Parker, R. L., and Constable, C.G., 1987, Occam's inversion: A practical algorithm for generating smooth models from electromagnetic sounding data: *Geophysics*, **52**, 289-300.
- Jones, A. G., 1992, Electrical conductivity of the continental lower crust, *in* Fountain, D. M., Arculus, R. J., and Kay, R. W., Eds., Continental Lower Crust: Elsevier Science Publ. Co., Inc., 81-143.
- Orange, A. S., 1989, Magnetotelluric Exploration for Hydrocarbons: *Proc. IEEE*, **77**, 287-317.
- Patro P., Uyeshima M., and **W. Siripunvaraporn**, 2010, Three-dimensional inversion of magnetic phase tensor, submitted *Geophysical Journal International*,
- Siripunvaraporn, W., Egbert, G. D., Eisel, M., and Unsworth, M., 1998, A High Resolution EM Survey of the San Andreas Fault (SAF): Local Conductivity Structure in a Regional Context: Eos, Trans. Am. Geophys. Union, **79**, 45, Fall Meet. Suppl., F227.
- Siripunvaraporn W. and Egbert, G. D., 2000, An Efficient Data-Subspace Inversion for Two-Dimensional Magnetotelluric Data, *Geophysics*, 65, 791-803.
- Siripunvaraporn, W., M. Uyeshima and G. Egbert, 2004, Three-dimensional inversion for Network-Magnetotelluric data, *Earth Planets Space*, 56, 893-902.
- Siripunvaraporn, W., G. Egbert, Y. Lenbury and M. Uyeshima, 2005, Three-Dimensional Magnetotelluric Inversion: Data Space Method, *Phys. Earth and Planet. Interior.*, 150, 3-14.
- Siripunvaraporn W. and G. Egbert, 2007, Data Space Conjugate Gradient Inversion for 2-D Magnetotelluric Data, Geophysical Journal International, **170**, 986-994.
- Siripunvaraporn W. and G. Egbert, 2009, WSINV3DMT: Vertical Magnetic Field Transfer Function Inversion and Parallel Implementation, *Physics of the Earth and Planetary Interiors* 173 (3-4), pp. 317-329
- Siripunvaraporn W., and W. Sarakorn, 2010, An efficient three-dimensional Magnetotelluric inversion: a mixed of the data space conjugate gradient method and the data space Occam's method, submitted to *Geophysical Journal International.*
- Tuncer V., M. Unsworth, W. Siripunvaraporn, J. Craven, 2006, Audio-magnetotelluric exploration for unconformity type uranium deposits at the McArthur River Mine, northern Saskatchewan (Canada), *Geophysics*, ***71,*** B201-209.
- Unsworth M., P. Bedrosian, M. Eisel, G. Egbert and W. Siripunvaraporn, 2000, Along strike variations in the electrical structure of the San Andreas Fault at Parkfield, California, *Geop. Res. Lett.*, **27**, 3021-3024.
- Vozoff, K., 1972, The Magnetotelluric Method in the Exploration of Sedimentary Basins: *Geophysics*, **37**, 98-141.

Output ที่ได้จากโครงการ

ผลงานตีพิมพ์ในวารสารวิชาการนานาชาติ (submitted and revised)

- **Siripunvaraporn W.**, and W. Sarakorn, 2010, An efficient three-dimensional Magnetotelluric inversion: a mixed of the data space conjugate gradient method and the data space Occam's method, submitted to *Geophysical Journal International.*
- Rung-Arunwan T. and **W. Siripunvaraporn**, 2010, An efficient modified hierarchical domain decomposition for 2-D Magnetotelluric forward modeling, submitted *Geophysical Journal International*, moderate revision.

ผลงานตีพิมพ์ในวารสารวิชาการนานาชาติ (published)

- **Siripunvaraporn W.** and G. Egbert, 2009, WSINV3DMT: Vertical Magnetic Field Transfer Function Inversion and Parallel Implementation, *Physics of the Earth and Planetary Interiors* 173 (3-4), pp. 317-329
- Kalscheuer, T., Pedersen, L.B., **Siripunvaraporn, W.**, 2008, Radiomagnetotelluric two-dimensional forward and inverse modelling accounting for displacement currents, *Geophysical Journal International* 175 (2), pp. 486-514
- Boonchaisuk, S., Vachiratienchai, C., **Siripunvaraporn, W.**, 2008, Two-dimensional direct current (DC) resistivity inversion: Data space Occam's approach, *Physics of the Earth and Planetary Interiors* 168 (3-4), pp. 204-211

Software

- **Siripunvaraporn W.**, 2010, WSMIX3DMT; Three-Dimensional inversion program based on a mixed of the data space conjugate gradient technique and the Occam's method. The code is written with Fortran 77/95.

การเสนอผลงานในที่ประชุมวิชาการนานาชาติบางส่วน

- **Weerachai Siripunvaraporn**, 2010, A combined DCG and OCCAM algorithm for three-dimensional Magnetotelluric Data, Japan Geosciences Union Meeting, 2010, Chiba, Japan, May 23 – 28.
- Noriko Tada, Kiyoshi Baba, **Weerachai Siripunvaraporn**, Makoto Uyeshima and Hisashi Utada, 2010, 3-D inversion of marine Magnetotelluric data, Japan Geosciences Union Meeting, 2010, Chiba, Japan, May 23 – 28.
- Noriko Tada, Kiyoshi Baba, **Weerachai Siripunvaraporn**, Makoto Uyeshima and Hisashi Utada, 2009, Toward 3-D inversion of seafloor MT data, Japan Geosciences Union Meeting 2009, Chiba, Japan, May 16 – 21.
- Thomas Kalscheuer, Laust B. Pedersen and **Weerachai Siripunvaraporn**, 2008, Radiomagnetotelluric two-dimensional forward and inverse modeling accounting for displacement currents, Induction Workshop #19, Beijing, China, Octorber 23-29.
- Noriko Tada, Kiyoshi Baba, **Weerachai Siripunvaraporn**, Makoto Uyeshima and Hisashi Utada, 2008, Modification of a 3-D Magnetotelluric forward code considering topography effect in order to estimate 3-D conductivity structures of oceanic upper mantle using inversion method, Induction Workshop #19, Beijing, China, Octorber 23-29.
- Uyeshima M, Ogawa T, Yamaguchi S, Murakami H, Toh H, Yoshimura R, Oshiman N, Tanbo T, Koyama S, Mochizuki H, Marutani Y, Usui Y, and **Siripunvaraporn W**, 2008, 3-D resistivity structure in the vicinity of the Atotsugawa fault revealed by a Network-MT survey, Induction Workshop #19, Beijing, China, Octorber 23-29.
- **Weerachai Siripunvaraporn**, Gary Egbert and Hishi Utada, 2008, Current version of WSINV3DMT, Induction Workshop #19, Beijing, China, Octorber 23-29.
- Weerachai Sarakorn, **Weerachai Siripunvaraporn** and Gary Egbert, 2008, Three-dimensional Magnetotelluric modeling using finite element method: Tetrahedral and

hexahedral elements, comparison to finite difference method, Induction Workshop #19, Beijing, China, Octorber 23-29.

- Gary D Egbert and **Weerachai Siripunvaraporn**, 2008, Hybrid Occam/Conjugate Gradient Methods for Practical Electromagnetic Inversion, Induction Workshop #19, Beijing, China, Octorber 23-29.
- Puwis Amatyakul and **Weerachai Siripunvaraporn**, 2008, A joint inversion of direct-current resistivity and Magnetotelluric data, Induction Workshop #19, Beijing, China, Octorber 23-29.
- Tawat Rung-Arunwan and **Weerachai Siripunvaraporn**, 2008, Domain decomposition for 2D Magnetotelluric forward modeling, Induction Workshop #19, Beijing, China, Octorber 23-29.
- Noriko Tada, **Weerachai Siripunvaraporn**, Makoto Uyeshima, Kiyoshi Baba, Hisashi Utada, 2008, Development of marine 3D MT inversion scheme, Japan Geoscience Union Meeting 2008, May 25-30, Chiba, Japan.
- Uyeshima, M., R. Yoshimura, T. Ogawa, S. Yamaguchi, H. Murakami, H. Toh, N. Oshiman, S. Koyama, T. Tanbo, Y. Usui, H. Mochizuki, W. Siripunvaraporn, and Research Group for Crustal Resistivity Structure in the NKTZ Concentrated Deformation Zone, 2008, 3-D resistivity structure beneath the Atotsugawa Fault zone revealed by the Network-MT observations, The Japan Geoscience Union Meeting 2008, May 25-30, Chiba, Japan

ภาคผนวก

ภาคผนวก ก. Manuscript

- **Siripunvaraporn W.**, and W. Sarakorn, 2010, An efficient three-dimensional Magnetotelluric inversion: a mixed of the data space conjugate gradient method and the data space Occam's method, submitted to *Geophysical Journal International.*

ภาคผนวก ข. Manuscript

- Rung-Arunwan T. and **W. Siripunvaraporn**, 2010, An efficient modified hierarchical domain decomposition for 2-D Magnetotelluric forward modeling, submitted *Geophysical Journal International*, moderate revision.

ภาคผนวก ค. Reprint

- **Siripunvaraporn W.** and G. Egbert, 2009, WSINV3DMT: Vertical Magnetic Field Transfer Function Inversion and Parallel Implementation, *Physics of the Earth and Planetary Interiors* 173 (3-4), pp. 317-329

ภาคผนวก ง. Reprint

- Kalscheuer, T., Pedersen, L.B., **Siripunvaraporn, W.**, 2008, Radiomagnetotelluric two-dimensional forward and inverse modelling accounting for displacement currents, *Geophysical Journal International* 175 (2), pp. 486-514

ภาคผนวก จ. Reprint

- Boonchaisuk, S., Vachiratienchai, C., **Siripunvaraporn, W.**, 2008, Two-dimensional direct current (DC) resistivity inversion: Data space Occam's approach, *Physics of the Earth and Planetary Interiors* 168 (3-4), pp. 204-211

ภาคผนวก ก. Manuscript

- **Siripunvaraporn W.**, and W. Sarakorn, 2010, An efficient three-dimensional Magnetotelluric inversion: a mixed of the data space conjugate gradient method and the data space Occam's method, submitted to *Geophysical Journal International.*

An efficient three-dimensional Magnetotelluric inversion: a mixed of the data space conjugate gradient method and the data space Occam's method

Weerachai Siripunvaraporn [1,2] and Weerachai Sarakorn[3]

[1]Department of Physics, Faculty of Science, Mahidol University, Rama 6 Rd., Rachatawee, Bangkok 10400, THAILAND.

[2] ThEP, Commission of Higher Education, 328 Si Ayuthaya Road, Bangkok 10400, THAILAND

[3]Department of Mathematics, Faculty of Science, Mahidol University, Rama 6 Rd., Rachatawee, Bangkok, 10400, THAILAND.

Abstract

In this paper, we start with the implementation and extension of the data space conjugate gradient (DCG) method previously developed for the two-dimension (2-D) to the three-dimension (3-D) Magnetotelluric (MT) data, and will be referred to as WSDCG3DMT. Synthetic experiments show that WSDCG3DMT usually spends computational time longer than the data space Occam's inversion (WSINV3DMT). However, memory requirement of WSDCG3DMT is only a fraction of WSINV3DMT. Knowledge and information gained from the synthetic studies of WSDCG3DMT has led to a creation of a mixed scheme (WSMIX3DMT) of the data space conjugate gradient and the data space Occam's methods. WSMIX3DMT is a slight modification of WSDCG3DMT but enhancing so that its computational time is several factors lower than both WSINV3DMT and WSDCG3DMT. Because WSMIX3DMT is a modification of WSDCG3DMT, its memory requirement is therefore a fraction of WSINV3DMT as in WSDCG3DMT. This makes WSMIX3DMT as the most efficient inversions. Computational performances and comparisons of all three methods are demonstrated with both synthetic and EXTECH field datasets.

1. Introduction

Recently, number of three-dimensional (3-D) magnetotelluric (MT) surveys is substantially increased worldwide (e.g., Tuncer et al., 2006; Patro and Egbert, 2008, among many others). This might be due to the fact that MT has increasingly accepted by many geophysicists and seismologists. Another factor is the improvements of the data acquisition units, the measurement sensors and their accessories. Examples of MT uses are for geothermal explorations (e.g., Heise et al., 2008; Árnason et al., 2010), volcanoes and tectonic studies (Uyeshima, 2007; Patro and Egbert, 2008; Hill et al., 2009; Ingham et al., 2009) and ore explorations (Tuncer et al., 2006; Queralt et al., 2007; Farquharson and Craven, 2008; Türkoğlu et al., 2009; Goldax and Kosteniuk, 2010). All of these have led to a higher demand for 3-D MT inversion codes for interpretation.

Currently, a number of 3-D MT inversion algorithms have been developed (e.g. Mackie & Madden 1993; Newman & Alumbaugh 2000; Zhdanov et al. 2000; Sasaki 2001; Mackie, personal communication 2002; Siripunvaraporn et al. 2004, 2005; Sasaki and Meju, 2006; Han et al., 2008; Lin et al., 2008,2009; Farquharson and Craven, 2008; Adveed and Adveed, 2009; Siripunvaraporn et al., 2009). All algorithms are designed to find "best" model that fits the data but also "geologically" interpretable. One of the 3-D algorithms (and the only one currently available to the MT communities) is the WSINV3DMT program by Siripunvaraporn et al. (2005; 2009). The algorithm's idea was based on the Occam's style inversion introduced for 1-D MT data by Constable et al. (1987). Occam's inversion is known for its robust calculation and its efficiency. However, its disadvantage is the large memory requirements, and the extensive computational time, particularly when applying to 2-D and 3-D modeling (Siripunvaraporn and Egbert, 2000; Siripunvaraporn et al., 2005).

To reduce both storage and calculation time, Siripunvaraporn and Egbert (2000) and Siripunvaraporn et al. (2004; 2005) transformed the original Occam's inversion which is a model space method into the data space Occam's algorithm. The transformation makes it practical for 3-D MT inversion on most computers. However, WSINV3DMT still requires substantial memory to store the $N \times M$ sensitivity matrix, where N and M are the data and model parameters, respectively. Siripunvaraporn and Egbert (2007) used 2-D MT data to show that the large storage can be avoid by using a data space conjugate gradient (DCG) approach.

From the 2-D studies, Siripunvaraporn and Egbert (2007) concluded that the DCG method can significantly reduce the memory usage. However, its computational time can be longer than that of the data space Occam's algorithm. Computational time of the DCG method is controlled by the stopping criteria used inside the conjugate gradient (CG) algorithm when solving the normal equation (**Rx** = **b**). The CG solver is terminated when the relative error (r = ||**Rx** – **b**||/||**b**||) reaches a given tolerance r_{tol}. Smaller r_{tol} (e.g., $r_{tol} < 10^{-2}$) requires many number of CG iterations, while larger r_{tol} (e.g., $r_{tol} = 10^{-1}$) requires significantly less but can cause the inversion to fail to converge to the target misfit. Large number of CG iterations translates into longer CPU time. Our 2-D studies also showed that $r_{tol} = 10^{-2}$ is the optimal tolerance value. The model generated with $r_{tol} = 10^{-2}$ differs less than a percent from that generated with $r_{tol} = 10^{-8}$ but requires significantly less CPU time.

In addition, convergence rate of the DCG inversion also depends on the regularization parameter λ, which acts as a trade-off between the data norm and the model norm. Larger λ ($\lambda >$ 10) demands small number of CG iterations per inversion iteration. However, the inversion could not bring the misfit down to the desired misfit because large λ produces very smooth model. Smaller λ ($0.1 \leq \lambda \leq 10$) can reach the desired level of misfit but normally requires large number of CG iterations per inversion iteration. However, if λ is too small ($\lambda < 0.1$), DCG can break down. If it converges, it requires significantly large number of CG iterations and also produces "very rough and spurious" structures which is not geologically interpretable.

Here, we directly implement and extend the data space conjugate gradient (DCG) algorithm for the 3-D MT data. Hereafter, we will refer to the 3-D DCG method as WSDCG3DMT. Numerical experiments are performed on a synthetic data in a similar way as conducted in the 2-D experiments (Siripunvaraporn and Egbert, 2007). The objective is to verify whether the conclusions learned from the 2-D cases remain the same or different for the 3-D data. Knowledge gained from the synthetic studies has led us to a creation of a mixed scheme of the Occam's inversion and the DCG method. We will refer to a mixed scheme as WSMIX3DMT.

We start the paper with a brief review of the data space conjugate gradient method (WSDCG3DMT) and its necessary mathematics. More details on the data space Occam's inversion and the data space conjugate gradient method can be found in many previous publications (Siripunvaraporn and Egbert, 2000; Siripunvaraporn et al., 2005; Siripunvaraporn

and Egbert, 2007; and Boonchaisuk et al., 2008). Later, a mixed scheme (WSMIX3DMT) between the DCG method and the Occam method is introduced. Numerical experiments on both synthetic data and EXTECH data are performed with these three algorithms (WSINV3DMT, WSDCG3DMT and WSMIX3DMT). Comparisons in terms of computational time and memory are analyzed and discussed. A conclusion is given at the end.

2. Review of Data Space Conjugate Gradient Inversion

Consider a general objective functional Φ^m,

$$\Phi^m = \Phi_d + \lambda\Phi_m = (\mathbf{d} - \mathbf{F}[\mathbf{m}])^T \mathbf{C_d}^{-1} (\mathbf{d} - \mathbf{F}[\mathbf{m}]) + \lambda (\mathbf{m} - \mathbf{m_0})^T \mathbf{C_m}^{-1} (\mathbf{m} - \mathbf{m_0}), \quad (1)$$

where Φ_d a data norm, Φ_m a model norm, $\mathbf{m}$ the resistivity model of dimension M, $\mathbf{m_0}$ the prior model, $\mathbf{C_m}$ the model covariance matrix, $\mathbf{d}$ the observed data with dimension N, $\mathbf{F}[\mathbf{m}]$ the forward model response, $\mathbf{C_d}$ the data covariance matrix, and λ a regularization parameter.

To minimize (1) in a data space method, we start with the transformation of the model space objective functional (1) to a data space objective functional (2) by expressing a model as a linear combination of rows of the smoothed sensitivity matrix (Parker, 1994), or $\mathbf{m} - \mathbf{m_0} = \mathbf{C_m}\mathbf{J}^T\boldsymbol{\beta}$. Then, (1) becomes

$$\Phi^d = (\mathbf{đ} - \mathbf{J}\mathbf{C_m}^T\mathbf{J}^T\boldsymbol{\beta})^T \mathbf{C_d}^{-1} (\mathbf{đ} - \mathbf{J}\mathbf{C_m}^T\mathbf{J}^T\boldsymbol{\beta}) + \lambda (\boldsymbol{\beta}^T\mathbf{J}\mathbf{C_m}^T\mathbf{J}^T\boldsymbol{\beta}), \quad (2)$$

where $\mathbf{J} = \partial\mathbf{F}/\partial\mathbf{m}$ is an $N \times M$ sensitivity matrix, and $\mathbf{đ} = \mathbf{d} - \mathbf{F}[\mathbf{m}] + \mathbf{J}(\mathbf{m} - \mathbf{m_0})$. To minimize (2), $\mathbf{F}[\mathbf{m}_{k+1}]$ is linearized with the first order Taylor series expansion, as $\mathbf{F}[\mathbf{m}_{k+1}] = \mathbf{F}[\mathbf{m}_k] + \mathbf{J}_k (\mathbf{m}_{k+1} - \mathbf{m}_k)$, when $_k$ is an inversion iteration number. Differentiating (2) with respect to $\boldsymbol{\beta}$ and rearranging, an iterative sequence of approximate solutions can be obtained as,

$$\mathbf{m}_{k+1} - \mathbf{m_0} = \mathbf{C_m}\mathbf{J_k}^T \mathbf{C_d}^{-½} [\lambda \mathbf{I} + \mathbf{C_d}^{-½}\mathbf{J}_k\mathbf{C_m}\mathbf{J_k}^T \mathbf{C_d}^{-½}]^{-1} \mathbf{C_d}^{-½}\mathbf{đ}_k, \quad (3)$$

where $\mathbf{I}$ is an identity matrix.

There are two methods to solve (3). First method is to explicitly form $\mathbf{J}$ and $\mathbf{R} = [\lambda \mathbf{I} + \mathbf{C_d}^{-½} \mathbf{J}_k\mathbf{C_m}\mathbf{J_k}^T \mathbf{C_d}^{-½}]$ and store them in the computer memory. $\mathbf{R}$ will be factorized into lower and upper matrices (LU-factorization), and then solved with backward and forward substitutions.

This method is used in WSINV3DMT program for 3-D MT data (Siripunvaraporn et al., 2005; Siripunvaraporn and Egbert, 2009) and DASOCC for 2-D MT data (Siripunvaraporn and Egbert, 2000). This scheme requires substantial amount of RAM to store $N \times M$ **J** and also $N \times N$ **R** matrices. This could prohibit a run on very large data sets, particularly for 3-D cases.

Instead of forming and decomposing **R** as in WSINV3DMT, an alternative method is to solve (3) with an iterative solver. Because **R** is theoretically symmetric, (3) is commonly solved with a conjugate gradient (CG) method as in many MT inversion algorithms (see Mackie and Madden, 1993; Siripunvaraporn and Egbert, 2007; Lin et al., 2008). One clear advantage of using CG to solve (3) is that the large $N \times M$ sensitivity matrix **J** is not explicitly formed and stored in the computer memory. Only a product of **J** or $\mathbf{J}^T$ with an arbitrary vector is required by solving one forward problem per period (see Mackie and Madden, 1993; Newman and Alumbaugh, 2000; Rodi and Mackie, 2001; Siripunvaraporn and Egbert, 2007; Lin et al., 2008). Two routines to compute **Jp** and $\mathbf{J}^T\mathbf{q}$ are therefore implemented here for the 3-D problem, where **p** and **q** are general $M \times 1$ and $N \times 1$ vectors, respectively. This method is used in WSDCG3DMT.

The data space conjugate gradient algorithm and the routines to explicitly form **J** and to compute **Jp** and $\mathbf{J}^T\mathbf{q}$ are briefly described in the following sub-sections.

2.1 Data Space Conjugate Gradient Algorithm (WSDCG3DMT)

The data space conjugate gradient algorithm denoted as WSDCG3DMT has two iterative loops. The outer loop which is a main inversion loop is to minimize (2), while the inner loop is to minimize $\mathbf{Rx} = \mathbf{b}$ in (3) with a conjugate gradient (CG) method where $\mathbf{R} = [\lambda \mathbf{I} + \mathbf{C_d}^{-½}\mathbf{J}\mathbf{C_m}\mathbf{J}^T \mathbf{C_d}^{-½}]$, $\mathbf{b} = \mathbf{C_d}^{-½}\mathbf{đ}$ and $\mathbf{x} = \mathbf{C_d}^{½}\boldsymbol{\beta}$ (see Barrett et al., 1994 for Preconditioned Conjugate Gradient algorithm). The algorithm was summarized in Figure 2 of Siripunvaraporn and Egbert (2007), and is repeatedly presented below with more explanations.

Reading inputs and initializing variables.

Start DCG "outer" loop to minimize (2) : iteration k

1. Compute $\mathbf{đ}_k = \mathbf{d} - \mathbf{F}[\mathbf{m}_k] + \mathbf{J}_k(\mathbf{m}_k - \mathbf{m}_0)$

2. Start DCG "inner" loop by using CG to solve $\mathbf{R_k x} = \mathbf{b}$

   2.1 Initialization: $\mathbf{x}_{(0)} = 0$; $\mathbf{r}_{(0)} = \mathbf{b}$, where $\mathbf{r} = ||\mathbf{Rx} - \mathbf{b}||/||\mathbf{b}||$.

   for *icg* = 1,2,…,*ncgmax* or $||\mathbf{r}^T\mathbf{r}|| < r_{tol}$, where *icg* a CG iteration number, *ncgmax* a maximum number of CG iterations, and r_{tol} a stopping tolerance level.

