วิทยานิพนธ์

การประมาณค่าเมตริกซ์ความแปรปรวนร่วมสำหรับสมการลาปลาสที่ค่าขอบมีความสัมพันธ์ร่วม

COVARIANCE MATRIX INTERPOLATION FOR LAPLACE'S EQUATION WITH CORRELATED BOUNDARY VALUE

นายณัฐพงศ์ มูลผลึก

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

พ.ศ. 2550

ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมอุตสาหการ)

ปริญญา

วิศวกรรมอุตสาหการ	วิศวกรรมอุตสาหการ
สาขา	**ภาควิชา**

เรื่อง การประมาณค่าเมตริกซ์ความแปรปรวนร่วมสำหรับสมการลาปลาสที่ค่าขอบมีความสัมพันธ์ร่วม

Covariance Matrix Interpolation for Laplace's Equation with Correlated Boundary Value

นามผู้วิจัย นายณัฐพงศ์ มูลผลึก

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก .....

(รองศาสตราจารย์พีรยุทธ์ ชาญเศรษฐิกุล, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม .....

(ผู้ช่วยศาสตราจารย์จุฑา พิชิตลำเค็ญ, Ph.D.)

หัวหน้าภาควิชา .....

(รองศาสตราจารย์อนันต์ มุ่งวัฒนา, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์วินัย อาจคงหาญ, M.A.)

คณบดีบัณฑิตวิทยาลัย

วันที่ ..... **เดือน** ..... **พ.ศ.** .....

วิทยานิพนธ์

เรื่อง

การประมาณค่าเมตริกซ์ความแปรปรวนร่วมสำหรับสมการลาปลาสที่ค่าขอบมีความสัมพันธ์ร่วม

Covariance Matrix Interpolation for Laplace's Equation with Correlated Boundary Value

โดย

นายณัฐพงศ์ มูลผลึก

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมอุตสาหการ)
พ.ศ. 2550

ณัฐพงศ์ มูลผลึก 2550: การประมาณค่าเมตริกซ์ความแปรปรวนร่วมสำหรับสมการลาปลาสที่ค่าขอบมีความสัมพันธ์ร่วม ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมอุตสาหการ) สาขาวิศวกรรมอุตสาหการ ภาควิชาวิศวกรรมอุตสาหการ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: รองศาสตราจารย์พีรยุทธ์ ชาญเศรษฐิกุล, Ph.D. 158 หน้า

ในงานวิจัยนี้วิธีเชิงตัวเลขสำหรับประมาณค่าเมตริกซ์ความแปรปรวนร่วมของผลเฉลยสำหรับสมการลาปลาสภายใต้เงื่อนไขค่าขอบที่เป็นค่าสุ่มและมีความสัมพันธ์ร่วมได้ถูกนำมาศึกษา ทั้งนี้การประมาณค่าเมตริกซ์ความแปรปรวนร่วมพัฒนาจากระเบียบวิธีผลต่างสืบเนื่องประยุกต์ใช้แก้ปัญหาเชิงเลขของสมการอนุพันธ์ย่อยอันดับที่เกี่ยวข้อง โดยแบบจำลองเชิงตัวเลขถูกคำนวณบนโปรแกรม MATLAB และถูกทดสอบบนชุดของตัวอย่างปัญหาทั้งในสองมิติ และสามมิติ ในกรณีที่ตัวอย่างปัญหามีโดเมนขนาดใหญ่ระเบียบวิธีแบ่งส่วนโดเมนได้ถูกนำมาประยุกต์ใช้ ในงานวิจัยนี้แบบจำลองเชิงตัวเลขให้ผลการทดสอบที่มีประสิทธิภาพ โดยสามารถประมาณค่าเมตริกซ์ความแปรปรวนร่วมให้ค่าเปอร์เซ็นต์ความคลาดเคลื่อนสูงสุดจากผลเฉลยแม่นตรงที่ 0.162017 นอกจากนี้คอมพิวเตอร์เครื่องเดียวสามารถประมวลผลแบบจำลองได้เฉพาะในตัวอย่างปัญหาที่มีโดเมนขนาดเล็กโดยเฉพาะในปัญหา 3 มิติ ซึ่งตัวอย่างปัญหาที่ใหญ่ที่สุดที่สามารถคำนวณได้มีจำนวนจุดต่อขนาด $8 \times 8 \times 8$ ยิ่งไปกว่านั้นเมื่อประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมนเข้ากับแบบจำลองคอมพิวเตอร์ยิ่งต้องใช้เวลาในการประมวลผลเพิ่มขึ้นจากเดิม ในขณะเดียวกันรอบทำซ้ำในการคำนวณของแบบจำลองกลับลดลง

_____ _____ ___ / ___ / ___

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Nattapong Moonpluck 2007: Covariance Matrix Interpolation for Laplace's Equation with Correlated Boundary Value. Master of Engineering (Industrial Engineering), Major Field: Industrial Engineering, Department of Industrial Engineering.
Thesis Advisor: Associate Professor Peerayuth Charnsethikul, Ph.D. 158 pages.

In this work, a numerical method for interpolating the covariance matrix of solution for Laplace's equation under random correlated boundary conditions is proposed, illustrated and implemented. Computationally, the proposed approximation method is programmed in Matlab and tested on two and three dimensional problems. For larger scale problems, a domain decomposition approach is suggested. In the result, the numerical model determined approximate solution of covariance matrix effectively. The maximun absolute percent error with respect to an exact solution generated by the direct Fourier method is 0.162017. Besides, a single PC can handle the numerical model only on small domain sizes especially on three-dimensional problem, the maximum number of variables in the related linear equation system is 10^6 variables. Furthermore, a domain decomposition method applied to the model results in an increasing time consuming of a single computing facility. The time consuming increased, whereas the iteration used in the calculation of the model decreased.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

ขอกราบขอบพระคุณ รศ.ดร. พีรยุทธ์ ชาญเศรษฐิกุล ประธานที่ปรึกษาวิทยานิพนธ์ที่ให้โอกาสในการทำงานวิจัย ให้คำปรึกษาในการแก้ปัญหาของวิทยานิพนธ์ และให้ความเอาใจใส่เป็นอย่างดีทั้งที่ท่านก็มีนิสิตในความดูแลเกินโควตาอยู่แล้ว อีกทั้งขอกราบขอบพระคุณ รศ.ดร. อนันต์ มุ่งวัฒนา ประธานกรรมการสอบ ผศ.ดร. จุฑา พิชิตลำเค็ญ กรรมการสอบ และ อ.ดร. วิชัย รุ่งเรืองอนันต์ ผู้ทรงคุณวุฒิภายนอกที่ให้คำแนะนำ ปรับแก้จนวิทยานิพนธ์เสร็จสมบูรณ์

นอกจากนี้ขอกราบขอบพระคุณบิดา และมารดาข้าพเจ้าที่คอยให้การสนับสนุน ให้กำลังใจ และคอยห่วงใยในทุกเรื่อง ทั้งที่ท่านก็ไม่ได้เข้าใจในโปรแกรมคอมพิวเตอร์และคณิตศาสตร์ที่ข้าพเจ้าทำ อีกทั้งขอบคุณเพื่อนๆ ที่ให้คำแนะนำเกี่ยวกับวิทยานิพนธ์

ณัฐพงศ์ มูลผลึก

กันยายน 2550

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

คำอธิบายสัญลักษณ์และคำย่อ

ω	=	ตัวประกอบน้ำหนัก (weighting factor)
ε	=	ค่าความผิดพลาดที่ยอมได้ (stopping tolerance)
Δ	=	ช่วงของความยาวที่ถูกแบ่งย่อย (step size)
DDM	=	ระเบียบวิธีแบ่งส่วนโดเมน (domain decomposition method)
FDM	=	ระเบียบวิธีผลต่างสืบเนื่อง (finite difference method)
SOR	=	ระเบียบการทำซ้ำแบบผ่อนปรนเกินสืบเนื่อง (successive over – relaxation method)

การประมาณค่าเมตริกซ์ความแปรปรวนร่วมสำหรับสมการลาปลาสที่ค่าขอบมีความสัมพันธ์ร่วม

Covariance Matrix Interpolation for Laplace's Equation with Correlated Boundary Value

คำนำ

สมการลาปลาสได้ถูกนำมาประยุกต์ใช้ในอุตสาหกรรมอย่างแพร่หลาย โดยถูกนำมาอธิบายปรากฏการณ์ทางวิศวกรรม เช่น กระบวนการแพร่ของพลังงานเชิงกลและไฟฟ้า การนำความร้อนของโลหะหรือผลิตภัณฑ์ในสภาวะคงตัว โดยในกรณีนี้สมการลาปลาสถูกนำมาใช้ในการพยากรณ์ค่าอุณหภูมิภายในของชิ้นผลิตภัณฑ์เมื่อทราบอุณหภูมิที่บริเวณผิวของชิ้นผลิตภัณฑ์ ซึ่งช่วยลดการตรวจสอบชิ้นผลิตภัณฑ์แบบทำลาย เป็นต้น อย่างไรก็ตามกระบวนการทางอุตสาหกรรมได้ก่อให้เกิดความคลาดเคลื่อน(ความแปรปรวน)ในการผลิต นั่นคือผลิตภัณฑ์ที่ถูกผลิตขึ้นไม่สามารถควบคุมให้มีค่าอุณหภูมิเท่ากันได้ในทุกชิ้นผลิตภัณฑ์ ดังนั้นการแก้สมการลาปลาสเพื่อหาค่าอุณหภูมิภายในชิ้นผลิตภัณฑ์จึงต้องทำภายใต้เงื่อนไขค่าขอบที่มีความคลาดเคลื่อนหรือไม่แน่นอน ซึ่งนำไปสู่การแก้สมการเชิงอนุพันธ์ย่อยอันดับสี่เพื่อหาความแปรปรวนและความแปรปรวนร่วมระหว่างตัวแปรภายในของรูปร่างขอบเขตปัญหาภายใต้เงื่อนไขค่าขอบที่ไม่แน่นอนข้างต้น

โดยแท้จริงแล้วการแก้สมการลาปลาสสามารถทำได้โดยง่ายภายใต้เงื่อนไขค่าขอบที่เป็นฟังก์ชันเฉพาะเจาะจงอันนำไปสู่การหารูปแบบคำตอบปิด(Closed form solution) แต่ในกรณีที่เงื่อนไขค่าขอบมีความซับซ้อนการแก้สมการลาปลาสเพื่อหารูปแบบคำตอบปิดก็ไม่สามารถทำได้เช่นกัน โดยเฉพาะในทางปฏิบัติที่มีความไม่แน่นอนในเงื่อนไขค่าขอบมาเกี่ยวข้อง ด้วยเหตุนี้ระเบียบวิธีการจำลองทางตัวเลขจึงถูกนำมาประยุกต์ใช้เพื่อประมาณผลเฉลยของสมการลาปลาส โดยแบบจำลองเชิงตัวเลขได้ถูกทดสอบบนโปรแกรม MATLAB นอกจากนี้ในกรณีที่แบบจำลองเชิงตัวเลขมีขนาดโดเมนที่ใหญ่ คอมพิวเตอร์เครื่องเดียวไม่สามารถประมวลผลแบบจำลองดังกล่าวได้เนื่องจากข้อจำกัดของหน่วยความจำและความเร็วของหน่วยประมวลผลกลางของคอมพิวเตอร์

ดังนั้นระเบียบวิธีแบ่งส่วนโดเมนจึงถูกนำมาประยุกต์ใช้ โดยแบ่งโดเมนใหญ่ออกเป็นโดเมนย่อย จากนั้นแก้ระบบสมการของแต่ละโดเมนย่อยโดยอิสระแก่กัน

อย่างไรก็ตามเครื่องมือทาง Computer Aided Engineering (CAE) มีซอฟต์แวร์ที่มีจำหน่ายอยู่เช่น Solid Work, Finite Element Analysis (FEA), Computer Aided Three dimensional Interactive Application (CATIA) และ อื่นๆ มีโปรแกรมย่อยเพื่อแก้สมการลาปลาสแฝงอยู่ แต่จะใช้ได้เฉพาะการประเมินด้วยค่าเฉลี่ยเท่านั้น ถ้านำความไม่แน่นอนทางสถิติมาอธิบายเพิ่มเติมยังไม่ปรากฏโปรแกรมสำเร็จรูปที่ประเมินผลได้ชัดเจนปรากฏอยู่ ดังนั้นงานวิจัยนี้จึงถูกสร้างขึ้นเพื่อเป็นส่วนเติมเต็มในช่องว่างดังกล่าว

วัตถุประสงค์

1. เพื่อแก้ปัญหาเชิงเลขในการหาความแปรปรวนร่วมสำหรับสมการลาปลาสภายใต้เงื่อนไขค่าขอบที่เป็นค่าสุ่มและมีความสัมพันธ์ร่วม

2. เพื่อประยุกต์ใช้ระเบียบวิธีแบ่งส่วนในการจัดการกับแบบจำลองทางตัวเลขที่มีโดเมนขนาดใหญ่

ขอบเขตของงานวิจัย

1. รูปร่างขอบเขตปัญหาที่ใช้ในงานวิจัยนี้มีลักษณะเป็นแผ่นสี่เหลี่ยมจัตุรัส และกล่องสี่เหลี่ยมด้านเท่าในปัญหา 2 มิติ และ 3 มิติตามลำดับ

2. เงื่อนไขค่าขอบของปัญหาถูกสร้างขึ้นจากผลเฉลยแม่นตรง ทั้งนี้เพื่อนำผลเฉลยโดยประมาณของตัวแปรภายในที่ได้จากแบบจำลองมาเปรียบเทียบหาค่าความผิดพลาดกับผลเฉลยแม่นตรง

การตรวจเอกสาร

ในอดีตจนถึงปัจจุบันสมการลาปลาส(Chow , 2000) เป็นสมการเชิงอนุพันธ์ย่อยที่มีความสำคัญในการนำไปประยุกต์ใช้ทั้งในทางฟิสิกส์ และวิศวกรรม ทั้งนี้การแก้หาผลเฉลยของสมการลาปลาสในรูปอนุกรมนั้นต้องประยุกต์ใช้ระเบียบวิธีฟูเรียร์(Brown and Churchill , 2001) โดยการที่จะแก้สมการลาปลาสแล้วได้รับผลเฉลยในรูปของอนุกรมฟูเรียร์นั้น เงื่อนไขค่าขอบของปัญหาต้องสามารถแสดงในรูปของฟังก์ชันที่สามารถจัดผลเฉลยให้อยู่ในรูปแบบของอนุกรมฟูเรียร์ได้ อย่างไรก็ตามในทางปฏิบัติการหาฟังก์ชันเงื่อนไขค่าขอบดังกล่าวสามารถทำได้ยาก อีกทั้งโดยส่วนใหญ่ข้อมูลดิบเชิงตัวเลขของเงื่อนไขค่าขอบยังสามารถเก็บและวัดได้โดยเครื่องมือทางอุตสาหกรรม ดังในกรณีที่มีการใช้เครื่องมือวัดกวาดผ่านผิวขอบของผลิตภัณฑ์เพื่อวัดค่าอุณหภูมิ (Crank and Nicolson, 1947) จากนั้นใช้ข้อมูลการกระจายตัวของอุณภูมิที่วัดได้มาเป็นเงื่อนไขค่าขอบในการแก้สมการลาปลาส โดยใช้วิธีเชิงตัวเลขแก้หาตัวแปรที่ต้องการ(อุณหภูมิที่กระจายตัวภายในพื้นที่หรืออาณาบริเวณจำกัด) นอกจากนี้ยังมีกรณีที่คล้ายกันดังแสดงใน Hayt (2006) ที่มีการแก้สมการหาการกระจายตัวของสนามแม่เหล็กไฟฟ้า ยิ่งไปกว่านั้นถ้ากระบวนการข้างต้นถูกทำซ้ำในเชิงอุตสาหกรรม ข้อมูลการกระจายตัวของอุณหภูมิที่ผิวขอบที่เกิดจากกระบวนการทำซ้ำในหลายๆ รอบซึ่งมารถเก็บและแสดงได้ในเชิงสถิติ ด้วยเหตุนี้หัวใจสำคัญของปัญหาจึงไม่ได้อยู่ที่การแก้สมการหาค่าเฉลี่ยของตัวแปรเพียงอย่างเดียว หากแต่การกระจายที่แสดงในรูปของความแปรปรวนและความแปรปรวนร่วมในหมู่ตัวแปรต้องถูกคำนวณหาด้วย ทั้งนี้เพื่อใช้ข้อมูลดังกล่าวให้เป็นประโยชน์ในการจัดมาตรฐานคุณภาพผลิตภัณฑ์(Karniadakis, 2002; Charnsethikul, 2004) นอกจากนี้การสร้างแบบจำลองและระเบียบวิธีที่เกี่ยวข้องกับเงื่อนไขค่าขอบแบบสุ่ม(Random boundary value) ได้เป็นหัวเรื่องสำคัญที่ได้รับความสนใจในหมู่นักวิจัยในทศวรรษที่ผ่านมา โดยงานวิจัยเริ่มแรกในเรื่องนี้เกิดจากแคลคูลัสในลักษณะของสมการเชิงอนุพันธ์แบบสโทแคสติก (Stochastic) (Kao, 1997) และนอกไปจากนี้มีงานวิจัยในเชิงทฤษฎีและเชิงตัวเลขที่สามารถพบได้ใน Xia (1986) และ Prigarin and Winkler (2002) ตามลำดับ

โดยทั่วไปเป็นที่ทราบกันดีว่าแบบจำลองเชิงตัวเลขที่ใช้แก้สมการลาปลาสเพื่อหาผลเฉลยโดยประมาณนั้นมีขนาดใหญ่ โดยแบบจำลองดังกล่าวอยู่ในรูปของระบบสมการเชิงเส้นที่ประกอบด้วยเมตริกซ์มากเลขศูนย์(Sparse matrix) ซึ่งโดยปกติระบบสมการเหล่านี้สามารถแก้ได้เป็นผลสำเร็จโดยใช้ระเบียบวิธีการทำซ้ำ(Iterative methods) ดังแสดงใน Sadd (1996) อย่างไรก็ตาม

ในกรณีที่รูปร่างขอบเขตของปัญหาเป็นโดเมนในสามมิติ แบบจำลองเชิงตัวเลขจะกลายมาเป็นแบบจำลองขนาดใหญ่จนเกินความสามารถของคอมพิวเตอร์เครื่องเดียวจะรองรับได้ ด้วยเหตุนี้ระเบียบวิธีแบ่งส่วนโดเมน(Domain decomposition methods: DDM) จึงถูกนำมาประยุกต์ใช้เพื่อจัดการกับปัญหาข้างต้น โดยแบ่งส่วนโดเมนของรูปร่างขอบเขตปัญหาทั้งหมดออกเป็นโดเมนย่อยหลายโดเมน จากนั้นแก้ระบบสมการของแต่ละโดเมนย่อยโดยอิสระแก่กันโดยประมวลผลบนระบบคอมพิวเตอร์แบบขนาน ดังจะพบได้ใน Kamiya *et al*. (1996a; 1996b) และ Ehart *et al.* (2006) ในงานวิจัยนี้สมการที่สอดคล้องกับเงื่อนไขค่าขอบแบบสุ่มและมีความสัมพันธ์ร่วมได้ก่อให้เกิดแบบจำลองที่มีความซับซ้อน โดยแบบจำลองดังกล่าวประกอบด้วย 4 และ 6 โดเมนในกรณีปัญหาแบบ 2 และ 3 มิติตามลำดับ

ระเบียบวิธีประมาณผลเฉลยสมการลาปลาส

ความเข้าใจเบื้องต้นเกี่ยวกับวิธีเชิงตัวเลข(Numerical methods) ที่ใช้ในการประมาณผลเฉลยสมการลาปลาสดังจะแสดงต่อไปนี้สามารถพบได้ใน ปราโมทย์ (2544) โดยระเบียบวิธีแก้สมการเชิงอนุพันธ์แบบเอลลิปติก(Elliptic partial diferential equation)ที่อยู่ในรูปแบบสมการลาปลาสสามารถแสดงได้ดังต่อไปนี้

พิจารณาสมการลาปลาสดั้งเดิมใน 2 มิติ ดังนี้

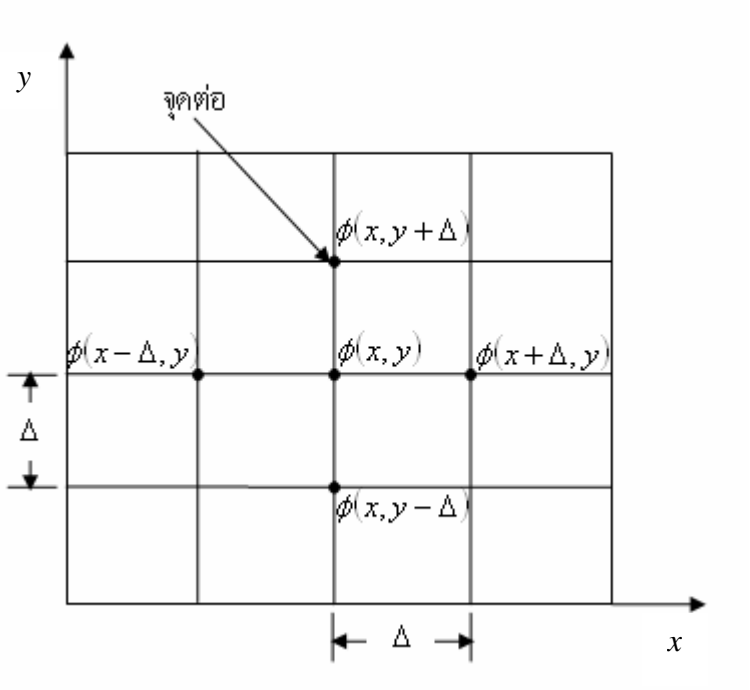
$$\frac{\partial^2 \phi(x,y)}{\partial x^2} + \frac{\partial^2 \phi(x,y)}{\partial y^2} = 0 \qquad (1)$$

โดยที่ $\phi(x,y)$ คือ ฟังก์ชันต่อเนื่องที่กระจายตัวภายในแผ่นสี่เหลี่ยมจัตุรัส ดังภาพที่ 1

จากนั้นแบ่งย่อยรูปร่างขอบเขตของปัญหาออกเป็นช่องๆ ที่มีขนาดเท่ากับ Δ ในทิศโคออร์ดิเนต x และ y ซึ่งจะได้ตารางที่ถูกเชื่อมด้วยจุดต่อ(Grid point)ดังแสดงในภาพที่ 1 อีกทั้งสมการเชิงอนุพันธ์ย่อยในสมการ (1) ต้องถูกแปลงให้อยู่ในรูปตัวไม่รู้ค่าที่จุดต่อ โดยในที่นี้ระเบียบวิธีผลต่างสืบเนื่อง (Finite difference method: FDM) ถูกนำมาใช้ในการหาผลเฉลยโดยประมาณของปัญหาข้างต้น โดยในขั้นแรกอนุพันธ์ย่อยอันดับสองในสมการ(1) ต้องถูกประมาณจากอนุกรมเทย์เลอร์อันดับ n ของ $f(x)$ ที่กระจายรอบจุด $x = a$ ดังนี้

$$f(x)=f(a)+f'(a)(x-a)+\frac{1}{2}f''(a)(x-a)^2+\ldots+\frac{1}{n!}f^{(n)}(a)(x-a)^n \qquad (2)$$

จากสมการ (2) จะเห็นได้ว่าฟังก์ชัน $f(x)$ สามารถหาค่าได้โดยที่รู้ค่าฟังก์ชันและอนุพันธ์ของฟังก์ชัน f ณ จุด a ในงานวิจัยนี้การประมาณค่าฟังก์ชันโดยอนุกรมเทย์เลอร์จะถูกใช้เพียง



ภาพที่ 1 การแบ่งย่อยรูปร่างขอบเขตปัญหาออกเป็นตารางสี่เหลี่ยมเพื่อใช้กับวิธีการผลต่างสืบเนื่อง

อนุพันธ์อันดับ 0 ถึง อันดับ 2 เท่านั้นเนื่องจาก อันดับ 3 จนถึงอันดับ n มีค่าน้อยมากจนสามารถตัดทิ้งได้ถ้า x และ a มีค่าใกล้กันมากๆ ($|x-a|<\varepsilon\to 0$) ในทำนองเดียวกันการประมาณค่าฟังก์ชัน f ณ จุด $x+\Delta$ และ $x-\Delta$ สามารถประมาณได้โดยที่รู้ค่าฟังก์ชันและอนุพันธ์ของฟังก์ชัน f ณ จุด x ซึ่งการประมาณโดยใช้อนุกรมเทย์เลอร์สามารถแสดงได้ดังนี้

$$f(x+\Delta)\cong f(x)+f'(x)(\Delta)+\frac{1}{2}f''(x)(\Delta)^2 \qquad (3)$$

$$f(x-\Delta)\cong f(x)-f'(x)(\Delta)+\frac{1}{2}f''(x)(\Delta)^2 \qquad (4)$$

นำสมการ (3) บวก (4) จะได้ค่าประมาณอนุพันธ์อันดับ 2 ณ จุด x ดังนี้

$$f''(x) \cong \frac{f(x+\Delta)+f(x-\Delta)-2f(x)}{(\Delta)^2} \tag{5}$$

จากสมการ (1) ค่าอนุพันธ์เชิงตัวเลขของฟังก์ชันค่าเฉลี่ย ณ โคออร์ดิเนต x และ y ใดๆ ในรูปร่างขอบเขตของปัญหาสามารถเขียนโดยใช้ความสัมพันธ์จากสมการ (5) ได้ดังนี้

$$\frac{\phi(x+\Delta, y)+\phi(x-\Delta, y)-2\phi(x, y)}{(\Delta)^2}+\frac{\phi(x, y+\Delta)+\phi(x, y-\Delta)-2\phi(x, y)}{(\Delta)^2} \cong 0 \tag{6}$$

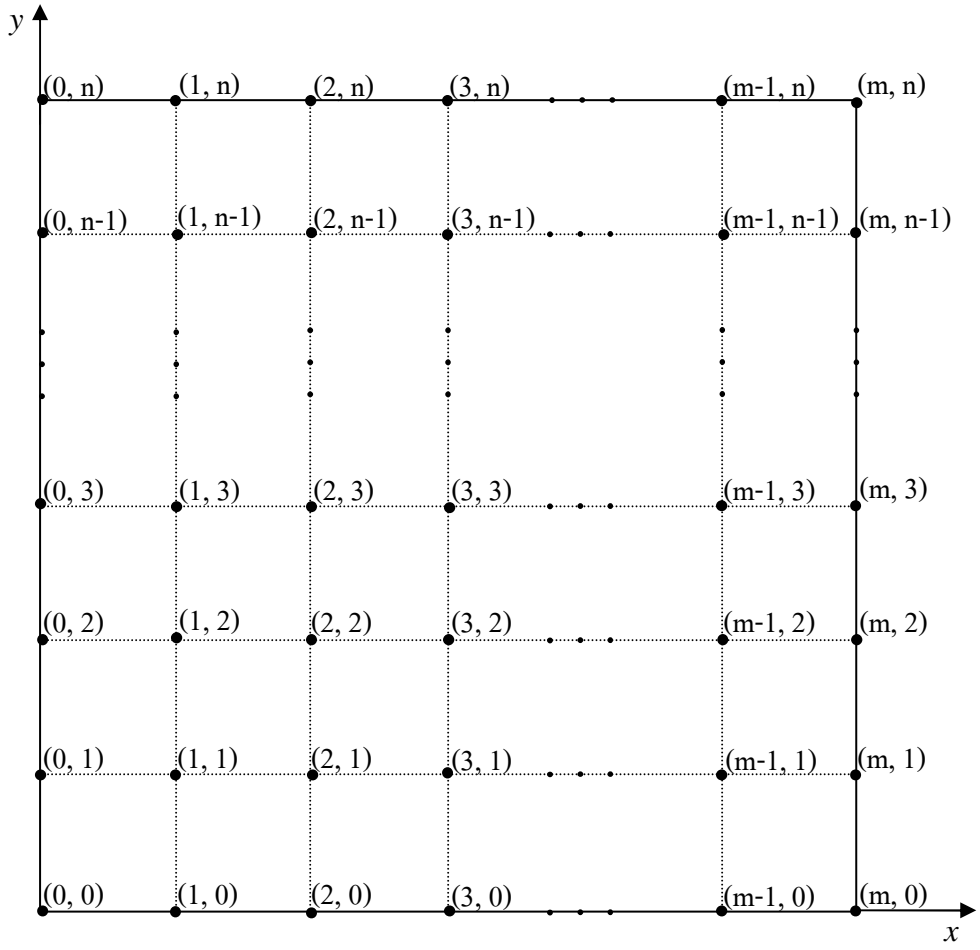
จัดรูปสมการ (6) ใหม่ ค่าอนุพันธ์เชิงตัวเลขของฟังก์ชันค่าเฉลี่ย ณ ตำแหน่ง x และ y ใดๆ สามารถแสดงได้ดังนี้

$$\phi(x, y) \cong \left(\frac{\phi(x+\Delta, y)+\phi(x-\Delta, y)+\phi(x, y+\Delta)+\phi(x, y-\Delta)}{4}\right) \tag{7}$$

จากฟังก์ชันค่าเฉลี่ยโดยประมาณในสมการ(7) เมื่อแทนค่าของฟังก์ชันดังกล่าวในโคออร์ดิเนต x และ y ใดๆ ลงในภาพที่ (1) ภายใต้เงื่อนไขค่าขอบที่กำหนดขึ้น ตัวแปรที่จุดต่อภายในของรูปร่างขอบเขตปัญหาสามารถคำนวณหาได้จากการแก้ระบบสมการเชิงเส้น โดยใช้ระเบียบวิธีการทำซ้ำคำนวณจนตัวแปรในโคออร์ดิเนต x และ y ณ จุดต่อภายในของรูปร่างเขตปัญหาลู่เข้าค่าที่แท้จริง

ทั้งนี้มีระเบียบวิธีจำนวนมากที่มีประสิทธิภาพในการแก้ระบบสมการเชิงเส้นดังกล่าว ยกตัวอย่างเช่น ระเบียบวิธีการกำจัดแบบเกาส์(Gauss elimination method) ระเบียบวิธีของเกาส์-ชอร์ดอง(Gauss-Jordan method) ระเบียบวิธีการทำเมตริกซ์ผกผัน(Matrix-Inverse) ระเบียบวิธีการแยกแบบแอลยู(LU decomposition method) ระเบียบวิธีการแยกแบบโชเลซกี ระเบียบการทำซ้ำแบบยาโคบี(Jacobi iteration method) ระเบียบวิธีการทำซ้ำแบบเกาส์-ไซเดล (Gauss-Seidel iteration method) และ ระเบียบการทำซ้ำแบบผ่อนปรนเกินสืบเนื่อง (Successive over – Relaxation method: SOR) เป็นต้น โดยในงานวิจัยนี้ระเบียบการทำซ้ำแบบ SOR ถูกเลือกนำมาใช้ ทั้งนี้เนื่องจากระเบียบวิธีการทำซ้ำดังกล่าวมีการกำหนดค่าตัวประกอบน้ำหนัก(Weighting factor)ให้กับ

ตัวแปรเพื่อให้ระบบสมการลู่เข้าสู่คำตอบไวขึ้น ซึ่งการทำซ้ำดังกล่าวเหมาะกับการแก้ระบบสมการขนาดใหญ่ อีกทั้งง่ายต่อการประดิษฐ์โปรแกรมนอกจากนี้การแก้ระบบสมการด้วยระเบียบวิธีการทำซ้ำนี้ยังช่วยลดความผิดพลาดที่เกิดจากการปัดเศษ นั่นคือคำนวณคำตอบของระบบสมการโดยการวนรอบทำซ้ำจนผลลัพธ์ลู่เข้าสู่คำตอบที่แท้จริงของระบบสมการ ซึ่งระเบียบวิธี SOR มีแนวคิดในการคำนวณดังต่อไปนี้ โดยในขั้นแรกแบ่งรูปร่างขอบเขตปัญหาออกเป็นช่องที่มีจำนวนจุดต่อ (Gird size) ขนาด m × n ดังภาพที่ 2



ภาพที่ 2 การแบ่งรูปร่างขอบเขตปัญหาออกเป็นช่องที่มีจำนวนจุดต่อ(Gird size) ขนาด m × n

จากนั้นแทนค่าอนุพันธ์เชิงตัวเลขของฟังก์ชันค่าเฉลี่ยในสมการ (7) ลงในจุดต่อภายในของรูปร่างขอบเขตปัญหาในภาพที่ 2 ภายใต้เงื่อนไขค่าขอบไดริคเร็ท (Dirichlet boundary condition) ซึ่งจะได้ระบบสมการสำหรับตัวแปรภายในรูปร่างขอบเขตปัญหา $\phi(i, j)$ ภายใต้เงื่อนไขค่าขอบทั้ง

4 ด้านคือ $b(0, j)$, $b(m, j)$, $b(i,0)$ และ $b(i,n)$ โดย i และ j คือตำแหน่งในทิศโคออร์ดิเนต x และ y ในภาพที่ 2 อีกทั้ง $i = 1, 2, 3, ..., m$ และ $j = 1, 2, 3, .., n$

$$\phi(1,1) = \frac{(\phi(2,1) + b(0,1) + \phi(1,2) + b(1,0))}{4}$$

$$\phi(2,1) = \frac{(\phi(3,1) + \phi(1,1) + \phi(2,2) + b(2,0))}{4}$$

.