   2.2 $\mathbf{z}_{(icg\text{-}1)} = \mathbf{r}_{(icg\text{-}1)}$

   2.3 $\delta_{(icg\text{-}1)} = \mathbf{r}^T_{(icg\text{-}1)}\, \mathbf{z}_{(icg\text{-}1)}$

   2.4 if $(icg = 1)$ $\mathbf{p}_{(1)} = \mathbf{z}_{(0)}$

   else

   $\beta_{(icg\text{-}1)} = \delta_{(icg\text{-}1)} / \delta_{(icg\text{-}2)}$

   $\mathbf{p}_{(icg)} = \mathbf{z}_{(icg\text{-}1)} + \beta_{(icg\text{-}1)}\, \mathbf{p}_{(icg\text{-}1)}$

   endif

   2.5 $\mathbf{q}_{(icg\text{-}1)} = \mathbf{R_k p}_{(icg)}$

   2.6 $\alpha_{(icg\text{-}1)} = \delta_{(icg\text{-}1)} / \mathbf{p}^T_{(icg)}\, \mathbf{q}_{(icg)}$

   2.7 $\mathbf{x}_{(icg)} = \mathbf{x}_{(icg\text{-}1)} + \alpha_{(icg)}\, \mathbf{p}_{(icg)}$

   2.8 $\mathbf{r}_{(icg)} = \mathbf{r}_{(icg\text{-}1)} - \alpha_{(icg\text{-}1)}\, \mathbf{q}_{(icg)}$

   2.9 if $(||\mathbf{r}^T\mathbf{r}|| < r_{tol})$ or $(icg > ncgmax)$, then stop CG iteration and go to 3, else go to 2.2.

   end *icg*

3. Compute $\mathbf{m}_{k+1} - \mathbf{m_0} = \mathbf{C_m J}_k\, \mathbf{C_d}^{-½}\mathbf{x}$

4. Compute $\mathbf{F}[\mathbf{m}_{k+1}]$ and RMS misfit $\|\mathbf{C_d}^{-½}(\mathbf{d} - \mathbf{F}[\mathbf{m}_{k+1})\|$

5. Check condition;

   5.1 exit if misfit below the desired level, go to 6;

   5.2 continue if misfit is greater than the desired level, go to 1;

6. End DCG outer loop.

Step 1 requires calling one forward routine for $\mathbf{F}[\mathbf{m}_k]$, and another call to compute $\mathbf{J}_k(\mathbf{m}_k - \mathbf{m}_0)$. On step 2.1, system (3) is already normalized, therefore there is no preconditioner here. Step 2.5 is a "key" for the CG solver. It requires two forward modeling calls to compute $\mathbf{s} = \mathbf{J}_k^T \mathbf{C_d}^{-½}\mathbf{p}_{(icg)}$ and $\mathbf{J}_k\mathbf{C_m}\mathbf{s}$. Step 3 demands one forward modeling call to compute $\mathbf{J}_k \mathbf{C_d}^{-½}\mathbf{x}$. Step 4 requires another forward modeling call to compute the model responses $\mathbf{F}[\mathbf{m}_{k+1}]$. Overall, numbers of forward modeling calls to compute the model response is two per outer loop iteration per period, and to compute a multiplication of $\mathbf{J}$ or $\mathbf{J}^T$ with a vector is $2 + 2N_{cg}$ per outer loop iteration per period, where N_{cg} is a number of CG iterations. A total number of forward modeling calls would therefore be $4 + 2N_{cg}$ per period per outer loop iteration.

2.2 Forward Modeling and Sensitivity Calculation

Given an electrical conductivity (σ) or resistivity (ρ) model, to yield MT responses at the surface, the electric fields ($\mathbf{E}$) are computed from the second order Maxwell's equation,

$$\nabla \times \nabla \times \mathbf{E} = i\omega\mu\sigma\mathbf{E}, \qquad (4)$$

where ω is an angular frequency and μ the magnetic permeability. Discretizing the model and applying the staggered grid finite difference approach to (4), we obtain a system of equations for a given period or frequency,

$$\mathbf{Se} = \mathbf{b}, \qquad (5)$$

where **e** represents the unknown internal electric fields, **b** a vector containing the terms associated with the boundary electric fields, and **S** a large sparse symmetric and complex coefficient matrix. System of equations (5) is solved with a quasi-minimum residual (QMR) method per period and per polarization as in Siripunvaraporn et al. (2002). Surface responses can then be obtained from a linear combination of a vector **a** associated at a measurement site and the computed electric fields,

$$\mathbf{F}[\mathbf{m}] = \mathbf{a}^T\mathbf{e} = \mathbf{a}^T\mathbf{S}^{-1}\mathbf{b}. \qquad (6)$$

To compute for the sensitivity $\mathbf{J} = \partial\mathbf{F}/\partial\mathbf{m}$ at a given period, equation (6) is differentiated with respect to the model **m**,

$$\mathbf{J} = \partial\mathbf{F}/\partial\mathbf{m} = \partial(\mathbf{a}^T\mathbf{e})/\,\partial\mathbf{m} = \mathbf{a}^T\mathbf{S}^{-1}\boldsymbol{\Theta} + \boldsymbol{\Psi}, \qquad (7)$$

where $\boldsymbol{\Theta} = \partial\mathbf{b}/\partial\mathbf{m} - (\partial\mathbf{S}/\partial\mathbf{m})\mathbf{e}$ and $\boldsymbol{\Psi} = (\partial\mathbf{a}^T/\partial\mathrm{m})\mathbf{e}$. The process to form **J** is straightforward by first constructing $\boldsymbol{\Theta}$, solving $\mathbf{S}^{-1}\boldsymbol{\Theta}$, multiplying the result with $\mathbf{a}^T$ and finally adding with $\boldsymbol{\Psi}$. With this technique, calculating $\mathbf{S}^{-1}\boldsymbol{\Theta}$ would require solving the system of equations (5) *M* times per period and per polarization (Rodi, 1976). This calculation can be very significant, particularly in 3-D cases.

To reduce number of forward callings, reciprocity property of the electromagnetic fields (see Rodi, 1976; Mackie and Madden, 1993; Siripunvaraporn and Egbert, 2000) is applied to (7). With the reciprocity, the process of computing **J** is modified by first solving $(\mathbf{a}^T\mathbf{S}^{-1})^T$, then multiplying the result with $\boldsymbol{\Theta}^T$ before finally adding with $\boldsymbol{\Psi}^T$. Using the reciprocity technique, computing $(\mathbf{a}^T\mathbf{S}^{-1})^T$ would require solving the system of equations (5) only N_s times per period and per polarization (Rodi, 1976; Siripunvaraporn and Egbert, 2000), where N_s is the number of observed stations which is typically a lot smaller than *M*, particularly in 3-D cases. The reciprocity theorem helps significantly decreasing the computational time of the program (Siripunvaraporn and Egbert, 2000).

2.3 Multiplication of J or $\mathbf{J}^T$ to any vectors

To compute the product of **J** with a given vector **p**, equation (7) becomes

$$\mathbf{Jp} = \mathbf{a}^T\mathbf{S}^{-1}\mathbf{\Theta p} + \mathbf{\Psi p}. \quad (8)$$

The process is started with a multiplication of $\mathbf{\Theta p}$, then solving $\mathbf{S}^{-1}\mathbf{\Theta p}$, multiplying the result with $\mathbf{a}^T$, and finally adding them with the product of $\mathbf{\Psi p}$. Similarly, to compute the product of $\mathbf{J}^T$ with a given vector $\mathbf{q}$, equation (7) also becomes

$$\mathbf{J}^T\mathbf{q} = \mathbf{\Theta}^T[\mathbf{S}^T]^{-1}\mathbf{aq} + \mathbf{\Psi}^T\mathbf{q}. \quad (9)$$

The process here is also straightforward. It starts with a multiplication of $\mathbf{aq}$, because $\mathbf{S} = \mathbf{S}^T$, then solving $\mathbf{S}^{-1}\mathbf{aq}$ and multiplying them with $\mathbf{\Theta}^T$, finally adding the result with $\mathbf{\Psi}^T\mathbf{q}$. Equation (8) and (9) show that each process requires solving the system of equations (5) only one times per period and per polarization. Storage for $\mathbf{J}$ matrix is not necessary for (8) and (9) but required for (7).

2.4 Theoretical Comparisons for Forming J and Its Multiplications

Both forming $\mathbf{J}$ and its multiplications ($\mathbf{Jp}$ or $\mathbf{J}^T\mathbf{q}$) require solving the same system of equations (5), but with different right hand sides. As in section 2.2 and 2.3, forming $\mathbf{J}$ requires solving (5) with $\mathbf{a}$ as the right hand side, while computing $\mathbf{Jp}$ and $\mathbf{J}^T\mathbf{q}$ have $\mathbf{\Theta p}$ and $\mathbf{aq}$, as their right hand sides, respectively. All vectors ($\mathbf{a}$, $\mathbf{\Theta p}$ and $\mathbf{aq}$) are sparse, but $\mathbf{\Theta p}$ and $\mathbf{aq}$ involve more non-zero terms than $\mathbf{a}$. Consequently, solving (5) with $\mathbf{\Theta p}$ and $\mathbf{aq}$ as the right hand sides will require larger number of QMR iterations than with just $\mathbf{a}$ as the right hand side to converge to the same accuracy level. Similar behavior was also occurred in 2-D cases. Because system of equations for 2-D cases is small, the difference is therefore not significant. However, for 3-D case, the difference in CPU time is noticeable and will be shown in the numerical experiments.

2.5 Parallel Implementation

Similar to WSINV3DMT (Siripunvaraporn and Egbert, 2009), we also implement our 3-D DCG code on a parallel system. Although memory is not an issue for the DCG method, its extensive runtime is still a big concern due to its numerous calls to the forward modeling routine. As in WSINV3DMT, we parallelize WSDCG3DMT over frequencies via MPI (Message Passing Interface) libraries. For DCG, the parallelization is relatively simple, just distributing the forward

modeling call of each period to each processor node when computing the forward response **F**[**m**], and calculating **Jp** and $\mathbf{J}^T\mathbf{q}$. The simplicity occurs because there is no need to form and store the cross-product **R** as in WSINV3DMT (Siripunvaraporn and Egbert, 2009).

3. Numerical Experiments on a Synthetic Data : WSDCG3DMT & WSINV3DMT

Here, before we introduce a mixed scheme of the data space conjugate gradient method and the Occam's inversion; we start with the repetitions of the same experiments we conducted with the 2-D MT data but now with the 3-D MT data. The goal of the experiments is to check whether the same conclusions derived from the 2-D studies can be gained. In addition, we also compare the results with WSINV3DMT in terms of computational time and memory.

Similar to Siripunvaraporn et al. (2005) and Siripunvaraporn and Egbert (2009), we use the same synthetic model to generate a synthetic dataset for testing our codes. The synthetic model consists of two anomalies, 1 Ω-m and 100 Ω-m buried next to each other inside a 10 Ω-m layer lying on top of a 100 Ω-m half-space as illustrated in Figure 1 (Figure 4 in Siripunvaraporn et al., 2005; Figure 3b in Siripunvaraporn and Egbert, 2009). The model mesh for the inversion was discretized at 28 × 28 × 21 (+7 air layers) in x, y and z, respectively. The full complex impedance data (Z_{xx}, Z_{xy}, Z_{yx} and Z_{yy}; i.e. $N_m = 4$) is generated for 40 MT sites ($N_s = 40$) located regularly covering the two anomalies (solid dots in Figure 1) and 16 periods from 0.031 to 1000 second ($N_p = 16$). Five percent Gaussian noise calculated from the data magnitude ($|Z_{xy}Z_{yx}|^{1/2}$) was added to the impedance data. With this configuration, model parameter M would be equal to 28 × 28 × 21 = 16,464, while data parameter N would be equal to 40 × 16 × 8 = 5,120. In this experiment, all runs can be performed on a serial machine; an Intel Core Two Duo 6400, 2.13 GHz machine with 2 GBytes of RAM. Bigger model mesh or dataset would prohibit a run on this serial machine for WSINV3DMT.

Our first test is to perform the WSDCG3DMT program with various λ (λ = 100, 10, 1, 0.1, 0.01) and two r_{tol} (10^{-1} and 10^{-2}) for the DCG inner loop or the CG loop. Convergence behaviors of WSDCG3DMT for various λ and different r_{tol} as a function of time are shown in Figure 2 in comparison to WSINV3DMT. An inverted model after four iterations from WSDCG3DMT (λ = 1 and $r_{tol} = 10^{-2}$) is shown in Figure 3. The inversion can recover both

anomalies and the underlying layer similar to the inverted result from WSINV3DMT (Figure 6 of Siripunvaraporn et al., 2005).

For larger λ (10 and 100) with $r_{tol} = 10^{-2}$, DCG cannot converge to the desired level of 1 RMS. It can only lower the misfit down in the first two iterations before idling. Similar to the 2-D tests, larger λ requires smaller number of CG iterations to solve the normal equation (3) per outer loop iteration. This is reflected in a small amount of computing time as shown in Figure 2 (cyan and blue colors). For smaller λ (1 and 0.1) with $r_{tol} = 10^{-2}$, DCG is able to converge to the desired 1 RMS in four iterations. However, in contrast to larger λ, it demands significantly large number of CG iterations to solve (3) per one outer loop iteration. This is shown by a large amount of computational time in Figure 2 (red and green), particularly for the first iteration.

Reducing number of CG iterations per main iteration would help decreasing a computer runtime. One way is to set r_{tol} to a larger value. Here, at 10^{-1}. In all λ cases with $r_{tol} = 10^{-1}$, DCG has difficulty to converge to the target misfit of 1 RMS as seen in dash-lines of Figure 2. Larger r_{tol} would only help reducing computing time but not the convergence. In contrast, setting r_{tol} to smaller values (e.g., at 10^{-3} or less), number of inversion iterations to converge to the desired misfit is the same as in the case of $r_{tol} = 10^{-2}$. Inverted model is also less than a percent difference. Major difference is at the number of CG iterations per main inversion iteration which is significantly larger for smaller r_{tol}. These experiments show that $r_{tol} = 10^{-2}$ is appeared to be an optimal tolerance level for terminating the CG iterations in the DCG inner loop.

For λ = 0.01 or smaller, DCG fails to converge from the start. The sign of the divergence can be observed or detected inside the CG solver after some number of CG iterations. This becomes a very important and useful information. We can use it as a criterion to decide the termination of the WSDCG3DMT code. Whenever a divergence inside the CG loop takes place, program is stopped. The cause for the divergence behavior inside the CG loop is probably due to the loss of the orthogonality of matrix **R**.

From all of these experiments, we can infer that both 2-D studies from Siripunvaraporn and Egbert (2007) and 3-D studies here yield almost the same conclusions. Optimal convergence occurs in the λ ranges between 0.1 and less than 10, and also with $r_{tol} = 10^{-2}$.

Computational performance in term of memory and CPU time of WSDCG3DMT is then compared with those from WSINV3DMT. Majority of the memory requirements for

WSINV3DMT is to store **J** and **R** matrices which can be approximated from $8NM+8NN$ with double precisions. This is about 1 GBytes in our test case. The code also requires less than 0.3 GBytes for storing **S**, $\partial\mathbf{S}/\partial\mathbf{m}$, and other parts for miscellaneous computations. For WSDCG3DMT, we do not store **J** and **R** in the memory. One GBytes of RAM is therefore not needed as in the case of WSINV3DMT. WSDCG3DMT requires only about 0.4 GBytes to store many different matrices and vectors. This is about the same as the memory used for the miscellaneous computations in WSINV3DMT.

In term of computational time, WSINV3DMT converges to the desired misfit within three iterations in about 300 minutes as shown in a black line of Figure 2, while WSDCG3DMT with $\lambda = 1$ and $\lambda = 0.1$ uses about 400 and 1600 minutes, respectively. This again shows that computational time of WSINV3DMT is less than that of converged WSDCG3DMT. Thus, in term of computational performance, one can clearly see that WSDCG3DMT has advantage in terms of memory. However, its computational time can be significantly greater than that of WSINV3DMT. A trade-off between computational time and memory used would be a factor for users to decide. This is also similar to the 2-D studies (Siripunvaraporn and Egbert, 2007).

In 2-D studies, we did not compare CPU time, but number of forward modeling calls of each algorithm. Here, similar analysis are performed for the 3-D cases. WSINV3DMT requires a fix number of callings at $N_pN_sN_m + N_p(N_\lambda+1)$ per inversion iteration to form the sensitivity and compute the misfit, where N_λ is a number of λ varied to search for the minimum misfit in each iteration of the Occam's inversion. In our experiments, for the first iteration, $N_\lambda = 5$, number of forward modeling calls for WSINV3DMT is therefore at 2,656. For WSDCG3DMT, in each iteration, number of forward modeling calls depends on a number of CG iterations (N_{cg}) in the DCG inner loop, and equal to $4N_p + 2N_pN_{cg}$ per inversion iteration as we previously discussed. In our experiments, for the case $\lambda = 1$ and $r_{tol} = 10^{-1}$, $N_{cg} = 47$ for the first iteration, number of forward modeling calls is then at 1,568.

Although number of forward modeling calls of WSDCG3DMT is about 1,000 less than WSINV3DMT, computational time is actually slightly longer for the first iteration of both methods as shown in Figure 2. This indicates that for each forward modeling call, WSDCG3DMT requires averagely longer runtime than that of WSINV3DMT. Because of more complicated right hand sides in the system of equation (5) when computing $\mathbf{Jp}$ or $\mathbf{J}^T\mathbf{q}$ than

forming **J**, as already stated in Section 2.4, it requires larger number of QMR iterations to converge to the solution. This study shows that to test the efficiency of the inversion, just counting number of forward modeling calls can be misleading (see Newman and Alumbaugh, 1997; Siripunvaraporn and Egbert, 2007).

Another interesting point for WSDCG3DMT is the reduction of the number of CG iterations per outer loop iteration when misfit becomes lower. For example, in the case $\lambda = 1$ and $r_{tol} = 10^{-2}$, N_{cg} = 108, 48, 25 and 21, respectively, from the first to forth iteration of the main inversion loop. This is reflected and shown with lesser CPU time for successive iterations in Figure 2. The reduction of number of CG iterations occurs on every case in our examples. When inverted solution gets closer to the "true" solution, normal equation (3) is probably lesser stiff and therefore become easily to solve.

4. The mixed scheme of the DCG and Occam's inversions (WSMIX3DMT)

Because DCG does not explicitly form and store the sensitivity matrix, DCG therefore requires significantly less memory than the Occam's inversion. However, the major drawback of the DCG method is its computational time which could be longer than the Occam's inversion. Here, we propose a new scheme which is a mixed concept of both DCG and Occam and a modification of the DCG method. Mathematics of the new scheme is in fact identical to the DCG method. Thus, it maintains the memory advantage of the DCG method over the Occam's style. However, we intentionally design so that the new scheme spends computational time less than both DCG and Occam. This would make the mixed scheme as the efficient inversion.

Assuming that the goal of the inversion is the same for both DCG and Occam that is to bring the misfit down to the desired level. One distinct feature between both methods is at the λ value. In Occam's inversion (Constable et al., 1987; Siripunvaraporn and Egbert, 2000; Siripunvaraporn et al., 2005), in every iteration, λ in equation (3) is varied in order to search for the model producing the "least" RMS misfit (see Siripunvaraporn and Egbert, 2000; Siripunvaraporn et al., 2005). With the Occam concept, λ is posed as both the step length and the regularization parameters. For the DCG method, λ is pre-selected and fixed in every iteration as shown in previous section in WSDCG3DMT. In DCG, λ therefore acts like a regularization or damping parameter.

In our mixed scheme, the algorithm is based mainly on the DCG method. However, λ is not fixed but varied as both step length and regularization parameter similar to the idea of the Occam's inversion. The difference from the Occam's method is we do not choose λ that minimize the RMS misfit, but we select λ that can both lower the misfit down and at the same time require small number of CG iterations per an outer loop iteration. The "optimal" λ is selected and varied based on our knowledge and experience gained from the studies in previous section 3. It is therefore not exactly the same philosophy as in the Occam's inversion, nor the DCG, but a mixed of both. This is why we refer to this method as a mixed DCG and Occam or in short WSMIX3DMT.

Based on earlier 3-D studies in section 3 and 2-D studies in Siripunvaraporn and Egbert (2007), r_{tol} for the inner CG loop is fixed at 10^{-2} as the optimal tolerance level for number of CG iterations. For early iterations, larger λ requires significantly smaller number of CG iterations than smaller λ and at the same time can lower the misfit down. We therefore choose to start our mixed scheme with large λ_{ini} (e.g., $\lambda_{ini} = 100$ or larger). To further decrease the misfit down, λ is automatically reduced by a factor of ε (e.g., $\varepsilon = 10$) in the next iteration. This automatic reduction is to avoid redundant computations as occurred when large λ is fixed (Figure 2). A reduction in λ was used before in Kelbert et al. (2008) but only when the misfit is not decreased in their non-linear conjugate gradient (NLCG) method. The automatic reduction in λ is continued successively for the next iterations until reaching λ_{min} (e.g., $\lambda_{min} = 0.1$). When λ below λ_{min}, it will set back to λ_{min}.

For example, $\lambda_{ini} = 100$, $\lambda_{min} = 0.1$ and $\varepsilon = 10$ is input in the first iteration. Values of λ for the 2nd, 3rd and 4th iterations would be 10, 1 and 0.1, respectively. If the inversion continues, 5th iteration and so on will have $\lambda = 0.1$. In addition, we also add a scheme to detect the divergence. Within N_{div} CG iterations (e.g., $N_{div} = 15$), if the divergence occurs, there is a high possibility that the inversion will fail to converge. If that happens, λ is automatically increased by a factor of ε and re-start the process again. This "extra" step may cause redundant computations but can help preventing the divergence inside the main inversion loop.

4.1 Numerical Experiments of WSMIX3DMT and Comparisons with WSINV3DMT and WSDCG3DMT

To check the efficiency of the WSMIX3DMT code, we apply it to the same synthetic data set generated from model in Figure 1. Four values of λ_{ini} are used (λ_{ini} = 10000, 1000, 100 and 10) with ε = 10. Figure 4 shows convergence rates from the WSMIX3DMT program with various initial λ_{ini}, in comparisons to those of WSINV3DMT (black) and WSDCG3DMT with λ = 1 (red). Figure 4 shows that all runs can converge to the desired level within 3-4 iterations. Most importantly, all WSMIX3DMT runs spend computational time less than both WSINV3DMT and fixed λ WSDCG3DMT. Inverted models from all runs with 1 RMS are similar to the inverted model plotted in Figure 3.

When λ_{ini} is too large (i.e. at 10000), redundant computation is occurred in the first iteration. Although the first iteration with λ_{ini} = 10000 runs very quick, it does not greatly reduce the misfit. When λ is decreased to 1000 in the next iteration. The misfit in this case is almost the same as starting the run with λ_{ini} = 1000. The first iteration of λ_{ini} = 10000 is therefore redundant and unnecessary. Starting the mixed inversion with $\lambda_{ini} \leq 10$ requires large computational time due to large number of CG iterations used in the first iteration. In addition, λ is decreased quickly to 1 and 0.1 in the next few iterations and would demand large number of CG iterations. In this case, we do not gain advantage of small number of CG iterations used from larger λ. It therefore become less effective as in WSDCG3DMT. Thus, we should avoid to start WSMIX3DMT with smaller λ or very large λ.

From the experiments, the "optimal" λ to start with would be around 100 to 1000 (Figure 4). Both cases spends computational time at about 100 minutes compared to 300 minutes of WSINV3DMT and 400 minutes of WSDCG3DMT. In addition, WSMIX3DMT requires memory the same as WSDCG3DMT, i.e. less than 0.4 Gbytes for this dataset, which is several factors less than WSINV3DMT. WSMIX3DMT which is a combination of DCG and Occam is the most efficient method compared to both WSINV3DMT and WSDCG3DMT.

Further studies show that ε around 10 is the optimal value. If ε too small, redundant computations can be occurred. If too large, WSMIX3DMT would not gain much advantage from smaller number of CG iterations when large λ used. This makes WSMIX3DMT less efficient.

5. Applications of WSMIX3DMT, WSDCG3DMT and WSINV3DMT to EXTECH data

To show the efficiency of our mixed scheme WSMIX3DMT in comparisons to the WSDCG3DMT and WSINV3DMT codes, we applied all three codes to the EXTECH dataset (Tuncer et al., 2006) conducting around the McArthur River mine, Saskatchewan, Canada (Figure 2 of Tuncer et al., 2006). The data consists of both impedance tensor (Z_{xx}, Z_{xy}, Z_{yx} and Z_{yy}) and the vertical magnetic field transfer function (VTF; T_{zx} and T_{zy}) for 131 stations and 16 periods (from 8000 Hz to 5 Hz). The data parameter N is therefore equal to 25,152. In all runs, minimum error bars for VTF is set at 15% of $(|T_{zx}|^2+|T_{zy}|^2)^{½}$ and 5% of $|Z_{xy}Z_{yx}|^{½}$ for off-diagonal and 50% for diagonal terms. A 1000 Ω m half-space is used as an initial model and a prior model ($\mathbf{m_0}$) and is discretized at 56 × 56 × 33 (+7 air layers). The model parameter M is therefore at 103,488.

To show the efficiency of the parallel codes, all runs are performed on a cluster computer which consists of 8 processor nodes with 8 GBytes in memory each. With 16 period data, two periods are distributed to compute on each processor node. In terms of memory, WSINV3DMT requires about 5 GBytes to store its two period sensitivities and the cross-product matrices. It also requires about 1 GBytes additional to store other necessary components. In contrast to WSINV3DMT, both WSDCG3DMT and WSMIX3DMT require less than 1 GBytes of RAM to perform the inversion of this EXTECH dataset. The EXTECH dataset and the model mesh used above are already at a maximum limitation of the cluster for WSINV3DMT. Because WSDCG3DMT and WSMIX3DMT use significantly less memory, they can therefore be applied on a bigger dataset and a bigger mesh on this cluster. However, here, same parameters are used for comparisons.