.

$$\phi(m-1,1) = \frac{(b(m,1) + \phi(m-2,1) + \phi(m-1,2) + b(m-1,0))}{4}$$

$$\phi(1,2) = \frac{(\phi(2,2) + b(0,2) + \phi(1,3) + \phi(1,1))}{4}$$

$$\phi(2,2) = \frac{(\phi(3,2) + \phi(1,2) + \phi(2,3) + \phi(2,1))}{4}$$

.

.

$$\phi(m-1,2) = \frac{(b(m,2) + \phi(m-2,2) + \phi(m-1,3) + \phi(m-1,1))}{4}$$

$$\phi(1,n-1) = \frac{(\phi(2,n-1) + b(0,n-1) + b(1,n) + \phi(1,n-2))}{4}$$

$$\phi(2,n-1) = \frac{(\phi(3,n-1) + \phi(1,n-1) + b(2,n) + \phi(2,n-2))}{4}$$

.

.

$$\phi(m-1,n-1) = \frac{(b(m,n-1) + \phi(m-2,n-1) + b(m-1,n) + \phi(m-1,n-2))}{4} \qquad (8)$$

โดยแท้จริงแล้วระเบียบวิธีการทำซ้ำแบบ SOR ได้ถูกดัดแปลงจากระเบียบวิธีการทำซ้ำแบบเกาส์-ไซเดลเพื่อให้เกิดอัตราการลู่เข้าสู่ผลลัพธ์ที่แท้จริงสูงขึ้น โดยการถ่วงน้ำหนักของค่าที่คำนวณได้จากการทำซ้ำครั้งใหม่และเก่า ดังนั้นเพื่อง่ายแก่การทำความเข้าใจระบบสมการ (8) จึงถูกคำนวณด้วยระเบียบวิธีการทำซ้ำแบบเกาส์-ไซเดลดังนี้ โดยในขั้นแรกเงื่อนไขเริ่มต้นสำหรับตัวแปรภายใน (ตัวแปรด้านซ้ายมือของสมการ (8)) ต้องถูกกำหนดขึ้นก่อน จากนั้นในรอบทำซ้ำแรกตัวแปร $\phi(1,1)$ จะถูกคำนวณจากตัวแปรที่ทราบค่า (เงื่อนไขเริ่มต้น และเงื่อนไขค่าขอบ) ด้านขวามือของสมการ และในรอบทำซ้ำเดียวกันนี้ตัวแปร $\phi(1,1)$ (ถูกคำนวณแล้ว) จะถูกส่งค่าต่อไปเพื่อใช้ในการคำนวณตัวแปร $\phi(2,1)$ ซึ่งขั้นตอนดังกล่าวจะถูกทำกับตัวแปรทุกตัวจนถึงตัวแปร $\phi(m-1,n-1)$ จึงจะสิ้นสุดการคำนวณในรอบทำซ้ำแรก จากนั้นขั้นตอนข้างต้นจะถูกคำนวณซ้ำในรอบทำซ้ำต่อไปจนกว่าตัวแปรภายใน(ตัวแปรด้านซ้ายมือสมการ)จะลู่เข้าสู่ผลลัพธ์ที่แท้จริงของระบบสมการ อย่างไรก็ตามระเบียบวิธีการทำซ้ำแบบ SOR แตกต่างจากระเบียบวิธีการทำซ้ำแบบเกาส์-ไซเดลตรงที่มีการกำหนดตัวประกอบน้ำหนักให้กับตัวแปรในรอบทำซ้ำครั้งเก่าและใหม่ นั่นคือ

$$\phi^{k+1}(i,j) = \omega\phi^{k+1}(i,j) + (1-\omega)\phi^{k}(i,j) \qquad (9)$$

โดยตัวยก k แทนรอบทำซ้ำที่ k ค่า $\phi^{k+1}(i,j)$ เป็นค่าที่คำนวณได้จากการทำซ้ำครั้งใหม่ด้วยระเบียบวิธีแบบเกาส์-ไซเดล และ ω คือ ตัวประกอบน้ำหนัก (Weighting factor) ซึ่งมีค่าระหว่าง 0 ถึง 2 หาก $\omega = 1$ ระเบียบวิธีนี้จะกลายเป็นระเบียบวิธีแบบเกาส์-ไซเดล หาก ω มีค่าระหว่าง 1 ถึง 2 แล้ว สมการ (9) ให้ความหมายว่าจะเพิ่มน้ำหนักลงบนผลลัพธ์ใหม่ที่คำนวณมากขึ้นเนื่องจากผลลัพธ์ใหม่กำลังลู่เข้าสู่ผลลัพธ์ที่ถูกต้องเพื่อให้การลู่เข้าเป็นไปได้เร็วยิ่งขึ้น กรณีเช่นนี้บางครั้งจึงถูกเรียกว่าเป็นการผ่อนเกิน (Over-relaxation) ในทางตรงข้ามหาก ω มีค่าระหว่าง 0 ถึง 1 ที่เรียกว่าเป็นการผ่อนปรนต่ำ (Under-relaxation) สมการ(9) จะหมายถึงการช่วยแก้ระบบสมการในขณะที่ผลลัพธ์กำลังเกิดการลู่ออกให้ลู่กลับเข้าสู่ผลลัพธ์ที่แท้จริง สำหรับในทางปฏิบัติ ค่า ω ที่ใช้โดยปกติมีค่าระหว่าง 1 ถึง 2 เพื่อให้เกิดการลู่เข้าสู่ผลลัพธ์ที่แท้จริงเร็วขึ้น อย่างไรก็ตามค่า ω ที่ดีที่สุดจะแตกต่างไปตามลักษณะของปัญหา โดย $\omega = 1.2$ มักจะถูกนำไปใช้ในการแก้หาผลเฉลยโดยประมาณของสมการลาปลาส(ปราโมทย์, 2544)

ในงานวิจัยนี้ผลลัพธ์ที่แท้จริงของระบบสมการจะลู่เข้าเมื่อเปอร์เซ็นต์ผลต่างสัมบูรณ์ของผลลัพธ์ในรอบทำซ้ำเก่าและใหม่มีค่าน้อยกว่าค่าความผิดพลาดที่ยอมได้(Stopping tolerance) นั่นคือ

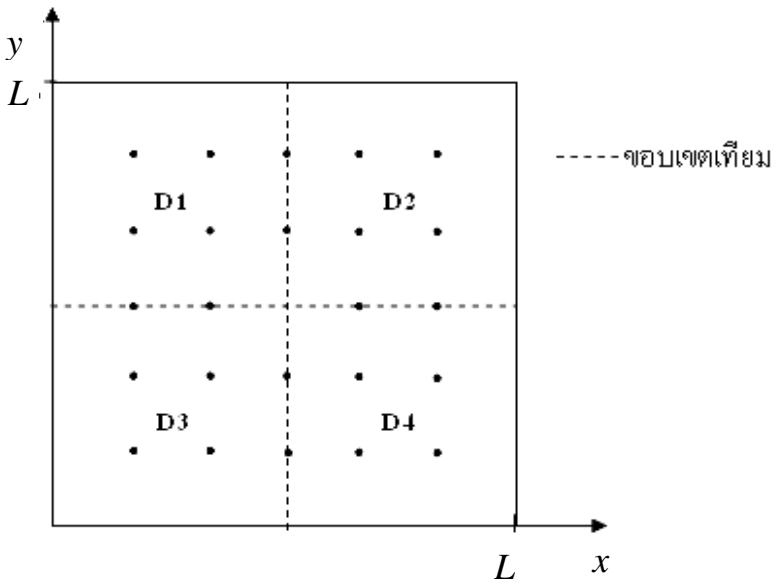
$$\left|\frac{\phi^{k+1}(i,j)-\phi^{k}(i,j)}{\phi^{k+1}(i,j)}\right|\times 100\% < \varepsilon$$

โดยตัวยก k คือ รอบทำซ้ำ ค่า $\phi^{k+1}(i,j)$ แสดงถึงค่าของตัวแปร ณ จุดต่อตำแหน่ง i และ j ในทิศโคออร์ดิเนต x และ y ในรอบทำซ้ำที่ $k+1$ และ ε คือค่าความผิดพลาดที่ยอมได้ นอกจากนี้ในกรณีที่แบบจำลองทางตัวเลขมีขนาดโดเมนที่ใหญ่ คอมพิวเตอร์เครื่องเดียวมีหน่วยความจำไม่เพียงพอที่จะประมวลผลแบบจำลองดังกล่าว ดังนั้นระเบียบวิธีแบ่งส่วนโดเมนจึงถูกนำมาใช้ในการแก้ปัญหาข้างต้น

ระเบียบวิธีแบ่งส่วนโดเมน

โดยทั่วไปปัญหาที่มีโดเมนขนาดใหญ่หรือปัญหาที่รูปร่างขอบเขตปัญหามีความซับซ้อน ระเบียบวิธีแบ่งส่วนโดเมนถูกนำมาใช้ในการแก้ปัญหาดังกล่าว(Ehart *et al.*, 2006) โดยเฉพาะกรณีโดเมนมีขนาดใหญ่จำนวนตัวแปรในระบบสมการเชิงเส้นของแบบจำลองเชิงตัวเลขมีจำนวนมาก ซึ่งการแก้ระบบสมการดังกล่าวอาจแก้ไม่ออกเนื่องจากขีดความสามารถของคอมพิวเตอร์เครื่องเดียวไม่เพียงพอ ด้วยเหตุนี้ระเบียบวิธีแบ่งส่วนโดเมนจึงถูกนำมาใช้ในการแก้ปัญหาข้างต้น โดยแบ่งรูปร่างขอบเขตปัญหาใหญ่ออกเป็นโดเมนย่อยหลายโดเมน จากนั้นแก้ระบบสมการของแต่ละโดเมนย่อยโดยอิสระแก่กัน โดยทั่วไประเบียบวิธีแบ่งส่วนโดเมนเป็นวิธีที่มีประสิทธิภาพมากเมื่อถูกนำไปประมวลผลบนระบบคอมพิวเตอร์แบบขนาน(Parallel computer system) โดยระเบียบวิธีแบ่งส่วนโดเมนมีแนวคิดคือ แบ่งส่วนรูปร่างขอบเขตของปัญหาใหญ่หรือโดเมนใหญ่ให้กลายเป็นโดเมนย่อยหลายๆ โดเมน โดยแต่ละโดเมนย่อยจะถูกแยกออกจากกันโดยขอบเขตเทียม(Artificial boundary) ดังการแบ่งส่วนโดเมนที่แสดงในภาพที่ 3 ซึ่งโดเมนใหญ่ถูกแยกออกเป็นโดเมนย่อย 4 โดเมน และเนื่องจากคอมพิวเตอร์เครื่องเดียวมีหน่วยความจำไม่เพียงพอที่จะแก้ระบบสมการของโดเมนใหญ่ ดังนั้นระบบสมการของแต่ละโดเมนย่อยจึงถูกคำนวณแยกโดยอิสระแก่กัน

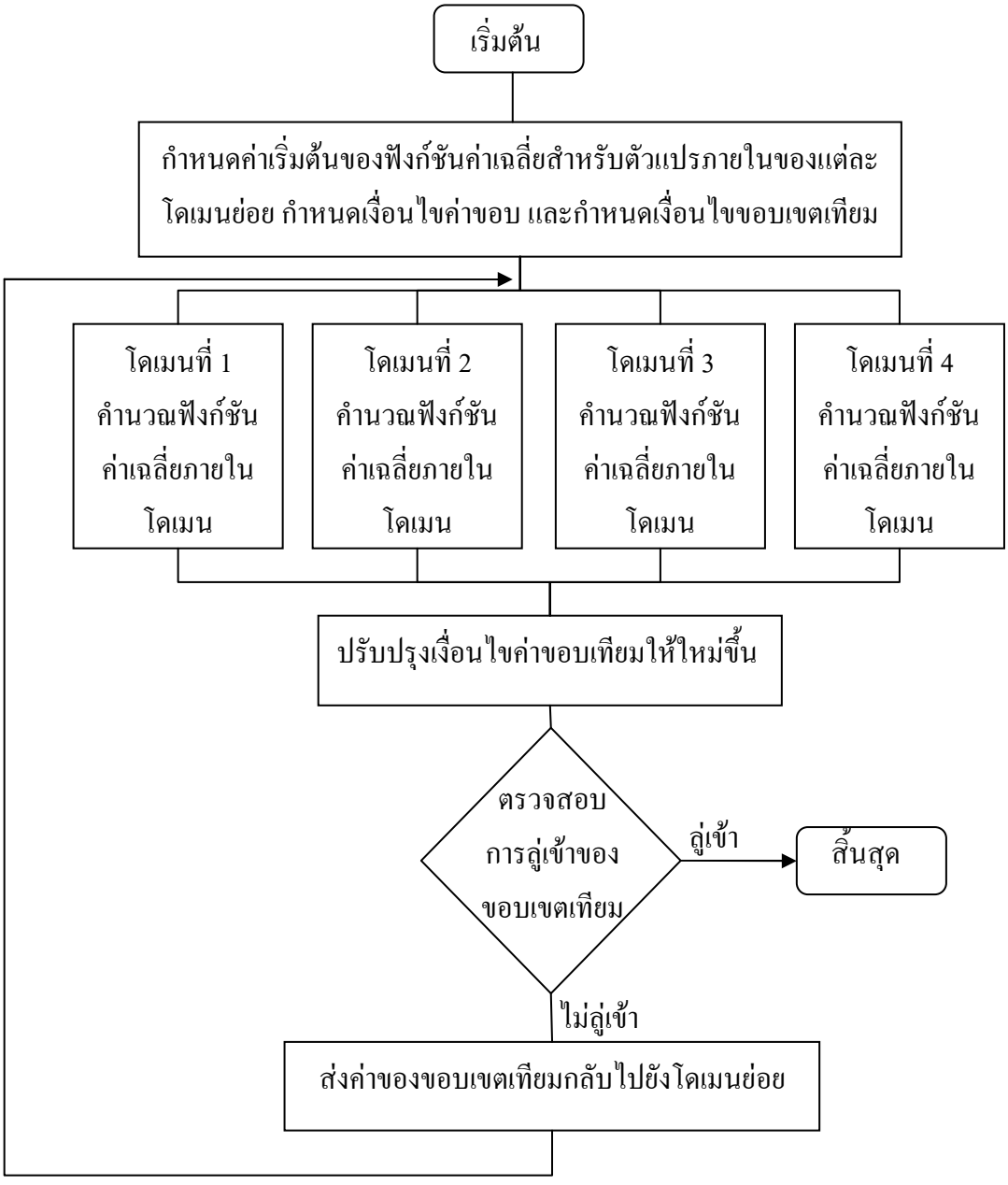
การประมาณฟังก์ชันค่าเฉลี่ยสำหรับสมการลาปลาสใน 2 มิติสามารถทำได้ดังขั้นตอนการคำนวณในภาพที่ 4 โดยในขั้นแรกกำหนดเงื่อนไขค่าขอบ กำหนดค่าเริ่มต้นของตัวแปรที่จุดต่อภายในของโดเมนย่อย และกำหนดค่าเริ่มต้นของตัวแปรในขอบเขตเทียม จากนั้นคำนวณค่าของตัวแปรค่าเฉลี่ยที่จุดต่อภายในของโดเมนย่อยโดยอิสระแก่กัน โดยใช้ความสัมพันธ์ในสมการ (7) ภายใต้เงื่อนไขค่าขอบและเงื่อนไขเริ่มต้นที่กำหนดไว้ข้างต้น หลังจากคำนวณค่าของตัวแปรที่จุดต่อภายในของโดเมนย่อยแล้ว ขั้นตอนต่อไปคือ ปรับปรุงเงื่อนไขค่าขอบเขตเทียมให้ใหม่ขึ้น โดยคำนวณจากทั้งตัวแปรที่จุดต่อที่อยู่ชิดกับขอบเขตเทียม และตัวแปรที่จุดต่อในขอบเขต



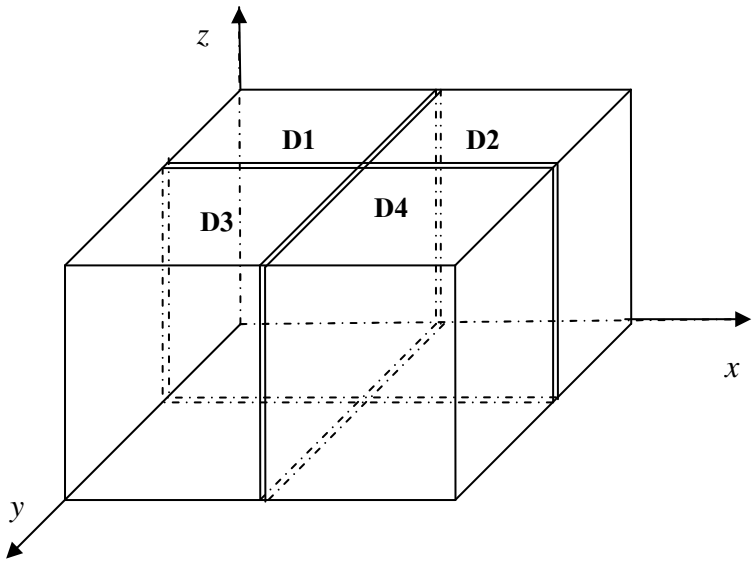
ภาพที่ 3 การแบ่งส่วนโดเมนในปัญหา 2 มิติ

เทียมด้วยกันเองดังความสัมพันธ์ในสมการ (7) ท้ายที่สุดตัวแปรในขอบเขตเทียมที่ถูกปรับปรุงแล้วต้องถูกนำมาตรวจสอบการลู่เข้า โดยพิจารณาจากเปอร์เซ็นต์ผลต่างสัมบูรณ์ของผลลัพธ์ในรอบทำซ้ำเก่าและใหม่ ถ้าเปอร์เซ็นต์ผลต่างสัมบูรณ์ของตัวแปรในขอบเขตเทียมทุกตัวมีค่าน้อยกว่าค่าความผิดพลาดที่ยอมได้ นั่นหมายความว่าระบบสมการในแต่ละโดเมนย่อยได้ลู่เข้าสู่ผลลัพธ์ที่แท้จริง ในทางตรงข้ามหากเปอร์เซ็นต์ผลต่างสัมบูรณ์ของตัวแปรในขอบเขตเทียมตัวใดตัวหนึ่งมีค่ามากกว่าค่าความผิดพลาดที่ยอมได้ นั่นคือระบบสมการของโดเมนย่อยที่ตัวแปรดังกล่าวเป็นสมาชิกอยู่ยังไม่ลู่เข้า ดังนั้นตัวแปรในขอบเขตเทียมข้างต้นต้องถูกส่งกลับไปเป็นเงื่อนไขค่าขอบของโดเมนย่อยอีกครั้ง และวนรอบทำซ้ำจนกว่าตัวแปรในขอบเขตเทียมทุกตัวมีค่าเปอร์เซ็นต์ผลต่างสัมบูรณ์ของผลลัพธ์ในรอบทำซ้ำเก่าและใหม่ มีค่าน้อยกว่าค่าความผิดพลาดที่ยอมได้

นอกจากนี้การประมาณฟังก์ชันค่าเฉลี่ยในปัญหา 3 มิติ สามารถทำได้ในทำนองเดียวกัน โดยแบ่งรูปร่างขอบเขตของปัญหาออกเป็น 4 โดเมนย่อยดังภาพที่ 5 โดยขั้นตอนการแก้ระบบสมการของแต่ละโดเมนย่อยในปัญหา 3 มิติ สามารถคำนวณได้ตามขั้นตอนดังภาพที่ 4 เช่นกัน

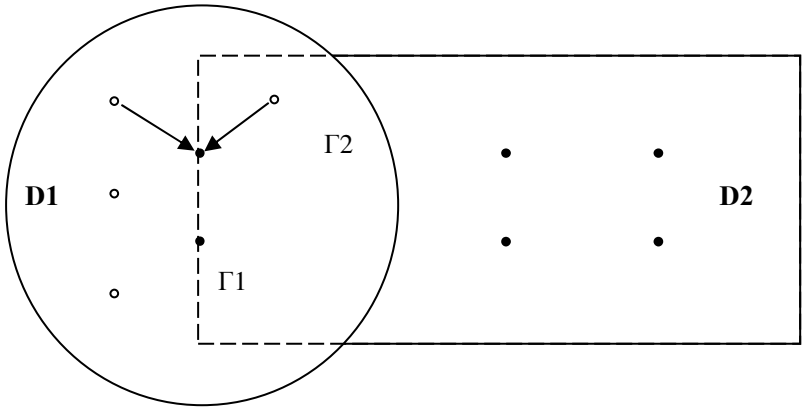


ภาพที่ 4 ขั้นตอนการคำนวณของระเบียบวิธีแบ่งส่วนโดเมนสำหรับฟังก์ชันค่าเฉลี่ย
ที่มา: Kamiya *et al*. (1996a)



ภาพที่ 5 การแบ่งส่วนโดเมนในปัญหา 3 มิติ

นอกจากนี้ในกรณีที่รูปร่างขอบเขตปัญหามีความซับซ้อน โดเมนใหญ่ต้องถูกแบ่งออกเป็นโดเมนย่อย และโดเมนย่อยดังกล่าวมีความจำเป็นต้องใช้ระเบียบวิธีหาผลเฉลยของคำตอบของสมการที่แตกต่างกัน เช่น โดเมนย่อยโดเมนหนึ่งมีความจำเป็นต้องใช้ระเบียบวิธีไฟไนต์เอลิเมนต์ (Finite element method) ในการแก้หาผลเฉลยของปัญหา และในขณะเดียวกันโดเมนย่อยอีกโดเมนต้องใช้ระเบียบวิธีผลต่างสืบเนื่อง เป็นต้น



ภาพที่ 6 โดเมนใหญ่ถูกแบ่งออกเป็นโดเมนย่อย D1 และ D2 ที่ซ้อนทับกันโดย $\Gamma 1$ และ $\Gamma 2$ คือ บริเวณที่โดเมนย่อยซ้อนทับกัน

ที่มา: Glimsdal *et al*. (2004)

จากเหตุผลข้างต้นระเบียบวิธีแบ่งส่วนโดเมนแบบโดเมนย่อยซ้อนทับกัน(Overlapping domain) ได้ถูกนำมาประยุกต์ใช้แก้ปัญหาดังกล่าวดังจะพบได้ใน Glimsdal *et al.* (2004) โดยแบ่งส่วนรูปร่างขอบเขตปัญหาออกเป็น 2 โดเมนย่อยที่ซ้อนทับกันดังภาพที่ 6 โดยจุดต่อที่แสดงด้วยวงกลมไม่แรเงาคือจุดต่อในโดเมนย่อย D1 ในขณะเดียวกันจุดต่อที่แสดงด้วยวงกลมแรเงาคือจุดต่อในโดเมนย่อย D2 และในกรณีที่โดเมนย่อยทั้งสองมีตำแหน่งของจุดต่อไม่ตรงกัน การปรับปรุงค่าตัวแปรบนขอบเขตเทียม(ตัวแปรบนเส้นประ)ให้ใหม่ขึ้นสามารถทำได้โดยประมาณจากจุดต่อข้างเคียงดังภาพที่ 6 โดยจุดต่อที่แสดงด้วยวงกลมแรงเงาบนเส้นประ(จุดต่อบนโดเมนย่อย D2 ที่ลูกศรชี้) ถูกประมาณจากจุดต่อข้างเคียงบนโดเมนย่อย D1(จุดต่อที่แสดงด้วยวงกลมไม่แรเงาที่ลูกศรชี้ออก)

ในทำนองเดียวกันเทคนิควิธี FDM ที่ใช้ในการประมาณฟังก์ชันค่าเฉลี่ยข้างต้นสามารถประยุกต์ใช้ในการประมาณฟังก์ชันความแปรปรวนร่วมได้เช่นกัน โดยวิธีการในการสร้างฟังก์ชันความแปรปรวนร่วมสามารถแสดงได้ดังในหัวข้อถัดไป

อุปกรณ์และวิธีการ

อุปกรณ์

1. คอมพิวเตอร์ส่วนบุคคล
   1.1 ระบบปฏิบัติการ Microsoft Windows XP
   1.2 หน่วยประมวลผลกลาง Intel Pentium 4 ความเร็ว 3.40 GHz
   1.3 หน่วยความจำ 512 MB
   1.4 ความจุของ Hard disk drive 80 GB

2. ซอฟต์แวร์
   2.1 โปรแกรม MATLAB 7.1

วิธีการ

เนื่องจากการผลิตในอุตสหกรรมมีความคลาดเคลื่อนมาเกี่ยวข้อง ทำให้การแก้สมการลาปลาสเพื่อหาตัวแปรภายในของรูปร่างขอบเขตปัญหาต้องทำภายใต้เงื่อนไงค่าขอบที่ไม่แน่นอน นั่นคือเงื่อนไขค่าขอบมีความแปรปรวนและความแปรปวนร่วมมาเกี่ยวข้อง ดังนั้นการแก้สมการลาปลาสเพื่อหาตัวแปรภายในของรูปร่างขอบเขตปัญหาจึงไม่ได้อยู่ที่การแก้หาค่าเฉลี่ยเพียงอย่างเดียว หากแต่ความแปรปรวนและความแปรปรวนร่วมของตัวแปรภายในของรูปร่างขอบเขตปัญหาจะต้องถูกคำนวณหาด้วย โดยเป็นที่ทราบกันดีว่าความแปรปรวนร่วมของตัวแปรตัวมันเอง ($\mathrm{cov}[\phi,\phi]$) ก็คือความแปรปรวน($V[\phi]$)

ในงานวิจัยนี้ฟังก์ชันความแปรปรวนร่วมถูกสร้างขึ้นด้วยระเบียบวิธีฟูเรียร์ ทั้งนี้ฟังก์ชันความแปรปรวนร่วมดังกล่าวจะถูกประมาณด้วยวิธีเชิงตัวเลข ภายใต้เงื่อนไขค่าขอบที่เป็นค่าสุ่มและมีความสัมพันธ์ร่วม(Random and correlated boundary condition) โดยประยุกต์จากความเข้าใจพื้นฐานของการประมาณฟังก์ชันค่าเฉลี่ย ด้วยเหตุนี้วิธีการในการคำนวณหาฟังก์ชันความแปรปรวนร่วมจึงถูกแสดงไว้ควบคู่กับการคำนวณหาฟังก์ชันค่าเฉลี่ย อีกทั้งแนวคิดของระเบียบวิธี