Convergence behaviors of the three methods are plotted in Figure 5 as a function of time in minutes. From Figure 5, WSINV3DMT requires about 870 minutes in 3 iterations to converge to its minimum at 1.52 RMS. After the 3rd iteration, the misfit is fluctuated above the minimum RMS. WSDCG3DMT with $\lambda = 1$ also requires 3 iterations to converge to 1.50 RMS but uses longer CPU time at about 1040 minutes. After the 3rd iteration, WSDCG3DMT increases its RMS to 1.57 in the 4th iteration and is terminated because of the divergence. With $\lambda < 0.5$, the WSDCG3DMT code diverges and fails after its first iteration.

For our mixed scheme, WSMIX3DMT with λ_{ini} = 100 can converge to 1.47 RMS slightly below the level of both WSINV3DMT and WSDCG3DMT in 3 iterations. Most importantly, the computational time is only about 450 minutes, about half of WSINV3DMT and WSDCG3DMT. At the 4th iteration when λ is reduced to 0.1, the scheme detected the divergence occurring inside the CG loop. The code is then re-started with a bigger λ = 1 on the 4th iteration. The process of increasing λ will cost some extra computational time. With the divergence detection scheme, the code can continue to run for several iterations.

After continuing the run, WSMIX3DMT can further reduce the misfit below the level that both WSINV3DMT and WSDCG3DMT can attain. At 5th iteration with λ = 1, the misfit is at the lowest RMS of 1.34. However, these 0.13 RMS difference from 3rd to 5th iteration require computational time almost 14 hours; about twice longer than the CPU time at the 3rd iteration. One can therefore stop at the 3rd iteration because the inverted models at the 3rd and 5th iteration are slightly different.

Convergence behavior from starting WSMIX3DMT with λ_{ini} = 1000 is redundant in early iterations similar to starting with λ_{ini} = 100, as shown in Figure 5. It therefore spends "extra" CPU time longer. Overall, it can still converge below 1.5 RMS within 500 minutes faster than both WSINV3DMT and WSDCGMT methods.

Inverted model from the 5th iteration of WSMIX3DMT starting with λ_{ini} = 100 is shown in Figure 6. It is similar to the inverted model from WSINV3DMT (Figure 11 of Siripunvaraporn and Egbert, 2009). Major differences are at the two conductors. Here, conductor on the eastern part of the profiles oriented in the NE-SW direction can be seen as shallow as 500 m depth. Northern conductor seems to be continuous from 800 m to 1.3 km depth. The difference of the two inverted models (Figure 6 here and Figure 11 of Siripunvaraporn and Egbert, 2009) and detail interpretation is beyond our scopes in this paper. For detail discussion of the EXTECH data set can be found in Tuncer et al. (2006) and Farquharson and Craven (2008).

6. Conclusions

In this paper, we implement and extend the data space conjugate gradient inversion for three-dimensional Magnetotelluric data (WSDCG3DMT). Numerical experiments on 3-D synthetic data show that WSDCG3DMT with some λ can converge to the desired level of misfit but often

spends longer computational time than the data space Occam's inversion (WSINV3DMT). However, because the whole sensitivity matrix is not explicitly formed and stored, its memory requirements are therefore minimal at a fraction of WSINV3DMT. This makes WSDCG3DMT practical for large to very large data set.

Based on the numerical experiments of WSDCG3DMT on synthetic data, number of CG iterations depends greatly on the λ values used. Larger λ usually requires smaller number of CG iterations per main inversion iteration but hardly converge to the "true" solution. Smaller λ requires larger number of CG iterations per main iteration but can converge to the desired level of misfit. However, if λ is too small, it can diverge. Computational time varies proportionally to the number of CG iterations. Thus, to use less CPU time, number of CG iterations per outer loop iteration must be minimized.

The information learned from the synthetic studies has inspired and led us to the creation of the mixed scheme of the Occam's and DCG methods or WSMIX3DMT. In DCG scheme, λ is fixed as a regularization parameter. In Occam's inversion, λ is varied as both step length and regularization parameters. In our mixed scheme, λ is varied but not in the same way as in the Occam's inversion. Instead of choosing λ that generates a model with smallest misfit as in Occam, we prefer λ that minimizes number of CG iterations but at the same time can reduce the misfit. With this strategy, λ should initially start from large value before reducing to smaller value for the next subsequent iterations. Our studies shows that λ between 100 to 1000 are the optimal λ to start with for the WSMIX3DMT code.

By applying all three algorithms (WSMIX3DMT, WSDCG3DMT and WSINV3DMT) on both synthetic and EXTECH field data, our mixed scheme (WSMIX3DMT) is significantly faster than both WSDCG3DMT and WSINV3DMT. Similar to WSDCG3DMT, it requires insignificant amount of memory. Because both computational time and memory performances are at minimum, we can conclude here that WSMIX3DMT is the most efficient inversion.

7. Acknowledgements

This research has been supported by the Thai Center of Excellence in Physics (ThEP) and by Thailand Research Fund (TRF: RMU5080025). The authors would like to thank XXX and XXX and the editor for their comments to help improve the manuscript. The authors would like to thank Jim Carven for allowing us to use EXTECH data sets

8. References

- Avdeev, D., and Avdeeva, A., 2009, 3D Magnetotelluric inversion using a limited-memory quasi-Newton optimization, Geophysics, 74, F45-F57.
- Árnason, K., Eysteinsson, H., Hersir, G.P., 2010, Joint 1D inversion of TEM and MT data and 3D inversion of MT data in the Hengill area, SW Iceland, Geothermics, 39, 13-34
- Barret, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and Van der Vorst, H., 1994, Templates for the solution of linear systems: Building blocks for iterative methods: Soc. Ind. Appl. Math.
- Boonchaisuk, S., C. Vachiratienchai and W. Siripunvaraporn, 2008, Two-dimensional direct current (DC) resistivity inversion: data space Occam's approach, Phys. Earth. Planetary Interiors, 168, 204-211.
- Constable, C. S., Parker, R. L., and Constable, C.G., 1987, Occam's inversion: A practical algorithm for generating smooth models from electromagnetic sounding data, *Geophysics*, 52: 289-300.
- Goldak, D. and Kosteniuk, P., 2010, 3D Inversion of Transient Magnetotelluric Data: an Example from Pasfield Lake, Saskatchewan, EGM 2010 International Workshop, 11-14 April, 2010. Capri, Italy.
- Farquharson C.G., and Craven, J.A., 2008, Three-dimensional inversion of Magnetotelluric data for mineral exploration: An example from the McArthur River uranium deposit, Saskatchewan, Canada., Jour. Applied. Geophysics., 68, 450-458.
- Han, N., Nam, M.J., Kim, H.J., Lee, T.J., Song, Y., Suh, J.H., 2008, Efficient three-dimensional inversion of Magnetotelluric data using approximate sensitivities, Geophys. J. Inter, 175, 477-485.

- Heise, W., Caldwell, T.G., Bibby, H.M., Bannister, S.C., 2008, Three-dimensional modelling of magnetotelluric data from the Rotokawa geothermal field, Taupo Volcanic Zone, New Zealand, Geophys. J. Inter, 173, 740-750.
- Hill, G.J., Caldwell, T.G., Heise, W., Chertkoff, D.G., Bibby, H.M., Burgess, M.K., Cull, J.P., Cas, R.A.F., 2009, distribution of melt beneath Mount St Helens and Mount Adams inferred from Magnetotelluric data, Nature Geoscience, 2, 785-789.
- Ingham, M.R., Bibby, H.M., Heise, W., Jones, K.A., Cairns, P., Dravitzki, S., Bennie, S.L., Caldwell, T.G., Ogawa, Y., 2009, A Magnetotelluric study of Mount Ruapehu volcano, New Zealand, 2009, Geophys. J. Inter., 179, 887-904.
- Kelbert, A., Egbert, G.D., and Schultz, A., 2008, Non-linear conjugate gradient inversion for global EM induction: resolution studies, Geophys. J. Int, 173, 365-381.
- Lin, C., Tan, H., and Tong, T., 2009, Parallel rapid relaxation inversion of 3D Magnetotelluric data, Applied Geophysics, 6, 77-83.
- Lin, C., Tan, H., and Tong, T., 2008, Three-dimensional conjugate gradient inversion of Magnetotelluric sounding data, Applied Geophys., 5, 314-321.
- Mackie, R. L., and Madden, T. R., 1993, Three-dimensional magnetotelluric inversion using conjugate gradients, *Geophys. J. Int*, 115: 215-229.
- Newman, G. A., and D. L. Alumbaugh, 1997, Three-dimensional massively parallel electromagnetic inversion – I. Theory, Geophys. J. Int, 128, 345-354.
- Newman, G. A., and D. L. Alumbaugh, 2000, Three-dimensional magnetotelluric inversion using non-linear conjugate gradients, Geophys. J. Int, 140, 410-424.
- Parker, R. L., 1994, Geophysical Inverse Theory, Princeton University Press.
- Patro, P.K. and Egbert, G.D., 2008, Regional conductivity structure of Cascadia: Preliminary results from 3D inversion of USArray transportable array Magnetotelluric data, Geophys. Res. Lett., 35, art. no. L20311.
- Rodi, W. L., 1976, A technique for improving the accuracy of finite element solutions for Magnetotelluric data, Geophys. J. Roy. Astr. Soc., **44**, 483-506.
- Sasaki, Y., 2001, Full 3D inversion of electromagnetic data on PC., *J. Appl. Geophys.*, 46, 45-54.

- Sasaki, Y. and Meju, M.A., 2006, Three-dimensional joint inversion for Magnetotelluric resistivity and static shift distributions in complex media, Jour. Geophys. Res. B: Solid Earth, 111, art. no. B05101.
- Siripunvaraporn, W. and Egbert, G., 2000. An effcient data-subspace inversion method for 2D magnetotelluric data, *Geophysics*, **65**(3), 791-803.
- Siripunvaraporn, W., Egbert, G., & Lenbury, Y., 2002. Numerical accuracy of magnetotelluric modeling: A comparison of finite difference approximations, *Earth Planets Space*, **54**(6), 721-725.
- Siripunvaraporn, W., M. Uyeshima and G. Egbert, 2004, Three-dimensional inversion for Network-Magnetotelluric data, *Earth Planets Space*, 56, 893-902.
- Siripunvaraporn, W., Egbert, G., Lenbury, Y., & Uyeshima, M., 2005, Three-dimensional Magnetotelluric inversion: data-space method, *Phys. Earth and Planetary Interiors*, **150**, 3-14.
- Siripunvaraporn, W., and Egbert, G., 2007, Data space conjugate gradient inversion for 2-D Magnetotelluric data, Geophys. Jour. Inter., 170, 986-994.
- Siripunvaraporn, W. and Egbert, G., 2009, WSINV3DMT: Vertical magnetic field transfer function inversion and parallel implementation, Phys. Earth. Planet. Interiors., 173, 317-329.
- Streich, R., 2009. 3D finite-difference frequency-domain modeling of controlled-source electromagnetic data: Direct solution and opti-mization for high accuracy, *Geophysics*, **74**(5), F95-F105.
- Tuncer, V., M. J. Unsworth, W. Siripunvaraporn, and J.A. Craven, 2006, Exploration for unconformity-type uranium deposits with audiomagnetotelluric data: A case study from the McArthur River mine, Saskatchewan, Canada, *Geophysics*, 71, B201-B209.
- Türkoğlu, E., Unsworth, M., Pana, D., 2009, Deep electrical structure of northern Alberta (Canada): Implications for diamond exploration, Canadian Jour. Of Earth Sciences, 46, 139-154.
- Uyeshima, M., 2007, EM monitoring of crustal processes including the use of the Network-MT observations, Surveys in Geophys., 28, 199-237.
- Zhdanov, M.S., S. Fang, G. Hursan, 2000, Electromagnetic inversion using quasi-linear approximation, *Geophysics*, 65, 1501-1513.

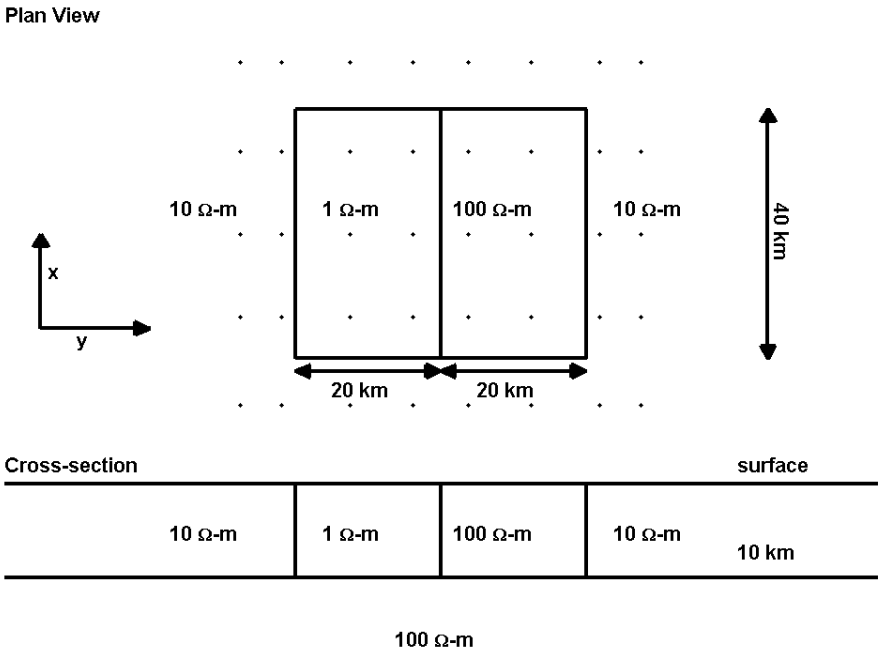


Figure 1. Two-block synthetic model used to test our inversions. The solid dots indicate the observational sites. A cross-section view in the lower panel is a profile cutting across the middle of the two anomalies in the upper panel, and is not to scale (after Siripunvaraporn et al., 2005; and Siripunvaraporn and Egbert, 2009).

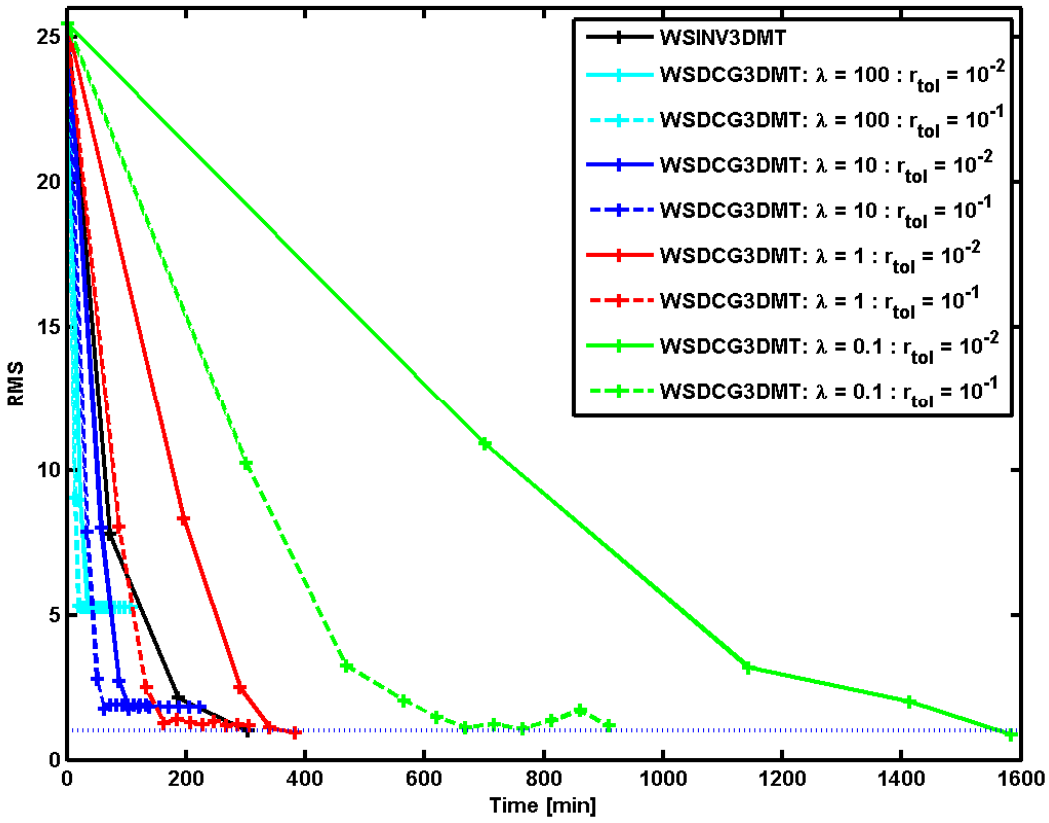


Figure 2. Convergence rates of WSINV3DMT (black) and WSDCG3DMT from various λs and r_{tol} to the synthetic dataset generated from a model in Figure 1. Dash line for $r_{tol} = 10^{-1}$. Solid line for $r_{tol} = 10^{-2}$. Each plus symbol indicates one iteration.

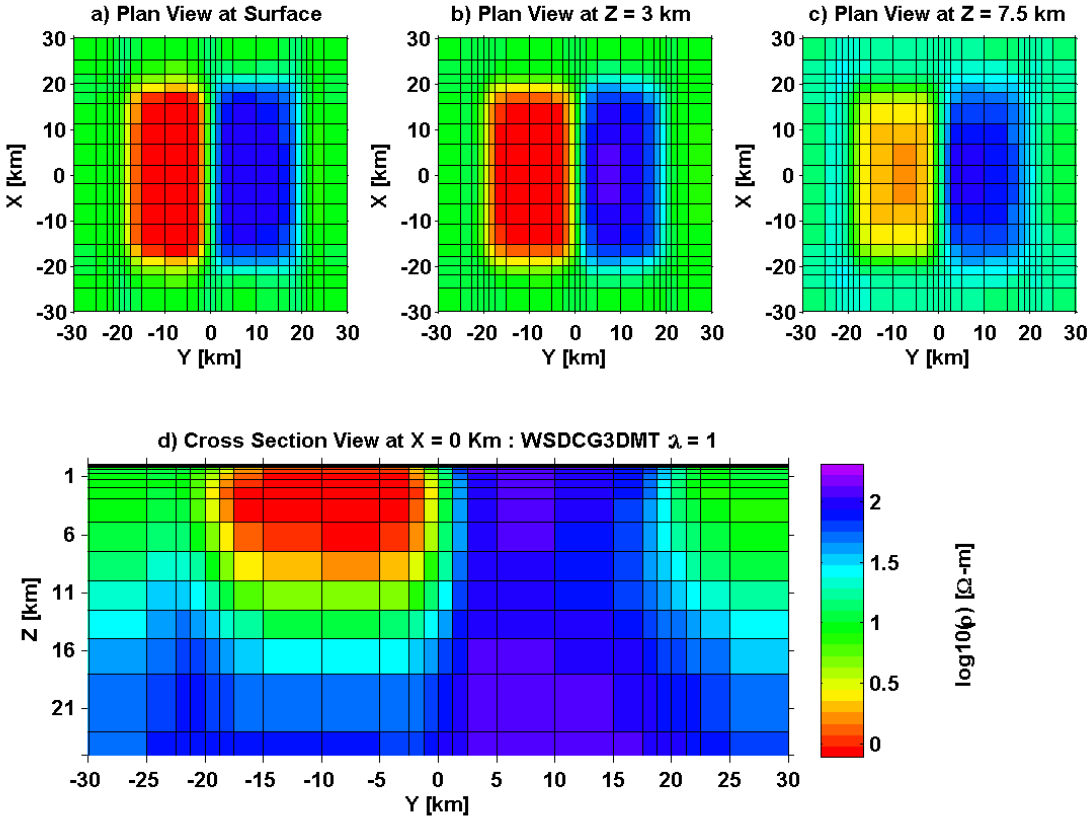


Figure 3. An inverted model from WSDCG3DMT with λ = 1. The synthetic data is generated from the model in Figure 1. The top panels (a)–(c) is a plan view at the surface, at 3 km and at 7.5 km depth, and the bottom panel (d) is a cross-section view cutting across the two anomalies at $X = 0$ km. The solution is shown only in the central area around the anomalies, not for the full model domain.

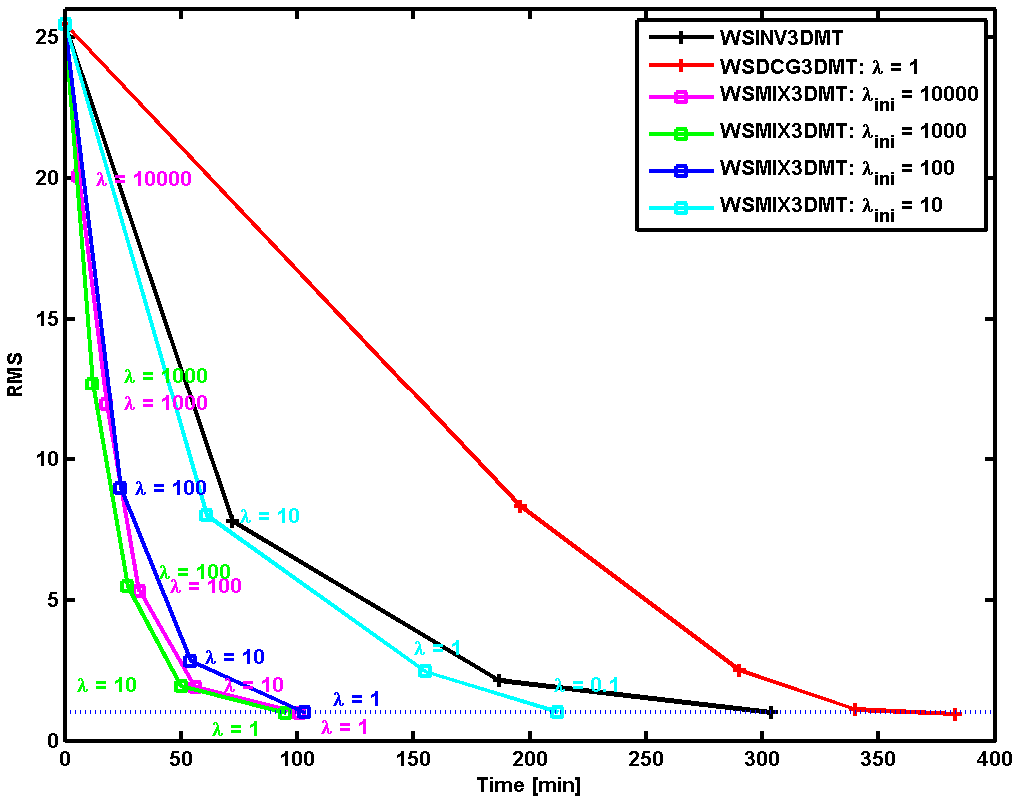


Figure 4. Convergence rates from WSINV3DMT (black), WSDCG3DMT with $\lambda = 1$ (red) and WSMIX3DMT with different initial λ_{ini} to the synthetic data generated from a model in Figure 1. Each square or plus symbol indicates one iteration. λ used in each iteration for WSMIX3DMT is printed next to its square symbols.

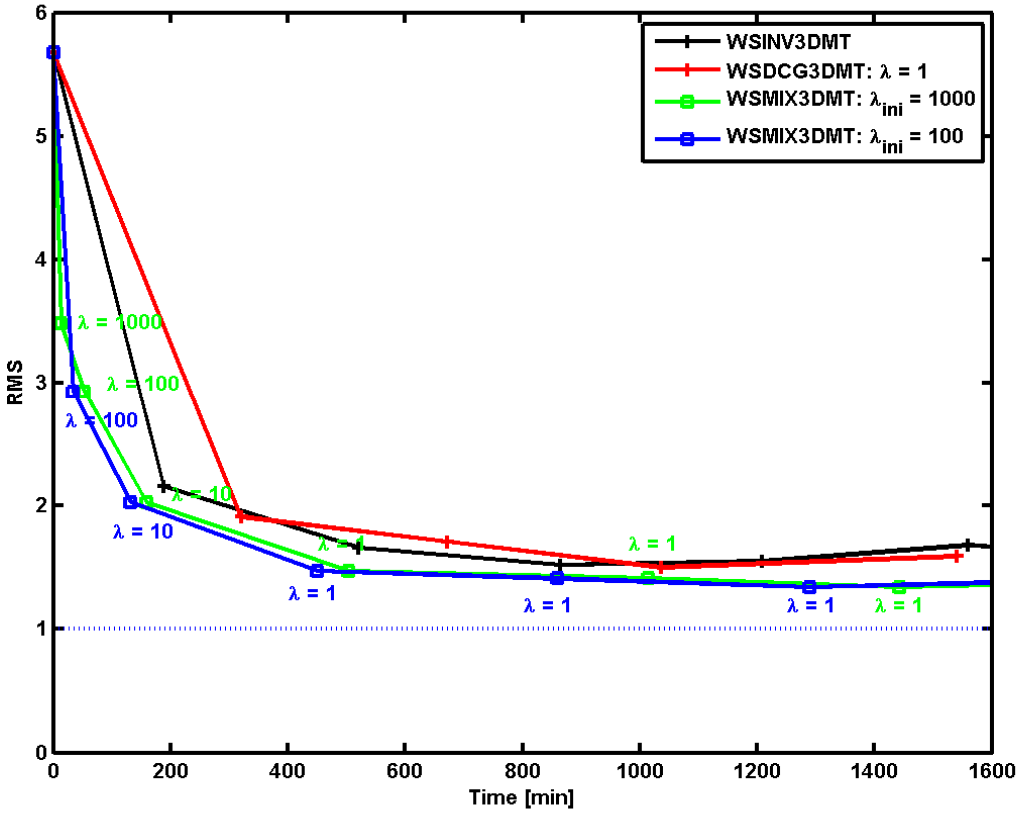


Figure 5. Convergence rates from WSINV3DMT (black), WSDCG3DMT with λ = 1 (red) and WSMIX3DMT with initial λ_{ini} = 1000 (green) and λ_{ini} = 100 (blue) to the EXTECH field dataset. Each square or plus symbol indicates one iteration. λ used in each iteration for WSMIX3DMT is printed next to its square symbols.