แบ่งส่วนโดเมนสำหรับประมาณฟังก์ชันความแปรปรวนร่วมได้ถูกแสดงไว้ในหัวข้อนี้ด้วย ทั้งนี้วิธีประมาณค่าฟังก์ชันความแปรปรวนร่วมและฟังก์ชันค่าเฉลี่ยทั้งในปัญหา 2 และ 3 มิติ สามารถแสดงได้ดังนี้

1. การประมาณค่าฟังก์ชันความแปรปรวนร่วมในปัญหา 2 มิติ

พิจารณาสมการลาปลาสดั้งเดิมใน 2 มิติในสมการ (1) ในรูปร่างขอบเขตปัญหาดังภาพที่ 7 ซึ่งผลเฉลยทั่วไปของฟังก์ชันค่าเฉลี่ยสำหรับสมการลาปลาสดั้งเดิมในสมการ (1) สามารถแสดงได้ดังนี้

$$\phi(x,y) = (c_1 e^{\sqrt{k}x} + c_2 e^{-\sqrt{k}x}) \cdot (c_3 \sin\sqrt{k}\,y + c_4 \cos\sqrt{k}\,y) \qquad (10)$$

โดย c_1 c_2 c_3 c_4 และ k คือ ค่าคงที่ ทั้งนี้ค่าคงที่เหล่านี้เป็นค่าคงที่เฉพาะที่สามารถคำนวณหาได้ภายใต้เงื่อนไขค่าขอบแบบคงที่(Deterministic boundary condition) ดังนี้

1. เงื่อนไขค่าขอบด้านบน $\phi(x,L),\ 0 \le x \le L$

2. เงื่อนไขค่าขอบด้านล่าง $\phi(x,0),\ 0 \le x \le L$

3. เงื่อนไขค่าขอบด้านซ้าย $\phi(0,y),\ 0 \le y \le L$

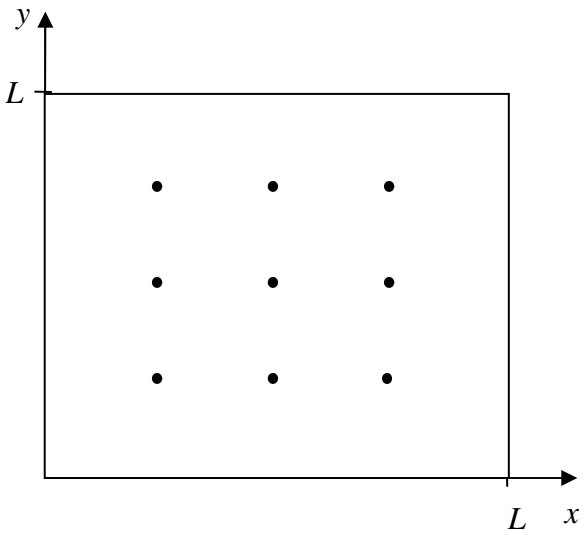
4. เงื่อนไขค่าขอบด้านขวา $\phi(L,y),\ 0 \le y \le L$

ในการคำนวณหาฟังก์ชันความแปรปรวนร่วมภายใต้เงื่อนไขค่าขอบที่เป็นค่าสุ่มและมีความสัมพันธ์ร่วม ในขั้นแรกพิจารณาสมการลาปลาสดั้งเดิมของสมการ(1) ใน 4 โดเมนดังแสดงในสมการ (11) และสมการ (12) ดังนี้

$$\frac{\partial^2 \phi_1(x_1, y_1)}{\partial x_1^2} + \frac{\partial^2 \phi_1(x_1, y_1)}{\partial y_1^2} = 0 \tag{11}$$

$$\frac{\partial^2 \phi_2(x_2, y_2)}{\partial x_2^2} + \frac{\partial^2 \phi_2(x_2, y_2)}{\partial y_2^2} = 0 \tag{12}$$

เพื่อความสะดวกในการคำนวณเขียน $\phi_1(x_1, y_1)$ ในรูปของ ϕ_1 และเขียน $\phi_2(x_2, y_2)$ ในรูปของ ϕ_2



ภาพที่ 7 ฟังก์ชันต่อเนื่อง $\phi(x, y)$ ณ จุดใดๆ กระจายตัวภายในแผ่นสี่เหลี่ยมจัตุรัส

จากนั้น นำสมการ(11) คูณกับสมการ (12) จะได้สมการเชิงอนุพันธ์ย่อยอันดับ 4 ดังแสดงดังต่อไปนี้

$$\frac{\partial^4 \phi_1 \phi_2}{\partial x_1^2 \partial x_2^2} + \frac{\partial^4 \phi_1 \phi_2}{\partial x_1^2 \partial y_2^2} + \frac{\partial^4 \phi_1 \phi_2}{\partial y_1^2 \partial x_2^2} + \frac{\partial^4 \phi_1 \phi_2}{\partial y_1^2 \partial y_2^2} = 0 \tag{13}$$

จากความสัมพันธ์ของค่าคาดหมาย $E[\phi_1 \cdot \phi_2] = \mathrm{cov}[\phi_1, \phi_2] + E[\phi_1] \cdot E[\phi_2]$ หลังจากประยุกต์ใช้ตัวดำเนินการค่าคาดหมายบนสองข้างของสมการ (13) รูปใหม่ของสมการ (13) สามารถเขียนให้อยู่ในรูปดังต่อไปนี้

$$\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial x_1^2 \partial x_2^2}+\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial x_1^2 \partial y_2^2}+\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial y_1^2 \partial x_2^2}+\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial y_1^2 \partial y_2^2}+$$

$$\left(\frac{\partial^2 E[\phi_1]}{\partial x_1^2}+\frac{\partial^2 E[\phi_1]}{\partial y_1^2}\right)\cdot\left(\frac{\partial^2 E[\phi_2]}{\partial x_2^2}+\frac{\partial^2 E[\phi_2]}{\partial y_2^2}\right)=0 \tag{14}$$

$$\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial x_1^2 \partial x_2^2}+\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial x_1^2 \partial y_2^2}+\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial y_1^2 \partial x_2^2}+\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial y_1^2 \partial y_2^2}=0 \tag{15}$$

ปัญหาค่าขอบเชิงเส้น(Linear boundary value problem) สามารถแก้ได้โดยใช้ระเบียบวิธีฟูเรียร์(Fourier method) โดยอาศัยหลักของซุปเปอร์โพซิชัน(Principle of superposition) ซึ่งผลเฉลยของปัญหาค่าขอบเชิงเส้นจะถูกเขียนให้อยู่ในรูปอนุกรม จากนั้นจัดรูปอนุกรมของผลเฉลยข้างต้นให้อยู่ในรูปของอนุกรมฟูเรียร์ แต่ในกรณีที่ปัญหามีความซับซ้อนการจัดรูปผลเฉลยให้อยู่ในรูปของอนุกรมฟูเรียร์ก็ไม่สามารถทำได้เช่นกัน ดังนั้นสมการ (15) สามารถแก้ได้โดยตรงโดยใช้ระเบียบวิธีฟูเรียร์ดังแสดงในสมการ (16) (17) (18) และ (19) โดยในขั้นแรกกำหนดให้

$$\operatorname{cov}[\phi_1,\phi_2]=f_1(x_1)g_1(y_1)f_2(x_2)g_2(y_2) \tag{16}$$

จากนั้น แทนสมการ(16) เข้าไปในสมการ (15) ซึ่งได้ความสัมพันธ์ดังนี้

$$f_1''(x_1)g_1(y_1)f_2''(x_2)g_2(y_2)+f_1''(x_1)g_1(y_1)f_2(x_2)g_2''(y_2)+$$

$$f_1(x_1)g_1''(y_1)f_2''(x_2)g_2(y_2)+f_1(x_1)g_1''(y_1)f_2(x_2)g_2''(y_2)=0 \tag{17}$$

นำสมการ(17) หารด้วย สมการ (16) และหลังจากจัดรูปพจน์ในสมการ รูปใหม่ของสมการสามารถแสดงได้ในรูปผลคูณดังนี้

$$\left(\frac{f_1''(x_1)}{f_1(x_1)}+\frac{g_1''(y_1)}{g_1(y_1)}\right)\cdot\left(\frac{f_2''(x_2)}{f_2(x_2)}+\frac{g_2''(y_2)}{g_2(y_2)}\right)=0 \tag{18}$$

ดังนั้น ผลเฉลยทั่วไปของฟังก์ชันความแปรปรวนร่วมสำหรับสมการด้านบนสามารถแสดงได้ในรูปแบบต่อไปนี้

$$\operatorname{cov}[\phi_1,\phi_2)] = \left(c_1 e^{\sqrt{k_2}x_1} + c_2 e^{-\sqrt{k_2}x_1}\right)\cdot\left(c_3 \sin\sqrt{k_2}\,y_1 + c_4 \cos\sqrt{k_2}\,y_1\right)\cdot$$

$$\left(c_5 e^{\sqrt{k_4}x_2} + c_6 e^{-\sqrt{k_4}x_2}\right)\cdot\left(c_7 \sin\sqrt{k_3}\,y_2 + c_8 \cos\sqrt{k_3}\,y_2\right) \qquad (19)$$

โดยที่ c_1 c_2 c_3 c_4 c_5 c_6 c_7 k_1 k_2 k_3 และ k_4 คือ ค่าคงที่ และ $k_4 = k_1 - k_3$ ในที่นี้ปัญหาหลักอยู่ที่การแก้หาค่าคงที่ในสมการ(19) ซึ่งพจน์ของค่าคงที่ดังกล่าวล้วนขึ้นอยู่กับเงื่อนไขค่าขอบที่ให้มา ทั้งนี้เป็นที่ทราบกันดีว่าไม่มีวิธีที่เฉพาะเจาะจงในการแก้หาค่าคงที่ข้างต้น ด้วยเหตุนี้วิธีเชิงตัวเลขจึงเป็นวิธีที่นิยมใช้ในการหาผลเฉลยโดยประมาณของปัญหาโดยเฉพาะในทางปฏิบัติ ในงานวิจัยนี้ระเบียบวิธี FDM ได้ถูกนำมาประยุกต์ใช้ในการหาผลเฉลยโดยประมาณของปัญหานี้ โดยในขั้นต้นประยุกต์ใช้อนุกรมเทย์เลอร์อันดับ 2 เพื่อหาค่าอนุพันธ์โดยประมาณของฟังก์ชันค่าเฉลี่ยสำหรับสมการลาปลาสดั้งเดิมในสมการ (1) ซึ่งรูปแบบที่รู้จักกันดีของสมการดังกล่าวสามารถแสดงได้ดังสมการ (7)

ในทำนองเดียวกันอนุพันธ์ย่อยอันดับ 4 ของฟังก์ชันความแปรปรวนร่วมในสมการ (15) สามารถประมาณได้ โดยใช้อนุพันธ์ย่อยอันดับ 2 ในสมการ (5) แทนเข้าไปในสมการ (15) ซึ่งหลังจากกระจายและรวมพจน์แล้ว รูปใหม่ของสมการ (15) สามารถแสดงได้ดังนี้

$$\operatorname{cov}[\phi_1(x_1,y_1),\phi_2(x_2,y_2)] \cong \{\operatorname{cov}[\phi_1(x_1+\Delta,y_1),\phi_2(x_2+\Delta,y_2)] +$$

$$\operatorname{cov}[\phi_1(x_1+\Delta,y_1),\phi_2(x_2-\Delta,y_2)] + \operatorname{cov}[\phi_1(x_1-\Delta,y_1),\phi_2(x_2-\Delta,y_2)] +$$

$$\operatorname{cov}[\phi_1(x_1+\Delta,y_1),\phi_2(x_2,y_2+\Delta)] + \operatorname{cov}[\phi_1(x_1-\Delta,y_1),\phi_2(x_2,y_2+\Delta)] +$$

$$\operatorname{cov}[\phi_1(x_1+\Delta,y_1),\phi_2(x_2,y_2-\Delta)] + \operatorname{cov}[\phi_1(x_1-\Delta,y_1),\phi_2(x_2,y_2-\Delta)] +$$

$$\operatorname{cov}[\phi_1(x_1,y_1+\Delta),\phi_2(x_2+\Delta,y_2)] + \operatorname{cov}[\phi_1(x_1,y_1-\Delta),\phi_2(x_2+\Delta,y_2)] +$$

$$\operatorname{cov}[\phi_1(x_1, y_1+\Delta), \phi_2(x_2-\Delta, y_2)] + \operatorname{cov}[\phi_1(x_1, y_1-\Delta), \phi_2(x_2-\Delta, y_2)] +$$

$$\operatorname{cov}[\phi_1(x_1, y_1+\Delta), \phi_2(x_2, y_2+\Delta)] + \operatorname{cov}[\phi_1(x_1, y_1-\Delta), \phi_2(x_2, y_2+\Delta)] +$$

$$\operatorname{cov}[\phi_1(x_1, y_1+\Delta), \phi_2(x_2, y_2-\Delta)] + \operatorname{cov}[\phi_1(x_1, y_1-\Delta), \phi_2(x_2, y_2-\Delta)] -$$

$$4\operatorname{cov}[\phi_1(x_1+\Delta, y_1), \phi_2(x_2, y_2)] - 4\operatorname{cov}[\phi_1(x_1-\Delta, y_1), \phi_2(x_2, y_2)] -$$

$$4\operatorname{cov}[\phi_1(x_1, y_1+\Delta), \phi_2(x_2, y_2)] - 4\operatorname{cov}[\phi_1(x_1, y_1-\Delta), \phi_2(x_2, y_2)] -$$

$$4\operatorname{cov}[\phi_1(x_1, y_1), \phi_2(x_2+\Delta, y_2)] - 4\operatorname{cov}[\phi_1(x_1, y_1), \phi_2(x_2-\Delta, y_2)] +$$

$$\operatorname{cov}[\phi_1(x_1-\Delta, y_1), \phi_2(x_2+\Delta, y_2)] - 4\operatorname{cov}[\phi_1(x_1, y_1), \phi_2(x_2, y_2+\Delta)] -$$

$$4\operatorname{cov}[\phi_1(x_1, y_1), \phi_2(x_2, y_2-\Delta)]\}/(-16) \qquad (20)$$

ค่าประมาณของฟังก์ชันความแปรปรวนร่วมระหว่างคู่จุด $\phi_1(x_1, y_1)$ และ $\phi_2(x_2, y_2)$ ในสมการ (20) เป็นการประมาณจากความแปรปรวนร่วมระหว่างคู่จุดทุกคู่รอบจุด $\phi_1(x_1, y_1)$ และ $\phi_2(x_2, y_2)$ ดังแสดงในภาพที่ 8

ทั้งนี้สมการ(19) และ สมการ (20) สามารถแก้ได้เมื่อเงื่อนไขค่าขอบที่เป็นค่าสุ่มและมีความสัมพันธ์ร่วมของคู่จุดที่ผิวขอบของรูปร่างขอบเขตปัญหาทั้ง 10 เงื่อนไขต่อไปนี้ถูกกำหนดขึ้น

1. เงื่อนไขค่าขอบด้านซ้ายกับด้านซ้าย $\operatorname{cov}[\phi_1(0, y_1), \phi_2(0, y_2)],\ \forall y_1, y_2$

2. เงื่อนไขค่าขอบด้านซ้ายกับด้านขวา $\operatorname{cov}[\phi_1(0, y_1), \phi_2(L, y_2)],\ \forall y_1, y_2$

3. เงื่อนไขค่าขอบด้านซ้ายกับด้านบน $\operatorname{cov}[\phi_1(0, y_1), \phi_2(x_2, L)],\ \forall y_1, x_2$

4. เงื่อนไขค่าขอบด้านซ้ายกับด้านล่าง $\operatorname{cov}[\phi_1(0, y_1), \phi_2(x_2, 0)],\ \forall y_1, x_2$

5. เงื่อนไขค่าขอบด้านขวากับด้านขวา $\text{cov}[\phi_1(L, y_1), \phi_2(L, y_2)],\ \forall y_1, y_2$

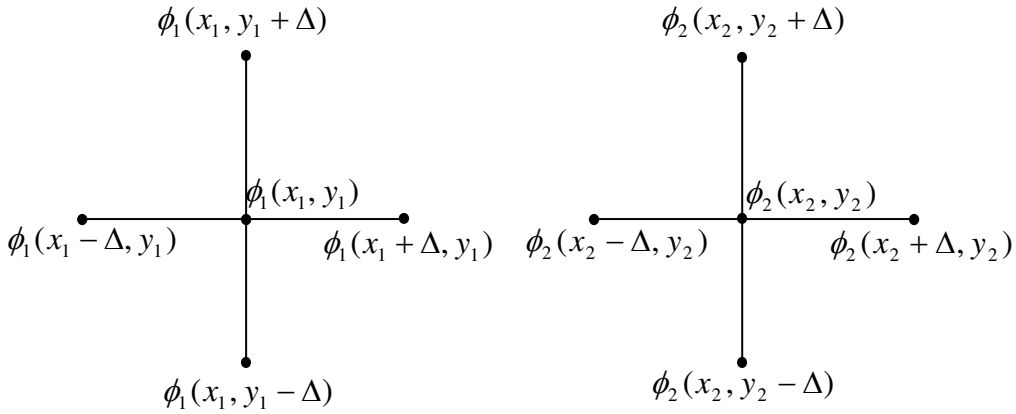
6. เงื่อนไขค่าขอบด้านขวากับด้านบน $\text{cov}[\phi_1(L, y_1), \phi_2(x_2, L)],\ \forall y_1, x_2$

7. เงื่อนไขค่าขอบด้านขวากับด้านล่าง $\text{cov}[\phi_1(L, y_1), \phi_2(x_2, 0)],\ \forall y_1, x_2$

8. เงื่อนไขค่าขอบด้านบนกับด้านบน $\text{cov}[\phi_1(x_1, L), \phi_2(x_2, L)],\ \forall x_1, x_2$

9. เงื่อนไขค่าขอบด้านบนกับด้านล่าง $\text{cov}[\phi_1(x_1, L), \phi_2(x_2, 0)],\ \forall x_1, x_2$

10. เงื่อนไขค่าขอบด้านล่างกับด้านล่าง $\text{cov}[\phi_1(x_1, 0), \phi_2(x_2, 0)],\ \forall x_1, x_2$



$$0 \le x_1, x_2 \le L \quad \text{และ} \quad 0 \le y_1, y_2 \le L$$

ภาพที่ 8 การประมาณค่าฟังก์ชัน $\text{cov}[\phi_1(x_1, y_1), \phi_2(x_2, y_2)]$

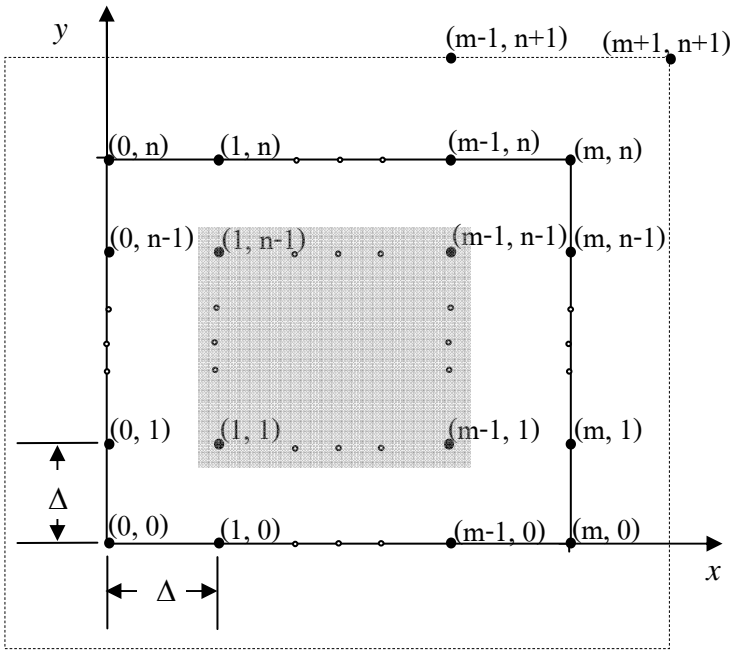
ในงานวิจัยนี้ฟังก์ชันความแปรปรวนร่วมในสมการ(21) มีลักษณะเป็นแบบไม่สมมาตร (Asymmetric covariance) นั่นคือ $\text{cov}[\phi_1, \phi_2)] \neq \text{cov}[\phi_2, \phi_1]$ ซึ่งความไม่สมมาตรดังกล่าวส่งผลให้ความแปรปรวนร่วมในแบบจำลองเชิงตัวเลขมีความไม่สมมาตรด้วย ดังนั้นสมการความสัมพันธ์ด้านล่างจึงถูกนำมาประยุกต์ใช้เพื่อปรับให้ฟังก์ชันความแปรปรวนร่วมของแบบจำลองมีความสมมาตร

$$\text{cov}[\phi_1, \phi_2)] = \frac{\text{cov}^*[\phi_1, \phi_2] + \text{cov}^*[\phi_2, \phi_1]}{2}$$

โดยที่ $\text{cov}^*[\phi_1, \phi_2]$ และ $\text{cov}^*[\phi_2, \phi_1]$ คือ ฟังก์ชันความแปรปรวนร่วมแบบไม่สมมาตร

เพื่อให้เกิดความเข้าใจที่ดีขึ้นพิจารณาการคำนวณหาฟังก์ชันความแปรปรวนร่วม โดยประมาณสามารถแสดงได้ดังขั้นตอนต่อไปนี้

ขั้นตอนที่ 1 แบ่งย่อยรูปร่างขอบเขตปัญหากล่องที่มีความกว้างขนาด m และยาวขนาด n ออกเป็นตารางที่มีจำนวนจุดต่อขนาด (m+2) ×(n+2) ใดๆ ดังภาพที่ 9 โดยเส้นทึบแสดงถึงผิวขอบของรูปร่างขอบเขตปัญหา



ภาพที่ 9 การแบ่งรูปร่างขอบเขตปัญหาด้วยจำนวนจุดต่อขนาด (m+2) ×(n+2) ในการหาฟังก์ชันความแปรปรวนร่วม

ขั้นตอนที่ 2 กำหนดค่าคงที่ในผลเฉลยทั่วไปของฟังก์ชันความแปรปรวนร่วมใน 2 มิติ (สมการ (19)) ซึ่งจะได้ผลเฉลยแม่นตรงเพื่อนำไปสร้างเงื่อนไขค่าขอบที่เป็นค่าสุ่มและมีความสัมพันธ์ร่วมทั้ง 10 เงื่อนไข คือ $\text{cov}[\phi_1(0, y_1), \phi_2(0, y_2)]$ $\text{cov}[\phi_1(0, y_1), \phi_2(m, y_2)]$ $\text{cov}[\phi_1(0, y_1), \phi_2(x_2, n)]$ $\text{cov}[\phi_1(0, y_1), \phi_2(x_2, 0)]$ $\text{cov}[\phi_1(m, y_1), \phi_2(m, y_2)]$ $\text{cov}[\phi_1(m, y_1), \phi_2(x_2, n)]$ $\text{cov}[\phi_1(m, y_1), \phi_2(x_2, 0)]$ $\text{cov}[\phi_1(x_1, n), \phi_2(x_2, n)]$ $\text{cov}[\phi_1(x_1, n), \phi_2(x_2, 0)]$ และ $\text{cov}[\phi_1(x_1, 0), \phi_2(x_2, 0)]$

ขั้นตอนที่ 3 กำหนดค่าเริ่มต้นสำหรับความแปรปรวนร่วมระหว่างตัวแปรที่จุดต่อดังต่อไปนี้

3.1 กำหนดค่าความแปรปรวนร่วมระหว่างตัวแปรที่จุดต่อบนผิวขอบของรูปร่างขอบเขตปัญหา(จุดต่อบนเส้นทึบ) และตัวแปรที่จุดต่อภายในรูปร่างขอบเขตปัญหา(จุดต่อในบริเวณแรที่เงา) โดยกำหนดให้มีค่าเริ่มต้นเท่ากับค่าคงที่ค่าหนึ่ง

3.2 กำหนดค่าความแปรปรวนร่วมระหว่างตัวแปรที่จุดต่อภายในของรูปร่างขอบเขตปัญหา(จุดต่อในบริเวณแรที่เงา) ด้วยกันเอง โดยกำหนดให้มีค่าเริ่มต้นเท่ากับค่าคงที่ค่าหนึ่ง

3.3 กำหนดค่าความแปรปรวนร่วมระหว่างตัวแปรที่จุดต่อบนผิวนอกของรูปร่างขอบเขตปัญหา(จุดต่อบนเส้นประ) กับตัวแปรที่จุดต่อใดๆ ในรูปร่างขอบเขตปัญหาให้มีค่าตายตัวเท่ากับศูนย์(ตัวแปรไม่มีความสัมพันธ์ร่วมกัน) ทั้งนี้เนื่องจากการคำนวณค่าความแปรปรวนร่วมบางค่าต้องเกี่ยวข้องกับตัวแปรที่ผิวนอก เช่น การคำนวณค่าความแปรปรวนร่วมระหว่างตัวแปรที่จุดต่อ (1, 1) และ (m-1, n) ต้องคำนวณจากความแปรปรวนร่วมทุกคู่จุดโดยรอบจุดต่อ(1, 1) และ (m-1, n) ดังภาพที่ 8 นั่นคือค่าความแปรปรวนร่วมระหว่างตัวแปรที่จุดต่อ (m-1, n+1) กับตัวแปรที่จุดต่อใดๆ ต้องถูกกำหนดให้มีค่าเท่ากับศูนย์ เป็นต้น

ขั้นตอนที่ 4 คำนวณหาค่าความแปรปรวนร่วมระหว่างตัวแปรทั้งหมดบนจุดต่อขนาด $m+1 \times n+1$ (ทุกจุดต่อบนกล่องสี่เหลี่ยมที่มีกรอบเป็นเส้นทึบ) โดยเริ่มต้นที่จุดต่อ (0, 0) และสิ้นสุดที่จุดต่อ (m, n)โดยที่เงื่อนไขค่าขอบที่เป็นค่าสุ่มและมีความสัมพันธ์ร่วมที่กำหนดไว้ในขั้นตอนที่ 2 ต้องไม่ถูกเปลี่ยนแปลง ทั้งนี้การคำนวณหาค่าความแปรปรวนร่วมระหว่างตัวแปรบนจุดต่อขนาด $m+1 \times n+1$ ดังกล่าวสามารถคำนวณหาได้โดยระเบียบวิธีการทำซ้ำ โดยสร้างระบบสมการเชิงเส้น

ให้ตัวแปรที่ต้องการหาค่าอยู่ด้านซ้ายมือของสมการในลักษณะเดียวกันกับสมการ (8) แต่ในกรณีการหาค่าความแปรปรวนร่วมนี้ตัวแปรด้านขวามือของระบบสมการเชิงเส้นข้างต้นต้องถูกสร้างขึ้นจากความสัมพันธ์ในสมการ (20) ซึ่งตัวแปรที่ต้องการคำนวณหา(ตัวแปรด้านซ้ายมือของระบบสมการเชิงเส้น) คือ $\text{cov}[\phi_1(0,0),\phi_2(0,0)]$, $\text{cov}[\phi_1(0,0),\phi_2(1,0)]$, $\text{cov}[\phi_1(0,0),\phi_2(2,0)]$, ..., $\text{cov}[\phi_1(0,0),\phi_2(m-1,0)]$, $\text{cov}[\phi_1(0,0),\phi_2(m,0)]$, $\text{cov}[\phi_1(0,0),\phi_2(0,1)]$, $\text{cov}[\phi_1(0,0),\phi_2(1,1)]$, ..., $\text{cov}[\phi_1(0,0),\phi_2(m-1,1)]$, $\text{cov}[\phi_1(0,0),\phi_2(m,1)]$, ..., $\text{cov}[\phi_1(m,n),\phi_2(m-1,n)]$ และ $\text{cov}[\phi_1(m,n),\phi_2(m,n)]$ จากนั้นใช้ระเบียบวิธีการทำซ้ำแบบ SOR แก้ระบบสมการเชิงเส้นข้างต้น โดยวนรอบทำซ้ำจนกว่าค่าความแปรปรวนร่วมของตัวแปรบนจุดต่อขนาด (m+1) × (n+1) ทุกตัวลู่เข้า นั่นคือค่าความแปรปรวนร่วมของตัวแปรบนจุดต่อในรอบทำซ้ำครั้งเก่าและใหม่มีค่าเปอร์เซ็นต์ผลต่างสัมบูรณ์น้อยกว่าค่าความผิดพลาดที่ยอมได้ ในทำนองเดียวกันการคำนวณหาค่าฟังก์ชันความแปรปรวนร่วมในปัญหา 3 มิติ สามารถทำได้โดยใช้ขั้นตอนทั้ง 4 ขั้นตอนที่กล่าวมา ดังแสดงในโปรแกรมที่ 1 ในภาคผนวก

2. การประมาณค่าฟังก์ชันความแปรปรวนร่วมในปัญหา 3 มิติ

พิจารณาสมการลาปลาสดั้งเดิมใน 3 มิติ

$$\frac{\partial^2\phi(x,y,z)}{\partial x^2}+\frac{\partial^2\phi(x,y,z)}{\partial y^2}+\frac{\partial^2\phi(x,y,z)}{\partial z^2}=0 \qquad (21)$$

$$0 \le x,y,z \le L$$

โดยที่ $\phi(x,y,z)$ คือ ฟังก์ชันต่อเนื่องที่กระจายตัวภายในกล่องสี่เหลี่ยมด้านเท่า ดังภาพที่ 10

ผลเฉลยทั่วไปของฟังก์ชันค่าเฉลี่ยสำหรับสมการลาปลาสดั้งเดิมในสมการ (21) สามารถแสดงได้ดังนี้

$$\phi(x,y,z)=\left(c_1e^{k_1x}+c_2e^{-k_1x}\right)(c_3\cos k_2y+c_4\sin k_2y)(c_5\cos k_3z+c_6\sin k_3z) \qquad (22)$$

โดย $c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ k_1\ k_2$ และ k_3 เป็นค่าคงที่ โดยที่ $k_1^2 = k_2^2 + k_3^2$ ทั้งนี้ค่าคงที่เหล่านี้เป็นค่าคงที่เฉพาะเจาะจงที่สามารถคำนวณหาได้ภายใต้เงื่อนไขค่าขอบแบบคงที่ทั้ง 6 เงื่อนไขดังนี้

1. เงื่อนไขค่าขอบด้านซ้าย $\phi(0, y, z)$

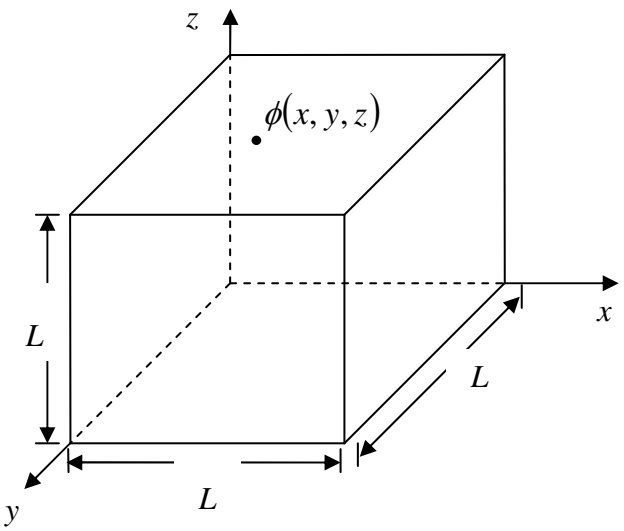
2. เงื่อนไขค่าขอบด้านขวา $\phi(L, y, z)$

3. เงื่อนไขค่าขอบด้านหน้า $\phi(x, L, z)$

4. เงื่อนไขค่าขอบด้านหลัง $\phi(x, 0, z)$

5. เงื่อนไขค่าขอบด้านบน $\phi(x, y, L)$

6. เงื่อนไขค่าขอบด้านล่าง $\phi(x, y, 0)$



ภาพที่ 10 ฟังก์ชันต่อเนื่อง $\phi(x, y, z)$ กระจายตัวภายในกล่องสี่เหลี่ยมด้านเท่า

ในการแก้สมการหาฟังก์ชันความแปรปรวนร่วมของสมการลาปลาสภายใต้เงื่อนไขค่าขอบที่เป็นค่าสุ่มและมีความสัมพันธ์ร่วม ขั้นแรกพิจารณาสมการลาปลาสดั้งเดิมในสมการ (21) ใน 6 โดเมนดังต่อไปนี้

$$\frac{\partial^2 \phi_1(x_1, y_1, z_1)}{\partial x_1^2} + \frac{\partial^2 \phi_1(x_1, y_1, z_1)}{\partial y_1^2} + \frac{\partial^2 \phi_1(x_1, y_1, z_1)}{\partial z_1^2} = 0 \tag{23}$$

$$\frac{\partial^2 \phi_2(x_2, y_2, z_2)}{\partial x_2^2} + \frac{\partial^2 \phi_2(x_2, y_2, z_2)}{\partial y_2^2} + \frac{\partial^2 \phi_2(x_2, y_2, z_2)}{\partial z_2^2} = 0 \tag{24}$$

เพื่อความสะดวกในการคำนวณเขียน $\phi_1(x_1, y_1, z_1)$ ในรูปของ ϕ_1 และเขียน $\phi_2(x_2, y_2, z_2)$ ในรูปของ ϕ_2

จากนั้น นำสมการ(23) คูณกับสมการ (24) จะได้สมการเชิงอนุพันธ์ย่อยอันดับ 4 ดังนี้

$$\frac{\partial^4 \phi_1\phi_2}{\partial x_1^2 \partial x_2^2} + \frac{\partial^4 \phi_1\phi_2}{\partial x_1^2 \partial y_2^2} + \frac{\partial^4 \phi_1\phi_2}{\partial x_1^2 \partial z_2^2} + \frac{\partial^4 \phi_1\phi_2}{\partial y_1^2 \partial x_2^2} + \frac{\partial^4 \phi_1\phi_2}{\partial y_1^2 \partial y_2^2} + \frac{\partial^4 \phi_1\phi_2}{\partial y_1^2 \partial z_2^2} +$$

$$\frac{\partial^4 \phi_1\phi_2}{\partial z_1^2 \partial x_2^2} + \frac{\partial^4 \phi_1\phi_2}{\partial z_1^2 \partial y_2^2} + \frac{\partial^4 \phi_1\phi_2}{\partial z_1^2 \partial z_2^2} = 0 \tag{25}$$

หลังจากประยุกต์ใช้ตัวดำเนินการค่าคาดหมายบนสองข้างของสมการ (25) รูปใหม่ของสมการดังกล่าวสามารถเขียนให้อยู่ในรูปดังต่อไปนี้

$$\frac{\partial^4 \operatorname{cov}[\phi_1, \phi_2]}{\partial x_1^2 \partial x_2^2} + \frac{\partial^4 \operatorname{cov}[\phi_1, \phi_2]}{\partial x_1^2 \partial y_2^2} + \frac{\partial^4 \operatorname{cov}[\phi_1, \phi_2]}{\partial x_1^2 \partial z_2^2} +$$

$$\frac{\partial^4 \operatorname{cov}[\phi_1, \phi_2]}{\partial y_1^2 \partial x_2^2} + \frac{\partial^4 \operatorname{cov}[\phi_1, \phi_2]}{\partial y_1^2 \partial y_2^2} + \frac{\partial^4 \operatorname{cov}[\phi_1, \phi_2]}{\partial y_1^2 \partial z_2^2} +$$

$$\frac{\partial^4 \operatorname{cov}[\phi_1, \phi_2]}{\partial z_1^2 \partial x_2^2} + \frac{\partial^4 \operatorname{cov}[\phi_1, \phi_2]}{\partial z_1^2 \partial y_2^2} + \frac{\partial^4 \operatorname{cov}[\phi_1, \phi_2]}{\partial z_1^2 \partial z_2^2} +$$

$$\left(\frac{\partial^2 E[\phi_1]}{\partial x_1^2}+\frac{\partial^2 E[\phi_1]}{\partial y_1^2}+\frac{\partial^2 E[\phi_1]}{\partial z_1^2}\right)\cdot\left(\frac{\partial^2 E[\phi_2]}{\partial x_2^2}+\frac{\partial^2 E[\phi_2]}{\partial y_2^2}+\frac{\partial^2 E[\phi_2]}{\partial z_2^2}\right)=0 \qquad (26)$$

$$\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial x_1^2\partial x_2^2}+\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial x_1^2\partial y_2^2}+\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial x_1^2\partial z_2^2}+$$

$$\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial y_1^2\partial x_2^2}+\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial y_1^2\partial y_2^2}+\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial y_1^2\partial z_2^2}+$$

$$\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial z_1^2\partial x_2^2}+\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial z_1^2\partial y_2^2}+\frac{\partial^4 \operatorname{cov}[\phi_1,\phi_2]}{\partial z_1^2\partial z_2^2}=0 \qquad (27)$$

สมการ (27) สามารถแก้ได้โดยตรงโดยใช้ระเบียบวิธีฟูเรียร์ดังนี้โดยในขั้นแรกกำหนดให้

$$\operatorname{cov}[\phi_1,\phi_2] = f_1(x_1)\cdot g_1(y_1)\cdot h_1(z_1)\cdot f_2(x_2)\cdot g_2(y_2)\cdot h_2(z_2) \qquad (28)$$

จากนั้น แทนสมการ(28) เข้าไปในสมการ (27) ซึ่งจะได้ความสัมพันธ์ดังแสดงในสมการ (29) ดังนี้

$$f_1''(x_1)\cdot g_1(y_1)\cdot h_1(z_1)\cdot f_2''(x_2)\cdot g_2(y_2)\cdot h_2(z_2) + f_1''(x_1)\cdot g_1(y_1)\cdot h_1(z_1)\cdot f_2(x_2)\cdot g_2''(y_2)\cdot h_2(z_2) +$$

$$f_1''(x_1)\cdot g_1(y_1)\cdot h_1(z_1)\cdot f_2(x_2)\cdot g_2(y_2)\cdot h_2''(z_2) + f_1(x_1)\cdot g_1''(y_1)\cdot h_1(z_1)\cdot f_2''(x_2)\cdot g_2(y_2)\cdot h_2(z_2) +$$

$$f_1(x_1)\cdot g_1''(y_1)\cdot h_1(z_1)\cdot f_2(x_2)\cdot g_2''(y_2)\cdot h_2(z_2) + f_1(x_1)\cdot g_1''(y_1)\cdot h_1(z_1)\cdot f_2(x_2)\cdot g_2(y_2)\cdot h_2''(z_2) +$$

$$f_1(x_1)\cdot g_1(y_1)\cdot h_1''(z_1)\cdot f_2''(x_2)\cdot g_2(y_2)\cdot h_2(z_2) + f_1(x_1)\cdot g_1(y_1)\cdot h_1''(z_1)\cdot f_2(x_2)\cdot g_2''(y_2)\cdot h_2(z_2) +$$

$$f_1(x_1)\cdot g_1(y_1)\cdot h_1''(z_1)\cdot f_2(x_2)\cdot g_2(y_2)\cdot h_2''(z_2) = 0 \qquad (29)$$

นำสมการ(29) หารด้วย สมการ (28) และหลังจากจัดรูปพจน์ในสมการจะได้

$$\left(\frac{f_1''(x_1)}{f_1(x_1)}+\frac{g_1''(y_1)}{g_1(y_1)}+\frac{h_1''(z_1)}{h_1(z_1)}\right)\cdot\left(\frac{f_2''(x_2)}{f_2(x_2)}+\frac{g_2''(y_2)}{g_2(y_2)}+\frac{h_2''(z_2)}{h_2(z_2)}\right)=0 \tag{30}$$

ผลเฉลยทั่วไปของฟังก์ชันความแปรปรวนร่วมสำหรับสมการ(30) สามารถแสดงได้ในรูปแบบต่อไปนี้

$$\begin{aligned}\operatorname{cov}[\phi_1(x_1,y_1,z_1),\phi_2(x_2,y_2,z_2)] &= \left(c_1e^{\sqrt{k_2}x_1}+c_2e^{-\sqrt{k_2}x_1}\right)\cdot\left(c_3e^{\sqrt{k_3}y_1}+c_4e^{-\sqrt{k_3}y_1}\right)\cdot \\ &\left(c_5e^{\sqrt{k_4}z_1}+c_6e^{-\sqrt{k_4}z_1}\right)\cdot\left(c_7\sin\sqrt{k_5}x_2+c_8\cos\sqrt{k_5}x_2\right)\cdot \\ &\left(c_9e^{\sqrt{k_6}y_2}+c_{10}e^{-\sqrt{k_6}y_2}\right)\cdot\left(c_{11}e^{\sqrt{k_7}z_2}+c_{12}e^{-\sqrt{k_7}z_2}\right)\end{aligned} \tag{31}$$

โดย $c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7\ c_8\ c_9\ c_{10}\ c_{11}\ c_{12}\ k_1\ k_2\ k_3\ k_4\ k_5\ k_6$ และ k_7 เป็นค่าคงที่ โดยที่ $k_1=k_2+k_3+k_4$ และ $k_5=k_6+k_7$ ในทำนองเดียวกับปัญหาแบบ 2 มิติดังแสดงในหัวข้อก่อนหน้านี้ ระเบียบวิธี FDM ได้ถูกนำมาประยุกต์ใช้ในการหาผลเฉลยโดยประมาณของฟังก์ชันความแปรปรวนร่วมในปัญหา 3 มิติ โดยในขั้นต้นประยุกต์ใช้อนุกรมเทย์เลอร์อันดับ 2 เพื่อหาค่าอนุพันธ์โดยประมาณของฟังก์ชันค่าเฉลี่ยสำหรับสมการลาปลาสดั้งเดิมในสมการ (21) ซึ่งฟังก์ชันค่าเฉลี่ยโดยประมาณในปัญหา 3 มิติสามารถแสดงได้ดังนี้

$$\begin{aligned}\phi(x,y,z) \cong \{&\phi(x+\Delta,y,z)+\phi(x-\Delta,y,z)+\phi(x,y+\Delta,z)+\phi(x,y-\Delta,z)+ \\ &\phi(x,y,z+\Delta)+\phi(x,y,z-\Delta)\}/\{6\}\end{aligned} \tag{32}$$

ในทำนองเดียวกันอนุพันธ์ย่อยอันดับ 4 ของฟังก์ชันความแปรปรวนร่วมในสมการ (27) สามารถประมาณได้ โดยใช้อนุพันธ์ย่อยอันดับ 2 ในสมการ (5) แทนเข้าไปในสมการ (27) ซึ่งหลังจากกระจายและรวมพจน์แล้ว รูปใหม่ของสมการ (27) สามารถแสดงได้ดังนี้

$$\operatorname{cov}[\phi_1(x_1, y_1, z_1), \phi_2(x_2, y_2, z_2)] \cong \{\operatorname{cov}[\phi_1(x_1 + \Delta, y_1, z_1), \phi_2(x_2 + \Delta, y_2, z_2)] +$$

$$\operatorname{cov}[\phi_1(x_1 - \Delta, y_1, z_1), \phi_2(x_2 + \Delta, y_2, z_2)] + \operatorname{cov}[\phi_1(x_1 + \Delta, y_1, z_1), \phi_2(x_2 - \Delta, y_2, z_2)] +$$

$$\operatorname{cov}[\phi_1(x_1 - \Delta, y_1, z_1), \phi_2(x_2 - \Delta, y_2, z_2)] + \operatorname{cov}[\phi_1(x_1 + \Delta, y_1, z_1), \phi_2(x_2, y_2 + \Delta, z_2)] +$$

$$\operatorname{cov}[\phi_1(x_1 - \Delta, y_1, z_1), \phi_2(x_2, y_2 + \Delta, z_2)] + \operatorname{cov}[\phi_1(x_1 + \Delta, y_1, z_1), \phi_2(x_2, y_2 - \Delta, z_2)] +$$

$$\operatorname{cov}[\phi_1(x_1 - \Delta, y_1, z_1), \phi_2(x_2, y_2 - \Delta, z_2)] + \operatorname{cov}[\phi_1(x_1 + \Delta, y_1, z_1), \phi_2(x_2, y_2, z_2 + \Delta)] +$$

$$\operatorname{cov}[\phi_1(x_1 - \Delta, y_1, z_1), \phi_2(x_2, y_2, z_2 + \Delta)] + \operatorname{cov}[\phi_1(x_1 + \Delta, y_1, z_1), \phi_2(x_2, y_2, z_2 - \Delta)] +$$

$$\operatorname{cov}[\phi_1(x_1 - \Delta, y_1, z_1), \phi_2(x_2, y_2, z_2 - \Delta)] + \operatorname{cov}[\phi_1(x_1, y_1 + \Delta, z_1), \phi_2(x_2 + \Delta, y_2, z_2)] +$$

$$\operatorname{cov}[\phi_1(x_1, y_1 - \Delta, z_1), \phi_2(x_2 + \Delta, y_2, z_2)] + \operatorname{cov}[\phi_1(x_1, y_1 + \Delta, z_1), \phi_2(x_2 - \Delta, y_2, z_2)] +$$

$$\operatorname{cov}[\phi_1(x_1, y_1 - \Delta, z_1), \phi_2(x_2 - \Delta, y_2, z_2)] + \operatorname{cov}[\phi_1(x_1, y_1 + \Delta, z_1), \phi_2(x_2, y_2 + \Delta, z_2)] +$$

$$\operatorname{cov}[\phi_1(x_1, y_1 - \Delta, z_1), \phi_2(x_2, y_2 + \Delta, z_2)] + \operatorname{cov}[\phi_1(x_1, y_1 + \Delta, z_1), \phi_2(x_2, y_2 - \Delta, z_2)] +$$

$$\operatorname{cov}[\phi_1(x_1, y_1 - \Delta, z_1), \phi_2(x_2, y_2 - \Delta, z_2)] + \operatorname{cov}[\phi_1(x_1, y_1 + \Delta, z_1), \phi_2(x_2, y_2, z_2 + \Delta)] +$$

$$\operatorname{cov}[\phi_1(x_1, y_1 - \Delta, z_1), \phi_2(x_2, y_2, z_2 + \Delta)] + \operatorname{cov}[\phi_1(x_1, y_1 + \Delta, z_1), \phi_2(x_2, y_2, z_2 - \Delta)] +$$

$$\operatorname{cov}[\phi_1(x_1, y_1 - \Delta, z_1), \phi_2(x_2, y_2, z_2 - \Delta)] + \operatorname{cov}[\phi_1(x_1, y_1, z_1 + \Delta), \phi_2(x_2 + \Delta, y_2, z_2)] +$$

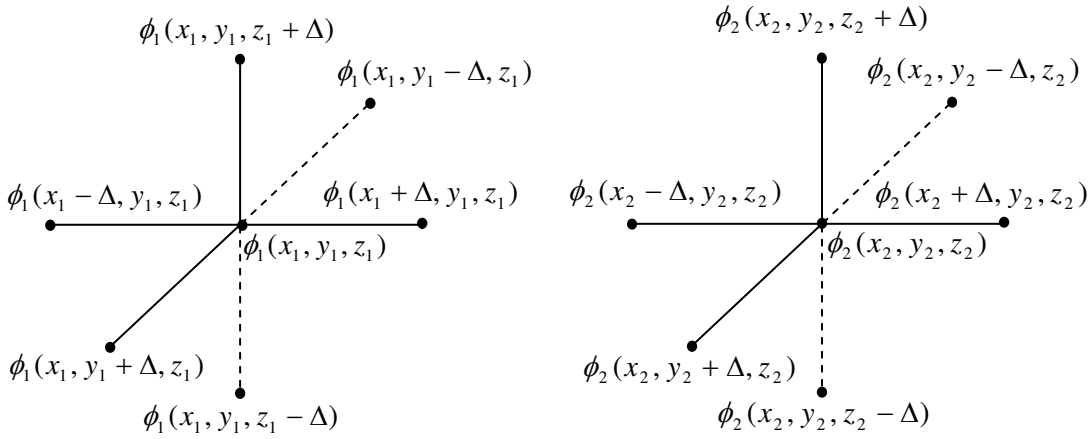
$$\operatorname{cov}[\phi_1(x_1, y_1, z_1 - \Delta), \phi_2(x_2 + \Delta, y_2, z_2)] + \operatorname{cov}[\phi_1(x_1, y_1, z_1 + \Delta), \phi_2(x_2 - \Delta, y_2, z_2)] +$$

$$\operatorname{cov}[\phi_1(x_1, y_1, z_1 - \Delta), \phi_2(x_2 - \Delta, y_2, z_2)] + \operatorname{cov}[\phi_1(x_1, y_1, z_1 + \Delta), \phi_2(x_2, y_2 + \Delta, z_2)] +$$

$$\begin{aligned}
&\operatorname{cov}[\phi_1(x_1, y_1, z_1 - \Delta), \phi_2(x_2, y_2 + \Delta, z_2)] + \operatorname{cov}[\phi_1(x_1, y_1, z_1 + \Delta), \phi_2(x_2, y_2 - \Delta, z_2)] + \\
&\operatorname{cov}[\phi_1(x_1, y_1, z_1 - \Delta), \phi_2(x_2, y_2 - \Delta, z_2)] + \operatorname{cov}[\phi_1(x_1, y_1, z_1 + \Delta), \phi_2(x_2, y_2, z_2 + \Delta)] + \\
&\operatorname{cov}[\phi_1(x_1, y_1, z_1 - \Delta), \phi_2(x_2, y_2, z_2 + \Delta)] + \operatorname{cov}[\phi_1(x_1, y_1, z_1 + \Delta), \phi_2(x_2, y_2, z_2 - \Delta)] + \\
&\operatorname{cov}[\phi_1(x_1, y_1, z_1 - \Delta), \phi_2(x_2, y_2, z_2 - \Delta)] - 6\operatorname{cov}[\phi_1(x_1 + \Delta, y_1, z_1), \phi_2(x_2, y_2, z_2)] - \\
&6\operatorname{cov}[\phi_1(x_1 - \Delta, y_1, z_1), \phi_2(x_2, y_2, z_2)] - 6\operatorname{cov}[\phi_1(x_1, y_1 + \Delta, z_1), \phi_2(x_2, y_2, z_2)] - \\
&6\operatorname{cov}[\phi_1(x_1, y_1 - \Delta, z_1), \phi_2(x_2, y_2, z_2)] - 6\operatorname{cov}[\phi_1(x_1, y_1, z_1 + \Delta), \phi_2(x_2, y_2, z_2)] - \\
&6\operatorname{cov}[\phi_1(x_1, y_1, z_1 - \Delta), \phi_2(x_2, y_2, z_2)] - 6\operatorname{cov}[\phi_1(x_1, y_1, z_1), \phi_2(x_2 + \Delta, y_2, z_2)] - \\
&6\operatorname{cov}[\phi_1(x_1, y_1, z_1), \phi_2(x_2 - \Delta, y_2, z_2)] - 6\operatorname{cov}[\phi_1(x_1, y_1, z_1), \phi_2(x_2, y_2 + \Delta, z_2)] - \\
&6\operatorname{cov}[\phi_1(x_1, y_1, z_1), \phi_2(x_2, y_2 - \Delta, z_2)] - 6\operatorname{cov}[\phi_1(x_1, y_1, z_1), \phi_2(x_2, y_2, z_2 + \Delta)] - \\
&6\operatorname{cov}[\phi_1(x_1, y_1, z_1), \phi_2(x_2, y_2, z_2 - \Delta)]\}/\{-36\} \qquad (33)
\end{aligned}$$

$$0 \le x_1, y_1, z_1, x_2, y_2, z_2 \le L$$

ค่าประมาณของฟังก์ชันความแปรปรวนร่วมระหว่างคู่จุด $\phi_1(x_1, y_1, z_1)$ และ $\phi_2(x_2, y_2, z_2)$ ในสมการ (33) เป็นการประมาณจากความแปรปรวนร่วมระหว่างคู่จุดทุกคู่รอบจุด $\phi_1(x_1, y_1, z_1)$ และ $\phi_2(x_2, y_2, z_2)$ ดังแสดงในภาพที่ 11



$0 \le x_1, y_1, z_1 \le L$ และ $0 \le x_2, y_2, z_2 \le L$

ภาพที่ 11 การประมาณค่าฟังก์ชัน $\text{cov}[\phi_1(x_1, y_1, z_1), \phi_2(x_2, y_2, z_2)]$

ทั้งนี้สมการ(31) และ สมการ (33) จะสามารถแก้ได้เมื่อเงื่อนไขค่าขอบที่เป็นค่าสุ่มและมีความสัมพันธ์ร่วมของคู่จุดที่ผิวขอบของรูปร่างขอบเขตปัญหาทั้ง 21 เงื่อนไขต่อไปนี้ถูกกำหนดขึ้น

1. ด้านซ้าย และ ด้านซ้าย $\text{cov}[\phi_1(0, y_1, z_1), \phi_2(0, y_2, z_2)],\ \forall y_1, z_1, y_2, z_2$
2. ด้านซ้าย และด้านขวา $\text{cov}[\phi_1(0, y_1, z_1), \phi_2(L, y_2, z_2)],\ \forall y_1, z_1, y_2, z_2$
3. ด้านซ้าย และด้านหน้า $\text{cov}[\phi_1(0, y_1, z_1), \phi_2(x_2, L, z_2)],\ \forall y_1, z_1, x_2, z_2$
4. ด้านซ้าย และด้านหลัง $\text{cov}[\phi_1(0, y_1, z_1), \phi_2(x_2, 0, z_2)],\ \forall y_1, z_1, x_2, z_2$
5. ด้านซ้าย และด้านบน $\text{cov}[\phi_1(0, y_1, z_1), \phi_2(x_2, y_2, L)],\ \forall y_1, z_1, x_2, y_2$
6. ด้านซ้าย และด้านล่าง $\text{cov}[\phi_1(0, y_1, z_1), \phi_2(x_2, y_2, 0)],\ \forall y_1, z_1, x_2, y_2$
7. ด้านขวา และด้านขวา $\text{cov}[\phi_1(L, y_1, z_1), \phi_2(L, y_2, z_2)],\ \forall y_1, z_1, y_2, z_2$

8. ด้านขวา และด้านหน้า $\text{cov}[\phi_1(L, y_1, z_1), \phi_2(x_2, L, z_2)],\ \forall y_1, z_1, x_2, z_2$

9. ด้านขวา และด้านหลัง $\text{cov}[\phi_1(L, y_1, z_1), \phi_2(x_2, 0, z_2)],\ \forall y_1, z_1, x_2, z_2$

10. ด้านขวา และด้านบน $\text{cov}[\phi_1(L, y_1, z_1), \phi_2(x_2, y_2, L)],\ \forall y_1, z_1, x_2, y_2$

11. ด้านขวา และด้านล่าง $\text{cov}[\phi_1(L, y_1, z_1), \phi_2(x_2, y_2, 0)],\ \forall y_1, z_1, x_2, y_2$

12. ด้านหน้า และด้านหน้า $\text{cov}[\phi_1(x_1, L, z_1), \phi_2(x_2, L, z_2)],\ \forall x_1, z_1, x_2, z_2$

13. ด้านหน้า และด้านหลัง $\text{cov}[\phi_1(x_1, L, z_1), \phi_2(x_2, 0, z_2)],\ \forall x_1, z_1, x_2, z_2$

14. ด้านหน้า และด้านบน $\text{cov}[\phi_1(x_1, L, z_1), \phi_2(x_2, y_2, L)],\ \forall x_1, z_1, x_2, y_2$

15. ด้านหน้า และด้านล่าง $\text{cov}[\phi_1(x_1, L, z_1), \phi_2(x_2, y_2, 0)],\ \forall x_1, z_1, x_2, y_2$

16. ด้านหลัง และด้านหลัง $\text{cov}[\phi_1(x_1, 0, z_1), \phi_2(x_2, 0, z_2)],\ \forall x_1, z_1, x_2, z_2$

17. ด้านหลัง และด้านบน $\text{cov}[\phi_1(x_1, 0, z_1), \phi_2(x_2, y_2, L)],\ \forall x_1, z_1, x_2, y_2$

18. ด้านหลัง และด้านล่าง $\text{cov}[\phi_1(x_1, 0, z_1), \phi_2(x_2, y_2, 0)],\ \forall x_1, z_1, x_2, y_2$

19. ด้านบน และด้านบน $\text{cov}[\phi_1(x_1, y_1, L), \phi_2(x_2, y_2, L)],\ \forall x_1, y_1, x_2, y_2$

20. ด้านบน และด้านล่าง $\text{cov}[\phi_1(x_1, y_1, L), \phi_2(x_2, y_2, 0)],\ \forall x_1, y_1, x_2, y_2$

21. ด้านล่าง และด้านล่าง $\text{cov}[\phi_1(x_1, y_1, 0), \phi_2(x_2, y_2, 0)],\ \forall x_1, y_1, x_2, y_2$

ในทำนองเดียวกับปัญหา 2 มิติ ฟังก์ชันความแปรปรวนร่วมในสมการ (31) เป็นแบบไม่สมมาตร ดังนั้นฟังก์ชันความแปรปรวนร่วมในสมการ(31) จึงต้องถูกนำมาปรับแก้ให้เกิดความสมมาตรเพื่อนำไปใช้ในการคำนวณของแบบจำลอง ดังแสดงในโปรแกรมที่ 2 ในภาคผนวก นอกจากนี้ในกรณีที่แบบจำลองเชิงตัวเลขที่ใช้ในการประมาณค่าฟังก์ชันความแปรปรวนร่วมข้างต้นมีขนาดใหญ่จนคอมพิวเตอร์เพียงเครื่องเดียวไม่มีหน่วยความจำเพียงพอที่จะใช้ในการประมวลผลข้อมูลเหล่านี้ ดังนั้นระเบียบการแบ่งส่วนโดเมนจึงถูกนำมาประยุกต์ใช้เพื่อแก้ปัญหาข้างต้น

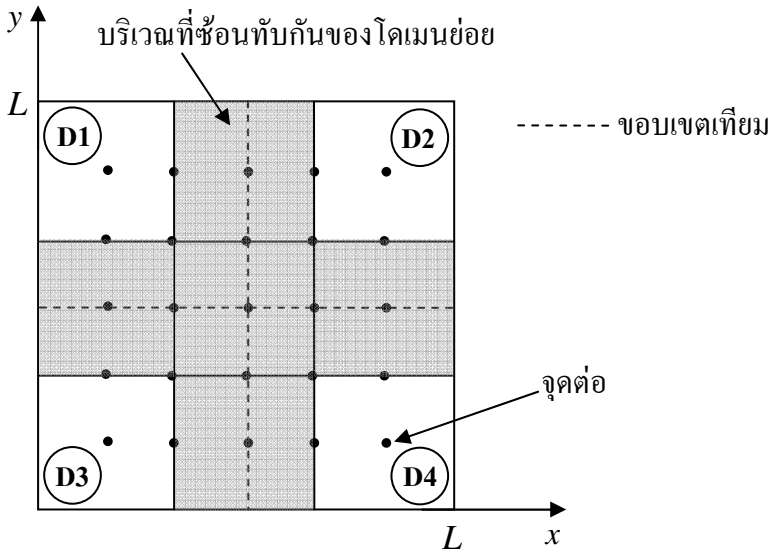
3. การประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมนในการประมาณค่าฟังก์ชันความแปรปรวนร่วม