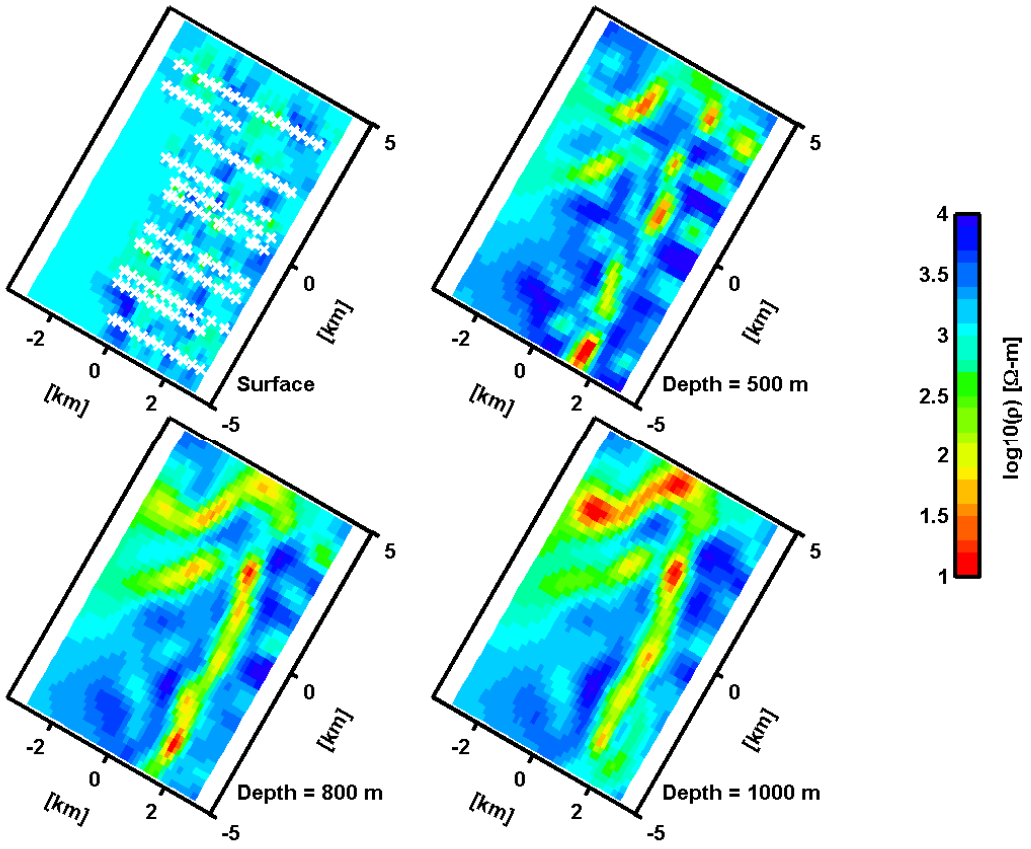


Figure 6. The inverse solution at various depths from the 5th iteration of the WSMIX3DMT method with initial λ_{ini} = 100. The EXTECH data used here consists of both vertical magnetic transfer function and full impedance tensor at 131 sites and 16 periods. The cross-symbols indicate the locations of the stations.

ภาคผนวก ข. Manuscript

Rung-Arunwan T. and **W. Siripunvaraporn**, 2010, An efficient modified hierarchical domain decomposition for 2-D Magnetotelluric forward modeling, submitted *Geophysical Journal International*, moderate revision.

An Efficient Modified Hierarchical Domain Decomposition for 2-D Magnetotelluric Forward Modeling

Tawat Rung-Arunwan[1,2] and Weerachai Siripunvaraporn[1,2]

[1]Department of Physics, Faculty of Science, Mahidol University, Rama 6 Rd., Rachatawee, Bangkok, 10400, THAILAND

[2] ThEP, Commission of Higher Education, 328 Si Ayuthaya Road, Bangkok 10400, THAILAND

Abstract

We use 2-D Magnetotelluric (MT) problems as a feasibility study to demonstrate that the 3-D MT modeling can be solved with a direct solver, even on a standard single processor PC. The scheme used is the hierarchical domain decomposition (HDD) method in which a global computational domain is uniformly split into many smaller non-overlapping subdomains. However, to make it more efficient, two modifications are made to the standard HDD method. Instead of three levels as in the standard HDD method, we classify the unknowns into four classes: the interiors, the horizontal and vertical interfaces and the intersections taking advantages of the finite-difference approximation. Four sets of smaller systems of equations are successively solved with a direct method (an LU factorization). The separation helps overcoming the memory overburden of a direct solver while remain computationally effective. To further enhance the speed of the code, a red-black ordering is applied to solve the horizontal and vertical interface reduced systems.

Numerical experiments on 2-D MT problem running on a single processor machine shows that CPU time and memory used are almost constant for any resistivity models, frequencies and modes as long as the model size remain the same. This is a clear advantage of our algorithm. Number of subdomains is a major factor controlling computational efficiency. Here, we also introduce a “memory map”, a tool we can use to pre-select “optimized” subdomains. Our 2-D experiments also shows that by splitting a domain with the optimized subdomains, this modified scheme can outperform the standard FD method in both CPU time and memory even running on a serial machine.

1. Introduction

To obtain magnetotelluric (MT) responses, the second order Maxwell's equation in either electric field or magnetic field is solved via three commonly used approaches: finite difference (FD) method (e.g. Mackie et al., 1994; Smith, 1996; Siripunvaraporn et al., 2002; Siripunvaraporn et al., 2005), finite element (FE) method (e.g. Wannamaker et al., 1987; Zyserman et al., 1999; Zyserman and Santos, 2000; Mitsuhata and Uchida, 2004;), and integral equation (IE) technique (e.g. Wannamaker, 1991; Xiong, 1992; Avdeed and Avdeeva, 2009). For complicated and geologically realistic two-dimensional (2D) and three-dimensional (3D) model, FD or FE methods are generally more efficient and robust than IE technique. In the past decades, FD method has gained more popularity due to its simplicity in technique and also its accuracy in solution.

In many problems, when model domain becomes very large, particularly in 3-D problems, solving the system of equations with the direct method is impractical in term of memory requirement (see Ben-Hadj-Ali et al., 2008 for 3-D frequency-domain full-waveform tomography; Streich, 2009 for 3-D MT;). The system is then alternatively solved with the iterative solvers (e.g. Bi-Conjugate Gradient (BiCG) method in Smith, 1996 and Xiong, 1999; Quasi Minimum Residual (QMR) in Siripunvaraporn et al., 2002; Preconditioned Conjugate Gradient (PCG) in Siripunvaraporn and Egbert, 2000; Minimum Residual Method (MRM) in Mackie et al., 1994). In many practical MT cases, the electrical resistivity model can be geologically complicated resulting in large conditioned number and therefore long computational time (see Patro and Egbert, 2009). Occasionally, the iterative solvers may become stagnant after many thousand of iterations and sometimes fail to converge. The calculated solution will therefore not be accurate and could mislead an interpretation if applied inside an inversion.

In high conditioned number case, being able to solve a problem with a direct solver is very crucial, if applicable. With direct method, accuracy is guarantee. Computational time is also controllable, because theoretically it is almost constant for any frequencies, modes or polarizations and resistivity models as long as the model domain remains the same size. In addition, the factorization used when solving the system can be re-used many times when

computing the sensitivity or Jacobian matrix (see Siripunvaraporn and Egbert, 2000) inside the inversion algorithm. In 3-D MT cases, the direct solver is still not practical with recent computer technology (see Streich, 2009). However, here we use the 2-D study to demonstrate that the 3-D problem can be efficiently solved with a direct solver even on a serial machine if the modified hierarchical domain decomposition developed in this paper is applied to.

Instead of computing on a large domain, a global domain can be splitting into several smaller local domains or subdomains. The solution on the global domain is then solved through the smaller systems of each subdomain. This technique is generally known as the domain decomposition (DD) technique. It is considered as a powerful tool in many large scale engineering problems (e.g. Lu and Shen, 1997; Bitzarakis et al., 1997; Larsson, 1999; Yin et al., 2002; Basermann et al., 2005; Lu et al., 2008; Wang et al., 2008;) and also in various multidimensional geophysical problems (e.g. Xiong, 1999; Zyserman et al., 1999; Zyserman and Santos, 2000; Xie et al., 2000; Pain et al., 2002; Ben-Hadj-Ali et al., 2008; Sourbier et al., 2008; Takei et al., 2010).

The domain decomposition method can be mainly classified into two categories: the overlapping technique where some region of the subdomain overlapping with the others (e.g. Xiong, 1999; Peng et al., 2009) and the non-overlapping method where neighboring subdomains share the same sub-boundaries (e.g. Lu and Shen, 1997; Zyserman et al., 1999; Zyserman and Santos, 2000; Lu et al., 2008; Wang et al., 2008). Comparison of the overlapping and the non-overlapping methods is mentioned in Chan and Goovaerts (1992) and Rice et al. (2000). Various schemes are used to solve the domain decomposition problems, such as the Schwartz algorithms (see Cai et al., 1998), Schur complement approach (see Smith et al., 1996; Saad, 2003; Zhang, 2005), the hierarchical domain decomposition approach (Smith et al., 1996; Takei et al., 2010), balancing domain decomposition method (Mandel, 1993), the interface relaxation methods (see Rice et al., 2000) among many other techniques.

In electromagnetic induction of the Earth, there are only a few papers demonstrating the use of domain decomposition method to solve MT forward problems. Zyserman et al. (1999) and Zyserman and Santos (2000) applied non-overlapping domain decomposition technique to 2-D and 3-D cases, respectively. In their techniques, sub-problems are iteratively solved via the interfaces enforced by the equivalent Robin-type transmission conditions. The memory requirement is significantly diminished due to no appearance of a large global matrix. Computational time is also greatly reduced when solving in the parallel computation (Zyserman and Santos, 2000). Although, the technique has proven to be numerically superior in the parallel system, the technique may not be suitable for serial computation. Xiong (1999) applied adaptive Schwartz overlapping domain decomposition technique for 3-D controlled source electromagnetic forward problems. In his method, all subdomains share overlapping regions. Each subdomain is independently solved and then updated from neighboring subdomains until the solution converges. The memory is significantly reduced. However, its total computational run time becomes larger than solving the whole system on single node processor (Xiong, 1999). Both schemes (Xiong, 1999; Zyserman et al., 1999; and Zyserman and Santos, 2000) show that efficiency in terms of computational time of the domain decomposition method can only be gained if running on parallel system. They are inferior if running on a serial machine.

In this paper, we investigated another method based on the hierarchical domain decomposition (HDD). Similar to other domain decomposition methods, the global domain is subdivided into many smaller subdomains. System of equations for each subdomain is separately formed and linked to the other via the interfaces. The hierarchical domain decomposition method can be directly applied to the MT problems both parallel and serial computations. Application of HDD on a parallel system is straightforward. Similar to others, calculation of each subdomain is performed separately on each processor node. A single interface system is then distributed to all processors for calculation. Theoretically, efficiency can be expected from applying the code to the parallel system. However, in practice, this parallel scheme requires substantial amount of communication time to exchange data among processors, particularly when solving the interface system. Efficiency is therefore platform-dependent. In this paper, we only illustrate the parallel algorithm but prefer not to demonstrate it numerically because our 2-D domain problem is “too” small for current

computer technology. The parallel algorithm will be later demonstrated on a bigger 3-D problem as a future research. In addition, this parallelization is not our main challenge. Our major challenge is the efficiency enhancement of HDD on a serial machine, not through a multi-processor machine.

Similar to other domain decomposition methods for MT problems (Xiong, 1999; Zyserman et al., 1999; and Zyserman and Santos, 2000), efficiency of HDD on a serial computation is low. However, in this paper, two modifications are developed and applied to the hierarchical domain decomposition method to increase its efficiency. First modification is the separation of interfaces into vertical and horizontal interfaces. This is natural for the finite-difference approximation scheme. Second modification is the application of red-black ordering to the re-ordered interface systems. With the two modifications, we will show that the modified HDD code for 2-D MT problems performs better than the conventional method even on a serial machine. Because we use a direct solver to solve system of equations, this 2-D experiment is also a feasibility study for future 3-D problems to demonstrate that the direct solver can be used to solve 3-D system of equations even with a serial calculation. These are therefore our main objectives for this paper.

Efficient modified HDD on a serial computation can also be applied to the parallel system. However, instead of parallelizing over subdomains, we parallelize over frequency. Calculation of MT responses of each frequency is performed serially on one processor. Thus, all frequencies are solved simultaneously but separately on multi-processor machines. This is used frequently in 3-D inversion algorithms (see Siripunvaraporn et al., 2004; 2005; Siripunvaraporn and Egbert, 2009; Siripunvaraporn and Sarakorn, 2010). In addition, this scheme does not require substantial amount of communication time between processors. It is therefore perfectly fit with the PC cluster platform which can be easily and cheaply built.

In addition, a major decisive factor that controls the efficiency of the modified HDD method is the number of subdomains. Selecting subdomains can be a trial and error processes. To

avoid wasting time to this process, here we introduce a “memory map” to help choosing “optimized” subdomains that yields the “best” computational performance. Memory map is pre-generated from several combinations of subdomains. Number of subdomains can be selected from the region of low memory in the memory map. This strategy often guarantees a faster CPU time than the standard method. The concept of memory map is new and first introduced here.

In the following, we first review the standard FD approach to solve a global domain problem. We then describes the basic idea of the hierarchical domain decomposition (HDD) and its parallel implementation. Then we describe the two modifications which help speeding up the HDD method on a serial calculation. Validations and numerical examples are given next along with the discussion. Conclusion are given at the end. Hereafter, we will refer to the standard finite difference for a global domain as FD2D, and to our modified hierarchical domain decomposition as MHDD2D.

2. Magnetotelluric forward modeling : Finite difference approach

Given an electrical conductivity (σ) or resistivity (ρ) model, to yield MT responses at the surface, the electric fields (**E**) are computed from the second order Maxwell's equation,

$$\nabla \times \nabla \times \mathbf{E} = i\omega\mu\sigma\mathbf{E}\,, \qquad (1a)$$

for the transverse electric field (TE) mode, while the magnetic fields (H) are solved from,

$$\nabla \times \rho\nabla \times \mathbf{H} = i\omega\mu\mathbf{H}\,, \qquad (1b)$$

for the transverse magnetic field (TM) mode, where ω is an angular frequency and μ the magnetic permeability. With finite difference approach, the conductivity or resistivity model is first discretized into many rectangular grids. An example of non-uniform grid discretization is shown in Figure 1. The unknown electric fields or magnetic fields are defined on the nodes (black dots) inside the domain, while the fields on the boundaries (left, right, top and bottom) are obtained from 1-D calculations. After applying finite difference to (1a) or (1b) and rearranging equation, both modes yield similar system of equations,

$$\mathbf{A}\mathbf{x} = \mathbf{b}\,, \qquad (2)$$

where **x** represents the unknown internal electric or magnetic fields; **b** a vector containing the term associated with the boundary fields; and **A** a coefficient matrix which is large sparse five-banded symmetric and complex only on the diagonal (Siripunvaraporn and Egbert, 2000). Equation (2) for 2-D problem can be solved either directly or iteratively such as preconditioned conjugate gradient (PCG) method (Siripunvaraporn and Egbert, 2000). One of our aims is to demonstrate the use a direct solver for 3-D problem. An LU-factorization is therefore applied here to solve all systems of equations from FD2D and MHDD2D.

After calculating the electric fields, the magnetic fields can be calculated from solving the first order Maxwell's equation, the Faraday's law. MT responses are then computed from the ratio of electric to magnetic fields at the surface.

3. Hierarchical Domain Decomposition method

An alternative method to solve (2) is via the domain decomposition method. There are many different domain decomposition techniques. Here, we applied the hierarchical domain decomposition (HDD) method which is a non-overlapping technique to our 2-D MT problems. We start this section by describing the basic idea of the HDD method.

In every domain decomposition techniques, the model domain is split into several smaller subdomains. For simplicity, example mesh in Figure 1 is redrawn as in Figure 2 with uniform space, and is uniformly partitioned into 3 × 4 subdomains only as an illustration. The unknown electric or magnetic fields located at the nodes can be classified into three "hierarchical" types: (1) the interiors (●), (2) the interfaces (■and ▲) and (3) the intersections (✕) from lowest to highest level, as shown in Figure 2. The intersections are defined as the highest level because they separate the interfaces. Similarly, the interfaces separate the interiors, so they are defined the next lower level.The interiors are therefore the lowest. With this configuration, the intersections must be solved first. Once the intersections are obtained, the interfaces can be successively calculated from the intersections. Similarly, the interiors can be successively computed from the interfaces. This hierarchical classification is slightly different from the "classic" Schur complement method (see Smith et al., 1996; Saad, 2003; Zhang, 2005;). In Schur complement method, the unknown fields are classified only the interiors and the interfaces.

For 2-D MT problem, assuming that the model domain is equally divided into $p \times q$ (= r) subdomains where p and q are number of subdomains in z- and y- directions, respectively, and r is the total number of subdomains. These partitions will yield a total of l interiors (or l/r for each subdomain), total of m interfaces and n intersections. Specifically, an inner subdomain i which has $l_{zi} \times l_{yi}$ (= l/r) interiors would have $2l_{zi} + 2l_{yi}$ interfaces, and 4 intersections, while outer or boundary subdomains would have less depending on their locations. By using Figure 1 and Figure 2 as an example, the model in Figure 1 is discretized into 12 × 20 grids, which is later decomposed into 3 × 4 (=12) subdomains. In this example, there would be a total of 209 unknowns inside a global domain. When partitioning into 3 × 4 subdomains, an inner subdomain would then have 12 interiors, 14 interfaces and 4 intersections. The total numbers of interiors, interfaces and intersections are 144, 59 and 6, respectively.

By organizing the unknowns into three levels, the system of equations (2) can be reordered according to this configuration as follows,

$$\begin{pmatrix} \mathbf{F} & \mathbf{D} & \mathbf{0} \\ \mathbf{D}^{\mathrm{T}} & \mathbf{G} & \mathbf{E} \\ \mathbf{0} & \mathbf{E}^{\mathrm{T}} & \mathbf{H} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \\ \mathbf{h} \end{pmatrix} , \qquad (3)$$

where **F**, **G** and **H** are $l \times l$ global interior coefficient matrix, $m \times m$ global interface coefficient matrix, and $n \times n$ intersection coefficient matrices, respectively. Global interior matrix **F** composes of many smaller $l/r \times l/r$ local interior sub-matrix $\mathbf{F}_i$ where $i = 1$ to r. Each $\mathbf{F}_i$ corresponds to a coupling within the interior elements inside the i subdomain. Global interface matrix **G** gathers all coefficients corresponding to an interaction between the interface elements, while **H** is diagonal matrix associating with the intersection elements. The inter-coupling coefficients between the interiors and interfaces are given in **D** with a dimension of $l \times m$, and between the interfaces and intersections are given in **E** with a dimension of $m \times n$. There is no coupling between the interiors and the intersections in our 2-D MT case as shown in Figure 2. Vectors **f**, **g** and **h** are domain boundary fields associated with the interiors (**u**), interfaces (**v**) and intersections (**w**), respectively. Figure 2 shows that there are no boundary fields that belong to the intersections. Therefore, **h** = **0** in our 2-D problems.

According to the hierarchical domain decomposition technique, equation (3) can be decomposed into two reduced systems: the interior-interface reduced system and the interface-intersection reduced system. The interior-interface reduced system is derived from the coupling between the interiors and interfaces,

$$\begin{pmatrix} \mathbf{F} & \mathbf{D} \\ \mathbf{D}^T & \mathbf{G} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g} - \mathbf{E}\mathbf{w} \end{pmatrix}, \qquad (4)$$

while the interface-intersection reduced system is from the coupling between the interfaces and intersections,

$$\begin{pmatrix} \mathbf{S} & \mathbf{E} \\ \mathbf{E}^T & \mathbf{H} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{w} \end{pmatrix} = \begin{pmatrix} \mathbf{g}' \\ \mathbf{h} \end{pmatrix}, \qquad (5)$$

where the interface Schur complement matrix $\mathbf{S} = \mathbf{G} - \mathbf{D}^T\mathbf{F}^{-1}\mathbf{D}$ and $\mathbf{g}' = \mathbf{g} - \mathbf{D}^T\mathbf{F}^{-1}\mathbf{f}$. The unknowns are then successively solved from the highest to the lowest level. The intersections **w** are solved first from

$$\mathbf{H'w = h'}, \qquad (6)$$

where the intersection Schur complement matrix $\mathbf{H' = H - E^T S^{-1} E}$, and its right-hand side $\mathbf{h' = h - E^T S^{-1} g'}$. Once solving the intersections, the interfaces **v** and the interiors **u** can then be consecutively solved from

$$\mathbf{Sv = g' - Ew}, \qquad (7)$$

and

$$\mathbf{F_i u_i = f_i - D_i v}. \qquad (8)$$

Algorithm of the standard HDD method can be summarized below after decomposing the global domain into several subdomains.

1. Form $\mathbf{F_i}$, $\mathbf{f_i}$, $\mathbf{D_i}$ and factorize $\mathbf{F_i}$ of each subdomain.
2. Compute $\mathbf{D_i^T F_i^{-1} D_i}$ and $\mathbf{D_i^T F_i^{-1} f_i}$ of each subdomain.
3. Form **G**, **g**, **H**, **h** and **E**.
4. Construct $\mathbf{S = G - \sum D_i^T F_i^{-1} D_i}$ and $\mathbf{g' = g - \sum D_i^T F_i^{-1} f_i}$.
5. Factorize **S**.
6. Build $\mathbf{H' = H - E^T S^{-1} E}$ and $\mathbf{h' = h - E^T S^{-1} g'}$.
7. Solve $\mathbf{H'w = h'}$.
8. Solve $\mathbf{Sv = g' - Ew}$.
9. Solve $\mathbf{F_i u_i = f_i - D_i v}$.
10. Merge $\mathbf{u_i}$, **v** and **w** as a solution for the system of equations (2).

The intersection Schur complement matrix **H'** (step 7) is dense, but its dimension, $n \times n$, is relatively small and therefore would not require a lot of computations. Similarly, the classical Schur method has a similar dense matrix but with a dimension equal to numbers of interfaces and intersections, i.e. $m+n \times m+n$. Thus, the hierarchical domain decomposition method yields a significant smaller dense matrix. The interface Schur complement matrix **S**, in the hierarchical case, is not dense but sparse matrix. Example of its sparse pattern is shown in Figure 3a) from subdomains of Figure 2.

All equations including equation (6), (7) and (8) are solved with a direct method (here, an LU-factorization). To construct $\mathbf{S} = \mathbf{G} - \mathbf{D}^T\mathbf{F}^{-1}\mathbf{D}$ and $\mathbf{H'} = \mathbf{H} - \mathbf{E}^T\mathbf{S}^{-1}\mathbf{E}$ in step 4 and 6, after factorizations, **F** and **S** systems are solved with a series of different right hand sides: $\mathbf{D}^T$ and $\mathbf{E}^T$ for m times and n times, respectively. Solving each system just one time requires relatively small amount of computational resources, both memory and CPU time. However, as showing in the algorithm above, both systems are solved several times. Computational time for numerous solving (step 2, 4 and 6) plus factorizations (step 1, 5 and 7) can be more than just solving one large global system (equation 2) on a serial machine. This statement is correctly confirmed in Xiong (1999) and also in our MT numerical experiments in the next section. Once all main matrices are obtained; equation (6) and (7) is solved just one to obtain **w** and **v** in step 7 and 8, respectively. Equation (8) is then consecutively solved to obtain the interiors **u** within each subdomain in step 9. If each subdomain is equally discretized, this is equivalent as solving equation (8) r times.

Because domain decomposition is not highly efficient on a serial machine, another way of using domain decomposition on a serial computation is to modify the hierarchical matrix (3) and used it as a preconditioner when solving the system with the iterative solvers (e.g., Bitzarakis et al., 1997; Larsson, 1999; Benedetti et al., 2009; Grasedyck et al., 2009).

3.1 Parallel Implementation of HDD

Most parallel domain decomposition algorithms distribute computations of each subdomain to each processor (see examples in Xiong, 1999; Zyserman et al., 1999; and Zyserman and Santos, 2000). In this parallel scheme, step 1, 2 and 9 of each subdomain are performed

separately on each processor. After calculations, all results are sent to the master node. The bottleneck of this parallelization occurs from step 3 to 8. The most difficult parts for parallelization are to factorize **S** in step 5, to construct $\mathbf{H'} = \mathbf{H} - \mathbf{E}^T\mathbf{S}^{-1}\mathbf{E}$ and $\mathbf{h'} = \mathbf{h} - \mathbf{E}^T\mathbf{S}^{-1}\mathbf{g'}$ in step 6 and to solve $\mathbf{Sv} = \mathbf{g'} - \mathbf{Ew}$ in step 8. Once distributing **S** to all processors, this process requires a lot of communication time among processors when factorizing and solving system of equations. Efficiency of this parallel scheme would depend significantly on the parallel algorithms which also depend on computer architectures (see Lu and Shen, 1997; Kocak and Akay, 2001). Many massive parallel manufacturers have provided their own efficient parallel algorithms to solve system of equations. These algorithms show best performance only on their own platforms.

However, this conventional parallel scheme could be a problem for PC cluster platform or distributed memory systems. Efficiency would be relatively low if switch or hub used to communicate among processors is slow regardless of how efficient the algorithm is. Parallel implementation is not the purposes of our paper as previously described. We therefore opt not to show the numerical experiments of HDD on parallel systems. Experiments with 3-D MT problems would be an interesting research to pursue which is beyond our scope here.

4. Modified hierarchical domain decomposition method

Earlier numerical experiments on single processor machine show that a straightforward application of the HDD method to the 2-D MT problems requires less memory storage than standard method. However, its computational time becomes longer. In order to make the hierarchical domain decomposition method more efficient on a serial machine for our 2-D MT problem, two modifications are necessary. First, the separation of the interfaces into vertical and horizontal interfaces will break the larger interface system into two smaller vertical and horizontal interface systems which would lead to a memory reduction. Second, the red-black ordering technique is applied inside the horizontal and vertical interface systems to further help decreasing the computational time.

Taking advantage of the rectangular discretization of the FD approximation, the interfaces can be further classified into two types: the horizontal interfaces (■ in Figure 2) and the

vertical interfaces (▲ in Figure 2). Number of interfaces (m) is then divided into number of horizontal interfaces (m_h) and number of vertical interfaces (m_v) where $m = m_h + m_v$. The system of equations (3) can then be reassembled as follows,

$$\begin{pmatrix} \mathbf{F} & \mathbf{D_H} & \mathbf{D_V} & \mathbf{0} \\ \mathbf{D_H^T} & \mathbf{G_H} & \mathbf{0} & \mathbf{E_H} \\ \mathbf{D_V^T} & \mathbf{0} & \mathbf{G_V} & \mathbf{E_V} \\ \mathbf{0} & \mathbf{E_H^T} & \mathbf{E_V^T} & \mathbf{H} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v_H} \\ \mathbf{v_V} \\ \mathbf{w} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g_H} \\ \mathbf{g_V} \\ \mathbf{h} \end{pmatrix} , \qquad (9)$$

where $_H$ and $_V$ represent horizontal and vertical interfaces, respectively. The main difference from the original hierarchical domain decomposition would be at the separation of **G** matrix into $\mathbf{G_H}$ and $\mathbf{G_V}$, where $\mathbf{G_H}$ gathers all coefficients corresponding to a coupling between the horizontal interfaces, and similarly for $\mathbf{G_V}$ corresponding to a coupling between the vertical interfaces. With new classification, both vertical interfaces ($\mathbf{v_V}$) and horizontal interfaces ($\mathbf{v_H}$) are situated in the middle level between the intersection (**w**) and the interior (**u**) which are the highest and lowest, respectively. The interior-interface and interface-intersection reduced systems in equation (4) and (5) become

$$\begin{pmatrix} \mathbf{F} & \mathbf{D_H} & \mathbf{D_v} \\ \mathbf{D_H^T} & \mathbf{G_H} & \mathbf{0} \\ \mathbf{D_V^T} & \mathbf{0} & \mathbf{G_v} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v_H} \\ \mathbf{v_V} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g_H - E_H w} \\ \mathbf{g_V - E_V w} \end{pmatrix} , \qquad (10)$$

and

$$\begin{pmatrix} \mathbf{S_{HH}} & \mathbf{S_{HV}} & \mathbf{E_H} \\ \mathbf{S_{VH}} & \mathbf{S_{VV}} & \mathbf{E_V} \\ \mathbf{E_H^T} & \mathbf{E_V^T} & \mathbf{H} \end{pmatrix} \begin{pmatrix} \mathbf{v_H} \\ \mathbf{v_V} \\ \mathbf{w} \end{pmatrix} = \begin{pmatrix} \mathbf{g'_H} \\ \mathbf{g'_V} \\ \mathbf{h} \end{pmatrix} , \qquad (11)$$

respectively. Here, the interface Schur complement matrix **S** is decomposed into $\mathbf{S_{HH}}$, $\mathbf{S_{HV}}$, $\mathbf{S_{VH}}$ and $\mathbf{S_{VV}}$ as follow,

$$\begin{pmatrix} \mathbf{S}_{HH} & \mathbf{S}_{HV} \\ \mathbf{S}_{VH} & \mathbf{S}_{VV} \end{pmatrix} = \begin{pmatrix} \mathbf{G}_H & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_V \end{pmatrix} - \begin{pmatrix} \mathbf{D}_H^T \\ \mathbf{D}_V^T \end{pmatrix} \mathbf{F}^{-1} \begin{pmatrix} \mathbf{D}_H & \mathbf{D}_V \end{pmatrix}, \qquad (12)$$

and

$$\begin{pmatrix} \mathbf{g'}_H \\ \mathbf{g'}_V \end{pmatrix} = \begin{pmatrix} \mathbf{g}_H \\ \mathbf{g}_V \end{pmatrix} - \begin{pmatrix} \mathbf{D}_H^T \\ \mathbf{D}_V^T \end{pmatrix} \mathbf{F}^{-1} \mathbf{f} . \qquad (13)$$

Example of the sparsity pattern of the modified Schur interface (12) is shown in Figure 3b) to be compared with the original Schur interface matrix **S** (Figure 3a). Similar to the original hierarchical domain decomposition, the unknown fields are successively solved from the highest level to the lowest level. The intersections **w** will be solved first from

$$\mathbf{H'w} = \mathbf{h'}, \qquad (14)$$

where, $\mathbf{H'} = \mathbf{H} - \begin{pmatrix} \mathbf{E}_H^T & \mathbf{E}_V^T \end{pmatrix} \begin{pmatrix} \mathbf{S}_{HH} & \mathbf{S}_{HV} \\ \mathbf{S}_{VH} & \mathbf{S}_{VV} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{E}_H \\ \mathbf{E}_V \end{pmatrix}$, and its right-hand side $\mathbf{h'} = \mathbf{h} - \begin{pmatrix} \mathbf{E}_H^T & \mathbf{E}_V^T \end{pmatrix} \begin{pmatrix} \mathbf{S}_{HH} & \mathbf{S}_{HV} \\ \mathbf{S}_{VH} & \mathbf{S}_{VV} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{g'}_H \\ \mathbf{g'}_V \end{pmatrix}$. After solving the intersections **w**, the vertical interfaces $\mathbf{v}_V$ and the horizontal interfaces $\mathbf{v}_H$ can be split and solved separately as,

$$(\mathbf{S}_{VV} - \mathbf{S}_{VH}\mathbf{S}_{HH}^{-1}\mathbf{S}_{HV})\mathbf{v}_V = \mathbf{g'}_V - \mathbf{E}_V\mathbf{w} - \mathbf{S}_{VH}\mathbf{S}_{HH}^{-1}(\mathbf{g'}_H - \mathbf{E}_H\mathbf{w}), \qquad (15)$$

and,

$$\mathbf{S}_{HH}\mathbf{v}_H = \mathbf{g'}_H - \mathbf{E}_H\mathbf{w} - \mathbf{S}_{HV}\mathbf{v}_V . \qquad (16)$$

Dimension of $\mathbf{S}_{HH}$ and $\mathbf{S}_{VV}$ from (15) and (16) are $m_h \times m_h$ and $m_v \times m_v$, respectively, which are smaller than $m \times m$ **S** matrix of (7). They are therefore faster to solve and less memory storage. This is one clear advantage of classifying the interfaces into the horizontal and

vertical interfaces. After obtaining **w** and **v**, the interiors **u** can then be consecutively solved from

$$\mathbf{F}_i\mathbf{u}_i = \mathbf{f}_i - \mathbf{D}_{Hi}\mathbf{v}_H - \mathbf{D}_{Vi}\mathbf{v}_V \,. \qquad (17)$$

To further increase the efficiency of our modified scheme, red-black coloring technique (See Press et al., 1992 and Saad, 2003) is applied to (15) and (16) to help reducing the computational time. Under the red-black ordering, the unknowns inside of $\mathbf{S}_{VV}$ and $\mathbf{S}_{HH}$ are classified into red and black unknowns. The idea of Schur complement is again applied to this coloring system of the interfaces. The reduced systems are then derived and recursively solved to the red and to the black systems. This modification demonstrates the application of Schur domain decomposition inside the hierarchical domain decomposition (see Rung-Arunwan, 2010 for further detail).

With both modifications, the modified hierarchical domain decomposition (MHDD2D) can outperform the FD2D code even running on a serial computational machine as showing in the next section.

5. Numerical Experiments

In this section, we first validate that the responses from our modified hierarchical domain decomposition method (MHDD2D) are as accurate as those from FD2D. Next, computational costs on a single processor are measured with different combinations of subdomains. A memory map is then introduced as a strategy to select an "optimized" number of subdomain where computational costs are minimized (i.e., relatively faster or at least equivalently to FD2D, but with a fraction of memory).

5.1 Validation Tests

To validate the MHDD2D approach, we show the apparent resistivities and phases of both TM and TE modes at three frequencies calculated from the model shown in Figure 1. The calculated responses from our MHDD2D approach are directly compared to those obtained from FD2D as in section 2. In this test, the model and air of Figure 1 is non-uniformly discretized into 80 × 240 grids in *z*- and *y*-direction, respectively. For FD2D method, the unknown to be solved is 18,881. For MHDD2D, the model domain is uniformly split into 4 × 8 (*z*- and *y*- direction, respectively) subdomains. With this 4 × 8 subdomains, the 18,881 unknowns will be divided into 551 interiors for each subdomain (or a total of 17,632 interiors), 696 horizontal interfaces and 532 vertical interfaces, and 21 intersections. Total memory requirement of MHDD2D is about 21.7 Mbytes, which is approximately one-third of FD2D (about 71.09 Mbytes). Memory estimation will be discussed in subsection 5.2.1.

Figure 4 shows that the calculated responses from both FD2D and MHDD2D are perfectly identical on both modes. Their difference is in the round-off level which is insignificant. This is expected since both methods solve the same system of equation, except that the MHDD2D method splits the computational domain into many smaller subdomains, and then solves smaller systems. In addition, we have performed validation tests on various synthetic models and real model (see inverted model from real data in Siripunvaraporn and Egbert, 2000) with several combinations of subdomains. All validation tests show that there is no difference from both methods (Rung-Arunwan, 2010). These have validated our MHDD2D method for both TM and TE modes.

5.2 Comparisons of Computational Efficiency

Next, to prove the efficiency of our modified domain decomposition scheme, we ran the code on several synthetic 2-D models and also real "inverted" model (from Siripunvaraporn and Egbert, 2000) for both TM and TE modes. Because a direct method (LU-factorization) is used to solve all systems of equations, computational time and memory requirements are no difference among different models, modes (TM or TE) and frequency if domain size is the same. Model of Figure 1 is therefore used as a representative to demonstrate the effectiveness of our code.

Model and air of Figure 1 is discretized into three size meshes: 40 × 120 (small), 80 × 240 (medium) and 120 × 360 (large). These three meshes are then uniformly subdivided into $p \times q$ subdomains, where p and q are numbers of subdomains in z-dir and y-dir, respectively, starting from 2. Estimated memory usage and actual calculation time for each combination of subdomains for each mesh are compared with those from FD2D. Comparison results are plotted and shown in Figure 5 for 40 × 120 mesh, Figure 6 for 80 × 240 mesh and Figure 7 for 120 × 360 mesh. Relative CPU time and memory (both in percents) are calculated from $(\text{time}_{\text{MHDD2D}}\text{-time}_{\text{FD2D}})*100/\text{time}_{\text{FD2D}}$ and $(\text{mem}_{\text{MHDD2D}}\text{-mem}_{\text{FD2D}})*100/\text{mem}_{\text{FD2D}}$, respectively. Positive relative time and relative memory indicate that MHDD2D is less efficiency than FD2D and therefore spend more calculation time and require more memory, while negative reflects the opposite, i.e. MHDD2D is more efficient. Actual memory usage of FD2D are 8.77 Mbytes, 71.09 Mbytes and 240.97 Mbytes for small, medium and large, respectively, while actual CPU time on an Intel Core Two Duo 6400, 2.13 GHz machine are 0.08 second, 1.12 second and 4.16 second, respectively. Actual CPU time and memory used of MHDD2D can thus be inferred from these actual values of FD2D and the maps shown in Figure 5, 6 and 7, respectively.

5.2.1 "Memory Map" and Memory Comparison

Total memory usage of MHDD2D can be calculated from numbers of subdomains in z-dir (p) and y-dir (q), number of interiors (l/r) for each subdomain, numbers of horizontal interfaces (m_h) and vertical interfaces (m_v) and number of intersections (n). However, it is quite complicated to express in a simple formula. It is therefore pre-estimated from the allocated variables inside the code to produce the "memory map" before running the actual code. Memory map displays minimum memory used for different combinations of subdomains as shown in Figure 5a, 6a and 7a. The concept of memory map is very useful and will be demonstrated in later subsection.

In contrast to MHDD2D, total memory usage for FD2D can be easily estimated from $(N_y\text{-}1)(N_z\text{-}1)(3N_z\text{+}1)*16$ where N_y and N_z is grid discretization in y-dir and z-dir, respectively.

Multiplication with 16 is required because complex double precision is used. Because a large global matrix (equation 2) of FD2D is broken into many smaller sub-matrices (equation 9) for MHDD2D, memory requirement for different combinations of subdomains should therefore be less than that of FD2D. This is evidently shown in Figure 5a, 6a and 7a, where negative percentage is all over the map indicating less memory requirement of MHDD2D. However, total memory usage varies according to numbers of subdomains used in both directions.

From all three figures, there are two cases where memory usage is relatively large (but still less than FD2D). First case is when the domain is divided into "large" numbers of subdomains. When number of subdomains become large (e.g., 20 × 30 subdomains in Figure 7a), number of interiors per subdomain is small (see Table 1), but total number of interfaces are high (Table 1). More memory is therefore required to store and solve those interface coefficient matrices ($\mathbf{G_H}$, $\mathbf{G_V}$, $\mathbf{S_{HH}}$, $\mathbf{S_{HV}}$, $\mathbf{S_{VH}}$ and $\mathbf{S_{VV}}$ in 10 and 11). Although intersections ($\mathbf{H}$) also increase, it would not significantly affect. In contrast, when small number of subdomains used (e.g., 3 × 3 subdomains in Figure 7a), total numbers of interfaces in both directions are small (see Table 1), but number of interiors per subdomain becomes very high (Table 1). Large number of interiors causes matrix $\mathbf{F_i}$ (equation 10) of each subdomain to require more memory to store and solve the system of equations (equation 13 and 17). Note that we use LU decomposition to solve all systems of equations. Some "extra" memory is therefore required to fill the empty band of the sparse matrix. This extra memory has already been accounted for in Figure 5a, 6a and 7a.

5.2.2 Comparisons of CPU time

Calculation time cannot be pre-estimated as the memory usage, it can only be obtained from running the actual code on the computer. Relative CPU time from small, middle and large meshes are shown in Figure 5b, 6b and 7b, respectively, from different combinations of subdomains. They are obtained from running on a single processor machine; here, an Intel Core Two Duo 6400, 2.13 GHz machine. Different machines or architectures may result differently. However, patterns of relative CPU time should remain approximately the same.

For small 40 × 120 mesh, relative CPU time of MHDD2D is at least 30% more than that of FD2D in every combination of subdomains (Figure 5b). Although a larger system of equations (equation 2) is broken into many smaller systems (equation 9), successively solving a series of these smaller systems (see equation 4-6, and 10-17) can outperform solving a global system of FD2D. This reflects in larger CPU time as shown with all positive in Figure 5b. Although there is no benefit of MHDD2D for smaller 40 × 120 meshes in term of CPU time, better efficiency can be gained up to 20% from larger meshes as shown with negative zones in Figure 6b for 80 × 240 mesh and in Figure 7b for 120 × 360 mesh. This shows that when grid discretization becomes large, MHDD2D will become more effective, even with a serial computation. This conclusion is significant, especially for future implementing the idea of MHDD2D to 3-D cases. In 3-D, the discretization mesh would be clearly a lot larger than what we used in 2-D case.

5.3 Optimized Number of Subdomains : Pre-Selection

Figure 5a, 6a and 7a show that there are regions where memory requirement is "minimum". The minimized memory zones have the centers at 5 × 6 subdomains for 40 × 120 mesh (Figure 5a), at 8 × 8 subdomains for 80 × 240 mesh (Figure 6a) and at 10 × 9 subdomains for 120 × 360 mesh (Figure 7a). The interiors, horizontal interfaces, vertical interfaces and intersections for these three subdomains are given in Table 2.

By matching Figure 5a, 6a and 7a to Figure 5b, 6b and 7b, respectively, we found that the minimized memory zones are coincidently occurred almost the same regions as the minimized CPU time zone. Both areas will be referred to as the "optimized" regions, because both memory and CPU time are least used. In this "optimized" regions, numbers of interiors, horizontal interfaces, vertical interfaces and intersections are properly justified or balancing (as shown in Table 2), so that solving and storing $\mathbf{F_i}$, $\mathbf{G_H}$, $\mathbf{G_V}$, $\mathbf{S_{HH}}$ and $\mathbf{S_{VV}}$ and $\mathbf{H}$ matrices are relatively fast and less memory requirement. Larger or smaller number of subdomains would cause an unbalance to these numbers. Larger number of subdomains would increase the interface sizes, while smaller number of subdomains would increase the interior size. Both cases would produce a large matrix, which would dominate both calculation time and memory usage.

The agreement between the optimized CPU time and memory usage has lead to the idea of subdomain selection. Usually, choosing number of subdomains that yields least CPU time and smallest memory requirement would be a trial and error strategy. Here, we propose to select the "optimized" subdomains from the memory map, shown in Figure 5a, 6a and 7a. Because memory usage can be pre-estimated from the variable allocations inside the code, this number can be printed out and plotted in a map from different combinations of subdomains. The optimized subdomains can therefore be chosen from the region of "least" memory requirement. There would be a higher chance that CPU time performance of MHDD2D would be better than FD2D if choosing subdomains from this region. When implementing MHDD2D to 3-D case, similar technique can be used to avoid trial and error selections.

5.4 Comparison of modified and non-modified hierarchical domain decomposition methods

For the original hierarchical domain decomposition technique, memory requirements for **F** and **H** matrices in (4) and (5) are identical to those in (10) and (11) for our modified hierarchical domain decomposition. However, interface matrices, **G** and **S** in (4) and (5) (Figure 3a), depends on the sum of horizontal interfaces and vertical interfaces ($m = m_h + m_v$). These matrices are therefore larger than $\mathbf{G_H}$, $\mathbf{G_V}$, $\mathbf{S_{HH}}$ and $\mathbf{S_{VV}}$ in (10) and (11) (Figure 3b) for the modified scheme around 20-50% depending on the number of subdomains (r). Memory requirement for non-modified hierarchical domain decomposition would therefore up to 50% more than the modified case from our 2-D study, but it is still less than FD2D.

In term of computational time, the standard hierarchical domain decomposition would require about the same CPU time to solve $\mathbf{F_i}$ and **H** systems of equations. However, our 2-D study reveal that for the interface parts, larger **G** and **S** in (4) and (5) of the non-modified code requires solving time slightly more or less than solving smaller $\mathbf{G_H}$, $\mathbf{G_V}$, $\mathbf{S_{HH}}$ and $\mathbf{S_{VV}}$ in (10) and (11) of the modified code. Not much can be gained in terms of CPU time in this part, but a lot more in terms of memory. However, by reducing the larger **G** into $\mathbf{G_H}$ and $\mathbf{G_V}$ (from Figure 3a to 3b), red-black ordering can be easily applied for solving $\mathbf{G_H}$ and $\mathbf{G_V}$, but not

directly to **G** in (4). With the red-black ordering, about 10-50% depending on a combination of subdomains can be gained comparing to the original HDD method for the 2-D case. Red-black ordering can be easily implementing in 3-D case as well, this would help further decreasing the computational time.

6. Conclusions

We have demonstrated the efficiency of the MHDD2D code for 2-D MT forward modeling. MHDD2D is a modified version of the hierarchical domain decomposition method. The original scheme begins by dividing a global computational domain into several subdomains. Then, the unknown nodes are classified into three different kinds: interiors, interfaces and intersections. A global system of equations is re-organized according to these configurations producing three sets of smaller systems of equations. The intersection reduced system of equations is solved first to obtain the intersections. The calculated intersections are then used in the right hand-side of the interface systems of equations to compute the interfaces. Similarly, the calculated interfaces are input in the interior systems of equations to compute the interiors inside each subdomain.

Normally, HDD is applied on a parallel system. Efficiency of the HDD method on a serial machine is very low comparing to the conventional method. To enhance the efficiency of the hierarchical method on single processor computer, the interfaces of the standard hierarchical domain decomposition method is further separated into horizontal interfaces and vertical interfaces by taking an advantage of the rectangular discretization of the finite difference. Our modified version will then have four sets of smaller systems of equations, instead of three as in the original version. The division of the interfaces into horizontal and vertical interfaces helps substantially decreasing the size of memory usage. However, it does little help in computing time. Red-black coloring is then applied to substantially reduce the computational time of the code.

By running MHDD2D with several combinations of subdomains on single processor machine, the optimized subdomains can be selected from the memory map generated prior

the run. Dividing the global domain with the optimized subdomains, MHDD2D can run up to 20-30% faster and require up to 70% less memory than FD2D on sing processor machine. This conclusion is very crucial. It indicates that the same hierarchical domain decomposition algorithm can be extended and applied to 3-D problem. By applying modified HDD method to 3-D case, 3-D forward problem can now be solved with a direct method, even on standard single processor PC. With the direct solver, its factorized matrices can be re-used several times with different right-hand sides. This will help speeding up the sensitivity calculation in the 3-D inversion process. Most importantly for a direct solver, computational time is controllable and independent of frequencies, modes and resistivities, as long as the domain size remains the same.

7. References

- Avdeev, D. & Avdeeva, A., 2009, 3D Magnetotelluric inversion using a limited-memory quasi-Newton optimization, *Geophysics*, 74 (3), F45-F57.
- Basermann, A., Jaekel, U., Nordhausen, M., & Hachiya, K., 2005. Parallel iterative solvers for sparse linear systems in circuit simulation, *Future Generation Computer Systems*, **21**(8), 1275-1284.
- Ben-Hadj-Ali, H., Operto, S., Virieux, J., & Sourbier, F., 2008. 3D frequency-domain full-waveform tomography based on a domain decomposition forward problem, *SEG Technical Program Expanded Abstracts*, **27**(1), 1945-1949.
- Benedetti, I., Milazzo, A., & Aliabadi, M.H., 2009, A fast dual boundary element method for 3D anisotropic crack problems, *Inter. Jour. Numer. Methods. Eng.*, 80,1356-1378.
- Bitzarakis, S., Papadrakakis, M., & Kotsopulos, A., 1997. Parallel solution techniques in computational structural mechanics, *Computer Methods in Applied Mechanics and Engineering*, **148**(1-2), 75-104.
- Cai, X.C., Casarin, M.A., Elliot Jr., F.W., & Widlund, O.B., 1998, Overlapping Schwartz Algorithms for solving Helmholtz equation, *Contemporary Math*, 218, 437-445.
- Chan, T.F. & Goovaerts, D., 1992, On the relationship between overlapping and nonoverlapping domain decomposition methods, *SIAM J. Matrix Anal. Appl*., 13, 663-670.
- Grasedyck, L., Kriemann, R., & Le Borne, S, 2009, Domain decomposition based H-LU preconditioning, *Numerische Mathematik*, 112 (4), 565-600.