ในงานวิจัยนี้การแก้สมการหาผลเฉลยโดยประมาณของฟังก์ชันความแปรปรวนร่วมสำหรับสมการลาปลาสเป็นปัญหาที่ซับซ้อน โดยเฉพาะกรณีที่โดเมนมีขนาดใหญ่ที่ต้องใช้ระเบียบวิธีแบ่งส่วนโดเมนเข้ามาช่วยในการแก้ปัญหา จากความซับซ้อนข้างต้นระเบียบวิธีแบ่งส่วนโดเมนแบบโดเมนย่อยซ้อนทับกัน (Overlapping domain) จึงถูกนำมาประยุกต์ใช้ในประมาณฟังก์ชันความแปรปรวนร่วมทั้งในปัญหา 2 และ 3 มิติ โดยรูปร่างขอบเขตปัญหาถูกแบ่งย่อยออกเป็น 4 โดเมนย่อยที่ซ้อนทับกันดังแสดงในภาพที่ 12 โดยแท้จริงแล้วบริเวณที่ซ้อนทับกันของโดเมนย่อยจำเป็นต้องครอบคลุมจุดต่อให้น้อยที่สุดเท่าที่เป็นไปได้ ทั้งนี้เพื่อประหยัดทรัพยากรของคอมพิวเตอร์ที่ใช้ในการคำนวณ ในขณะเดียวกันจำนวนจุดต่อที่น้อยดังกล่าวกลับทำให้ค่าความคลาดเคลื่อนในการคำนวณมากขึ้น (Min and Yang, 2006) อย่างไรก็ตามค่าความคลาดเคลื่อนไม่ได้ลดลงเสมอไปเมื่อจำนวนจุดต่อ (ในบริเวณที่โดเมนย่อยซ้อนทับกัน) มากขึ้น แต่เมื่อจุดต่อข้างต้นมีจำนวนเข้าใกล้ครึ่งหนึ่งของจำนวนจุดต่อในโดเมนย่อย ค่าความคลาดเคลื่อนจะมีค่าน้อยที่สุด ดังนั้นงานวิจัยนี้จึงเลือกใช้จุดต่อในบริเวณที่โดเมนย่อยซ้อนทับกันจำนวน 2 จุดต่อเท่านั้น ทั้งนี้เพื่อประหยัดหน่วยความจำของคอมพิวเตอร์ โดยประยุกต์ใช้ทั้งในปัญหา 2 และ 3 มิติ

การประมาณฟังก์ชันความแปรปรวนร่วมในปัญหา 2 มิติ รูปร่างขอบเขตปัญหาถูกแบ่งออกเป็น 4 โดเมนย่อยดังภาพที่ 12 นั่นคือค่าความแปรปรวนร่วมภายในโดเมนใหญ่ต้องถูกคำนวณแยกจากกัน นอกจากนี้เนื่องจากฟังก์ชันความแปรปรวนร่วมที่ถูกสร้างขึ้นจากระเบียบวิธีฟูเรียร์มีคุณสมบัติไม่สมมาตร ดังนั้นการคำนวณค่าความแปรปรวนร่วมจึงต้องถูกคำนวณบนคู่ของโดเมนย่อยทุกคู่ที่เป็นไปได้ ซึ่งในที่นี้ค่าความแปรปรวนร่วมระหว่างจุดบนคู่โดเมนย่อยทั้ง 16 คู่โดเมนต่อไปนี้ต้องถูกคำนวณหา

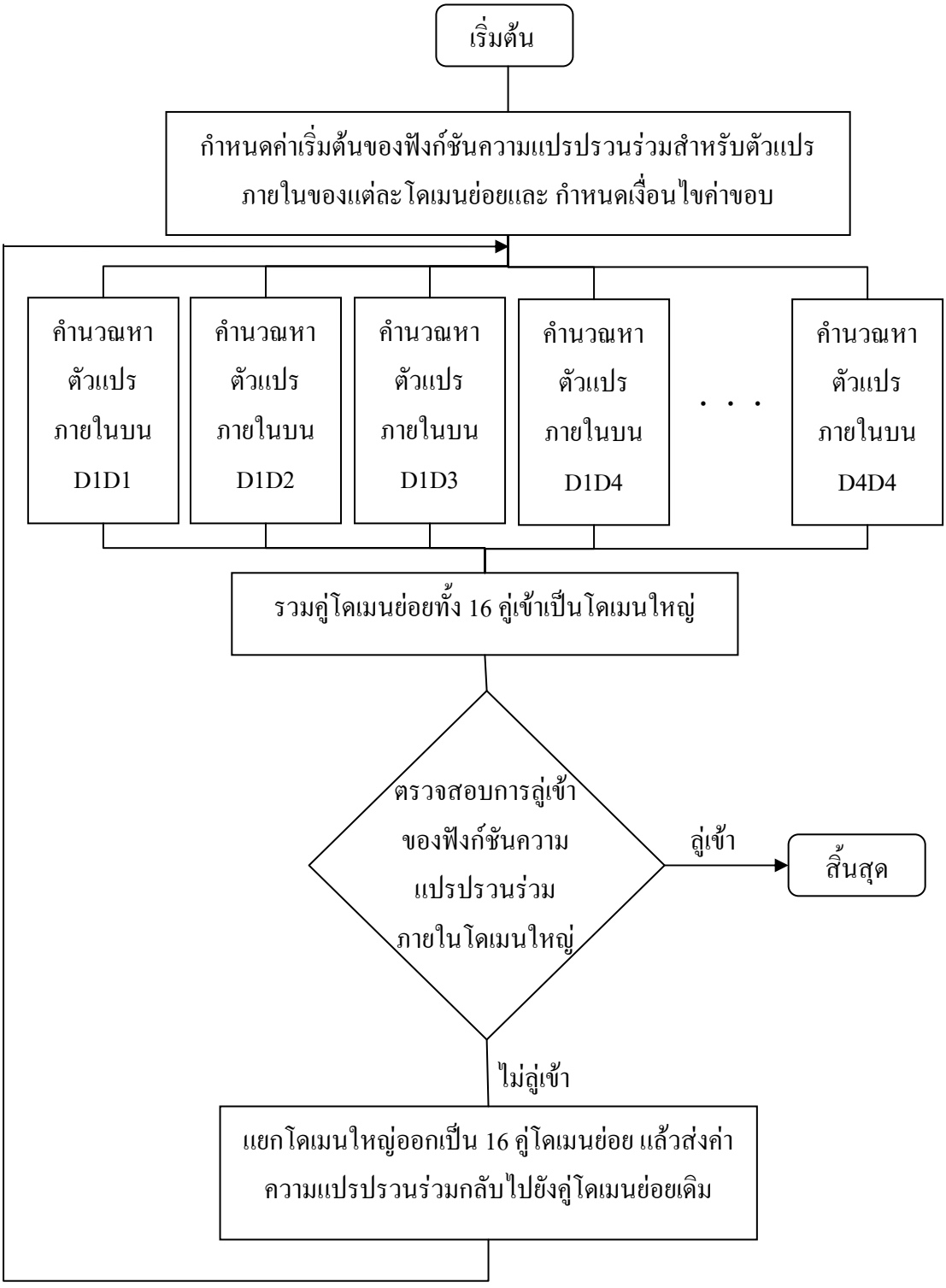
1. คู่โดเมนย่อย D1D1
2. คู่โดเมนย่อย D1D2
3. คู่โดเมนย่อย D1D3
4. คู่โดเมนย่อย D1D4
5. คู่โดเมนย่อย D2D1
6. คู่โดเมนย่อย D2D2
7. คู่โดเมนย่อย D2D3
8. คู่โดเมนย่อย D2D4
9. คู่โดเมนย่อย D3D1
10. คู่โดเมนย่อย D3D2
11. คู่โดเมนย่อย D3D3
12. คู่โดเมนย่อย D3D4
13. คู่โดเมนย่อย D4D1
14. คู่โดเมนย่อย D4D2
15. คู่โดเมนย่อย D4D3

16. คู่โดเมนย่อย D4D4



ภาพที่ 12 การแบ่งรูปร่างขอบเขตปัญหาออกเป็นโดเมนย่อยที่ซ้อนทับกันในปัญหา 2 มิติ

ทั้งนี้ความแปรปรวนร่วมระหว่างจุดบนคู่โดเมนย่อยทั้ง 16 คู่โดเมนสามารถคำนวณได้ดังขั้นตอนในภาพที่ 13 โดยในขั้นแรกกำหนดค่าเริ่มต้นของฟังก์ชันความแปรปรวนร่วมสำหรับตัวแปรภายในของแต่ละคู่โดเมนย่อยและ กำหนดเงื่อนไขค่าขอบ จากนั้นเงื่อนไขเหล่านี้จะถูกใช้ในขั้นตอนการคำนวณฟังก์ชันความแปรปรวนร่วมสำหรับตัวแปรภายในบนคู่โดเมนย่อยทั้ง 16 คู่โดเมนโดยอิสระแก่กัน โดยใช้ความสัมพันธ์ในสมการ (20) จากนั้นค่าความแปรปรวนร่วมในแต่ละคู่โดเมนย่อยทั้ง 16 คู่โดเมนจะถูกรวมเข้าเป็นโดเมนหลักอีกครั้ง ทั้งนี้เพื่อปรับปรุงค่าความแปรปรวนร่วมสำหรับตัวแปรภายในของรูปร่างขอบเขตปัญหาทั้งหมดให้ใหม่ขึ้น ท้ายที่สุดตรวจสอบการลู่เข้าของค่าความแปรปรวนร่วมภายในโดเมนใหญ่ โดยตรวจสอบว่าเปอร์เซ็นต์ผลต่างสัมบูรณ์ของผลลัพธ์ในรอบทำซ้ำเก่าและใหม่มีค่าน้อยกว่าค่าความผิดพลาดที่ยอมได้หรือไม่ ถ้าเปอร์เซ็นต์ผลต่างสัมบูรณ์ดังกล่าวน้อยกว่าค่าความผิดพลาดที่ยอมได้ นั่นคือระบบสมการของแบบจำลองลู่เข้าสู่ผลลัพธ์ที่แท้จริง ในทางตรงข้ามถ้าเปอร์เซ็นต์ผลต่างสัมบูรณ์ของค่าความแปรปรวนร่วมภายในของโดเมนใหญ่ตัวใดตัวหนึ่ง มีค่ามากกว่าค่าความผิดพลาดที่ยอมได้ นั่นคือระบบสมการยังไม่ลู่เข้า ดังนั้นโดเมนใหญ่ต้องถูกแบ่งออกเป็นคู่โดเมนย่อย 16 คู่อีกครั้ง ทั้งนี้เพื่อส่งกลับค่าความแปรปรวนร่วมที่คำนวณได้ไปคำนวณซ้ำอีกครั้ง และวนรอบทำซ้ำจนกว่าผลลัพธ์จะลู่เข้า

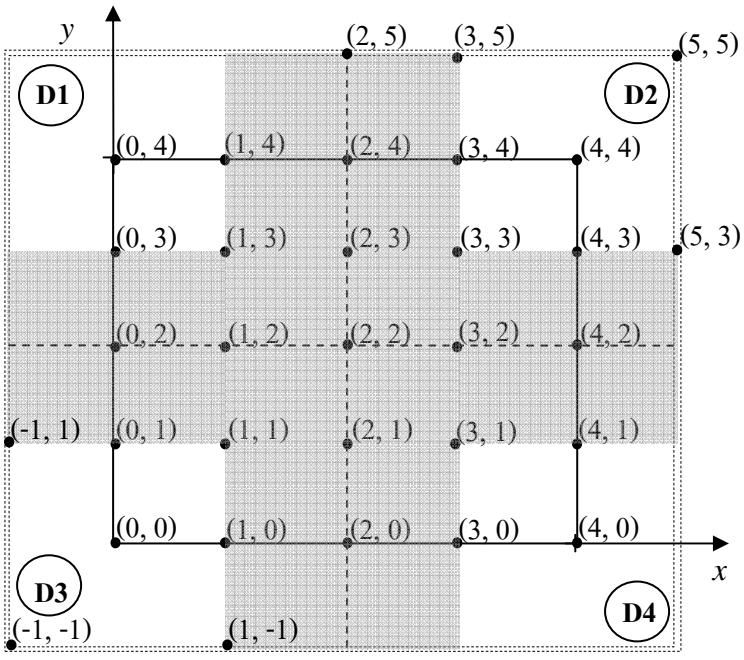


ภาพที่ 13 ขั้นตอนการคำนวณระเบียบวิธีแบ่งส่วนโดเมนสำหรับฟังก์ชันความแปรปรวนร่วม

ตัวอย่างการคำนวณค่าฟังก์ชันความแปรปรวนร่วมกรณีประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมนแบบโดเมนย่อยซ้อนทับกันในปัญหา 2 มิติ สามารถแสดงได้ดังนี้

ขั้นตอนที่ 1 แบ่งย่อยรูปร่างขอบเขตปัญหากล่องที่มีความกว้างและยาวขนาด 4 หน่วย ออกเป็นตารางที่มีจำนวนจุดต่อขนาด 6×6 ดังภาพที่ 14 โดยเส้นทึบแสดงถึงผิวขอบของรูปร่างขอบเขตปัญหา

ขั้นตอนที่ 2 กำหนดค่าคงที่ในผลเฉลยทั่วไปของฟังก์ชันความแปรปรวนร่วมใน 2 มิติ (สมการ (19)) ซึ่งจะได้ผลเฉลยแม่นตรงเพื่อนำไปสร้างเงื่อนไขค่าขอบที่เป็นค่าสุ่มและมีความสัมพันธ์ร่วมทั้ง 10 เงื่อนไข คือ $\mathrm{cov}[\phi_1(0,y_1),\phi_2(0,y_2)]$ $\mathrm{cov}[\phi_1(0,y_1),\phi_2(4,y_2)]$ $\mathrm{cov}[\phi_1(0,y_1),\phi_2(x_2,4)]$ $\mathrm{cov}[\phi_1(0,y_1),\phi_2(x_2,0)]$ $\mathrm{cov}[\phi_1(4,y_1),\phi_2(4,y_2)]$ $\mathrm{cov}[\phi_1(4,y_1),\phi_2(x_2,4)]$ $\mathrm{cov}[\phi_1(4,y_1),\phi_2(x_2,0)]$ $\mathrm{cov}[\phi_1(x_1,4),\phi_2(x_2,4)]$ $\mathrm{cov}[\phi_1(x_1,4),\phi_2(x_2,0)]$ และ $\mathrm{cov}[\phi_1(x_1,0),\phi_2(x_2,0)]$ โดยที่ $x_1,y_1,x_2,y_2\in\{1,2,3,4\}$



ภาพที่ 14 การแบ่งรูปร่างขอบเขตปัญหาออกเป็น 4 โดเมนย่อยที่ซ้อนทับกันภายใต้จำนวนจุดต่อขนาด 6 × 6

ขั้นตอนที่ 3 กำหนดค่าเริ่มต้นสำหรับความแปรปรวนร่วมระหว่างตัวแปรที่จุดต่อ ดังต่อไปนี้

3.1 กำหนดค่าความแปรปรวนร่วมระหว่างตัวแปรที่จุดต่อบนผิวขอบของรูปร่างขอบเขตปัญหา(จุดต่อบนเส้นทึบ) และตัวแปรที่จุดต่อภายในของรูปร่างขอบเขตปัญหาโดยกำหนดให้มีค่าเริ่มต้นเท่ากับค่าคงที่ค่าหนึ่ง

3.2 กำหนดค่าความแปรปรวนร่วมระหว่างตัวแปรที่จุดต่อภายในของรูปร่างขอบเขตปัญหาด้วยกันเอง โดยกำหนดให้มีค่าเริ่มต้นเท่ากับค่าคงที่ค่าหนึ่ง

3.3 กำหนดค่าความแปรปรวนร่วมระหว่างตัวแปรที่จุดต่อบนผิวนอกของรูปร่างขอบเขตปัญหา(จุดต่อบนเส้นประคู่) กับตัวแปรที่จุดต่อใดๆ ในรูปร่างขอบเขตปัญหาให้มีค่าตายตัวเท่ากับศูนย์(ตัวแปรไม่มีความสัมพันธ์ร่วมกัน) ทั้งนี้เนื่องจากการคำนวณค่าความแปรปรวนร่วมบางค่าต้องเกี่ยวข้องกับตัวแปรที่ผิวนอก เช่น การคำนวณค่าความแปรปรวนร่วมระหว่างตัวแปรที่จุดต่อ(1, 1) และ (3, 4) ต้องคำนวณจากความแปรปรวนร่วมทุกคู่จุดโดยรอบจุดต่อ(1, 1) และ (3, 4) ดังภาพที่ 8 นั่นคือค่าความแปรปรวนร่วมระหว่างตัวแปรที่จุดต่อ (3, 5) กับตัวแปรที่จุดต่อใดๆ ต้องถูกกำหนดให้มีค่าเท่ากับศูนย์ เป็นต้น

ขั้นตอนที่ 4 แบ่งส่วนรูปร่างขอบเขตปัญหาออกเป็น 4 โดเมนย่อยดังภาพที่ 14 โดยพื้นที่ที่แรเงาคือบริเวณที่ซ้อนทับกันของโดเมนย่อย ทั้งนี้กำหนดให้แต่ละโดเมนย่อยมีอาณาบริเวณที่ซ้อนทับกันครอบคลุมจุดต่อจำนวน 2 จุด โดยโดเมนย่อยทั้ง 4 โดเมนย่อยครอบคลุมจุดต่อดังต่อไปนี้

โดเมนย่อย D1 เริ่มต้นและสิ้นสุดที่จุดต่อ (-1, 1) และ (3, 5) ตามลำดับ

โดเมนย่อย D2 เริ่มต้นและสิ้นสุดที่จุดต่อ (1, 1) และ (5, 5) ตามลำดับ

โดเมนย่อย D3 เริ่มต้นและสิ้นสุดที่จุดต่อ (-1, -1) และ (3, 3) ตามลำดับ

โดเมนย่อย D4 เริ่มต้นและสิ้นสุดที่จุดต่อ (1, -1) และ (5, 3) ตามลำดับ

ขั้นตอนที่ 5 คำนวณหาค่าความแปรปรวนร่วมของตัวแปรบนขอบเขตปัญหาจริง(เริ่มต้นที่จุดต่อ (0, 0) และสิ้นสุดที่จุดต่อ (4, 4)) นั่นคือ คำนวณค่าความแปรปรวนร่วมระหว่างตัวแปรบนโดเมนย่อยที่ครอบคลุมจุดต่อดังแสดงต่อไปนี้

โดเมนย่อย D1 เริ่มต้นและสิ้นสุดที่จุดต่อ (0, 1) และ (3, 4) ตามลำดับ

โดเมนย่อย D2 เริ่มต้นและสิ้นสุดที่จุดต่อ (1, 1) และ (4, 4) ตามลำดับ

โดเมนย่อย D3 เริ่มต้นและสิ้นสุดที่จุดต่อ (0, 0) และ (3, 3) ตามลำดับ

โดเมนย่อย D4 เริ่มต้นและสิ้นสุดที่จุดต่อ (1, 0) และ (4, 3) ตามลำดับ

โดยการคำนวณหาค่าความแปรปรวนร่วมระหว่างตัวแปรบนคู่โดเมนย่อยข้างต้นสามารถคำนวณได้โดยใช้ความสัมพันธ์ในสมการ (20) ดังนี้

1. คู่โดเมนย่อย D1D1 เช่น $\mathrm{cov}[\phi_1(0,2),\phi_2(1,2)]$

2. คู่โดเมนย่อย D1D2 เช่น $\mathrm{cov}[\phi_1(0,2),\phi_2(4,4)]$

3. คู่โดเมนย่อย D1D3 เช่น $\mathrm{cov}[\phi_1(0,2),\phi_2(1,1)]$

4. คู่โดเมนย่อย D1D4 เช่น $\mathrm{cov}[\phi_1(0,2),\phi_2(3,1)]$

5. คู่โดเมนย่อย D2D1 เช่น $\mathrm{cov}[\phi_1(1,3),\phi_2(1,2)]$

6. คู่โดเมนย่อย D2D2 เช่น $\mathrm{cov}[\phi_1(1,3),\phi_2(4,4)]$

7. คู่โดเมนย่อย D2D3 เช่น $\mathrm{cov}[\phi_1(1,3),\phi_2(1,1)]$

8. คู่โดเมนย่อย D2D4 เช่น $\mathrm{cov}[\phi_1(1,3),\phi_2(3,1)]$

9. คู่โดเมนย่อย D3D1 เช่น $\mathrm{cov}[\phi_1(0,0),\phi_2(1,2)]$

10. คู่โดเมนย่อย D3D2 เช่น $\mathrm{cov}[\phi_1(0,0),\phi_2(4,4)]$

11. คู่โดเมนย่อย D3D3 เช่น $\mathrm{cov}[\phi_1(0,0),\phi_2(1,1)]$

12. คู่โดเมนย่อย D3D4 เช่น $\mathrm{cov}[\phi_1(0,0),\phi_2(3,1)]$

13. คู่โดเมนย่อย D4D1 เช่น $\mathrm{cov}[\phi_1(4,0),\phi_2(1,2)]$

14. คู่โดเมนย่อย D4D2 เช่น $\mathrm{cov}[\phi_1(4,0),\phi_2(4,4)]$

15. คู่โดเมนย่อย D4D3 เช่น $\mathrm{cov}[\phi_1(4,0),\phi_2(1,1)]$

16. คู่โดเมนย่อย D4D4 เช่น $\mathrm{cov}[\phi_1(4,0),\phi_2(3,1)]$

ขั้นตอนที่ 6 นำคู่โดเมนย่อยทั้ง 16 คู่โดเมนย่อยมารวมเข้าด้วยกันอีกครั้ง ซึ่งจะได้โดเมนใหญ่ที่มีจุดเริ่มต้นและสิ้นสุดที่จุดต่อ (-1, -1) และ (5, 5) ตามลำดับ

ขั้นตอนที่ 7 ใช้ระเบียบวิธีการทำซ้ำโดยวนรอบทำซ้ำจนกว่าค่าความแปรปรวนร่วมระหว่างตัวแปรบนจุดต่อในขั้นตอนที่ 6 ทุกตัวแปรลู่เข้า ดังแสดงในโปรแกรมที่ 3 ในภาคผนวก

ในทำนองเดียวกันการประมาณฟังก์ชันความแปรปรวนร่วมสำหรับสมการลาปลาสในปัญหา 3 มิติ สามารถทำได้โดยแบ่งรูปร่างขอบเขตปัญหาออกเป็น 4 โดเมนย่อยดังภาพที่ 5 โดยที่โดเมนย่อยแต่ละโดเมนย่อยซ้อนทับกันด้วยจำนวนจุดต่อ 2 จุด ในลักษณะเดียวกับภาพที่ 12 โดยขั้นตอนการคำนวณของระเบียบวิธีแบ่งส่วนโดเมนแบบโดเมนย่อยซ้อนทับกันสามารถประยุกต์ใช้จากภาพที่ 13 อีกทั้งการคำนวณฟังก์ชันความแปรปรวนร่วมสำหรับตัวแปรภายในของคู่โดเมนย่อยสามารถคำนวณได้จากความสัมพันธ์ในสมการ (33) ดังแสดงในโปรแกรมที่ 4 ในภาคผนวก

เนื่องจากในงานวิจัยนี้ระเบียบวิธีแบ่งส่วนโดเมนถูกประมวลผลบนคอมพิวเตอร์เพียงเครื่องเดียว ซึ่งระบบสมการในโดเมนย่อยไม่ได้ถูกคำนวณโดยอิสระต่อกันโดยแท้จริง(ตัวแปรในโดเมนย่อยแต่ละโดเมนย่อยสามารถติดต่อกันได้) อีกทั้งคอมพิวเตอร์ต้องแบ่งหน่วยความจำไว้จำตัวแปรของทุกโดเมนย่อย ซึ่งทำให้คอมพิวเตอร์ขาดศักยภาพอย่างเต็มที่ในการประมวลผลข้อมูลของโดเมนย่อยอันหนึ่งอันใด ด้วยเหตุนี้เพื่อแก้ปัญหาข้างต้นตัวแปรในแต่ละโดเมนย่อยหลังจากที่ถูกคำนวณแล้วต้องถูกบันทึกเก็บค่าตัวแปรไว้ในไฟล์ .mat และหลังจากที่ตัวแปรถูกบันทึกลงไฟล์แล้วตัวแปรดังกล่าวต้องถูกลบออกจากหน่วยความจำของคอมพิวเตอร์ โดยเหลือไว้เพียงแต่ไฟล์ที่เก็บค่าตัวแปรของโดเมนย่อยไว้เท่านั้น และในคราวจำเป็นตัวแปรที่ถูกเก็บค่าไว้ในไฟล์ดังกล่าวจะถูกดึงออกมาใช้ในการคำนวณในแต่ละรอบทำซ้ำ โดยอ่านค่าจากไฟล์ .mat ที่เก็บตัวแปรนั้นไว้

ผลและวิจารณ์

ผล

ในงานวิจัยนี้แบบจำลองเชิงตัวเลขสำหรับประมาณผลเฉลยของสมการลาปลาสได้ถูกสร้างขึ้น โดยใช้การกระจายอนุกรมเทย์เลอร์อันดับ 2 ในการหาอนุพันธ์โดยประมาณของสมการ ทั้งนี้เพื่อนำอนุพันธ์ดังกล่าวมาใช้ในการประมาณฟังก์ชันค่าเฉลี่ยและฟังก์ชันความแปรปรวนร่วมดังแสดงในสมการ (7) (20) (32) และ (33) โดยค่าคงที่ที่จำเป็นของผลเฉลยทั่วไปในสมการ (10) (19) (22) และ (31) ได้ถูกกำหนดขึ้นเพื่อสร้างผลเฉลยแม่นตรง ทั้งนี้เพื่อนำผลเฉลยแม่นตรงดังกล่าวมาเป็นเงื่อนไขค่าขอบสำหรับสมการ (7) (20) (32) และ (33) และนำไปใช้ในการประมาณค่าตัวแปรภายในของรูปร่างขอบเขตปัญหา โดยรูปร่างขอบเขตปัญหามีลักษณะเป็นแผ่นสี่เหลี่ยมจัตุรัสและกล่องสี่เหลี่ยมด้านเท่าที่มีความยาวในแต่ละโดเมนขนาด L ซึ่งความยาวโดเมนดังกล่าวถูกแบ่งย่อยด้วยจำนวนจุดต่อ(Grid size) ขนาด n × n และ n × n × n ในปัญหา 2 และ 3 มิติตามลำดับ นอกจากนี้ระเบียบวิธีการทำซ้ำแบบ SOR ที่มีการกำหนดค่าตัวประกอบน้ำหนักให้กับตัวแปรได้ถูกนำมาประยุกต์ใช้ในการแก้ระบบสมการเชิงเส้นของแบบจำลอง โดยระบบสมการดังกล่าวจะลู่เข้าเมื่อผลเฉลยโดยประมาณในรอบทำซ้ำครั้งเก่าและใหม่มีเปอร์เซ็นต์ผลต่างสัมบูรณ์น้อยกว่าค่าความผิดพลาดที่ยอมได้

ในงานวิจัยนี้ประเด็นปัญหาหลักอยู่ที่การประมาณผลเฉลยของฟังก์ชันความแปรปรวนร่วมสำหรับสมการลาปลาส และการประมาณผลเฉลยของฟังก์ชันค่าเฉลี่ยสำหรับสมการลาปลาสเป็นปัญหาที่ได้ถูกศึกษาโดยนักวิจัยจำนวนมาก หากแต่งานวิจัยนี้ได้นำวิธีการประมาณผลเฉลยของฟังก์ชันค่าเฉลี่ยสำหรับสมการลาปลาสมาศึกษาและทดสอบ เพื่อเป็นพื้นฐานในการประยุกต์ใช้เพื่อแก้สมการหาผลเฉลยโดยประมาณของฟังก์ชันความแปรปรวนร่วมสำหรับสมการลาปลาส ซึ่งผลเฉลยโดยประมาณของฟังก์ชันค่าเฉลี่ยและฟังก์ชันความแปรปรวนร่วมของตัวแปรภายในบนรูปร่างขอบเขตปัญหาได้ถูกนำมาเปรียบเทียบกับผลเฉลยแม่นตรง(สมการ (10) (19) (22) และ (31) ที่มีการกำหนดค่าคงที่แล้ว) โดยมีจุดประสงค์เพื่อคำนวณหาความคลาดเคลื่อนที่อยู่ในรูปของเปอร์เซ็นต์ผลต่างสัมบูรณ์ ซึ่งแบบจำลองได้ถูกทดสอบในโปรแกรม MATLAB บนคอมพิวเตอร์ความเร็ว CPU ขนาด 3.40 GHz และ RAM ขนาด 512 MB

ตารางที่ 1 การประมาณฟังก์ชันค่าเฉลี่ยในปัญหา 2 มิติ

ขนาดกริด	จำนวนรอบการทำซ้ำ	เวลาที่ใช้	ความผิดพลาดน้อยที่สุด (%)	ความผิดพลาดสูงสุด (%)	ความผิดพลาดเฉลี่ย (%)
5×5	15	< 1 วินาที	0.0000044111	0.0000072960	0.0000053433
25×25	783	< 1 วินาที	0.0000000208	0.0000011783	0.0000005332
125×125	21,006	1 นาที 4 วินาที	0.0000000096	0.0000010072	0.0000004152
625×625	532,048	10 ชั่วโมง 30 นาที 14 วินาที	0.0000000001	0.0000010004	0.0000004067

ตัวประกอบนำหนัก = 1.2

ค่าความผิดพลาดที่ยอมได้ = 0.000005

L = 0.1

ตารางที่ 2 การประมาณฟังก์ชันค่าเฉลี่ยในปัญหา 3 มิติ

ขนาดกริด	จำนวนรอบการทำซ้ำ	เวลาที่ใช้	ความผิดพลาดน้อยที่สุด (%)	ความผิดพลาดสูงสุด (%)	ความผิดพลาดเฉลี่ย (%)
5×5×5	35	< 1 วินาที	0.08428552	0.36244551	0.21293101
15×15×15	492	3 วินาที	0.00899020	0.46461480	0.20916624
45×45×45	5,108	2 นาที 58 วินาที	0.00097039	0.48787299	0.20740308
135×135×135	49,420	11 ชั่วโมง 2 นาที 56 วินาที	0.00010683	0.49468184	0.20682425

ตัวประกอบนำหนัก = 1.2

ค่าความผิดพลาดที่ยอมได้ = 0.000005

L = 0.1

ตารางที่ 3 การประมาณฟังก์ชันความแปรปรวนร่วมในปัญหา 2 มิติ

ขนาดกริด	จำนวนรอบการทำซ้ำ	เวลาที่ใช้	ความผิดพลาดน้อยที่สุด (%)	ความผิดพลาดสูงสุด (%)	ความผิดพลาดเฉลี่ย (%)
5×5	192	< 1 วินาที	0.085829	0.140679	0.102439
10×10	3,466	10 วินาที	0.029900	0.142977	0.085486
15×15	17,076	8 นาที 20 วินาที	0.015236	0.146810	0.080174
20×20	52,500	55 นาที 04 วินาที	0.009350	0.146406	0.077604
25×25	125,460	6 ชั่วโมง 25 นาที 10 วินาที	0.006376	0.147197	0.076091

ตัวประกอบนำหนัก = 1.2

ค่าความผิดพลาดที่ยอมได้ = 0.0000005

L = 0.1

ตารางที่ 4 การประมาณฟังก์ชันความแปรปรวนร่วมในปัญหา 3 มิติ

ขนาดกริด	จำนวนรอบการทำซ้ำ	เวลาที่ใช้	ความผิดพลาดน้อยที่สุด (%)	ความผิดพลาดสูงสุด (%)	ความผิดพลาดเฉลี่ย (%)
5×5×5	208	14 นาที 2 วินาที	0.080767	0.154495	0.101184
6×6×6	478	1 ชั่วโมง 30 นาที 07 วินาที	0.058842	0.145411	0.094214
7×7×7	936	7 ชั่วโมง 19 นาที 14 วินาที	0.044140	0.161983	0.089174
8×8×8	6,225	31 ชั่วโมง 12 นาที 15 วินาที	0.034128	0.156403	0.085396

ตัวประกอบนำหนัก = 1.2

ค่าความผิดพลาดที่ยอมได้ = 0.000005

L = 0.1

ในกรณีที่รูปร่างขอบเขตปัญหามีขนาดใหญ่คอมพิวเตอร์เครื่องเดียวมีหน่วยความจำไม่เพียงพอที่จะประมวลผลระบบสมการของแบบจำลอง ดังนั้นระเบียบวิธีแบ่งส่วนโดเมนจึงถูกนำมาประยุกต์ใช้ โดยแบ่งรูปร่างขอบเขตของปัญหาออกเป็น 4 โดเมนย่อย ซึ่งผลการทดสอบแบบจำลองกรณีที่ประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมนเข้ากับการประมาณฟังก์ชันค่าเฉลี่ยและฟังก์ชันความแปรปรวนร่วมสามารถแสดงได้ดังตารางที่ 5 – 8 ดังนี้

ตารางที่ 5 การประมาณฟังก์ชันค่าเฉลี่ยในปัญหา 2 มิติ กรณีประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมน

ขนาดกริด	จำนวนรอบการทำซ้ำ	เวลาที่ใช้	ความผิดพลาดน้อยที่สุด (%)	ความผิดพลาดสูงสุด (%)	ความผิดพลาดเฉลี่ย (%)
5×5	23	< 1 วินาที	0.0000045940	0.0000077859	0.0000060290
25×25	1,126	27 วินาที	0.0000001514	0.0000101200	0.0000047960
125×125	28,803	41 นาที 36 วินาที	0.0000000344	0.0000576988	0.0000241157
625×625	35,681	22 ชั่วโมง 55 นาที 16 วินาที	0.0000002043	0.1166937947	0.0468896257

ตัวประกอบนำหนัก= 1.2

ค่าความผิดพลาดที่ยอมได้= 0.000005

L = 0.1

ตารางที่ 6 การประมาณฟังก์ชันค่าเฉลี่ยในปัญหา 3 มิติ กรณีประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมน

ขนาดกริด	จำนวนรอบการทำซ้ำ	เวลาที่ใช้	ความผิดพลาดน้อยที่สุด (%)	ความผิดพลาดสูงสุด (%)	ความผิดพลาดเฉลี่ย (%)
5×5×5	24	< 1 วินาที	0.02651350	0.05145599	0.03625609
15×15×15	420	14 วินาที	0.00321756	0.05576967	0.02551091
45×45×45	4,431	22 นาที 58 วินาที	0.00035592	0.05615801	0.02190013
135×135×135	41,877	87 ชั่วโมง 31 นาที 53 วินาที	0.00003949	0.05620841	0.02074268

ตัวประกอบนำหนัก = 1.2

ค่าความผิดพลาดที่ยอมได้ = 0.000005

L = 0.1

ตารางที่ 7 การประมาณฟังก์ชันความแปรปรวนร่วมในปัญหา 2 มิติ กรณีประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมน

ขนาดกริด	จำนวนรอบการทำซ้ำ	เวลาที่ใช้	ความผิดพลาดน้อยที่สุด (%)	ความผิดพลาดสูงสุด (%)	ความผิดพลาดเฉลี่ย (%)
5×5	148	2 วินาที	0.86022	0.141428	0.102880
10×10	2,446	31 นาที 19 วินาที	0.029923	0.143682	0.085740
15×15	11,860	9 ชั่วโมง 16 นาที 05 วินาที	0.015242	0.147480	0.080370
20×20	36,231	77 ชั่วโมง 43 นาที 07 วินาที	0.009352	0.147063	0.077776

ตัวประกอบนำหนัก = 1.2

ค่าความผิดพลาดที่ยอมได้ = 0.005

L = 0.1

ตารางที่ 8 การประมาณฟังก์ชันความแปรปรวนร่วมในปัญหา 3 มิติ กรณีประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมน

ขนาดกริด	จำนวนรอบการทำซ้ำ	เวลาที่ใช้	ความผิดพลาดน้อยที่สุด (%)	ความผิดพลาดสูงสุด (%)	ความผิดพลาดเฉลี่ย (%)
5×5×5	204	2 ชั่วโมง 10 min 46 วินาที	0.080777	0.154571	0.101218
6×6×6	455	11 ชั่วโมง 22 นาที 06 วินาที	0.058845	0.145462	0.094232
7×7×7	876	47 ชั่วโมง 27 นาที 19 วินาที	0.044141	0.162017	0.089185
8×8×8	1,528	161 ชั่วโมง 34 นาที 21 วินาที	0.034128	0.156430	0.085403

ตัวประกอบนำหนัก = 1.2

ค่าความผิดพลาดที่ยอมได้ = 0.005

L = 0.1

นอกจากนี้เพื่อเปรียบเทียบประสิทธิภาพของแบบจำลองภายใต้จำนวนโดเมนย่อยที่แตกต่างกัน การประมาณฟังก์ชันค่าเฉลี่ยในปัญหา 2 มิติจึงถูกนำมาเป็นปัญหาตัวอย่างในการศึกษา โดยแบ่งรูปร่างขอบเขตปัญหาที่เป็นแผ่นสี่เหลี่ยมจัตุรัสออกเป็นโดเมนย่อยขนาดเท่ากันจำนวน $n \times n$ โดเมนย่อย ซึ่งผลการเปรียบเทียบสามารถแสดงได้ดังตารางที่ 9

ตารางที่ 9 การเปรียบเทียบประสิทธิภาพของการแบ่งโดเมนย่อยในการประมาณฟังก์ชันค่าเฉลี่ยในปัญหา 2 มิติ

จำนวนโดเมนย่อย	ขนาดกริดของโดเมนย่อย	จำนวนรอบการทำซ้ำ	เวลาที่ใช้	ความผิดพลาดน้อยที่สุด (%)	ความผิดพลาดสูงสุด (%)	ความผิดพลาดเฉลี่ย (%)
9	281×281	54,818	59 ชั่วโมง 12 นาที 05 วินาที	0.00000021	0.20700326	0.08477370
16	211×211	51,987	63 ชั่วโมง 08 นาที 44 วินาที	0.00000048	0.24053419	0.09855944
25	169×169	52,612	72 ชั่วโมง 29 นาที 35 วินาที	0.00000001	0.23273895	0.09567423
36	141×141	51,494	81 ชั่วโมง 02 นาที 23 วินาที	0.00000021	0.24693786	0.10167499
49	121×121	51,630	94 ชั่วโมง 25 นาที 11 วินาที	0.00000065	0.24517897	0.10123608

ขนาดกริดของโดเมนหลัก = 841×841

ค่าความผิดพลาดที่ยอมได้ = 0.000005

ตัวประกอบนำหนัก = 1.2

L = 0.1

วิจารณ์

โดยทั่วไปเมื่อความยาวของโดเมนถูกแบ่งย่อยด้วยจำนวนจุดต่อ (Grid size) ที่มากขึ้น ความละเอียดในการคำนวณย่อมมากขึ้น ความคลาดเคลื่อนในการคำนวณต้องมีค่าลดลงตามอันดับความผิดพลาด $O(\Delta h^2)$ โดย h คือ ช่วงของความยาวที่ถูกแบ่งย่อย (Step size) ซึ่งจากตารางที่ 1- 9 จะเห็นได้ว่าแบบจำลองเชิงตัวเลขสามารถประมาณผลเฉลยของสมการลาปลาสได้อย่างมีประสิทธิภาพภายใต้ระเบียบวิธีการทำซ้ำแบบ SOR โดยค่าเปอร์เซ็นต์ความคลาดเคลื่อนสูงสุดอยู่ที่ 0.162017 เปอร์เซ็นต์ ในตารางที่ 8 ภายใต้จำนวนจุดต่อขนาด 7×7×7 นอกจากนี้ความแม่นยำของแบบจำลองในการประมาณผลเฉลยในตารางที่ 1- 9 ยังขึ้นกับค่าความผิดพลาดที่ยอมได้ โดยค่าความผิดพลาดที่ยอมได้ยิ่งน้อย ค่าความคลาดเคลื่อนในการประมาณของแบบจำลองยิ่งน้อยลงตามไปด้วย แต่ในอีกแง่มุมหนึ่งค่าความผิดพลาดที่ยอมได้ที่มีค่าน้อยดังกล่าวกลับทำให้คอมพิวเตอร์ต้องวนรอบทำซ้ำแก้ระบบสมการมากรอบขึ้นด้วยเช่นกัน

เนื่องจากในแบบจำลองเชิงตัวเลขที่ใช้ประมาณฟังก์ชันความแปรปรวนร่วมในปัญหา 2 และ 3 มิติ รูปร่างขอบเขตของปัญหาถูกแบ่งย่อยด้วยจำนวนจุดต่อขนาด $n \times n$ และ $n \times n \times n$ ตามลำดับ ด้วยเหตุนี้การแก้ระบบสมการเชิงเส้นต้องเกี่ยวข้องกับตัวแปรความแปรปรวนร่วมในระบบสมการจำนวน n^4 และ n^6 ตัวแปรในปัญหา 2 และ 3 มิติ ตามลำดับ ดังนั้นแบบจำลองดังกล่าวจึงสามารถทดสอบได้เฉพาะในกรณีที่ปัญหามีจำนวนจุดต่อน้อยเท่านั้น โดยเฉพาะในปัญหา 3 มิติ ดังจะเห็นได้จากตารางที่ 4 และ 8 โดยจำนวนจุดต่อที่มากที่สุดที่คอมพิวเตอร์หนึ่งเครื่องสามารถรองรับได้ คือ $8 \times 8 \times 8$ หรือ คิดเป็นตัวแปรในระบบสมการเชิงเส้นจำนวน 10^6 ตัวแปร ยิ่งไปกว่านั้นเมื่อประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมนเข้ากับปัญหาข้างต้น คอมพิวเตอร์ยิ่งต้องใช้เวลาในการประมวลผลมากขึ้น ทั้งนี้เนื่องจากระเบียบวิธีแบ่งส่วนโดเมนแบบโดเมนย่อยซ้อนทับกันที่ใช้ในงานวิจัยนี้ต้องมีการปรับปรุงค่าตัวแปรทุกตัวของแต่ละโดเมนย่อยในแต่ละรอบทำซ้ำ และระเบียบวิธีการทำซ้ำ SOR จะหยุดการคำนวณเมื่อตัวแปรดังกล่าวทุกตัวลู่เข้า จากสาเหตุที่กล่าวมาข้างต้นถึงแม้แบบจำลองสามารถประมาณค่าผลเฉลยได้แม่นยำเนื่องจากมีการปรับปรุงค่าข้อมูลทุกตัวในทุกโดเมนย่อยในแต่ละรอบทำซ้ำ แต่ในอีกแง่มุมหนึ่งระเบียบวิธีแบ่งส่วนโดเมนดังกล่าวมีผลทำให้คอมพิวเตอร์ต้องใช้เวลาประมวลผลนานด้วยเช่นกัน

จากผลการคำนวณในตารางที่ 5-8 จะเห็นได้ว่าแบบจำลองกรณีประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมน ใช้เวลาในการคำนวณเพิ่มขึ้นจากแบบจำลองกรณีที่ยังไม่มีการประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมน (ผลการคำนวณในตารางที่ 1-4) ซึ่งเวลาที่เพิ่มขึ้นดังกล่าวเป็นผลมาจากการที่คอมพิวเตอร์ต้องใช้เวลาส่วนหนึ่งในการสื่อสารหรือส่งต่อข้อมูลระหว่างโดเมนย่อยด้วยกันเอง อย่างไรก็ตามถ้าแบบจำลองที่ประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมนถูกนำไปประมวลผลบนระบบคอมพิวเตอร์แบบขนานจริง โดยใช้คอมพิวเตอร์1 เครื่องประมวลผลระบบสมการของ 1 โดเมนย่อย (การประมาณค่าฟังก์ชันความแปรปรวนร่วม) ซึ่งจากการทดสอบจับเวลาการประมวลของการประมาณฟังก์ชันความแปรปรวนร่วมที่จำนวนจุดต่อขนาด 15×15 พบว่าในการคำนวณหาค่าความแปรปรวนร่วมระหว่างตัวแปรบนคู่ของโดเมนย่อย 1 คู่ คอมพิวเตอร์ต้องใช้เวลาคำนวณเท่ากับ 19 นาที 22 วินาที / จำนวนรอบการทำซ้ำ หรือคอมพิวเตอร์ใช้เวลาในการประมวลผลทั้งหมด 37 นาที 59 วินาที ซึ่งคล้อยตามผลทดสอบในตารางที่ 7 นอกจากนี้ในกรณีที่จำนวนโดเมนย่อยมีจำนวนมากขึ้นภายใต้รูปร่างขอบเขตปัญหาขนาดเท่าเดิม ดังผลการทดสอบในตารางที่ 9 จะเห็นได้ว่าความคลาดเคลื่อนที่แสดงในรูปของเปอร์เซ็นต์ผลต่างสัมบูรณ์ไม่จำเป็นต้องมีค่าลดลงเสมอไปเมื่อจำนวนโดเมนย่อยเพิ่มขึ้น แต่เป็นธรรมดาที่เวลารวมของการคำนวณมากขึ้นเมื่อจำนวนโดเมนย่อยมากขึ้น ทั้งนี้เนื่องจากการที่โดเมนย่อยมีจำนวนมากขึ้นคอมพิวเตอร์ย่อมต้องใช้เวลาในการสื่อสารหรือส่งต่อข้อมูลระหว่างโดเมนย่อยด้วยกันเองมากขึ้นด้วยเช่นกัน ดังนั้นระเบียบวิธีแบ่งส่วนโดเมนในงานวิจัยนี้จึงแบ่งย่อยรูปร่างขอบเขตปัญหาออกเป็น 4 โดเมนย่อยเท่านั้น

สรุปและข้อเสนอแนะ

สรุป

ในงานวิจัยนี้แบบจำลองเชิงตัวเลขและระเบียบวิธีที่ใช้ในการแก้หาผลเฉลยโดยประมาณของสมการลาปลาสภายใต้เงื่อนไขค่าขอบที่เป็นค่าสุ่มและมีความสัมพันธ์ร่วมได้ถูกนำมาศึกษา โดยระเบียบวิธีดังกล่าวเป็นวิธีที่มีประสิทธิผลหลังถูกทดสอบกับผลเฉลยแม่นตรงที่สร้างขึ้นจากระเบียบวิธีฟูเรียร์ อย่างไรก็ตามในกรณีที่ปัญหามีความซับซ้อนขึ้นโดยเฉพาะในปัญหา 3 มิติ แบบจำลองเชิงตัวเลขสามารถประยุกต์ใช้ได้กับปัญหาขนาดเล็กที่มีจำนวนจุดต่อน้อยเท่านั้น โดยแบบจำลองที่ถูกประมวลผลบนคอมพิวเตอร์เครื่องเดียวดังกล่าวสามารถแก้ระบบสมการเชิงเส้นที่ประกอบไปด้วยตัวแปรสูงสุดจำนวน 10^6 ตัวแปร และให้ผลการคำนวณคลาดเคลื่อนสูงสุด 0.162017 เปอร์เซ็นต์ นอกจากนี้ระเบียบวิธีแบ่งส่วนโดเมนสามารถนำมาประยุกต์ใช้ร่วมกับการประมาณผลเฉลยสมการลาปลาสได้อย่างแม่นยำ และระเบียบวิธีแบ่งส่วนโดเมนที่ประมวลผลบนคอมพิวเตอร์เครื่องเดียวในงานวิจัยนี้ เป็นเพียงการทดสอบเพื่อนำไปสู่การประยุกต์ใช้จริงบนระบบคอมพิวเตอร์แบบขนาน อย่างไรก็ตามโปรแกรมสำหรับประมาณผลเฉลยของฟังก์ชันความแปรปรวนร่วมสำหรับสมการลาปลาสที่ถูกสร้างขึ้นบนโปรแกรม Matlab ยังต้องประสบกับปัญหาในการใช้งานจริง นั่นคือ การป้อนเงื่อนไขค่าขอบแบบสุ่มและมีความสัมพันธ์ร่วมมีความยุ่งยากและต้องใช้เวลานาน

ข้อเสนอแนะ

ระเบียบวิธีแบ่งส่วนโดเมนแบบโดเมนย่อยซ้อนทับกัน (Overlapping domain) ที่ใช้ในการประมาณฟังก์ชันความแปรปรวนร่วมทำให้คอมพิวเตอร์สิ้นเปลืองทรัพยากรในการคำนวณ ทั้งนี้เพราะระเบียบวิธีดังกล่าวต้องประกอบคู่โดเมนย่อยเข้าด้วยกันเพื่อปรับปรุงค่าตัวแปรให้ใหม่ ดังนั้นระเบียบวิธีแบ่งส่วนโดเมนแบบโดเมนไม่ซ้อนทับกัน (Non-Overlapping domain) ควรถูกนำมาทดสอบเปรียบเทียบกัน นอกจากนี้วิธีการประมาณความแปรปรวนร่วมอันดับสองด้วยแบบจำลองเชิงตัวเลขในงานวิจัยนี้ สามารถนำไปขยายผลเพื่อประยุกต์ใช้ในการประมาณค่าความแปรปรวนร่วมอันดับสูงกว่าสองได้

เอกสารและสิ่งอ้างอิง

ปราโมทย์ เดชะอำไพ. 2544. **ระเบียบวิธีเชิงตัวเลขในงานวิศวกรรม.** พิมพ์ครั้งที่ 3. สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, กรุงเทพฯ.

Brown, J.W. and R.V. Churchill. 2001. **Fourier Series and Boundary Value Problems.** 6th ed. McGraw-Hill , Inc. New York.

Chow, T.L. 2000. **Mathematical Methods for Physicists.** 1st ed. Cambridge University Press. New York.

Charnsethikul, P. 2004. Solving (n×n) Linear Equations with Parameters Covariances. **International Journal of Computational Methods**, 1(3):471-489.

Crank, J. and P. Nicolson. 1947. A Practical Method for Numerical Evaluation of Solutions of P.D.E. of the Heat Conduction Type, **Proc. Cambridge Phil. Soc.**, 43: 50-67.

Erhart, K., E. Divo and A. J. Kassab. 2006. A Parallel Domain Decomposition Boundary Element Method Approach for the Solution of Large-Scale Transient Heat Conduction Problems. **Engineering Analysis with Boundary Elements** 30:553-563

Glimsdal, S., G.K. Pedersen and H.P. Langtangen. 2004. An Investigation of Overlapping Domain Decomposition Methods for One-Dimensional Dispersive Long Wave Equations. **Advances in Water Resources** (27): 1111-1133.

Kamiya, N., H. Iwase and E. Kita. 1996. Parallel Implementation of Boundary Element Method with Domain Decomposition. **Engineering Analysis with Boundary Elements** 18:209-216

Kamiya, N., H. Iwase and E. Kita. 1996. Performance evaluation of parallel boundary element analysis by domain decomposition method. **Engineering Analysis with Boundary Elements** 18:217-222

Kao, E. P.C. 1997. **An Introduction to Stochastic Processes**. Duxbury Press, ITP.

Karniadakis, G. 2002. Quantifying Uncertainty in CFD. **Journal of Fluids Engineering-Transactions of the ASME**, 124(1): 2-3.

Min, T. and D. Yang. 2006. Parallel Finite Difference Schemes for Heat Equation Based upon Overlapping Domain Decomposition. **Applied Mathematics and Computation** (186): 1276-1292.

Prigarin, S. M. and G. Winkler. 2002. Numerical Solution of Boundary Value Problems for Stochastic Differential Equations on the Basis of the Gibbs Sampler, Working Paper, **ibb.gsf.de/preprints/2002/pp02-27.pdf**

Sadd, Y. 1996. **Iterative Methods for Sparse Linearsystems.** PSW Publishing company. Boston

Xia, N. M. 1986. Linear Random Boundary Value Problems containing Weakly Correlated Forcing Function, **International Journal of Mathematics and Mathematical Sciences**, 9(3):497-516

ภาคผนวก

โปรแกรมประมาณผลเฉลยของฟังก์ชันความแปรปรวนร่วมสำหรับสมการลาปลาส

ในงานวิจัยนี้ฟังก์ชันความแปรปรวนร่วมสำหรับสมการลาปลาสถูกสร้างขึ้นจากระเบียบวิธีฟูเรียร์ และฟังก์ชันดังกล่าวถูกประมาณโดยแบบจำลองเชิงตัวเลขทั้งในปัญหา 2 มิติ และ 3 มิติ อีกทั้งในกรณีที่แบบจำลองมีขนาดใหญ่ ระเบียบวิธีแบ่งส่วนโดเมนแบบโดเมนย่อยซ้อนทับกันได้ถูกนำมาประยุกต์ใช้ในการประมาณฟังก์ชันดังกล่าวทั้งในปัญหา 2 มิติ และ 3 มิติ ดังนั้นโปรแกรมสำหรับประมาณฟังก์ชันความแปรปรวนร่วมทั้ง 4 โปรแกรมข้างต้นจึงสามารถแสดงได้ดังนี้

1. โปรแกรมประมาณฟังก์ชันความแปรปรวนร่วมสำหรับสมการลาปลาสในปัญหา 2 มิติ

```
function cov_approx=covariance(qqqq)

tic
start=datestr(now);          % Marking starting time of calculation
% -------------------------INPUT VARIABLE ------------------------
% Setting any constant in exact solution
c1=1; c2=1; c3=1; c4=1; c5=1; c6=1; c7=1; c8=1; k1=2; k2=1; k3=1; k4=k1-k3;

%Setting number of grids to form nxn array
n=27;
% Setting Lenght of x and y axis
% x1=0,...,(j-3)*delta,...,L          y1=0,...,(i-3)*delta,...,L
% x2=0,...,(b-3)*delta,...,L          y2-0,...,(a-3)*delta,...,L

L=0.1;
stt=0.0000005;               % Stopping tolerance of Relaxation method
delta=L/(n-3);
% Setting weighting-factor of Relaxation method
WF=1.2;     % 0 < WF < 2
exactV=inline('( c1*exp(sqrt(k2)*x1)+ c2*exp(-sqrt(k2)*x1) )*
```

```
                ( c3*sin(sqrt(k2)*y1) + c4*cos(sqrt(k2)*y1) )*
                 (c5*exp(sqrt(k4)*x2)+c6*exp(-sqrt(k4)*x2))*
                ( c7*exp(sqrt(k3)*y2) + c8*exp(-sqrt(k3)*y2) )',
                  'x1','y1','x2','y2','c1','c2','c3','c4','c5','c6','c7','c8','k1','k2','k3','k4');

% To find covariance at boundary points, using exact solution is required
for i=1:n
   for j=1:n
      for a=1:n
         for b=1:n
           x1=(j-2)*delta;     y1=(i-2)*delta;     x2=(b-2)*delta;     y2=(a-2)*delta;

         %Setting innitial guessing value for covariance of inner grids
             cov(i,j,a,b)=10;
         if i==1|j==1|a==1|b==1|i==n|j==n|a==n|b==n
             cov(i,j,a,b)=0;

          % Setting boundary conditions by using exat solution
         elseif (j==2&b==2)                                  % 1) Left&Left
              cov(i,j,a,b) =exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
         elseif (j==2&a==n-1)|(i==n-1&b==2)                  % 2) Left&Lower or Lower&Left
              cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
         elseif (j==2&b==n-1)|(j==n-1&b==2)                  % 3) Left&Right or Right&Left
              cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
         elseif (j==2&a==2)|(i==2&b==2)                      % 4) Left&Upper or Upper&Left
               cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
         elseif (i==n-1&a==n-1)                              % 5) Lower&Lower
              cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);

         elseif (i==n-1&b==n-1)|(j==n-1&a==n-1)              % 6) Lower&Right or Right&Lower
```

```
            cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
        elseif (i==n-1&a==2)|(i==2&a==n-1)          % 7) Lower&Upper or Upper&Lower
            cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
        elseif (j==n-1&b==n-1)                      % 8) Right&Right
            cov(i,j,a,b) =exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
        elseif (j==n-1&a==2)|(i==2&b==n-1)          % 9) Right&Upper or Upper&Right
            cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
        elseif (i==2&a==2)                          % 10) Upper&Upper
            cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
        end
      end
    end
  end
end

% To generate symmetry covariance at boundaries of rectangular box
 for i=1:n
   for j=1:n
     for a=1:n
       for b=1:n
         if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
           (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|(j==n-1&a==n-1)|
           (i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|(j==n-1&a==2)|(i==2&b==n-1)|
           (i==2&a==2)
           cov(i,j,a,b)=(cov(i,j,a,b)+cov(a,b,i,j))/2;
           cov(a,b,i,j)=cov(i,j,a,b);
         end
       end
     end
   end
```

```
 end

% Ussing Successive over-relaxation method to find covariance of inner grids
err=1+stt;
iteration=0;

while err>stt
   err=0;
     for i=2:n-1
        for j=2:n-1
           for a=2:n-1
              for b=2:n-1

     cov_new=( cov(i+1,j,a+1,b) + cov(i+1,j,a-1,b) + cov(i+1,j,a,b+1) + cov(i+1,j,a,b-1) +
                  cov(i-1,j,a+1,b) + cov(i-1,j,a-1,b) + cov(i-1,j,a,b+1) + cov(i-1,j,a,b-1) +
                  cov(i,j+1,a+1,b) + cov(i,j+1,a-1,b) + cov(i,j+1,a,b+1) + cov(i,j+1,a,b-1) +
                  cov(i,j-1,a+1,b) + cov(i,j-1,a-1,b) + cov(i,j-1,a,b+1) + cov(i,j-1,a,b-1) –
                  4*cov(i+1,j,a,b) -4*cov(i-1,j,a,b) -4*cov(i,j+1,a,b) -4*cov(i,j-1,a,b) –
                  4*cov(i,j,a+1,b)-4*cov(i,j,a-1,b) -4*cov(i,j,a,b+1)-4*cov(i,j,a,b-1) )/(-16.0);

    % Unchange boundary conditions
     if   (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
        (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|(j==n-1&a==n-1)|
        (i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|(j==n-1&a==2)|(i==2&b==n-1)|
        (i==2&a==2)
         cov_new=cov(i,j,a,b);
      end

    err=((abs(cov_new-cov(i,j,a,b))*100)/cov_new )+err;
    cov(i,j,a,b)=WF*cov_new+(1-WF)*cov(i,j,a,b);
```

```
            end
          end
        end
      end
      iteration=iteration+1;
end

stop=datestr(now);          % Marking ending time of calculation
t1=toc;
tot=sec2hms(t1);

% ----------------------FINDNIG AN ERROR OF THE MODEL -----------------------------
% EXACTSOLUTION
for i=1:n
      for j=1:n
        for a=1:n
          for b=1:n
          x1=(j-2)*delta;    y1=(i-2)*delta;    x2=(b-2)*delta;    y2=(a-2)*delta;
          cov_exact(i,j,a,b) =exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
          end
        end
      end
end

%  To generate symmetry covariance of an exact solution
 for i=1:n
    for j=1:n
      for a=1:n
        for b=1:n
```

```
                cov_exact(i,j,a,b)=(cov_exact(i,j,a,b)+cov_exact(a,b,i,j))/2;
                cov_exact(a,b,i,j)=cov_exact(i,j,a,b);
            end
        end
    end
 end

app=cov(3:n-2,3:n-2,3:n-2,3:n-2);
exact=cov_exact(3:n-2,3:n-2,3:n-2,3:n-2);
abs_error=(abs(app-exact)*100)./exact;
 error=reshape(abs_error,1,(n-4)^4);
min_error=min(error);
max_error=max(error);
average_error=mean(error);

% ------------------------To save an output into file cov_app2D.txt-----------------------------------
fid=fopen('cov_app2D.txt','a');
fprintf(fid,'-------------------------------------------------------------------------\n');
fprintf(fid,'n= %-5.2f\n',n);
fprintf(fid,'Stopping Tolerance= %-2.10f\n',stt);
fprintf(fid,'Iteration= %-5.2f\n',iteration);
fprintf(fid,'\n');

 % Showing solution
for i=[3,n-2]
   for j=[3,n-2]
      for a=[3,n-2]
         for b=[3,n-2]
            x1=(j-2)*delta;    y1=(i-2)*delta;    x2=(b-2)*delta;    y2=(a-2)*delta;
```

```
            fprintf(fid,'cov[t1(%-2.8f, %-2.8f), t2(%-2.8f, %-2.8f)]= %-3.5f\n',
                                                                    x1,y1,x2,y2,cov(i,j,a,b));
          end
        end
      end
end

fprintf(fid,'Starting time:\n');
              fprintf(fid,'%s\n',start);
fprintf(fid,'Ending time:\n');
              fprintf(fid,'%s\n',stop);
fprintf(fid,'-----------------------------\n');
fprintf(fid,'\n');
fprintf(fid,'Min Error:   %-3.6f  percent\n',min_error);
fprintf(fid,'Max Error:   %-3.6f  percent\n',max_error);
fprintf(fid,'Avg Error:   %-3.6f  percent\n',average_error);
fprintf(fid,'Total time:  %f  \n',tot);
fprintf(fid,'-------------------------------------------------------------------------\n');
fclose(fid)
end
```