- Kocak, S., & Akay, H.U., 2001, Parallel Schur complement method for large-scale systems on distributed memory computers, *Applied Mathematical Modelling*, 25, 873-886.
- Larsson, E., 1999. A Domain Decomposition Method for the Helmholtz Equation in a Multilayer Domain, *SIAM Journal on Scientific Computing*, **20**(5), 1713-1731.
- Lu, Y. & Shen, C., 1997. A domain decomposition finite-difference method for parallel numerical implementation of time-dependent Maxwell's equations, *IEEE Transactions on Antennas and Propagation*, **45**(3), 556-562.
- Lu, Z., An, X., & Hong, W., 2008. A fast domain decomposition method for solving three-dimensional large-scale electromagnetic problems, *IEEE Transactions on Antennas and Propagation*, **56**(8 I), 2200-2210.
- Mackie, R., Smith, J., & Madden, T., 1994. Three-dimensional electromagnetic modeling using finite difference equations: The magne-totelluric example, *Radio Science*, **29**(4), 923-935.
- Mandel, J., 1993, Balancing domain decomposition, *Comm. Numer. Methods Engrg*, 9, 233-241.
- Mitsuhata, Y. & Uchida, T., 2004, 3D Magnetotelluric modeling using the T-Ω finite-element method, *Geophysics*, 69 (1), 108-119.
- Pain, C., Herwanger, J., Worthington, M., & De Oliveira, C., 2002. Effective multidimensional resistivity inversion using finite-element techniques, *Geophys. J Int.*, **151**(3), 710-728.
- Patro, P.K., & Egbert, G.D., 2008, Regional conductivity structure of Cascadia: Preliminary results from 3D inversion of USArray transportable array Magnetotelluric data, *Geophys. Res. Lett*., 35 (20), art. no. L20311.
- Peng, T., Sertel, K., & Volakis, J.L., 2009, Fully overlapping domain-decomposition for fast optimization of small antennas in large-scale composite media, *2009 Computational Electromagnetics International Workshop*, CEM 2009, art. no. 5228103, 77-81.
- Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P., 1992. *Numerical Recipes in FORTRAN: The art of scientific computing*, Cambridge University Press, 2nd edn.
- Rice, J.R., Tsompanopoulou, P., & Vavalis, E., 2000, Interface relaxation methods for elliptic differential equations, *Applied. Numer. Math*., 32, 219-245.

- Rung-Arunwan, T., 2010, An efficient modified hierarchical domain decomposition for 2D Magnetotelluric forward modeling, M.Sc. Thesis. Mahidol University.
- Saad, Y., *Iterative Methods for Sparse Linear Systems*, 2nd ed, SIAM, Philadelphia, 2003.
- Siripunvaraporn, W. & Egbert, G., 2000. An effcient data-subspace inversion method for 2D magnetotelluric data, *Geophysics*, **65**(3), 791-803.
- Siripunvaraporn, W., Egbert, G., & Lenbury, Y., 2002. Numerical accuracy of magnetotelluric modeling: A comparison of finite difference approximations, *Earth Planets Space*, **54**(6), 721-725.
- Siripunvaraporn, W., Uyeshima, M., & Egbert, G., 2004, Three-dimensional inversion for Network-Magnetotelluric data, *Earth Planets Space*, **56**, 893-902.
- Siripunvaraporn, W., Egbert, G., Lenbury, Y., & Uyeshima, M., 2005, Three-dimensional Magnetotelluric inversion: data-space method, *Phys. Earth and Planetary Interiors*, **150**, 3-14.
- Siripunvaraporn, W & Egbert, G., 2009, WSINV3DMT: Vertical magnetic field transfer function inversion and parallel implementation, *Phys. Earth and Planetary Interiors*, **173**, 317-329.
- Siripunvaraporn, W & Sarakorn, W., 2010, An efficient three-dimensional Magnetotelluric inversion: a mixed of the data space conjugate gradient method and the data space Occam's method, *submitted to GJI*.
- Smith, B., BjØrstad, P., & Gropp, W., 1996. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press.
- Smith, J. T., 1996. Conservative modeling of 3D electromagnetic felds, Part I: Properties and error analysis, *Geophysics*, **61**(5), 1308-1318.
- Sourbier, F., Haidar, A., Giraud, L., Operto, S., & Virieux, J., 2008. Frequency-domain full-waveform modeling using a hybrid direct-iterative solver based on a parallel domain decomposition method: A tool for 3D full-waveform inversion?, *SEG Technical Program Expanded Abstracts*, **27**(1), 2147-2151.
- Streich, R., 2009. 3D finite-difference frequency-domain modeling of controlled-source electromagnetic data: Direct solution and opti-mization for high accuracy, *Geophysics*, **74**(5), F95-F105.
- Takei, A., Sugimot, S.-I., Ogino, M., Yoshimura, S., & Kanayama, H., 2010. Large-scale analysis of high frequency electromagnetic field by hierarchical domain decomposition

method with direct method in subdomains, IEEE Transactions on Fundamentals and Materials, **130**(3), 239-246.

- Wang, B., Mittra, R., & Shao, W., 2008. A domain decomposition finite-difference method utilizing characteristic basis functions for solving electrostatic problems, *IEEE Transactions on Electromagnetic Compatibility*, **50**(4), 946-952.
- Wannamaker, P., 1991. Advances in three-dimensional magnetotelluric modeling using integral equations, *Geophysics*, **56**(11), 1716-1728.
- Wannamaker, P., Stodt, J., & Rijo, L., 1987. A stable finite element solution for two-dimensional magnetotelluric modelling, *Geophys. J. R. astr. Soc.,* **88**(1), 277-296.
- Xie, G., Li, J., Majer, E., Zuo, D., & Oristaglio, M., 2000. 3D electromagnetic modeling and nonlinear inversion, *Geophysics*, **65**(3), 804-822.
- Xiong, Z., 1992. Electromagnetic modeling of 3D structures by the method of system iteration using integral equations, *Geophysics*, **57**(12), 1556-1561.
- Xiong, Z., 1999. Domain decomposition for 3D electromagnetic modeling, *Earth Planets Space*, **51**(10), 1013-1018.
- Yin, L., Wang, J., & Hong, W., 2002. A novel algorithm based on the domain-decomposition method for the full-wave analysis of 3D electromagnetic problems, *IEEE Transactions on Microwave Theory and Techniques*, **50**(8), 2011-2017.
- Zhang, F., 2005, The Schur complement and its applications, *Springer,* ISBN 0387242716.
- Zyserman, F. & Santos, J., 2000. Parallel finite element algorithm with domain decomposition for three-dimensional magnetotelluric modelling, *Journal of Applied Geophysics*, **44**(4), 337-351.
- Zyserman, F., Guarracino, L., & Santos, J., 1999. A hybridized mixed finite element domain decomposed method for two dimensional magnetotelluric modelling, *Earth Planets Space*, **51**(4), 297-306.

7. Acknowledgements

This research has been supported by Thai Center of Excellence in Physics (ThEP) and by Thailand Research Fund (TRF: RMU5080025). The authors would like to thank Colin G. Farquharson, anonymous reviewer and the editor, Oliver Ritter, for their comments to help improve the manuscript

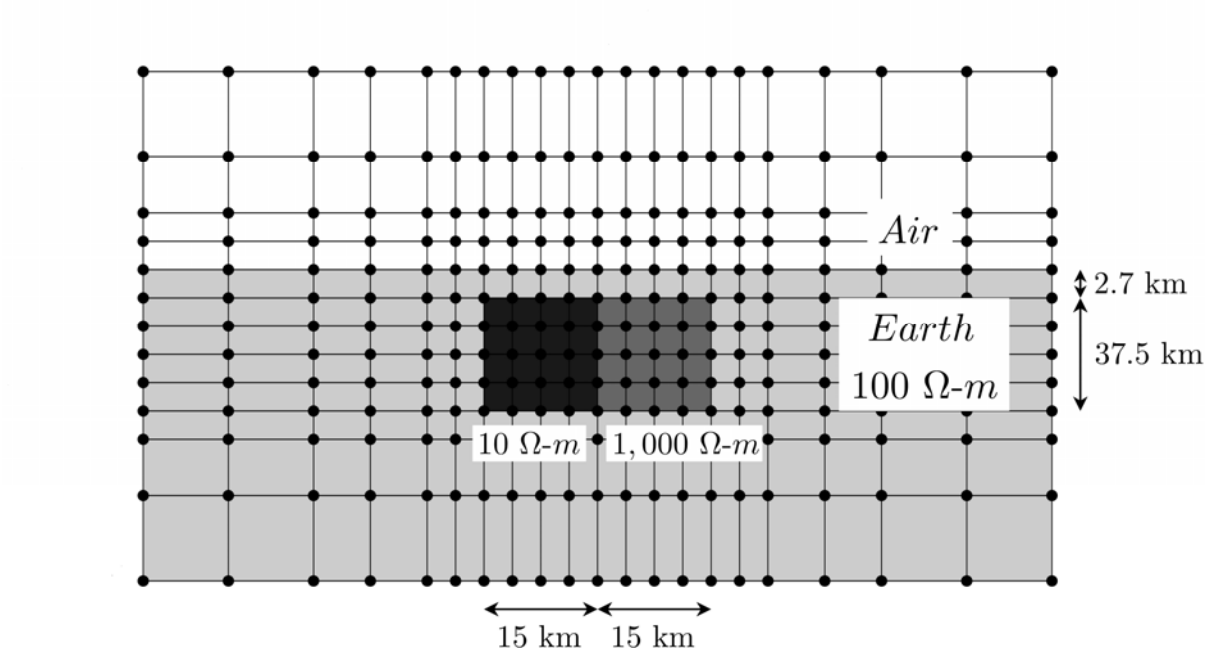


Figure 1. Model used to test the efficiency and accuracy of the modified hierarchical domain decomposition method. The model consists of two resistivity contrast blocks buried in a 100 $\Omega - m$ half-space. The left and right blocks are 10 $\Omega - m$ and 1,000 $\Omega - m$, respectively. This model is discretized into three finite difference meshes: 40 × 120, 80 × 240 and 120 × 360 and are used in the numerical experiment section. Discretization shown in this figure is merely an example to illustrate that the unknown fields are defined on the nodes (black dots).

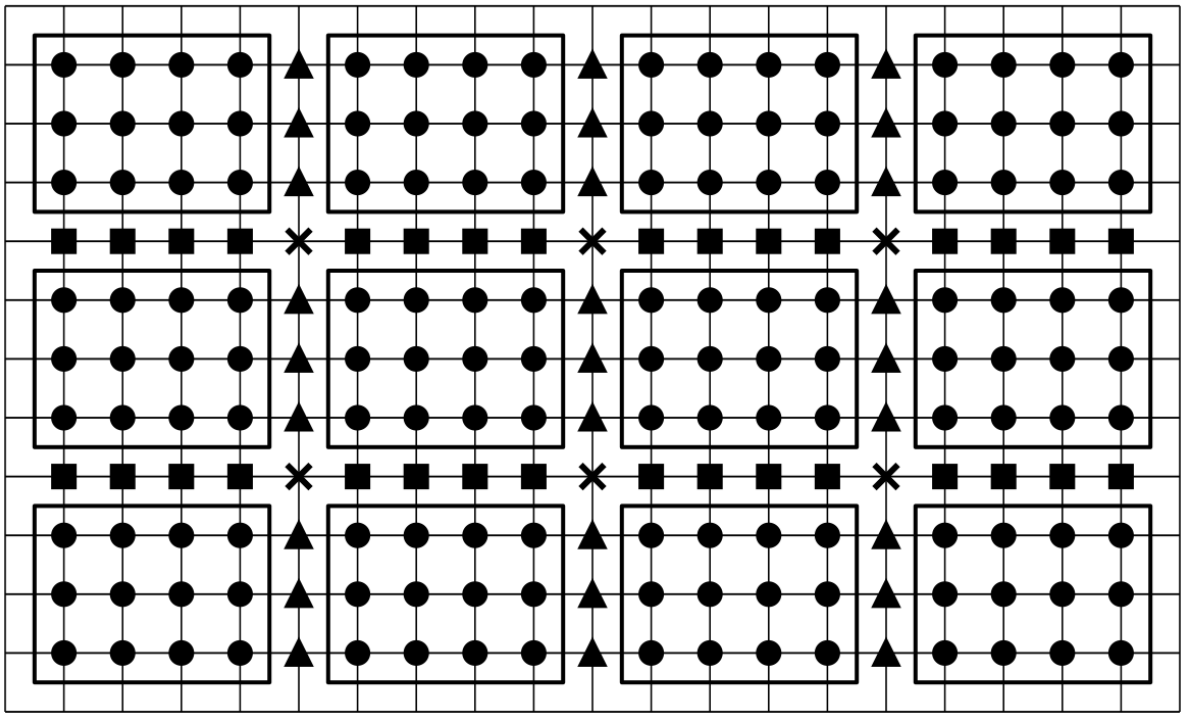


Figure 2. Example mesh of Figure 1 is uniformly redrawn, and subdivided into 3 × 4 subdomains as an illustration here. The interiors inside each subdomain are drawn with solid circle (•). The horizontal and vertical interfaces between subdomains are shown with solid rectangle (■) and solid triangle (▲), respectively. The intersections from four subdomains are plotted with solid cross (✖).

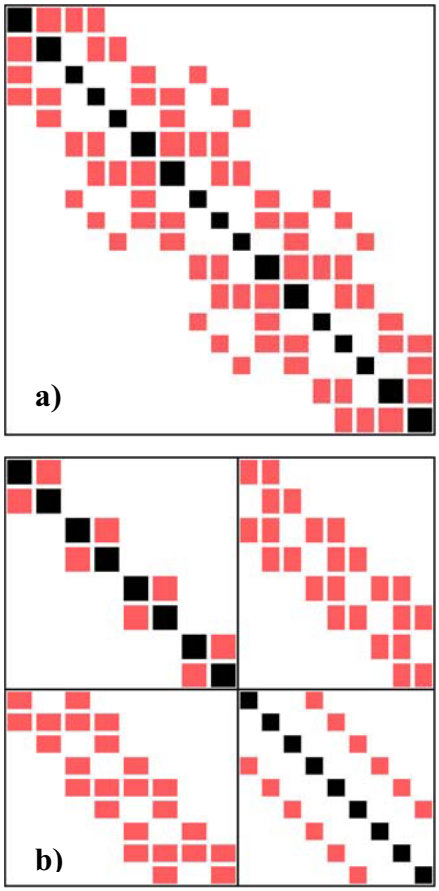


Figure 3. (a) Sparsity pattern of the Schur complement matrix **S** (equation 5) of the non-modified hierarchical domain decomposition. (b) Sparsity pattern of the Schur complement interface systems ($\mathbf{S_{HH}}$, $\mathbf{S_{HV}}$, $\mathbf{S_{VH}}$ and $\mathbf{S_{VV}}$ in equation 12) of the modified hierarchical domain decomposition.

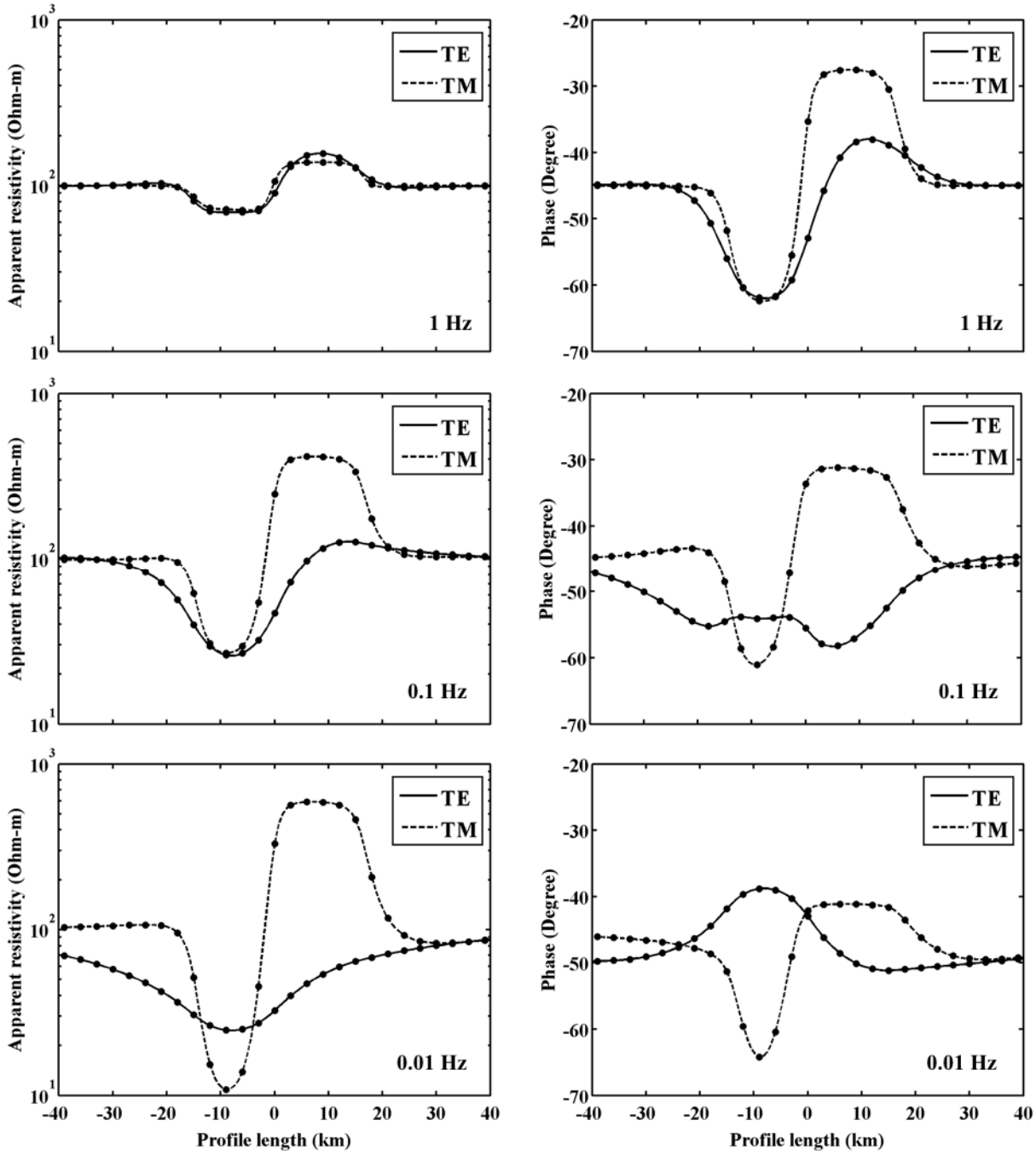


Figure 4. Apparent resistivities (Ohm-m) and phases (degree) of TM and TE modes from three different frequencies (1 Hz, 0.1 Hz and 0.01 Hz) across the model in Figure 1. Dots are from MHDD2D. Solid and dash lines are from TM and TE of FD2D, respectively. The differences of both responses from both methods are in the round-off level. This validates our MHDD2D code.

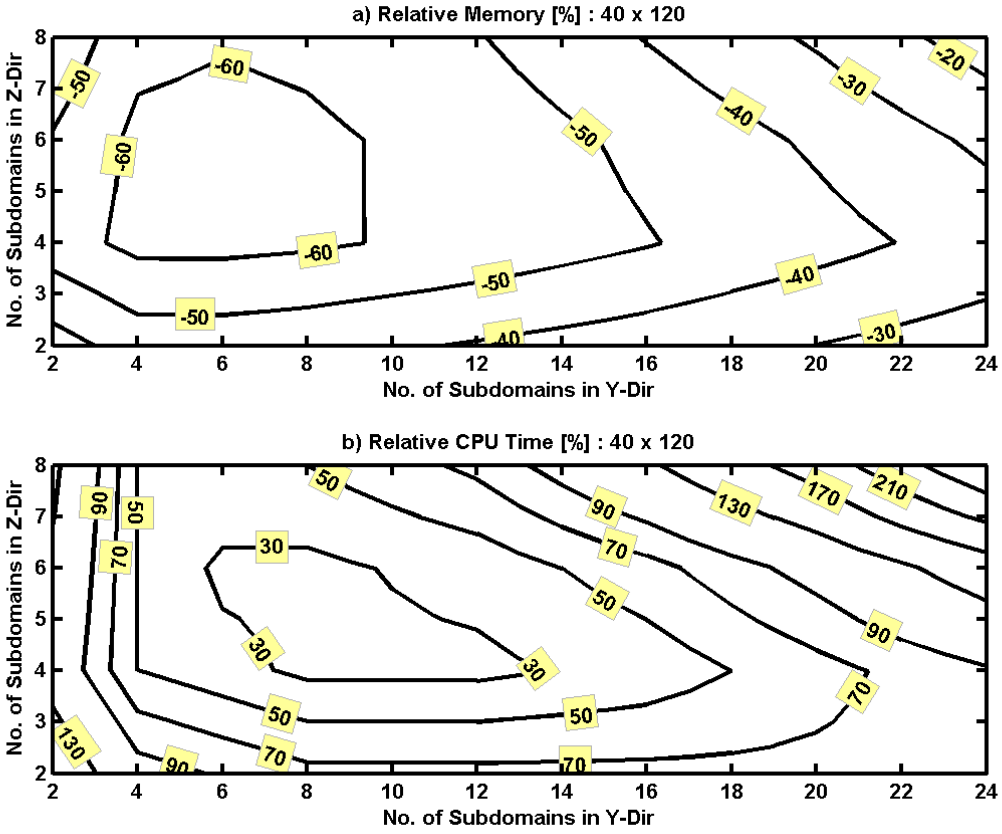


Figure 5. (a) Relative memory usage (in percent) and (b) relative CPU time (in percent) of MHDD2D to FD2D from several combinations of subdomains running on a 40 × 120 mesh. MHDD2D is more efficient than FD2D where larger negative percentage is presented, and less efficient where larger positive percentage.

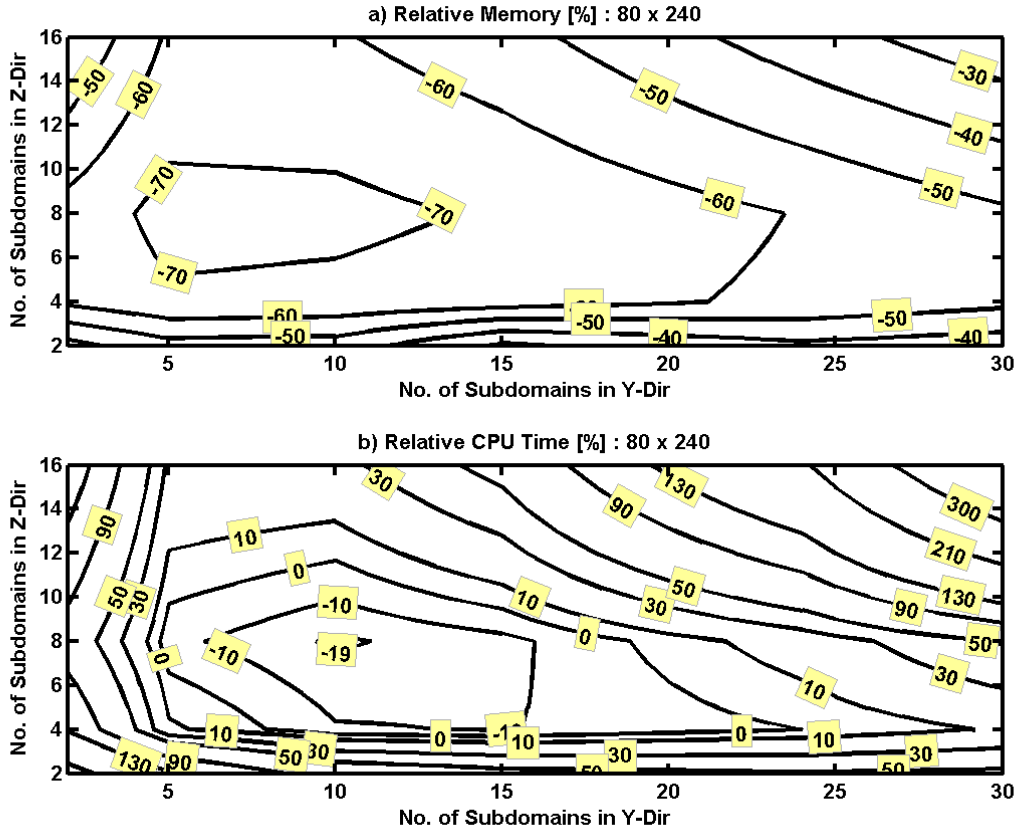


Figure 6. Same captions as in Figure 5 but for 80 × 240 mesh.