2. โปรแกรมประมาณฟังก์ชันความแปรปรวนร่วมสำหรับสมการลาปลาสในปัญหา 3 มิติ

```
function cov_approx_3D=cov_matrix(a)

% From Laplace's equation in 3D an exact solution can be shown in the
% following form
%   cov[t1(x1,y1,z1), t2(x2,y2,z2)] =        ( c1*exp(sqrt(k2)*x1)+ c2*exp(-sqrt(k2)*x1) )*
                                             ( c3*exp(sqrt(k3)*y1) + c4*exp(-sqrt(k3)*y1) ) *
                                             ( c5*exp(sqrt(k4)*z1) + c6*exp(-sqrt(k4)*z1) )*
                                             ( c7*sin(sqrt(k5)*x2) + c8*cos(sqrt(k5)*x2) )*
                                             ( c9*exp(sqrt(k6)*y2) + c10*exp(-sqrt(k6)*y2) )*
                                             ( c11*exp(sqrt(k7)*z2) + c12*exp(-sqrt(k7)*z2) );
tic
start=datestr(now);          % Marking starting time of calculation

% -------------------------INPUT VARIABLE -----------------------
% Setting any constant in exact solution
c1=1; c2=1; c3=1; c4=1; c5=1; c6=1; c7=1; c8=1; c9=1; c10=1; c11=1; c12=1;
k1=3; k2=1; k3=1; k4=k1-k2-k3; k5=2; k6=1; k7=k5-k6; % where k1>k2+k3 and k5>k6

%Setting number of grids to form n x n array
 n=7;

% Setting Lenght of x, y and z axis
% x1=0,...,(j-2)*delta,...,L    y1=0,...,(i-2)*delta,...,L    z1=0,...,(k-2)*delta,...,L
% x2=0,...,(b-2)*delta,...,L    y2=0,...,(a-2)*delta,...,L    z2=0,...,(c-2)*delta,...,L

L=0.1;               L is length of domain in x1, y1, z1, x2, y2 and z2 direction
delta=L/(n-3);
stt=500;             % Stopping tolerance of Relaxation method
```

```
% Setting weighting-factor of Relaxation method
WF=1.2;          % 0 < WF < 2

exactV=inline('  ( c1*exp(sqrt(k2)*x1)+ c2*exp(-sqrt(k2)*x1) )*
                 ( c3*exp(sqrt(k3)*y1) + c4*exp(-sqrt(k3)*y1) ) *
                 ( c5*exp(sqrt(k4)*z1) + c6*exp(-sqrt(k4)*z1) )*
                 ( c7*sin(sqrt(k5)*x2) + c8*cos(sqrt(k5)*x2) )*
                 ( c9*exp(sqrt(k6)*y2) + c10*exp(-sqrt(k6)*y2) )*
                 ( c11*exp(sqrt(k7)*z2) + c12*exp(-sqrt(k7)*z2) )',
                 'x1','y1','z1','x2','y2','z2','c1','c2','c3','c4','c5','c6','c7','c8','c9','c10','c11',
                 'c12','k1','k2','k3','k4','k5','k6','k7');

% Using exact solution in finding boundary conditions
 for i=1:n
   for j=1:n
      for k=1:n
         for a=1:n
            for b=1:n
               for c=1:n
            x1=(j-2)*delta;          y1=(i-2)*delta;          z1=(k-2)*delta;
            x2=(b-2)*delta;          y2=(a-2)*delta;          z2=(c-2)*delta;

            % Setting innitial guess value for inner grids
         cov(i,j,k,a,b,c)=100;


         if i==1|j==1|k==1|a==1|b==1|c==1|i==n|j==n|k==n|a==n|b==n|c==n
            cov(i,j,k,a,b,c)=0;
```

```
   % Setting boundary conditions from exat solution
elseif (j==2&b==2)                          % 1) left&left
   cov(i,j,k,a,b,c) =   exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (j==2&b==n-1)|(j==n-1&b==2)        % 2) left&right or right&left
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (j==2&a==n-1)|(i==n-1&b==2)        % 3) left&front or front&left
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (j==2&a==2)|(i==2&b==2)            % 4) left&rear or rear&left
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (j==2&c==n-1)|(k==n-1&b==2)        % 5) left&upper or upper&left
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (j==2&c==2)|(k==2&b==2)            % 6) left&lower or lower&left
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);



elseif (j==n-1&b==n-1)                      % 7) right&right
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);
```

```
elseif (j==n-1&a==n-1)|(i==n-1&b==n-1)  % 8) right&front or front&right
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (j==n-1&a==2)|(i==2&b==n-1)        % 9) right&rear or rear&right
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (j==n-1&c==n-1)|(k==n-1&b==n-1)  % 10) right&upper or upper&right
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (j==n-1&c==2)|(k==2&b==n-1)        % 11) right&lower or lower&right
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (i==n-1&a==n-1)                    % 12) front&front
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (i==n-1&a==2)|(i==2&a==n-1)      % 13) front&rear or rear&front
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);


elseif (i==n-1&c==n-1)|(k==n-1&a==n-1)  % 14) front&upper or upper&front
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);
```

```
elseif (i==n-1&c==2)|(k==2&a==n-1)      % 15) front&lower or lower&front
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (i==2&a==2)                      % 16) rear&rear
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (i==2&c==n-1)|(k==n-1&a==2)      % 17) rear&upper or upper&rear
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (i==2&c==2)|(k==2&a==2)          % 18) rear&lower or lower&rear
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                       k1,k2,k3,k4,k5,k6,k7);

elseif (k==n-1&c==n-1)                  % 19) upper&upper
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);

elseif (k==n-1&c==2)|(k==2&c==n-1)      % 20) upper&lower or lower&upper
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);


elseif (k==2&c==2)                      % 21) lower&lower
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                        k1,k2,k3,k4,k5,k6,k7);
```

```
        end
            end
          end
        end
      end
  end
end

% To generate symmetry covariance at boundaries of rectangular box
for i=1:n
   for j=1:n
      for k=1:n
         for a=1:n
            for b=1:n
               for c=1:n
                  if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|
                    (i==n-1&b==2)|(j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|
                    (k==n-1&b==2)|(j==2&c==2)|(k==2&b==2)|(j==n-1&b==n-1)|
                    (j==n-1&a==n-1)|(i==n-1&b==n-1)|(j==n-1&a==2)|(i==2&b==n-1)|
                    (j==n-1&c==n-1)|(k==n-1&b==n-1)|(j==n-1&c==2)|(k==2&b==n-1)|
                    (i==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(i==n-1&c==n-1)|
                    (k==n-1&a==n-1)|(i==n-1&c==2)|(k==2&a==n-1)|(i==2&a==2)|
                    (i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|(k==2&a==2)|
                    (k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
                     cov(i,j,k,a,b,c)=( cov(i,j,k,a,b,c)+ cov(a,b,c,i,j,k) )/2.0;
                     cov(a,b,c,i,j,k)=cov(i,j,k,a,b,c);
                  end
               end
            end
         end
```

```
        end
    end
end

% Ussing Successive over-relaxation method to find covariance of inner grids
err=1+stt;
iteration=0;

while err>stt
err=0;
for i=2:n-1
    for j=2:n-1
        for k=2:n-1
            for a=2:n-1
                for b=2:n-1
                    for c=2:n-1

 cov0=( cov(i+1,j,k,a+1,b,c)+ cov(i-1,j,k,a+1,b,c)+ cov(i+1,j,k,a-1,b,c)+ cov(i-1,j,k,a-1,b,c)+
            cov(i+1,j,k,a,b+1,c)+ cov(i-1,j,k,a,b+1,c)+ cov(i+1,j,k,a,b-1,c)+ cov(i-1,j,k,a,b-1,c) +
            cov(i+1,j,k,a,b,c+1)+ cov(i-1,j,k,a,b,c+1)+ cov(i+1,j,k,a,b,c-1)+ cov(i-1,j,k,a,b,c-1)+
            cov(i,j+1,k,a+1,b,c)+ cov(i,j-1,k,a+1,b,c)+ cov(i,j+1,k,a-1,b,c)+ cov(i,j-1,k,a-1,b,c) +
            cov(i,j+1,k,a,b+1,c)+ cov(i,j-1,k,a,b+1,c)+ cov(i,j+1,k,a,b-1,c)+ cov(i,j-1,k,a,b-1,c)+
            cov(i,j+1,k,a,b,c+1)+ cov(i,j-1,k,a,b,c+1)+ cov(i,j+1,k,a,b,c-1)+ cov(i,j-1,k,a,b,c-1)+
            cov(i,j,k+1,a+1,b,c)+ cov(i,j,k-1,a+1,b,c)+ cov(i,j,k+1,a-1,b,c)+ cov(i,j,k-1,a-1,b,c)+
            cov(i,j,k+1,a,b+1,c)+ cov(i,j,k-1,a,b+1,c)+ cov(i,j,k+1,a,b-1,c)+ cov(i,j,k-1,a,b-1,c)+
            cov(i,j,k+1,a,b,c+1)+ cov(i,j,k-1,a,b,c+1)+ cov(i,j,k+1,a,b,c-1)+ cov(i,j,k-1,a,b,c-1)-
            6*cov(i+1,j,k,a,b,c) -6*cov(i-1,j,k,a,b,c) -6*cov(i,j+1,k,a,b,c) -6*cov(i,j-1,k,a,b,c)-
            6*cov(i,j,k+1,a,b,c) -6*cov(i,j,k-1,a,b,c) -6*cov(i,j,k,a+1,b,c) -6*cov(i,j,k,a-1,b,c)-
            6*cov(i,j,k,a,b+1,c) -6*cov(i,j,k,a,b-1,c) -6*cov(i,j,k,a,b,c+1) -6*cov(i,j,k,a,b,c-1) )/(-36);
```

```
            % Unchange boundary conditions
      if    (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
            (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
            (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|(j==n-1&a==2)|
            (i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|(j==n-1&c==2)|(k==2&b==n-1)|
            (i==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|
            (i==n-1&c==2)|(k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|
            (i==2&c==2)|(k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|
            (k==2&c==2)
          cov0=cov(i,j,k,a,b,c);
      end

          err=( (abs(cov0-cov(i,j,k,a,b,c))*100)/cov0 )+err;
          cov(i,j,k,a,b,c)=WF*cov0+(1-WF)*cov(i,j,k,a,b,c);

               end
            end
          end
       end
    end
end

iteration=iteration+1;
end



stop=datestr(now);          % Marking ending time of calculation
t1=toc;
tot=sec2hms(t1);
```

```
% --------------------FINDNIG AN ERROR OF THE MODEL ----------------------------------
% EXACTSOLUTION
for i=1:n
      for j=1:n
         for k=1:n
            for a=1:n
               for b=1:n
                  for c=1:n
            x1=(j-2)*delta;    y1=(i-2)*delta;    z1=(k-2)*delta;
            x2=(b-2)*delta;    y2=(a-2)*delta;    z2=(c-2)*delta;

            cov_exact(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,
                                          c10,c11,c12,k1,k2,k3,k4,k5,k6,k7);
                  end
               end
            end
         end
      end
end

% To generate symmetry covariance of exact solution
for i=1:n
   for j=1:n
      for k=1:n
         for a=1:n
            for b=1:n
               for c=1:n
                     cov_exact(i,j,k,a,b,c)=( cov_exact(i,j,k,a,b,c)+ cov_exact(a,b,c,i,j,k) )/2.0;
                     cov_exact(a,b,c,i,j,k)=cov_exact(i,j,k,a,b,c);
```

```
                end
            end
        end
    end
  end
end

app=cov(3:n-2,3:n-2,3:n-2,3:n-2,3:n-2,3:n-2);
exact=cov_exact(3:n-2,3:n-2,3:n-2,3:n-2,3:n-2,3:n-2);
aps_error=(abs(app-exact)*100)./exact;
error=reshape(aps_error,1,(n-4)^6);
min_error=min(error);
max_error=max(error);
average_error=mean(error);

% -------------------------To save an output into file cov_app_3D.txt ------------------------------
fid=fopen('cov_app_3D.txt','a');
fprintf(fid,'\n');
fprintf(fid,'n= %-5.2f\n',n);
fprintf(fid,'Stopping Tolerance= %2.10f\n',stt);
fprintf(fid,'Iteration= %-5.2f\n',iteration);
fprintf(fid,'\n');

 % Showing solution
for i=[3,n-2]
   for j=[3,n-2]
      for k=[3,n-2]
         for a=[3,n-2]
            for b=[3,n-2]
               for c=[3,n-2]
```

```
                    x1=(j-2)*delta;     y1=(i-2)*delta;     z1=(k-2)*delta;
                    x2=(b-2)*delta;    y2=(a-2)*delta;     z2=(c-2)*delta;
              fprintf(fid,'cov[t1(%-2.8f, %-2.8f, %-2.8f), t2(%-2.8f, %-2.8f, %-2.8f)]=
                                  %-3.5f\n',x1,y1,z1,x2,y2,z2,cov(i,j,k,a,b,c));
                end
              end
            end
          end
        end
end

fprintf(fid,'-----------------------------\n');
fprintf(fid,'\n');
fprintf(fid,'Starting time:\n');
                fprintf(fid,'%s\n',start);
fprintf(fid,'Ending time:\n');
                fprintf(fid,'%s\n',stop);
fprintf(fid,'-----------------------------\n');
fprintf(fid,'\n');
fprintf(fid,'Min Error:   %-3.6f  percent\n',min_error);
fprintf(fid,'Max Error:   %-3.6f  percent\n',max_error);
fprintf(fid,'Avg Error:   %-3.6f  percent\n',average_error);
fprintf(fid,'Total time:  %f',tot);
fprintf(fid,'\n-------------------------------------------------------------------');
fprintf(fid,'\n-------------------------------------------------------------------');

fclose(fid)
```