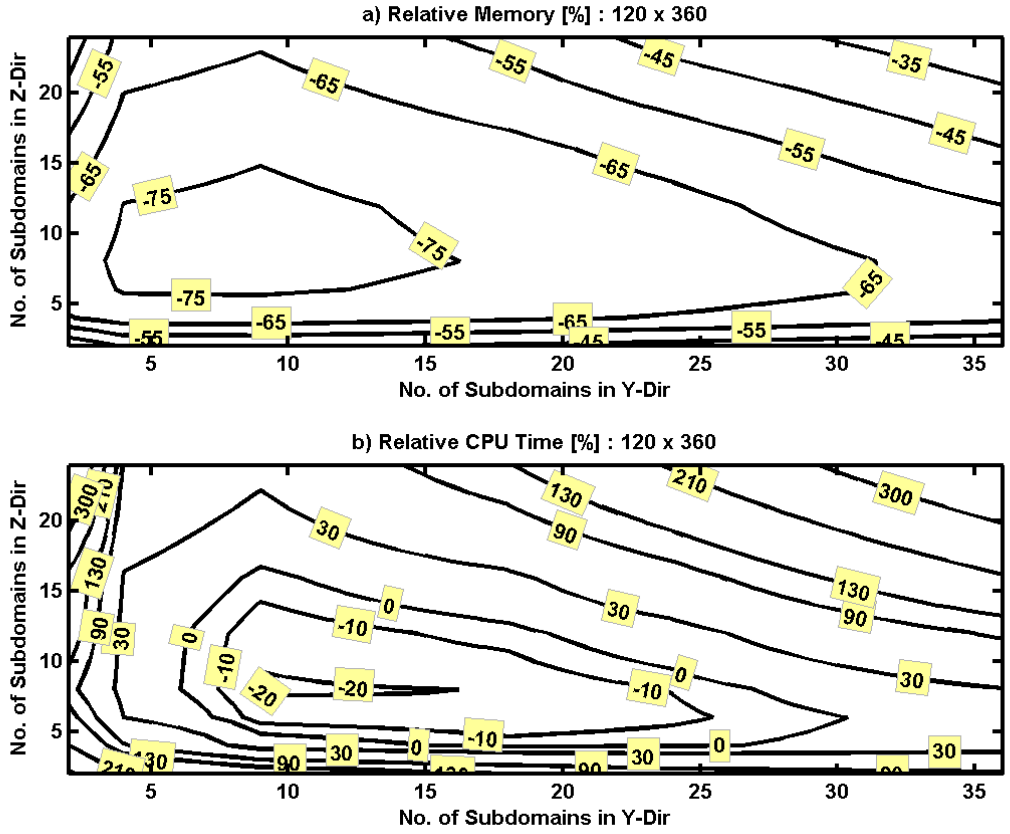


Figure 7. Same captions as in Figure 5 but for 120 × 360 mesh.

$p \times q$ subdomains	l/r (l)	n	m_h	m_v	m
3 × 3	4641 (41769)	4	714	234	948
10 × 9	429 (38610)	72	3159	880	4039
20 × 30	55 (33000)	551	6270	2900	9170

Table 1. Numbers of interiors per subdomain (l/r where l is total of interiors and $r = p \times q$), intersections (n), horizontal interfaces (m_h), vertical interfaces (m_v) and all interfaces (m) for three different numbers of subdomains running on a 120 × 360 mesh (Figure 7).

Center of optimized region	l/r (l)	n	m_h	m_v	m
5 × 6 subdomains on 40 × 120 mesh	133 (3990)	20	456	175	631
8 × 8 subdomains on 80 × 240 mesh	261 (16704)	49	1624	504	2128
10 × 9 subdomains on 120 × 360 mesh	429 (38610)	72	3159	880	4039

Table 2. Numbers of interiors per subdomain (l/r where l is total of interiors and $r = p \times q$), intersections (n), horizontal interfaces (m_h), vertical interfaces (m_v) and all interfaces (m) for 5 × 6 subdomains on 40 × 120 mesh (Figure 5), 8 × 8 subdomains on 80 × 240 mesh (Figure 6), and 10 × 9 subdomains on 120 × 360 mesh (Figure 7), respectively. These subdomains represent the center of optimized regions.

ภาคผนวก ค. Reprint

Siripunvaraporn W. and G. Egbert, 2009, WSINV3DMT: Vertical Magnetic Field Transfer Function Inversion and Parallel Implementation, *Physics of the Earth and Planetary Interiors* 173 (3-4), pp. 317-329

Physics of the Earth and Planetary Interiors 173 (2009) 317–329

Contents lists available at ScienceDirect

Physics of the Earth and Planetary Interiors

journal homepage: www.elsevier.com/locate/pepi

WSINV3DMT: Vertical magnetic field transfer function inversion and parallel implementation

Weerachai Siripunvaraporn [a,*], Gary Egbert [b]

[a] Department of Physics, Faculty of Science, Mahidol University, Rama VI Rd., Rachatawee, Bangkok 10400, Thailand
[b] College of Oceanic and Atmospheric Sciences, Oregon State University, Corvallis, OR 97331, USA

ARTICLE INFO

Article history:
Received 15 September 2008
Received in revised form 21 January 2009
Accepted 25 January 2009

Keywords:
Magnetotellurics
Vertical magnetic transfer function
Data space method
3-D inversion
Occam's inversion

ABSTRACT

We describe two extensions to the three-dimensional magnetotelluric inversion program WSINV3DMT (Siripunvaraporn, W., Egbert, G., Lenbury, Y., Uyeshima, M., 2005, Three-dimensional magnetotelluric inversion: data-space method. Phys. Earth Planet. Interiors 150, 3–14), including modifications to allow inversion of the vertical magnetic transfer functions (VTFs), and parallelization of the code. The parallel implementation, which is most appropriate for small clusters, uses MPI to distribute forward solutions for different frequencies, as well as some linear algebraic computations, over multiple processors. In addition to reducing run times, the parallelization reduces memory requirements by distributing storage of the sensitivity matrix. Both new features are tested on synthetic and real datasets, revealing nearly linear speedup for a small number of processors (up to 8). Experiments on synthetic examples show that the horizontal position and lateral conductivity contrasts of anomalies can be recovered by inverting VTFs alone. However, vertical positions and absolute amplitudes are not well constrained unless an accurate host resistivity is imposed *a priori*. On very simple synthetic models including VTFs in a joint inversion had little impact on the inverse solution computed with impedances alone. However, in experiments with real data, inverse solutions obtained from joint inversion of VTF and impedances, and from impedances alone, differed in important ways, suggesting that for structures with more realistic levels of complexity the VTFs will in general provide useful additional constraints.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

WSINV3DMT (Siripunvaraporn et al., 2005) has been developed to invert Magnetotelluric (MT) impedance tensor components for three-dimensional (3-D) Earth conductivity. It was made freely available to the MT research community in 2006 and has since become one of the standard tools for 3-D inversion and interpretation (e.g., Tuncer et al., 2006; Heise et al., 2008; among others). The inversion algorithm used closely follows the two-dimensional (2-D) data space Occam's inversion of Siripunvaraporn and Egbert (2000) which has also been widely used for 2-D interpretation (e.g., Pous et al., 2002; Oskooi and and Perdersen, 2005; Toh et al., 2006; among others). Here we describe extensions to this code, which we illustrate with tests on synthetic and real data.

We first briefly summarize WSINV3DMT; see Siripunvaraporn et al. (2005) for more technical details. The algorithm used is based on the classic Occam's inversion introduced by Constable et al. (1987) for the one-dimensional (1-D) MT and DC resistivity sounding problems. The Occam inversion seeks a minimum structure model (as defined by some model norm which penalizes roughness) subject to an appropriate fit to the data. The minimization is accomplished with a modified Gauss–Newton algorithm, in which the regularization parameter (which controls the tradeoff between model roughness and data fit) is also used for step length control (Parker, 1994). The main advantages of the Occam approach are its stability and robustness, and the fact that the scheme often converges to the desired misfit in a relatively small number of iterations (e.g., Siripunvaraporn and Egbert, 2000). Occam was extended to treat two-dimensional MT data by deGroot-Hedlin and Constable (1990), but for multi-dimensional inversion the originally proposed scheme can be computationally impractical, as the system of normal equations is explicitly formed and solved in the model space.

Siripunvaraporn and Egbert (2000) transformed the inverse problem into the data space (e.g., Parker, 1994). If the number of data (N) is small compared to the number of model parameters (M), as will typically be the case in 3-D, the data space variant requires a fraction of the CPU time and memory compared to a model space scheme. This data space Occam scheme forms the basis for the WSINV3DMT algorithm, which is summarized in Fig. 1.

The initial version of WSINV3DMT was only capable of inverting the impedance tensor $\mathbf{Z}$, the 2×2 complex frequency dependent

* Corresponding author. Tel.: +662 201 5770; fax: +662 354 7159.
E-mail address: scwsp@mahidol.ac.th (W. Siripunvaraporn).

0031-9201/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.pepi.2009.01.013

Nomenclature

$\mathbf{d}$	observed data
$\mathbf{C}_d$	data error
$\mathbf{m}_0$	initial and prior model
$\mathbf{C}_m$	model covariance
$\mathbf{m}_k$	model at k iteration
$\mathbf{J}_k$	$N \times M$ sensitivity matrix forming from $\mathbf{m}_k$
$\mathbf{F}[\mathbf{m}_k]$	forward responses of $\mathbf{m}_k$
$\mathbf{\Gamma}_k$	data space cross product matrix
$\mathbf{R}_k$	representer for k iteration
λ	Lagrange multiplier
N_s	number of stations
N_m	number of modes
N_p	number of periods
N	number of data $= N_s \times N_m \times N_p$
M	number of model parameters

transfer function relating electric to magnetic fields

$$\begin{bmatrix} E_x \\ E_y \end{bmatrix} = \begin{bmatrix} Z_{xx} & Z_{xy} \\ Z_{yx} & Z_{yy} \end{bmatrix} \begin{bmatrix} H_x \\ H_y \end{bmatrix}. \tag{1}$$

The impedance tensor is frequently used by itself for 3-D conductivity imaging (e.g., Tuncer et al., 2006; Heise et al., 2008; Patro and Egbert, 2008). However, modern MT field practice typically includes measurement of vertical magnetic fields (particularly at long periods, where a tri-axial magnetometer is used), and thence computation of vertical field transfer functions (VTFs)

$$H_z = \begin{bmatrix} T_{zx} & T_{zy} \end{bmatrix} \begin{bmatrix} H_x \\ H_y \end{bmatrix}. \tag{2}$$

The vertical magnetic field is only produced when there are lateral or horizontal variations of conductivity. Researchers have often used VTFs in the form of induction vectors (Parkinson, 1959) to indicate or point to the source of conductivity anomalies and to establish or verify geoelectic strike directions (e.g., Bedrosian et al., 2004; Uyeshima et al., 2005; Tuncer et al., 2006). A number of 2-D inversion codes (e.g., REBOCC of Siripunvaraporn and Egbert, 2000; and NLCG of Rodi and Mackie, 2001) allow inversion of VTFs (or "Tipper"), and these are often included along with TE and TM impedances in 2-D interpretations of MT profile data (e.g., Wannamaker et al., 1989; Wannamaer et al., 2008). Berdichevsky et al. (2003) studied VTFs using analytical and modeling studies, and concluded that inclusion of these additional induction transfer functions can substantially improve the reliability of geoelectrical models, because they are not affected as strongly by local distortion as the impedance tensor is.

Here, we describe the implementation of VTF inversion for the WSINV3DMT inversion code, and apply this to inversion of real and synthetic datasets. In addition, we describe implementation of a parallel version of WSINV3DMT, using MPI and parallelizing over frequencies to help reduce program execution times, which can be quite long for realistic modern datasets (e.g., Patro and Egbert, 2008).

The paper is organized as follows. First, we summarize the modifications to WSINV3D, for the most part omitting technical details. Next, we illustrate and test the new features on the same synthetic datasets previously used in Siripunvaraporn et al. (2005). Here we illustrate the speedup obtained with the parallelization, and explore the effectiveness of VTF data for recovering conductivity structures, alone, and in conjunction with impedance data. We then test the VTF inversion on the EXTECH dataset (Tuncer et al., 2006), comparing inverted models from only VTF data, from only impedance data, and from a joint inversion of both data types.

2. Implementation of WSINV3DMT to include the vertical magnetic transfer function

There are only two major modifications to the WSINV3DMT codes required to allow inversion of VTFs: adding the VTF computation to the forward modeling routine, and the corresponding modifications for the sensitivities of the real and imaginary parts of the VTFs.

In WSINV3DMT, the electric fields are calculated by solving the second order Maxwell's equation using a staggered grid finite difference numerical scheme (Siripunvaraporn et al., 2002). Magnetic field components can then be computed (on grid cell faces) from Faraday's law $\nabla \times \mathbf{E} = i\omega\mu\mathbf{H}$, and interpolated to the observation locations, which in the modified version of WSINV3D can be at any location on the surface. In order to compute the impedance tensor $\mathbf{Z}$ the forward equations are solved for two polarizations, and $\mathbf{Z}$ is calculated from the combination of horizontal electric and magnetic fields from both polarizations, as described in Siripunvaraporn et al. (2005).

The only modification required for the VTF is that the vertical magnetic field must also be computed at the observation location. As for the horizontal magnetic components, this is accomplished using Faraday's law, taking the curl of the horizontal $\mathbf{E}$ components on the model air–Earth interface, and interpolating the result (defined at cell centers) to the observation locations. Then, similarly to the impedance tensor, the vertical and horizontal magnetic fields computed from the solutions for both polarizations are combined to form the vertical magnetic field transfer function $\mathbf{T}$,

$$\begin{bmatrix} \boldsymbol{H}_z^1 & \boldsymbol{H}_z^2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{T}_{zx} & \boldsymbol{T}_{zy} \end{bmatrix} \begin{bmatrix} \boldsymbol{H}_x^1 & \boldsymbol{H}_x^2 \\ \boldsymbol{H}_y^1 & \boldsymbol{H}_y^2 \end{bmatrix} \tag{3}$$

Here $\boldsymbol{H}_z^1$ and $\boldsymbol{H}_z^2$ are the z-component of magnetic fields for the $\mathbf{E}_x$–$\mathbf{H}_y$ and $\mathbf{E}_y$–$\mathbf{H}_x$ polarizations, respectively, and similarly for other field components. For a joint inversion with impedance tensor, computing the vertical magnetic transfer function does not require any extra forward modeling calls, as all transfer functions are computed from the same solutions.

The sensitivity calculation for VTFs is essentially identical to that used for impedances, which is based on the reciprocity approach described in Rodi (1976), Newman and Alumbaugh (2000), and Siripunvaraporn et al. (2005). Briefly, the linearized data functional, which is represented by linear combinations of electric field solution components on cell edges surrounding the observation point, is used to force the adjoint equation, and the result is mapped to perturbations in the model parameter, as described in Siripunvaraporn et al. (2005). Only the first step requires modification, with the coefficients for the linearized functionals for $\mathbf{T}_{zx}$ and $\mathbf{T}_{zy}$ replacing those for $\mathbf{Z}_{xx}$ and $\mathbf{Z}_{xy}$. Details of this modification are straightforward, and are omitted here.

3. Parallel implementation with MPI

A major challenge in using WSINV3DMT, or for that matter, any 3-D MT inversion code, is that the program is very time consuming, especially when run with the sort of large dataset (and model domain) that justifies a 3-D interpretation. Run times exceeding a full month (on a single processor desktop computer, for the full inversion process, including multiple iterations of the outer loop of Fig. 1) have been reported when WSINV3D has been applied to even modest 3-D MT datasets (e.g., Patro and Egbert, 2008). These long run times primarily reflect the need for many forward modeling calls, each of which requires iterative

Serial WSINV3DMT algorithm:

1) Solve forward problem and compute misfit from model $\mathbf{m_0}$

2) Start WSINV3DMT outer loop iteration k:

2.1) For $i = 1$ to $N_s*N_m*N_p$

Call forward solver to form $\mathbf{J_{ki}}$ sensitivity for data i

End

2.2) Compute $\tilde{\mathbf{d}}_k = \mathbf{d} - \mathbf{F}[\mathbf{m}_k] + \mathbf{J}_k(\mathbf{m}_k - \mathbf{m}_0)$

2.3) Compute $\Gamma_k = \mathbf{C_d}^{-½}\mathbf{J}_k\mathbf{C_m}\mathbf{J}_k^{\mathrm{T}}\mathbf{C_d}^{-½}$

2.4) For various values of λs

2.4.1) Compute representer matrix $\mathbf{R}_k = [\lambda\,\mathbf{I} + \Gamma_k]$

2.4.2) Use Cholesky decomposition to solve

$\mathbf{m}_{k+1} - \mathbf{m_0} = \mathbf{C_m}\mathbf{J}_k\mathbf{C_d}^{-½}\mathbf{R}_k^{-1}\mathbf{C_d}^{-½}\tilde{\mathbf{d}}_k$

2.4.3) Solve forward problem and Compute misfit from model $\mathbf{m}_{k+1}$

2.4.4) Phase I :

Compare misfit from different λs to seek for minimum misfit

Phase II:

Compare norm from different λs to seek minimum norm

End

2.5) Exit when misfit less than desired level with minimum norm

End WSINV3DMT outer loop iteration

Fig. 1. Pseudo-code for serial WSINV3DMT (after Siripunvaraporn and Egbert, 2007).

solution of the large sparse linear system arising from discretization of Maxwell's equations. WSINV3D was developed as a serial code, to run on a single processor. An obvious way to speed up execution is to parallelize the code, and make use of the multiple processors which are increasingly common even in desktop computers.

There are several ways to redesign the codes to run on parallel system, and the most appropriate approach will depend on system architecture. For supercomputers or large clusters to make effective use of hundreds of processors it would be necessary to rewrite parts of the forward solver—e.g., parallelizing the iterative solver and preconditioner (e.g., Newman and Alumbaugh, 2000), or domain decomposition. Here, we consider a parallelization approach appropriate to small systems with a few to several tens of processors. Such small clusters and multi-processor workstations are now readily affordable and more widely available than supercomputers. To adapt WSINV3DMT for this class of systems, we parallelize over frequencies, adding calls to MPI (Message Passing Interface) library routines to the existing codes. In this way, we do not have to alter the core forward modeling and sensitivity calculation routines in any way. The parallel algorithm is summarized in Fig. 2.

Forward modeling and sensitivity calculations for each period are sent to one processor (Steps 2.1 and 2.2 in Fig. 2). If there are fewer processors than periods, each processor performs calculations for more than one period. With this simple parallelization, which requires minimal inter-processor communication, the computational time should be theoretically reduced by a factor P, the number of processors available. This parallel implementation also distributes storage of the sensitivity matrix over the available nodes. The $N \times M$ sensitivity matrix $\mathbf{J}$ requires $8NM$ bytes (in double precision), and the need to store this in RAM limits the size of datasets and model grids that can be practically treated. With the parallelization, memory required on each node is reduced to about two times $8NM/P$ (including temporary storage for cross product computations), allowing WSINV3D to be run for larger models grids and datasets.

With the sensitivities distributed over processors, formation of the cross product matrix $\mathbf{\Gamma} = \mathbf{J}\mathbf{C}_m^{-1}\mathbf{J}^T$ also requires MPI calls. We have implemented this in a fairly simple way, breaking $\mathbf{\Gamma}$ into P^2 blocks to be computed on the P processors (Step 2.3 in Fig. 2). Diagonal blocks $\mathbf{\Gamma}_{ii}$ are computed on the local processor where the corresponding block $\mathbf{J}_i$ of the sensitivity matrix (corresponding to one or more frequencies) is computed and stored. The off-diagonal blocks ($\mathbf{\Gamma}_{ij}$) can only be formed by sharing blocks of $\mathbf{J}$ between nodes. Since $\mathbf{\Gamma}$ is symmetric, only upper off-diagonal blocks ($j > i$) need be formed. On step k block $\mathbf{J}_j$, where $j = \mathrm{mod}(i + k, P)$ is sent to node i to compute $\mathbf{\Gamma}_{ij}$ where this block is stored. With this simple scheme the load is balanced and the number of steps required is approximately $(N_p + 1)/2$. Although computing the cross products requires significant communication time to share sensitivities between nodes, it can still significantly reduce the total computing time required to form $\mathbf{\Gamma}$ compared to single node processing.

In the data space Occam scheme used by WSINV3D the system of normal equations (Eq. (6) in Siripunvaraporn et al., 2005) must be solved for a series of trial values of the regularization parameter (about 3–7 from our experience) to find the optimal (in terms of data misfit and model norm) model parameter update. In the serial version of WSINV3D these dense systems are solved by Cholesky decomposition (Step 2.4.2 in Fig. 1). Parallel Cholesky decomposition subroutines are available (e.g., Choi and Moon, 1997), but these are generally tailored to a specific parallel architecture and are not easily adapted. To develop code that will be portable, and reasonably efficient on a generic multi-processor system, we have thus pursued a different strategy, using the easily parallelized preconditioned conjugate gradient (PCG) algorithm to solve the normal equations (Step 2.4.1.2 in Fig. 2). The major computation in the

Parallel WSINV3DMT algorithm for *P*-cluster PCs nodes:

0) Parent node distributes data to other nodes. Each node would then takes care a computational load of N_p/P data.

1) Each node separately solve forward problem and compute misfit from model $\mathbf{m_0}$

2) Start parallel WSINV3DMT outer loop iteration k:

2.1) In each node,

For $i = 1$ to $N_s * N_m$; Call forward solver to form local $\mathbf{J}_{ki}$ sensitivity for data i

2.2) Each node compute $\tilde{\mathbf{d}}_k = \mathbf{d} - \mathbf{F}[\mathbf{m}_k] + \mathbf{J}_k(\mathbf{m}_k - \mathbf{m}_0)$ separately.

2.3) To compute $\Gamma_k = \mathbf{C_d}^{-1/2}\mathbf{J}_k\mathbf{C_m}\mathbf{J}_k^T\mathbf{C_d}^{-1/2}$,

2.3.1) each node first computing local or diagonal Γ_{ii} from their local $\mathbf{J}_{ki}$

2.3.2) each node cyclically sending their local $\mathbf{J}_{kl}$ to others nodes to compute the off-diagonal Γ_{il}.

2.4) For various values of λs

2.4.1) If N is small (single node process)

2.4.1.1) Sending Γ_k from local nodes to parent nodes to compute global representer matrix $\mathbf{R}_k = [\lambda \mathbf{I} + \Gamma_k]$

2.4.1.2) On parent node, applying Cholesky decomposition to solve

$$\mathbf{m}_{k+1} - \mathbf{m}_0 = \mathbf{C_m}\mathbf{J}_k\mathbf{C_d}^{-1/2}\mathbf{R}_k^{-1}\mathbf{C_d}^{-1/2}\tilde{\mathbf{d}}_k$$

else if N is large (parallel process)

2.4.1.3) Local $\mathbf{R_k} = [\lambda \mathbf{I} + \Gamma_k]$ is formed in each node

2.4.1.4) Parallel iterative solver (PCG) is applied to solve

$$\mathbf{m}_{k+1} - \mathbf{m}_0 = \mathbf{C_m}\mathbf{J}_k\mathbf{C_d}^{-1/2}\mathbf{R}_k^{-1}\mathbf{C_d}^{-1/2}\tilde{\mathbf{d}}_k$$

2.4.2) Each node separately solve forward problem and compute misfit from model $\mathbf{m}_{k+1}$

2.4.3) On parent node

Phase I : Compare misfit from different λs to seek for minimum misfit

Phase II: Compare norm from different λs to seek minimum norm

2.5) Exit when misfit less than desired level with minimum norm

End parallel WSINV3DMT outer loop iteration

Fig. 2. Pseudo-code for parallel WSINV3DMT for cluster PCs system.

PCG algorithm is matrix–vector multiplication. This is readily parallelized by dividing the vectors and matrix into blocks, spreading computations for individual blocks over processors, and then gathering the results back to the master node. To simplify the algorithm we have distributed the full matrix to all computational nodes.

The preconditioner, based on the diagonals of the coefficient matrix, is also trivially parallelized. Because the coefficient matrices are dense, the parallel PCG scheme may not be efficient when N is small, since communication and other overhead may exceed the serial computational time. For smaller N, we therefore retain the option of solving the normal equations with a serial Cholesky decomposition, after all blocks $\mathbf{\Gamma}_{ij}$ are sent back to the parent node. The optimal choice of solution scheme (parallel or serial) for a specific value of N will depend on the cluster architecture. We give examples below where each approach is more efficient.

Once the new model $\mathbf{m}_{k+1}$ is obtained, the parallelized forward solver is called to compute the responses of each period, with the results gathered to the parent node to compute misfits (Step 2.4.2 in Fig. 2). All steps are repeated until an acceptable misfit and norm are achieved

4. Synthetic data examples

To illustrate the efficiency of the parallelized WSINV3D, and the effectiveness of the VTF inversion, we first consider inversion of synthetic datasets, revisiting the two synthetic examples previously used by Siripunvaraporn et al. (2005), reproduced in Fig. 3. The results of these tests are consistent with those obtained for other synthetic examples. Our basic test configuration is the two-block model (Fig. 3a) consisting of two anomalies, 1 Ω m and 100 Ω m located next to each other within a 10 Ω m host. The spatially homogeneous layer, which extends from the surface to 10 km depth, is underlain by a 100 Ω m half space. To test the efficiency of our parallel codes, and the VTF inversion, we generated VTF and impedance data at 16 periods (from 0.1 to 1000 s) for a total of 40 sites in a regular grid, as illustrated in Fig. 3a. Gaussian noise (5% of the data magnitude) was added to the generated data. The inversions for this case are performed on a $21 \times 28 \times 21$ (+7 air layers) mesh. The second model consists of a single conductive block (1 Ω m) buried in a 100 Ω m half-space (Fig. 3b), and responses were computed at 16 periods for 36 sites (Fig. 3b). The inversions

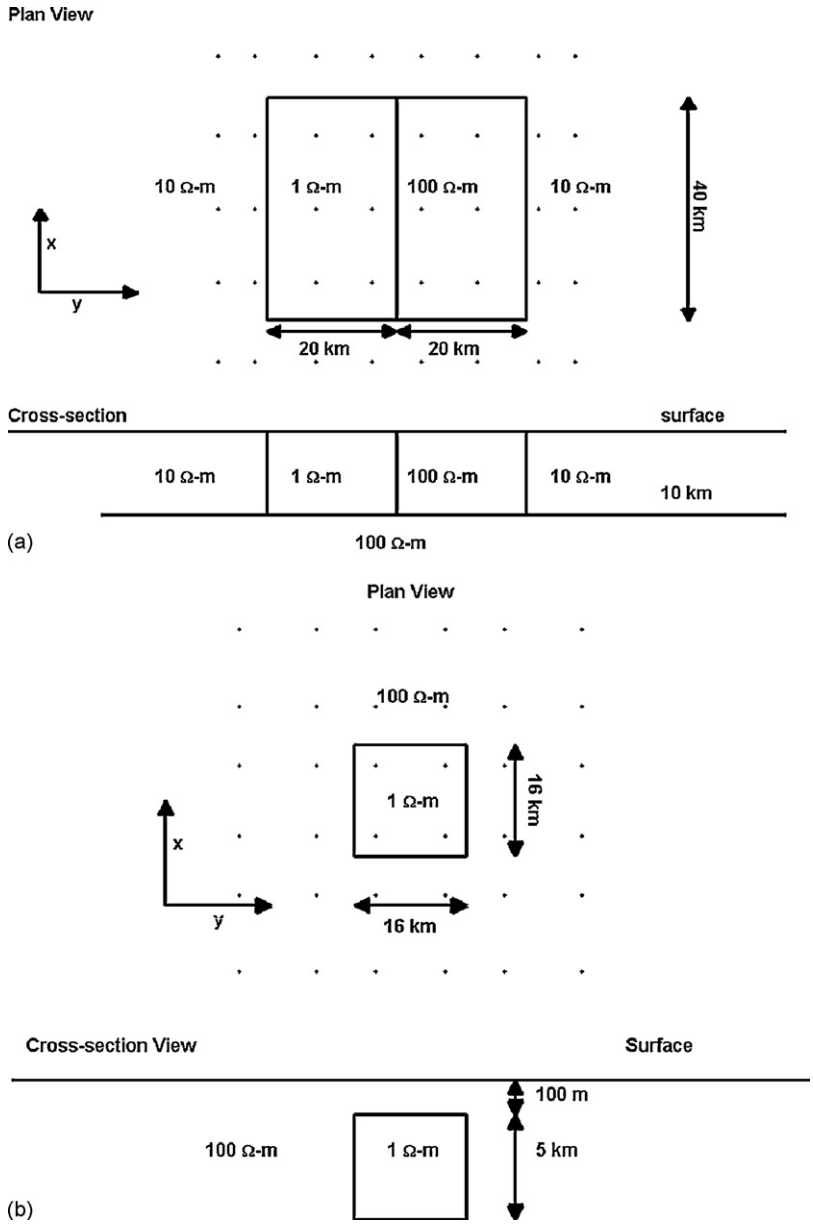


Fig. 3. Two synthetic models used to test our inversion. (a) Two-block synthetic model and (b) a single conductive block model. The solid dots indicate the observation sites. The cross-section view in the lower panel is a profile cutting across the middle of the model in the upper panel, and is not to scale (after Siripunvaraporn et al., 2005).

for the second case are performed on a $28 \times 28 \times 21$ (+7 air layers) mesh.

We first demonstrate the efficiency of the parallel version of WSINV3D, using both VTF and joint VTF/impedance datasets for tests. We then consider the effectiveness of VTF data for recovering conductivity variations, both alone, and in conjunction with impedances.

4.1. Parallel efficiency

We tested WSINV3DMT by running on 1, 4, 8 and 16 nodes for the first synthetic test case (Fig. 3a), with the 16 periods divided evenly among nodes (e.g., with 4 nodes, each solves for 4 periods). Tests were conducted on a small PC-clusters and a supercomputer (SGI Altix 4700) at the Earthquake Research Institute, University of Tokyo. To quantify efficiency of the parallel code, we display the speedup, defined as $S = T_1/T_P$, where T_1 is the execution time of the sequential WSINV3DMT algorithm and T_P is the execution time of the parallel version, run on P processors. The idealized maximum speedup is P. However, due to computational overhead, the need for some computations to be performed only on the master node, and the time required to exchange information between nodes, S will always be less than P. Fig. 4 displays speedup versus the number of nodes. Inversions of all data (i.e., VTF + impedance, $N = 40 \times 12 \times 16 = 7680$) are plotted with solid lines. Inversions of the VTF only dataset ($N = 40 \times 4 \times 16 = 2560$, or one third the size of the joint inversion dataset) are plotted as dashed lines. We also compare speedups achieved with the two approaches for solving the normal equations: speedups obtained with the single processor Cholesky decomposition are plotted as solid symbols, while those obtained with the parallel PCG algorithm are plotted as open symbols.

For the inversion of the VTF dataset for this very small test problem, actual (wall clock) run times were about 186 min on a single node machine, 82 min on 4 nodes, 46 min on 8 nodes and 34 min on 16 nodes, resulting in speedups of about 2.2 for 4 nodes, 4 for 8

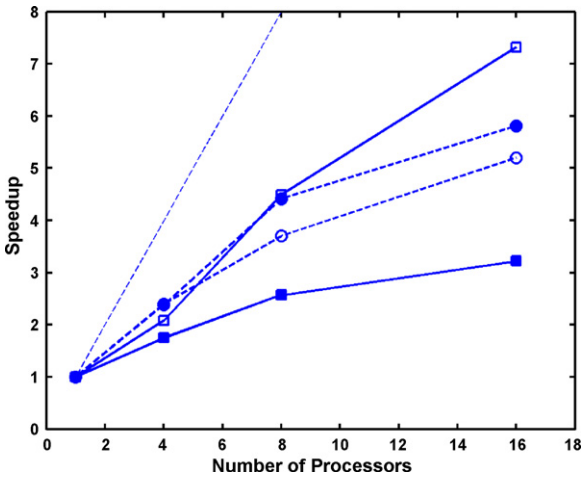


Fig. 4. Speedup versus the number of processors or nodes. Solid lines are the speedups from inversion using both VTF and impedance data ($N = 7680$). Dashed lines are the speedups from inversion using only VTF data ($N = 2560$). Results for the scheme which solves the normal equations by Cholesky decomposition on a single node (step 2.4.1.2 of Fig. 2) are plotted with solid symbols. The corresponding results with the parallel PCG solver (step 2.4.1.4 of Fig. 2) are plotted with open symbols. The thin-dashed line of slope one gives the ideal perfect speedup.

nodes and 5.4 for 16 nodes. Thus, as the number of nodes increases, the relative efficiency of additional nodes decreases. One reason for this is that the run time of the iterative forward modeling routine depends on the period of the data. Shorter periods typically require a larger number of iterations for convergence, and hence longer run times. Thus, some nodes are usually idle waiting for modeling computations to complete on other nodes, before moving on to the next step in the inversion. With fewer nodes some of the frequency-to-frequency variations average out, resulting in better balance.

Efficiencies are somewhat lower for the larger joint VTF/impedance dataset, when the serial Cholesky decomposition solver is used (solid line with solid square symbols of Fig. 4). Now the speedups are about 1.8, 2.6 and 3.2 for 4, 8 and 16 nodes, respectively, almost 50% below those achieved for the VTF only inversion. However, solving the normal equations with the parallel PCG solver (solid line with open square symbols in Fig. 4) significantly improves performance, increasing S to approximately 2, 4.5 and 7.3 for the three cases considered. In the VTF only case, where N is significantly smaller, both methods for solving the normal equations have similar performance (dashed lines in Fig. 4), and indeed the speedup is slightly greater when the single node Cholesky decomposition is used.

The difference between the two cases is readily understood. Operation counts for Cholesky decomposition scale as N^3 so computation times for the serial Cholesky decomposition in the all data case ($N = 7680$) are expected to be about 27 times greater than for the VTF only case ($N = 2560$). Other computational steps scale better with increasing N. For fixed model parameter size, total operation counts for the sensitivity calculations increase linearly in N, and formation of the cross product matrices increases as N^2. Thus, as the size of the dataset increases, run times required for the serial Cholesky decomposition step become increasingly significant, and at large enough N this step will control the overall runtime. Operation counts for a single iteration in the parallel PCG scheme scale as N^2, but overall runtimes will also depend on the number of iterations required. Although this should increase with N also, the dependence is weak, and so PCG becomes increasingly advantageous as N increases, particularly since computations for the PCG scheme can be distributed over the P processors.

The number of iterations for PCG also depends on the relative tolerance for the residual ($=||Ax - b||/||b||$) used to define convergence. We find that a tolerance of 10^{-4} results in models that are essentially identical to those obtained with the Cholesky decomposition technique. The number of iterations, and hence the run time of the parallel PCG scheme also depends on the condition number of the normal equations. For large values of the Lagrange multiplier (corresponding to a smoother model) the condition number is smaller, and the parallel solver thus converges in a small number of iterations. In contrast, when the Lagrange multiplier is very small (rough model) the parallel solver can require considerably more iterations, and solution times can exceed those for the serial Cholesky decomposition scheme. This occurred occasionally in our tests with the larger VTF/impedance dataset, but overall performance using the parallel PCG solver was much better when N is large enough.

We will not attempt to quantify more precisely how large N must be before the parallel approach to normal equation solution would be preferred. This will depend on the cluster architecture, especially on the sort of inter-processor communication used, since the parallel PCG solver requires substantial sharing of data.

In addition to reducing computational times, the parallel version also reduces the need for a large amount of memory on a single computer. Even for the small joint VTF/impedance inversion test example, about 1.5 GBytes are required for the representer and sensitivity matrices. In the parallel implementation, the required memory may be distributed over all of the nodes used. For example, with 16 nodes, each would require only 0.090 GBytes for storing the sensitivity matrix and forming cross products, almost 13 times less than required by the serial code. If the whole representer matrix is stored on a single processor (for the Cholesky decomposition, or to reduce the communication time between nodes for PCG) about 0.4 Gb are required on each node, still only a quarter required for a serial version.

The exact time speedup and per-node memory reduction factors will depend to some extent on the problem size, both in terms of model grid dimensions, and number of data. For larger problems, such as the real data EXTECH example considered below, similar performance gains were attained. For these larger problems, however, a speedup by a factor of roughly 7 means a run time that was perhaps 2–3 weeks on a single node is now reduced to 2–3 days, making inversion of realistic datasets considerably more practical. The practical impact of distributing memory is even greater. Total storage required by WSINV3D for the EXTECH example described below (joint inversion of the full impedance and VTFs) is at least 30 Gb, making this impractical on almost any shared memory machine.

4.2. Vertical magnetic transfer function inversion

We next consider the effectiveness of WSINV3DMT at correctly recovering resistivity when only VTF data are available. Because in practice one would not know *a priori* the correct background resistivity, we run the inversion using several prior (and starting) models. Inversion results for the synthetic VTF data from the test case of Fig. 3a are summarized in Figs. 5 and 6. Using a 50 Ω m half-space as a prior (this is intermediate between the true 10 Ω m upper layer background, and the 100 Ω m basement), inversion of VTF data reveals both the conductive body and the adjacent resistor, extending from near the surface to approximately 20 km depth. The calculated responses generated from the inverse solution of Fig. 5 fit the observed responses within 15% of the typical VTF amplitude (recall that 5% random noise was added to the synthetic data).

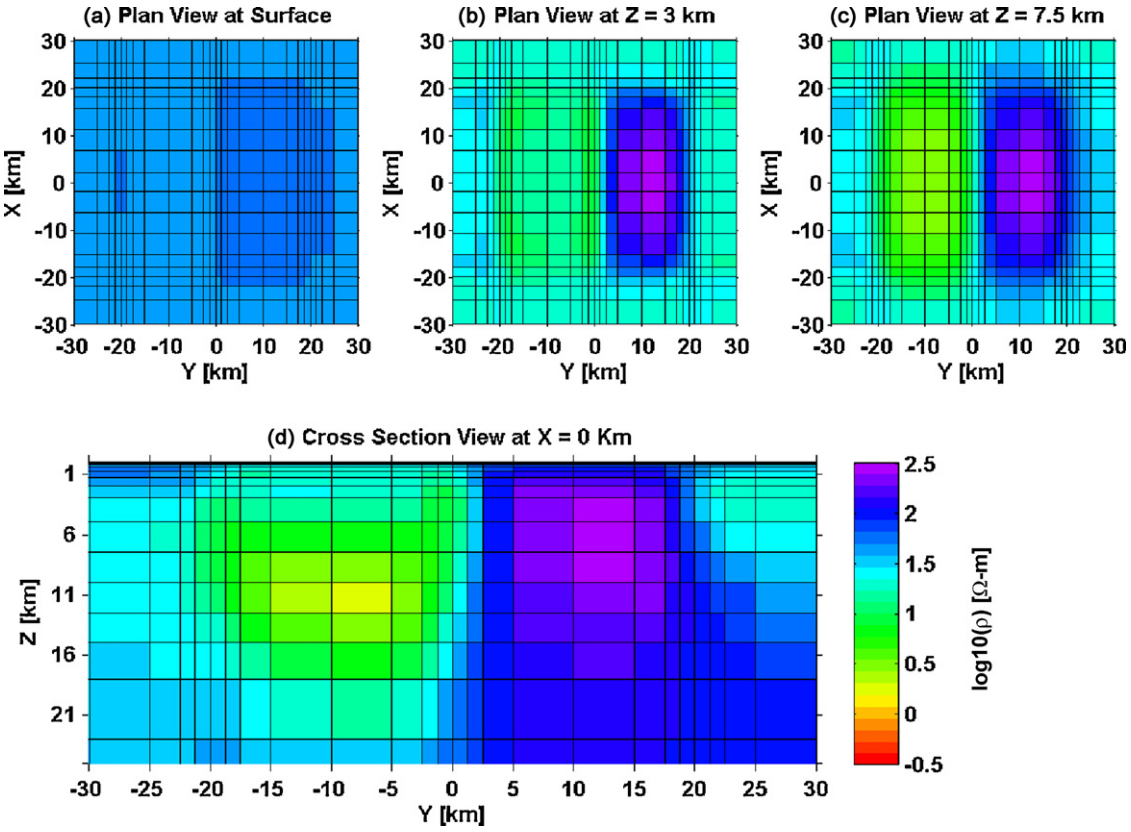


Fig. 5. An inverse solution from the VTF data alone after the 9th iterations with an RMS value of 1, fitting synthetic data generated from the model in Fig. 3a. The top panels (a)–(c) is a plan view at the surface, at 3 km and at 7.5 km depth, and the bottom panel (d) is a cross-section view cutting across the conductive block at $X = 0$ km. The solution is shown only in the central area around the anomalies, not for the full model domain.

Although both anomalies are detected in approximately the correct location, the true resistivities of Fig. 3a are not correctly estimated. However, calculating the average resistivity over the anomalous volumes we find for the inverse model of Fig. 5 an average resistivity of about 6.3 Ω m for the conductive anomaly, and of about 453 Ω m for the resistive body, while the background resistivity of the inverse model was changed only slightly from the 50 Ω m prior. Computing the volume average resistivity ratios from left to right in Fig. 5d, we obtain values of 7.9 (=50/6.3), 72 (=453/6.3) and 9 (=453/50), compared to the actual ratios (Fig. 3a) of 10 (=10/1), 100

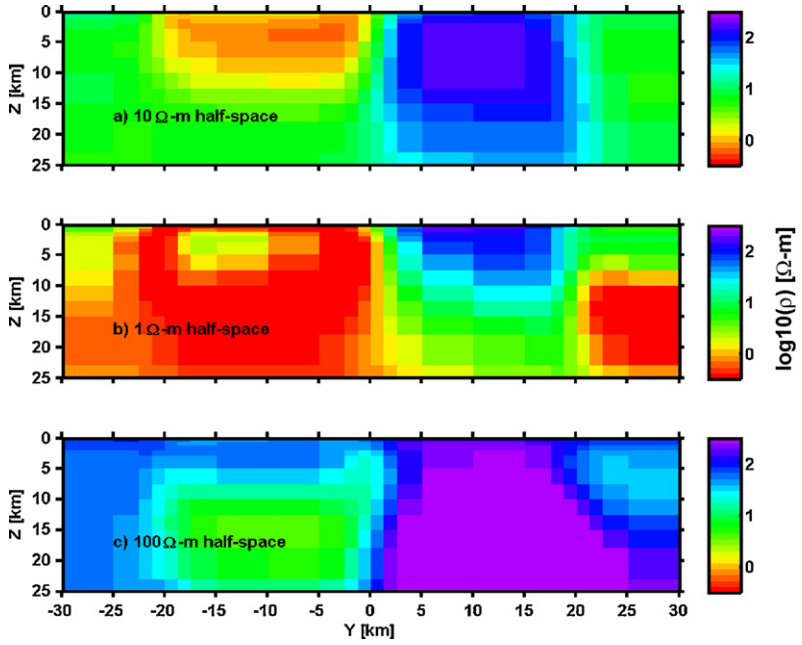


Fig. 6. Cross-sectional plots at $X = 0$ km (as in Fig. 5d) of the inverse solutions from VTF data alone, when the prior models are (a) 10 Ω m half-space, (b) 1 Ω m half-space and (c) 100 Ω m half-space.

(=100/1) and 10 (=100/10), respectively. The inversion thus results in roughly the correct structure, with approximately correct resistivity contrasts, but it does not recover the correct amplitude of either the background or the anomalies, or the actual depth extent of the anomalies.

To explore this issue further we ran the inversion on the same VTF dataset, using a range of values for the assumed half-space prior. Fig. 6 summarizes the results with cross-sectional plots of the inverse solutions at $X = 0$ km. When the prior model is the same as the correct background resistivity (i.e., a 10 Ω-m half-space in our example), the inversion quickly converges to the desired misfit within 4 iterations, even with error floors set to 5%. In this case, the inversion estimates the resistivity, and the depth extents, of the two anomalies quite well (Figs. 6a and 3a). However, the 100 Ω m basement resistivity (below 10 km depth in the synthetic test model of Fig. 3a) is not recovered—the prior resistivity of 10 Ω m remains unchanged at depth in the inverse solution. This again demonstrates that inversion of VTF data alone can only recover lateral resistivity contrasts, and is not effective at correcting resistivities, or their variations with depth.

Larger deviations of the prior model from the correct background result in even larger discrepancies in anomaly amplitudes and depths, but still generally allow the horizontal structure to be recovered. With a 1 Ω m half-space (Fig. 6b) data is fit to within 10%. Anomalies appear at very shallow depths (upper few km), with all features more conductive than their actual values. At greater depth, features with appropriate resistivity ratios are imaged, but the absolute levels are incorrectly estimated, and spurious structures appear. Using a 100 Ω m half-space as a prior, the VTF data can only be fit to within 20%. The basic structure is again recovered, but both anomalies are at greater depth (Fig. 6c) and have increased resistivity. The host resistivity is estimated to be slightly lower than the 100 Ω m starting value, but is still well above the correct value of 10 Ω m. As in the other cases, the basement resistivity remains the same as the prior model.

All of these experiments suggest that when only VTF data are available, to achieve the target misfit and recover correct amplitudes and depths, the inversion must be started with a prior model that is close to the correct host resistivity. However, even starting far from the correct background model, anomalies are recovered with the correct horizontal location and dimensions. This result is not unexpected since the vertical magnetic fields are generated where there are lateral discontinuities, but are not inherently sensitive to the profile of vertical conductivity structure.

In addition, resistivities of anomalous bodies scale with the assumed prior background (Fig. 6), and resistivity contrasts (i.e., ratios) can be close to actual values, especially if the assumed background resistivity is not too far off. However, the VTFs provide little intrinsic constraint on contrasts in the vertical direction, including the location of the top or the bottom of the anomalies. The inversion only gets these properties of the anomalies correct if something close to the correct background is used (Fig. 6a).

Performing similar experiments to those summarized in Fig. 6, but using impedance tensor data shows that these inversions are much less sensitive to the assumed prior model. This is consistent with the basic physics, as the ratio of electric to magnetic fields is intrinsically related to the resistivity profile. In spite of the well-known uncertainties in depth and absolute resistivity level that may result from local static distortions, there is by now ample evidence (e.g., Tuncer et al., 2006; Unsworth et al., 2000) that, with proper care, MT impedances can yield reliable information about conductivity-depth profiles. The same does not appear to be true in practice with VTF data, although theoretical analysis of idealized models suggests otherwise (Berdichevsky et al., 2003).

The above results suggest that VTF data will be most useful as an adjunct to impedance data, which can provide the necessary con-

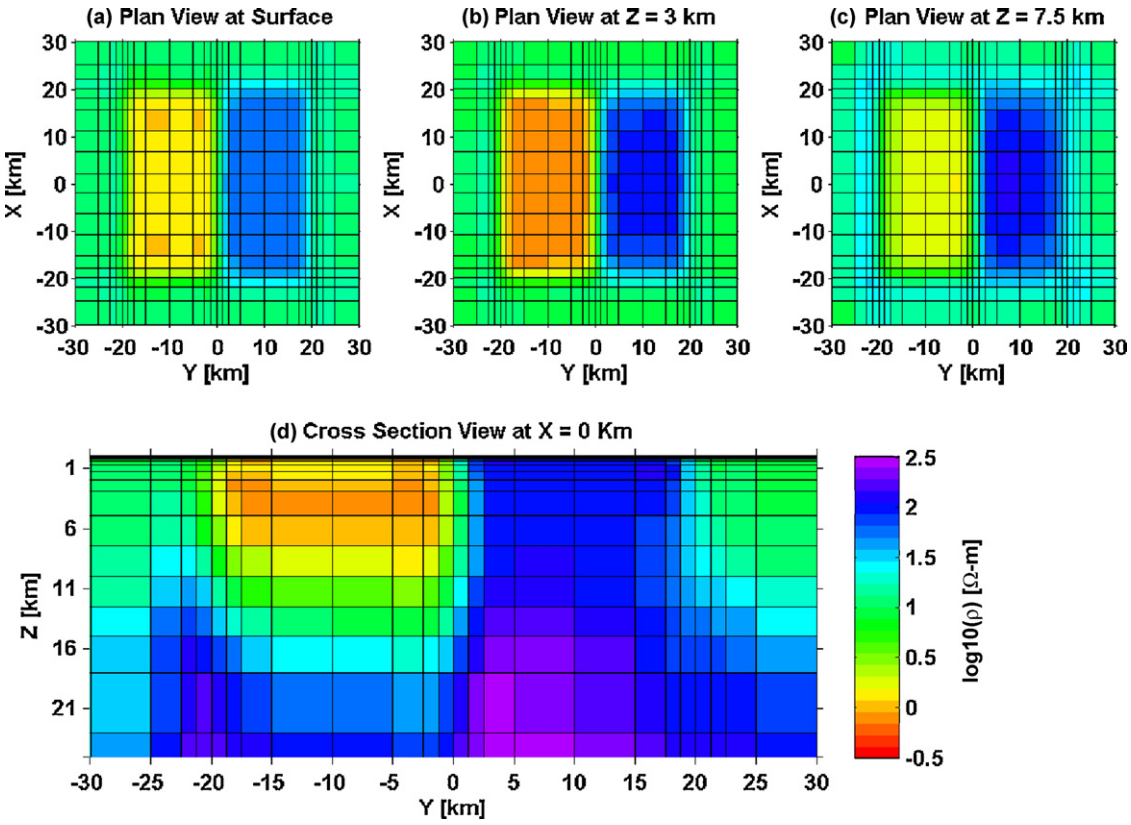


Fig. 7. Results from joint inversion of both VTF and impedance tensor data generated from the model in Fig. 3a. See caption of Fig. 4 for other details.