3. โปรแกรมประมาณฟังก์ชันความแปรปรวนร่วมสำหรับสมการลาปลาสในปัญหา 2 มิติ (กรณีประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมน)

```
function cov_approx=cov_dom_app(xxxx)

% An exact solution of Laplace's equation in 2-D can be shown in the following form
% cov(i,j,a,b) =  ( c1*exp(sqrt(k2)*x1) + c2*exp(-sqrt(k2)*x1) )*
%                 ( c3*sin(sqrt(k2)*y1) + c4*cos(sqrt(k2)*y1) )*
%                  ( c5*exp(sqrt(k4)*x2) + c6*exp(-sqrt(k4)*x2))*
%                  ( c7*exp(sqrt(k3)*y2) + c8*exp(-sqrt(k3)*y2) );

 start=datestr(now);        % Marking starting time of calculation
 tic

% ------------------------ INPUT VARIABLES --------------------------------
% Setting any constants in exact solution
c1=1; c2=1; c3=1; c4=1; c5=1; c6=1; c7=1; c8=1; k1=2; k2=1; k3=1; k4=k1-k3;

%Setting n-by-n array of whole domain
n=7;                % In order to equalise grid number in each sub-domain,n must be odd number
N=ceil(n/2);        % Setting NxN array of each sub-domain

% Setting length(L) of X and Y axis of whole domain
L=0.1;
LD=L/2;             % Setting length(LD)of X and Y axis of each sub-domain
delta=L/(n-3);
stt=0.0000005;    % Setting stopping tolerance

% Setting weighting-factor of Relaxation method
WF=1.2;             % 0 < WF < 2
```

```
 % ------------------- FINDING COVARIANCE IN EACH SUB-DOMAIN ---------------
exactV=   inline ('( c1*exp(sqrt(k2)*x1)+ c2*exp(-sqrt(k2)*x1) )*
                    ( c3*sin(sqrt(k2)*y1) + c4*cos(sqrt(k2)*y1) )*
                    (c5*exp(sqrt(k4)*x2)+c6*exp(-sqrt(k4)*x2))*
                    ( c7*exp(sqrt(k3)*y2) + c8*exp(-sqrt(k3)*y2) )',
                    'x1','y1','x2','y2','c1','c2','c3','c4','c5','c6','c7','c8','k1','k2','k3','k4');


%   |--------|--------|
%   |   D1  A2  D2    |
%   |----A1--|----A3--|
%   |   D3  A4  D4    |
%   |--------|--------|

% Using exact solution in finding boundary conditions
 for i=1:n
   for j=1:n
      for a=1:n
         for b=1:n
            x1=(j-2)*delta;    y1=(i-2)*delta;    x2=(b-2)*delta;    y2=(a-2)*delta;


          %Setting innitial guessing value for covariance of inner grids
              cov(i,j,a,b)=10;
          % Setting boundary conditions by using exat solution
          if i==1|j==1|a==1|b==1|i==n|j==n|a==n|b==n
              cov(i,j,a,b)=0;
          elseif (j==2&b==2)                                  % 1) Left&Left
              cov(i,j,a,b) =exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
          elseif (j==2&a==n-1)|(i==n-1&b==2)                  % 2) Left&Lower or Lower&Left
              cov(i,j,a,b) =exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
          elseif (j==2&b==n-1)|(j==n-1&b==2)                  % 3) Left&Right or Right&Left
              cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
          elseif (j==2&a==2)|(i==2&b==2)                      % 4) Left&Upper or Upper&Left
              cov(i,j,a,b) =exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
```

```
            elseif (i==n-1&a==n-1)                          % 5) Lower&Lower
                cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
            elseif (i==n-1&b==n-1)|(j==n-1&a==n-1)          % 6) Lower&Right or Right&Lower
                cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
            elseif (i==n-1&a==2)|(i==2&a==n-1)              % 7) Lower&Upper or Upper&Lower
               cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
            elseif (j==n-1&b==n-1)                          % 8) Right&Right
                cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
            elseif (j==n-1&a==2)|(i==2&b==n-1)              % 9) Right&Upper or Upper&Right
               cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
            elseif (i==2&a==2)                              % 10) Upper&Upper
               cov(i,j,a,b) = exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
            end
          end
        end
      end
    end

%   To generate symmetry covariance at boundaries of rectangular box
      for i=1:n
        for j=1:n
          for a=1:n
            for b=1:n
                cov(i,j,a,b)=(cov(i,j,a,b)+cov(a,b,i,j))/2;
                cov(a,b,i,j)=cov(i,j,a,b);
            end
          end
        end
      end
```

```
 % To Divide whole domain into 4 sub-domain
for i=1:n
   for j=1:n
      for a=1:n
           for b=1:n

                if     (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
                        covD1D1(i,j,a,b)=cov(i,j,a,b);
                end
                if  (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
                        covD1D2(i,j,a,b)=cov(i,j,a,b);
                end
                if  (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
                        covD1D3(i,j,a,b)=cov(i,j,a,b);
                end
                if  (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
                        covD1D4(i,j,a,b)=cov(i,j,a,b);
                end
                if  (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
                        covD2D1(i,j,a,b)=cov(i,j,a,b);
                end
                if  (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
                         covD2D2(i,j,a,b)=cov(i,j,a,b);
                end
                if  (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
                        covD2D3(i,j,a,b)=cov(i,j,a,b);
                end
                if  (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
                        covD2D4(i,j,a,b)=cov(i,j,a,b);
                end
```

```
                if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
                        covD3D1(i,j,a,b)=cov(i,j,a,b);
                end
                if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
                        covD3D2(i,j,a,b)=cov(i,j,a,b);
                end
                if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
                        covD3D3(i,j,a,b)=cov(i,j,a,b);
                end
                if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
                        covD3D4(i,j,a,b)=cov(i,j,a,b);
                end
                if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
                        covD4D1(i,j,a,b)=cov(i,j,a,b);
                end
                if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
                        covD4D2(i,j,a,b)=cov(i,j,a,b);
                end
                if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
                        covD4D3(i,j,a,b)=cov(i,j,a,b);
                end
                if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
                        covD4D4(i,j,a,b)=cov(i,j,a,b);
                end
            end
        end
    end
end
save('CD1D1.mat', 'covD1D1')            % Saving variables covD1D1 into file CD1D1.mat
save('CD1D2.mat', 'covD1D2')            % Saving variables covD1D2 into file CD1D2.mat
```

```
save('CD1D3.mat', 'covD1D3')            % Saving variables covD1D3 into file CD1D3.mat
save('CD1D4.mat', 'covD1D4')            % Saving variables covD1D4 into file CD1D4.mat

save('CD2D1.mat', 'covD2D1')            % Saving variables covD2D1 into file CD2D1.mat
save('CD2D2.mat', 'covD2D2')            % Saving variables covD2D2 into file CD2D2.mat
save('CD2D3.mat', 'covD2D3')            % Saving variables covD2D3 into file CD2D3.mat
save('CD2D4.mat', 'covD2D4')            % Saving variables covD2D4 into file CD2D4.mat

save('CD3D1.mat', 'covD3D1')            % Saving variables covD3D1 into file CD3D1.mat
save('CD3D2.mat', 'covD3D2')            % Saving variables covD3D2 into file CD3D2.mat
save('CD3D3.mat', 'covD3D3')            % Saving variables covD3D3 into file CD3D3.mat
save('CD3D4.mat', 'covD3D4')            % Saving variables covD3D4 into file CD3D4.mat

save('CD4D1.mat', 'covD4D1')            % Saving variables covD4D1 into file CD4D1.mat
save('CD4D2.mat', 'covD4D2')            % Saving variables covD4D2 into file CD4D2.mat
save('CD4D3.mat', 'covD4D3')            % Saving variables covD4D3 into file CD4D3.mat
save('CD4D4.mat', 'covD4D4')            % Saving variables covD4D4 into file CD4D4.mat
save('Ccov.mat', 'cov')                 % Saving variables cov into file Ccov.mat

% To clear the variables in each sub-domain from the memory
clear covD1D1 covD1D2 covD1D3 covD1D4 covD2D1 covD2D2 covD2D3 covD2D4 covD3D1
      covD3D2 covD3D3 covD3D4 covD4D1 covD4D2 covD4D3 covD4D4 cov

% ----------To calculate covariance of each sub-domain----------
err=1+stt;
iteration=0;
while err>stt
        % Calculating covariance between inner grid of each subdomain
```

```
% 1) D1D1
load ('CD1D1.mat', 'covD1D1')
load ('Ccov.mat', 'cov')
for i=2:N+1
    for j=2:N+1
        for a=2:N+1
            for b=2:N+1

              covD1D1(i,j,a,b)=( covD1D1(i+1,j,a+1,b) + covD1D1(i+1,j,a-1,b) +
covD1D1(i+1,j,a,b+1) + covD1D1(i+1,j,a,b-1) + covD1D1(i-1,j,a+1,b) + covD1D1(i-1,j,a-1,b) +
covD1D1(i-1,j,a,b+1) + covD1D1(i-1,j,a,b-1) + covD1D1(i,j+1,a+1,b) + covD1D1(i,j+1,a-1,b) +
covD1D1(i,j+1,a,b+1) + covD1D1(i,j+1,a,b-1) + covD1D1(i,j-1,a+1,b) + covD1D1(i,j-1,a-1,b) +
covD1D1(i,j-1,a,b+1) + covD1D1(i,j-1,a,b-1) -4*covD1D1(i+1,j,a,b) -4*covD1D1(i-1,j,a,b) -
4*covD1D1(i,j+1,a,b) -4*covD1D1(i,j-1,a,b) -4*covD1D1(i,j,a+1,b)-4*covD1D1(i,j,a-1,b) -
4*covD1D1(i,j,a,b+1)-4*covD1D1(i,j,a,b-1) )/(-16.0);
              % Unchange boundary conditions

            if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-
1&b==2)|(j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|(j==n-1&a==n-
1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|(j==n-1&a==2)|(i==2&b==n-
1)|(i==2&a==2)
              covD1D1(i,j,a,b)=cov(i,j,a,b);
                end
              end
            end
        end
end
save ('CD1D1.mat', 'covD1D1')
save ('Ccov.mat', 'cov')
clear covD1D1 cov
```

```
load ('CD1D2.mat', 'covD1D2')
load ('Ccov.mat', 'cov')
% 2) D1D2
for i=2:N+1
    for j=2:N+1
        for a=2:N+1
            for b=N-1:n-1
covD1D2(i,j,a,b)=( covD1D2(i+1,j,a+1,b) + covD1D2(i+1,j,a-1,b) + covD1D2(i+1,j,a,b+1) +
covD1D2(i+1,j,a,b-1) + covD1D2(i-1,j,a+1,b) + covD1D2(i-1,j,a-1,b) + covD1D2(i-1,j,a,b+1) +
covD1D2(i-1,j,a,b-1) + covD1D2(i,j+1,a+1,b) + covD1D2(i,j+1,a-1,b) + covD1D2(i,j+1,a,b+1) +
covD1D2(i,j+1,a,b-1) + covD1D2(i,j-1,a+1,b) + covD1D2(i,j-1,a-1,b) + covD1D2(i,j-1,a,b+1) +
covD1D2(i,j-1,a,b-1) -4*covD1D2(i+1,j,a,b) -4*covD1D2(i-1,j,a,b) -4*covD1D2(i,j+1,a,b) -
4*covD1D2(i,j-1,a,b) -4*covD1D2(i,j,a+1,b)-4*covD1D2(i,j,a-1,b) -4*covD1D2(i,j,a,b+1)-
4*covD1D2(i,j,a,b-1) )/(-16.0);

            % Unchange boundary conditions
        if    (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
              (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
              (j==n-1&a==n-1)|  (i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
              (j==n-1&a==2)|(i==2&b==n-1)| (i==2&a==2)
            covD1D2(i,j,a,b)=cov(i,j,a,b);
          end
            end
         end
      end
end
save ('CD1D2.mat', 'covD1D2')
save ('Ccov.mat', 'cov')
clear covD1D2 cov
```

```
load ('CD1D3.mat', 'covD1D3')
load ('Ccov.mat', 'cov')
% 3) D1D3
for i=2:N+1
    for j=2:N+1
        for a=N-1:n-1
            for b=2:N+1
covD1D3(i,j,a,b)=( covD1D3(i+1,j,a+1,b) + covD1D3(i+1,j,a-1,b) + covD1D3(i+1,j,a,b+1) +
covD1D3(i+1,j,a,b-1) + covD1D3(i-1,j,a+1,b) + covD1D3(i-1,j,a-1,b) + covD1D3(i-1,j,a,b+1) +
covD1D3(i-1,j,a,b-1) + covD1D3(i,j+1,a+1,b) + covD1D3(i,j+1,a-1,b) + covD1D3(i,j+1,a,b+1) +
covD1D3(i,j+1,a,b-1) + covD1D3(i,j-1,a+1,b) + covD1D3(i,j-1,a-1,b) + covD1D3(i,j-1,a,b+1) +
covD1D3(i,j-1,a,b-1) -4*covD1D3(i+1,j,a,b) -4*covD1D3(i-1,j,a,b) -4*covD1D3(i,j+1,a,b) -
4*covD1D3(i,j-1,a,b) -4*covD1D3(i,j,a+1,b)-4*covD1D3(i,j,a-1,b) -4*covD1D3(i,j,a,b+1)-
4*covD1D3(i,j,a,b-1) )/(-16.0);

            % Unchange boundary conditions
           if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
              (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
              (j==n-1&a==n-1)| (i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
              (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
            covD1D3(i,j,a,b)=cov(i,j,a,b);
           end
             end
           end
       end
end
save ('CD1D3.mat', 'covD1D3')
save ('Ccov.mat', 'cov')
clear covD1D3 cov
```

```
load ('CD1D4.mat', 'covD1D4')
load ('Ccov.mat', 'cov')
%4) D1D4
for i=2:N+1
    for j=2:N+1
        for a=N-1:n-1
            for b=N-1:n-1
covD1D4(i,j,a,b)=( covD1D4(i+1,j,a+1,b) + covD1D4(i+1,j,a-1,b) + covD1D4(i+1,j,a,b+1) +
covD1D4(i+1,j,a,b-1) + covD1D4(i-1,j,a+1,b) + covD1D4(i-1,j,a-1,b) + covD1D4(i-1,j,a,b+1) +
covD1D4(i-1,j,a,b-1) + covD1D4(i,j+1,a+1,b) + covD1D4(i,j+1,a-1,b) + covD1D4(i,j+1,a,b+1) +
covD1D4(i,j+1,a,b-1) + covD1D4(i,j-1,a+1,b) + covD1D4(i,j-1,a-1,b) + covD1D4(i,j-1,a,b+1) +
covD1D4(i,j-1,a,b-1) -4*covD1D4(i+1,j,a,b) -4*covD1D4(i-1,j,a,b) -4*covD1D4(i,j+1,a,b) -
4*covD1D4(i,j-1,a,b) -4*covD1D4(i,j,a+1,b)-4*covD1D4(i,j,a-1,b) -4*covD1D4(i,j,a,b+1)-
4*covD1D4(i,j,a,b-1) )/(-16.0);

            % Unchange boundary conditions
            if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
                (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
                (j==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
                (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
              covD1D4(i,j,a,b)=cov(i,j,a,b);
              end
               end
            end
        end
end
save ('CD1D4.mat', 'covD1D4')
save ('Ccov.mat', 'cov')
clear covD1D4 cov
```

```
load ('CD2D1.mat', 'covD2D1')
load ('Ccov.mat', 'cov')
%5) D2D1
for i=2:N+1
    for j=N-1:n-1
        for a=2:N+1
            for b=2:N+1
covD2D1(i,j,a,b)=( covD2D1(i+1,j,a+1,b) + covD2D1(i+1,j,a-1,b) + covD2D1(i+1,j,a,b+1) +
covD2D1(i+1,j,a,b-1) + covD2D1(i-1,j,a+1,b) + covD2D1(i-1,j,a-1,b) + covD2D1(i-1,j,a,b+1) +
covD2D1(i-1,j,a,b-1) + covD2D1(i,j+1,a+1,b) + covD2D1(i,j+1,a-1,b) + covD2D1(i,j+1,a,b+1) +
covD2D1(i,j+1,a,b-1) + covD2D1(i,j-1,a+1,b) + covD2D1(i,j-1,a-1,b) + covD2D1(i,j-1,a,b+1) +
covD2D1(i,j-1,a,b-1) -4*covD2D1(i+1,j,a,b) -4*covD2D1(i-1,j,a,b) -4*covD2D1(i,j+1,a,b) -
4*covD2D1(i,j-1,a,b) -4*covD2D1(i,j,a+1,b)-4*covD2D1(i,j,a-1,b) -4*covD2D1(i,j,a,b+1)-
4*covD2D1(i,j,a,b-1) )/(-16.0);

            % Unchange boundary conditions
            if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
                (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
                (j==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
                (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
              covD2D1(i,j,a,b)=cov(i,j,a,b);
             end
               end
            end
        end
    end
save ('CD2D1.mat', 'covD2D1')
save ('Ccov.mat', 'cov')
clear covD2D1 cov
```

```
load ('CD2D2.mat', 'covD2D2')
load ('Ccov.mat', 'cov')
%6) D2D2
for i=2:N+1
    for j=N-1:n-1
        for a=2:N+1
            for b=N-1:n-1
covD2D2(i,j,a,b)=( covD2D2(i+1,j,a+1,b) + covD2D2(i+1,j,a-1,b) + covD2D2(i+1,j,a,b+1) +
covD2D2(i+1,j,a,b-1) + covD2D2(i-1,j,a+1,b) + covD2D2(i-1,j,a-1,b) + covD2D2(i-1,j,a,b+1) +
covD2D2(i-1,j,a,b-1) + covD2D2(i,j+1,a+1,b) + covD2D2(i,j+1,a-1,b) + covD2D2(i,j+1,a,b+1) +
covD2D2(i,j+1,a,b-1) + covD2D2(i,j-1,a+1,b) + covD2D2(i,j-1,a-1,b) + covD2D2(i,j-1,a,b+1) +
covD2D2(i,j-1,a,b-1) -4*covD2D2(i+1,j,a,b) -4*covD2D2(i-1,j,a,b) -4*covD2D2(i,j+1,a,b) -
4*covD2D2(i,j-1,a,b) -4*covD2D2(i,j,a+1,b)-4*covD2D2(i,j,a-1,b) -4*covD2D2(i,j,a,b+1)-
4*covD2D2(i,j,a,b-1) )/(-16.0);

            % Unchange boundary conditions
            if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
                (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
                (j==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
                (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
              covD2D2(i,j,a,b)=cov(i,j,a,b);
            end
              end
            end
        end
    end
save ('CD2D2.mat', 'covD2D2')
save ('Ccov.mat', 'cov')
clear covD2D2 cov
```

```
load ('CD2D3.mat', 'covD2D3')
load ('Ccov.mat', 'cov')
%7) D2D3
for i=2:N+1
    for j=N-1:n-1
        for a=N-1:n-1
            for b=2:N+1
covD2D3(i,j,a,b)=( covD2D3(i+1,j,a+1,b) + covD2D3(i+1,j,a-1,b) + covD2D3(i+1,j,a,b+1) +
covD2D3(i+1,j,a,b-1) + covD2D3(i-1,j,a+1,b) + covD2D3(i-1,j,a-1,b) + covD2D3(i-1,j,a,b+1) +
covD2D3(i-1,j,a,b-1) + covD2D3(i,j+1,a+1,b) + covD2D3(i,j+1,a-1,b) + covD2D3(i,j+1,a,b+1) +
covD2D3(i,j+1,a,b-1) + covD2D3(i,j-1,a+1,b) + covD2D3(i,j-1,a-1,b) + covD2D3(i,j-1,a,b+1) +
covD2D3(i,j-1,a,b-1) -4*covD2D3(i+1,j,a,b) -4*covD2D3(i-1,j,a,b) -4*covD2D3(i,j+1,a,b) -
4*covD2D3(i,j-1,a,b) -4*covD2D3(i,j,a+1,b)-4*covD2D3(i,j,a-1,b) -4*covD2D3(i,j,a,b+1)-
4*covD2D3(i,j,a,b-1) )/(-16.0);

            % Unchange boundary conditions
        if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
            (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
            (j==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
            (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
         covD2D3(i,j,a,b)=cov(i,j,a,b);
        end
            end
        end
    end
end
save ('CD2D3.mat', 'covD2D3')
save ('Ccov.mat', 'cov')
clear covD2D3 cov
```

```
load ('CD2D4.mat', 'covD2D4')
load ('Ccov.mat', 'cov')
%8) D2D4
for i=2:N+1
    for j=N-1:n-1
        for a=N-1:n-1
            for b=N-1:n-1
covD2D4(i,j,a,b)=( covD2D4(i+1,j,a+1,b) + covD2D4(i+1,j,a-1,b) + covD2D4(i+1,j,a,b+1) +
covD2D4(i+1,j,a,b-1) + covD2D4(i-1,j,a+1,b) + covD2D4(i-1,j,a-1,b) + covD2D4(i-1,j,a,b+1) +
covD2D4(i-1,j,a,b-1) + covD2D4(i,j+1,a+1,b) + covD2D4(i,j+1,a-1,b) + covD2D4(i,j+1,a,b+1) +
covD2D4(i,j+1,a,b-1) + covD2D4(i,j-1,a+1,b) + covD2D4(i,j-1,a-1,b) + covD2D4(i,j-1,a,b+1) +
covD2D4(i,j-1,a,b-1) -4*covD2D4(i+1,j,a,b) -4*covD2D4(i-1,j,a,b) -4*covD2D4(i,j+1,a,b) -
4*covD2D4(i,j-1,a,b) -4*covD2D4(i,j,a+1,b)-4*covD2D4(i,j,a-1,b) -4*covD2D4(i,j,a,b+1)-
4*covD2D4(i,j,a,b-1) )/(-16.0);

              % Unchange boundary conditions
            if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
                (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
                (j==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
                (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
              covD2D4(i,j,a,b)=cov(i,j,a,b);
            end
              end
            end
        end
    end
save ('CD2D4.mat', 'covD2D4')
save ('Ccov.mat', 'cov')
clear covD2D4 cov
```

```
load ('CD3D1.mat', 'covD3D1')
load ('Ccov.mat', 'cov')
%9) D3D1
for i=N-1:n-1
    for j=2:N+1
        for a=2:N+1
            for b=2:N+1
covD3D1(i,j,a,b)=( covD3D1(i+1,j,a+1,b) + covD3D1(i+1,j,a-1,b) + covD3D1(i+1,j,a,b+1) +
covD3D1(i+1,j,a,b-1) + covD3D1(i-1,j,a+1,b) + covD3D1(i-1,j,a-1,b) + covD3D1(i-1,j,a,b+1) +
covD3D1(i-1,j,a,b-1) + covD3D1(i,j+1,a+1,b) + covD3D1(i,j+1,a-1,b) + covD3D1(i,j+1,a,b+1) +
covD3D1(i,j+1,a,b-1) + covD3D1(i,j-1,a+1,b) + covD3D1(i,j-1,a-1,b) + covD3D1(i,j-1,a,b+1) +
covD3D1(i,j-1,a,b-1) -4*covD3D1(i+1,j,a,b) -4*covD3D1(i-1,j,a,b) -4*covD3D1(i,j+1,a,b) -
4*covD3D1(i,j-1,a,b) -4*covD3D1(i,j,a+1,b)-4*covD3D1(i,j,a-1,b) -4*covD3D1(i,j,a,b+1)-
4*covD3D1(i,j,a,b-1) )/(-16.0);

                % Unchange boundary conditions
            if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
                (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
                (j==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
                (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
              covD3D1(i,j,a,b)=cov(i,j,a,b);
             end
               end
            end
        end
    end
save ('CD3D1.mat', 'covD3D1')
save ('Ccov.mat', 'cov')
clear covD3D1 cov
```

```
load ('CD3D2.mat', 'covD3D2')
load ('Ccov.mat', 'cov')
%10) D3D2
for i=N-1:n-1
    for j=2:N+1
        for a=2:N+1
            for b=N-1:n-1
covD3D2(i,j,a,b)=( covD3D2(i+1,j,a+1,b) + covD3D2(i+1,j,a-1,b) + covD3D2(i+1,j,a,b+1) +
covD3D2(i+1,j,a,b-1) + covD3D2(i-1,j,a+1,b) + covD3D2(i-1,j,a-1,b) + covD3D2(i-1,j,a,b+1) +
covD3D2(i-1,j,a,b-1) + covD3D2(i,j+1,a+1,b) + covD3D2(i,j+1,a-1,b) + covD3D2(i,j+1,a,b+1) +
covD3D2(i,j+1,a,b-1) + covD3D2(i,j-1,a+1,b) + covD3D2(i,j-1,a-1,b) + covD3D2(i,j-1,a,b+1) +
covD3D2(i,j-1,a,b-1) -4*covD3D2(i+1,j,a,b) -4*covD3D2(i-1,j,a,b) -4*covD3D2(i,j+1,a,b) -
4*covD3D2(i,j-1,a,b) -4*covD3D2(i,j,a+1,b)-4*covD3D2(i,j,a-1,b) -4*covD3D2(i,j,a,b+1)-
4*covD3D2(i,j,a,b-1) )/(-16.0);

            % Unchange boundary conditions
          if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
              (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
              (j==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
              (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
            covD3D2(i,j,a,b)=cov(i,j,a,b);
              end
            end
        end
    end
end
save ('CD3D2.mat', 'covD3D2')
save ('Ccov.mat', 'cov')
clear covD3D2 cov
```

```
load ('CD3D3.mat', 'covD3D3')
load ('Ccov.mat', 'cov')
%11) D3D3
for i=N-1:n-1
    for j=2:N+1
        for a=N-1:n-1
            for b=2:N+1
covD3D3(i,j,a,b)=( covD3D3(i+1,j,a+1,b) + covD3D3(i+1,j,a-1,b) + covD3D3(i+1,j,a,b+1) +
covD3D3(i+1,j,a,b-1) + covD3D3(i-1,j,a+1,b) + covD3D3(i-1,j,a-1,b) + covD3D3(i-1,j,a,b+1) +
covD3D3(i-1,j,a,b-1) + covD3D3(i,j+1,a+1,b) + covD3D3(i,j+1,a-1,b) + covD3D3(i,j+1,a,b+1) +
covD3D3(i,j+1,a,b-1) + covD3D3(i,j-1,a+1,b) + covD3D3(i,j-1,a-1,b) + covD3D3(i,j-1,a,b+1) +
covD3D3(i,j-1,a,b-1) -4*covD3D3(i+1,j,a,b) -4*covD3D3(i-1,j,a,b) -4*covD3D3(i,j+1,a,b) -
4*covD3D3(i,j-1,a,b) -4*covD3D3(i,j,a+1,b)-4*covD3D3(i,j,a-1,b) -4*covD3D3(i,j,a,b+1)-
4*covD3D3(i,j,a,b-1) )/(-16.0);

                % Unchange boundary conditions
            if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
                (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
                (j==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
                (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
              covD3D3(i,j,a,b)=cov(i,j,a,b);
                end
              end
            end
        end
    end
end
save ('CD3D3.mat', 'covD3D3')
save ('Ccov.mat', 'cov')
clear covD3D3 cov
```

```
load ('CD3D4.mat', 'covD3D4')
load ('Ccov.mat', 'cov')
%12) D3D4
for i=N-1:n-1
    for j=2:N+1
        for a=N-1:n-1
            for b=N-1:n-1
covD3D4(i,j,a,b)=( covD3D4(i+1,j,a+1,b) + covD3D4(i+1,j,a-1,b) + covD3D4(i+1,j,a,b+1) +
covD3D4(i+1,j,a,b-1) + covD3D4(i-1,j,a+1,b) + covD3D4(i-1,j,a-1,b) + covD3D4(i-1,j,a,b+1) +
covD3D4(i-1,j,a,b-1) + covD3D4(i,j+1,a+1,b) + covD3D4(i,j+1,a-1,b) + covD3D4(i,j+1,a,b+1) +
covD3D4(i,j+1,a,b-1) + covD3D4(i,j-1,a+1,b) + covD3D4(i,j-1,a-1,b) + covD3D4(i,j-1,a,b+1) +
covD3D4(i,j-1,a,b-1) -4*covD3D4(i+1,j,a,b) -4*covD3D4(i-1,j,a,b) -4*covD3D4(i,j+1,a,b) -
4*covD3D4(i,j-1,a,b) -4*covD3D4(i,j,a+1,b)-4*covD3D4(i,j,a-1,b) -4*covD3D4(i,j,a,b+1)-
4*covD3D4(i,j,a,b-1) )/(-16.0);

              % Unchange boundary conditions
            if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
                (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
                (j==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
                (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
              covD3D4(i,j,a,b)=cov(i,j,a,b);
                end
              end
            end
        end
    end
save ('CD3D4.mat', 'covD3D4')
save ('Ccov.mat', 'cov')
clear covD3D4 cov
```

```
load ('CD4D1.mat', 'covD4D1')
load ('Ccov.mat', 'cov')
%13) D4D1
for i=N-1:n-1
    for j=N-1:n-1
        for a=2:N+1
            for b=2:N+1
covD4D1(i,j,a,b)=( covD4D1(i+1,j,a+1,b) + covD4D1(i+1,j,a-1,b) + covD4D1(i+1,j,a,b+1) +
covD4D1(i+1,j,a,b-1) + covD4D1(i-1,j,a+1,b) + covD4D1(i-1,j,a-1,b) + covD4D1(i-1,j,a,b+1) +
covD4D1(i-1,j,a,b-1) + covD4D1(i,j+1,a+1,b) + covD4D1(i,j+1,a-1,b) + covD4D1(i,j+1,a,b+1) +
covD4D1(i,j+1,a,b-1) + covD4D1(i,j-1,a+1,b) + covD4D1(i,j-1,a-1,b) + covD4D1(i,j-1,a,b+1) +
covD4D1(i,j-1,a,b-1) -4*covD4D1(i+1,j,a,b) -4*covD4D1(i-1,j,a,b) -4*covD4D1(i,j+1,a,b) -
4*covD4D1(i,j-1,a,b) -4*covD4D1(i,j,a+1,b)-4*covD4D1(i,j,a-1,b) -4*covD4D1(i,j,a,b+1)-
4*covD4D1(i,j,a,b-1) )/(-16.0);

                % Unchange boundary conditions
            if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
                (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
                (j==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
                (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
              covD4D1(i,j,a,b)=cov(i,j,a,b);
                end
              end
            end
        end
    end
end
save ('CD4D1.mat', 'covD4D1')
save ('Ccov.mat', 'cov')
clear covD4D1 cov
```

```
load ('CD4D2.mat', 'covD4D2')
load ('Ccov.mat', 'cov')
%14) D4D2
for i=N-1:n-1
    for j=N-1:n-1
        for a=2:N+1
            for b=N-1:n-1
covD4D2(i,j,a,b)=( covD4D2(i+1,j,a+1,b) + covD4D2(i+1,j,a-1,b) + covD4D2(i+1,j,a,b+1) +
covD4D2(i+1,j,a,b-1) + covD4D2(i-1,j,a+1,b) + covD4D2(i-1,j,a-1,b) + covD4D2(i-1,j,a,b+1) +
covD4D2(i-1,j,a,b-1) + covD4D2(i,j+1,a+1,b) + covD4D2(i,j+1,a-1,b) + covD4D2(i,j+1,a,b+1) +
covD4D2(i,j+1,a,b-1) + covD4D2(i,j-1,a+1,b) + covD4D2(i,j-1,a-1,b) + covD4D2(i,j-1,a,b+1) +
covD4D2(i,j-1,a,b-1) -4*covD4D2(i+1,j,a,b) -4*covD4D2(i-1,j,a,b) -4*covD4D2(i,j+1,a,b) -
4*covD4D2(i,j-1,a,b) -4*covD4D2(i,j,a+1,b)-4*covD4D2(i,j,a-1,b) -4*covD4D2(i,j,a,b+1)-
4*covD4D2(i,j,a,b-1) )/(-16.0);

              % Unchange boundary conditions
            if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
                (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
                (j==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
                (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
              covD4D2(i,j,a,b)=cov(i,j,a,b);
                end
              end
            end
        end
    end
save ('CD4D2.mat', 'covD4D2')
save ('Ccov.mat', 'cov')
clear covD4D2 cov
```

```
load ('CD4D3.mat', 'covD4D3')
load ('Ccov.mat', 'cov')
%15) D4D3
for i=N-1:n-1
    for j=N-1:n-1
        for a=N-1:n-1
            for b=2:N+1
covD4D3(i,j,a,b)=( covD4D3(i+1,j,a+1,b) + covD4D3(i+1,j,a-1,b) + covD4D3(i+1,j,a,b+1) +
covD4D3(i+1,j,a,b-1) + covD4D3(i-1,j,a+1,b) + covD4D3(i-1,j,a-1,b) + covD4D3(i-1,j,a,b+1) +
covD4D3(i-1,j,a,b-1) + covD4D3(i,j+1,a+1,b) + covD4D3(i,j+1,a-1,b) + covD4D3(i,j+1,a,b+1) +
covD4D3(i,j+1,a,b-1) + covD4D3(i,j-1,a+1,b) + covD4D3(i,j-1,a-1,b) + covD4D3(i,j-1,a,b+1) +
covD4D3(i,j-1,a,b-1) -4*covD4D3(i+1,j,a,b) -4*covD4D3(i-1,j,a,b) -4*covD4D3(i,j+1,a,b) -
4*covD4D3(i,j-1,a,b) -4*covD4D3(i,j,a+1,b)-4*covD4D3(i,j,a-1,b) -4*covD4D3(i,j,a,b+1)-
4*covD4D3(i,j,a,b-1) )/(-16.0);

            % Unchange boundary conditions
           if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
              (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
              (j==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
              (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
            covD4D3(i,j,a,b)=cov(i,j,a,b);
                end
            end
        end
    end
end
save ('CD4D3.mat', 'covD4D3')
save ('Ccov.mat', 'cov')
clear covD4D3 cov
```

```
load ('CD4D4.mat', 'covD4D4')
load ('Ccov.mat', 'cov')
%16) D4D4
for i=N-1:n-1
    for j=N-1:n-1
        for a=N-1:n-1
            for b=N-1:n-1
covD4D4(i,j,a,b)=( covD4D4(i+1,j,a+1,b) + covD4D4(i+1,j,a-1,b) + covD4D4(i+1,j,a,b+1) +
covD4D4(i+1,j,a,b-1) + covD4D4(i-1,j,a+1,b) + covD4D4(i-1,j,a-1,b) + covD4D4(i-1,j,a,b+1) +
covD4D4(i-1,j,a,b-1) + covD4D4(i,j+1,a+1,b) + covD4D4(i,j+1,a-1,b) + covD4D4(i,j+1,a,b+1) +
covD4D4(i,j+1,a,b-1) + covD4D4(i,j-1,a+1,b) + covD4D4(i,j-1,a-1,b) + covD4D4(i,j-1,a,b+1) +
covD4D4(i,j-1,a,b-1) -4*covD4D4(i+1,j,a,b) -4*covD4D4(i-1,j,a,b) -4*covD4D4(i,j+1,a,b) -
4*covD4D4(i,j-1,a,b) -4*covD4D4(i,j,a+1,b)-4*covD4D4(i,j,a-1,b) -4*covD4D4(i,j,a,b+1)-
4*covD4D4(i,j,a,b-1) )/(-16.0);

              % Unchange boundary conditions
            if (j==2&b==2) | (j==2&a==n-1)|(i==n-1&b==2)| (j==2&b==n-1)|(j==n-1&b==2)|
                (j==2&a==2)|(i==2&b==2)|(i==n-1&a==n-1)|(i==n-1&b==n-1)|
                (j==n-1&a==n-1)|(i==n-1&a==2)|(i==2&a==n-1)|(j==n-1&b==n-1)|
                (j==n-1&a==2)|(i==2&b==n-1)|(i==2&a==2)
              covD4D4(i,j,a,b)=cov(i,j,a,b);
                end
              end
            end
        end
    end
save ('CD4D4.mat', 'covD4D4')
save ('Ccov.mat', 'cov')
clear covD4D4 cov
```

```
%       Composing each sub-domain into whole domain to check convergent condition
 load('CD1D1.mat', 'covD1D1')              % Loading variables covD1D1 from file CD1D1.mat
load('CD1D2.mat', 'covD1D2')               % Loading variables covD1D2 from file CD1D2.mat
load('CD1D3.mat', 'covD1D3')               % Loading variables covD1D3 from file CD1D3.mat
load('CD1D4.mat', 'covD1D4')               % Loading variables covD1D4 from file CD1D4.mat

load('CD2D1.mat', 'covD2D1')               % Loading variables covD2D1 from file CD2D1.mat
load('CD2D2.mat', 'covD2D2')               % Loading variables covD2D2 from file CD2D2.mat
load('CD2D3.mat', 'covD2D3')               % Loading variables covD2D3 from file CD2D3.mat
load('CD2D4.mat', 'covD2D4')               % Loading variables covD2D4 from file CD2D4.mat

load('CD3D1.mat', 'covD3D1')               % Loading variables covD3D1 from file CD3D1.mat
load('CD3D2.mat', 'covD3D2')               % Loading variables covD3D2 from file CD3D2.mat
load('CD3D3.mat', 'covD3D3')               % Loading variables covD3D3 from file CD3D3.mat
load('CD3D4.mat', 'covD3D4')               % Loading variables covD3D4 from file CD3D4.mat

load('CD4D1.mat', 'covD4D1')               % Loading variables covD4D1 from file CD4D1.mat
load('CD4D2.mat', 'covD4D2')               % Loading variables covD4D2 from file CD4D2.mat
load('CD4D3.mat', 'covD4D3')               % Loading variables covD4D3 from file CD4D3.mat
load('CD4D4.mat', 'covD4D4')               % Loading variables covD4D4 from file CD4D4.mat
load ('Ccov.mat', 'cov')

err=0;
for i=2:n-1
    for j=2:n-1
       for a=2:n-1
            for b=2:n-1

                  if     (i>=1&i<=N)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=1&b<=N)
                        cov_new(i,j,a,b)=covD1D1(i,j,a,b);
```

```
elseif (i>=1&i<=N)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=N&b<=n)
        cov_new(i,j,a,b)=covD1D2(i,j,a,b);
elseif (i>=1&i<=N)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=1&b<=N)
        cov_new(i,j,a,b)=covD1D3(i,j,a,b);
elseif (i>=1&i<=N)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=N&b<=n)
        cov_new(i,j,a,b)=covD1D4(i,j,a,b);
elseif (i>=1&i<=N)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=1&b<=N)
        cov_new(i,j,a,b)=covD2D1(i,j,a,b);
elseif (i>=1&i<=N)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=N&b<=n)
        cov_new(i,j,a,b)=covD2D2(i,j,a,b);
elseif (i>=1&i<=N)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=1&b<=N)
        cov_new(i,j,a,b)=covD2D3(i,j,a,b);
elseif (i>=1&i<=N)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=N&b<=n)
        cov_new(i,j,a,b)=covD2D4(i,j,a,b);
elseif (i>=N&i<=n)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=1&b<=N)
        cov_new(i,j,a,b)=covD3D1(i,j,a,b);
elseif (i>=N&i<=n)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=N&b<=n)
        cov_new(i,j,a,b)=covD3D2(i,j,a,b);
elseif (i>=N&i<=n)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=1&b<=N)
        cov_new(i,j,a,b)=covD3D3(i,j,a,b);
elseif (i>=N&i<=n)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=N&b<=n)
        cov_new(i,j,a,b)=covD3D4(i,j,a,b);
elseif (i>=N&i<=n)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=1&b<=N)
        cov_new(i,j,a,b)=covD4D1(i,j,a,b);
elseif (i>=N&i<=n)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=N&b<=n)
        cov_new(i,j,a,b)=covD4D2(i,j,a,b);
elseif (i>=N&i<=n)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=1&b<=N)
        cov_new(i,j,a,b)=covD4D3(i,j,a,b);
elseif (i>=N&i<=n)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=N&b<=n)
        cov_new(i,j,a,b)=covD4D4(i,j,a,b);
```

```
                    end

                % Ussing Successive over-relaxation method
                        err=( (abs(cov_new(i,j,a,b)-cov(i,j,a,b))*100)/cov_new(i,j,a,b) )+err;
                        cov(i,j,a,b)=WF*cov_new(i,j,a,b)+(1-WF)*cov(i,j,a,b);

            end
        end
    end
end
iteration=iteration+1;

% To Update covariance in each sub-domain
for i=2:n-1
  for j=2:n-1
    for a=2:n-1
        for b=2:n-1

            if    (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
                    covD1D1(i,j,a,b)=cov(i,j,a,b);
            end
            if (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
                    covD1D2(i,j,a,b)=cov(i,j,a,b);
            end
            if (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
                    covD1D3(i,j,a,b)=cov(i,j,a,b);
            end
            if (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
                    covD1D4(i,j,a,b)=cov(i,j,a,b);
            end
```

```
if (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
        covD2D1(i,j,a,b)=cov(i,j,a,b);
end
if (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
        covD2D2(i,j,a,b)=cov(i,j,a,b);
end
if (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
        covD2D3(i,j,a,b)=cov(i,j,a,b);
end
if (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
        covD2D4(i,j,a,b)=cov(i,j,a,b);
end
if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
        covD3D1(i,j,a,b)=cov(i,j,a,b);
end
if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
        covD3D2(i,j,a,b)=cov(i,j,a,b);
end
if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
        covD3D3(i,j,a,b)=cov(i,j,a,b);
end
if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
        covD3D4(i,j,a,b)=cov(i,j,a,b);
end
if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
        covD4D1(i,j,a,b)=cov(i,j,a,b);
end
```

```
                if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
                        covD4D2(i,j,a,b)=cov(i,j,a,b);
                end
                if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
                        covD4D3(i,j,a,b)=cov(i,j,a,b);
                end
                if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
                        covD4D4(i,j,a,b)=cov(i,j,a,b);
                end

            end
        end
    end
end

save('CD1D1.mat', 'covD1D1')          % Saving variables covD1D1 into file CD1D1.mat
save('CD1D2.mat', 'covD1D2')          % Saving variables covD1D2 into file CD1D2.mat
save('CD1D3.mat', 'covD1D3')          % Saving variables covD1D3 into file CD1D3.mat
save('CD1D4.mat', 'covD1D4')          % Saving variables covD1D4 into file CD1D4.mat

save('CD2D1.mat', 'covD2D1')          % Saving variables covD2D1 into file CD2D1.mat
save('CD2D2.mat', 'covD2D2')          % Saving variables covD2D2 into file CD2D2.mat
save('CD2D3.mat', 'covD2D3')          % Saving variables covD2D3 into file CD2D3.mat
save('CD2D4.mat', 'covD2D4')          % Saving variables covD2D4 into file CD2D4.mat

save('CD3D1.mat', 'covD3D1')          % Saving variables covD3D1 into file CD3D1.mat
save('CD3D2.mat', 'covD3D2')          % Saving variables covD3D2 into file CD3D2.mat
save('CD3D3.mat', 'covD3D3')          % Saving variables covD3D3 into file CD3D3.mat
save('CD3D4.mat', 'covD3D4')          % Saving variables covD3D4 into file CD3D4.mat
```

```
save('CD4D1.mat', 'covD4D1')          % Saving variables covD4D1 into file CD4D1.mat
save('CD4D2.mat', 'covD4D2')          % Saving variables covD4D2 into file CD4D2.mat
save('CD4D3.mat', 'covD4D3')           % Saving variables covD4D3 into file CD4D3.mat
save('CD4D4.mat', 'covD4D4')          % Saving variables covD4D4 into file CD4D4.mat
save('Ccov.mat', 'cov')               % Saving variables cov into file Ccov.mat

clear covD1D2 covD1D3 covD1D4 covD2D1 covD2D2 covD2D3 covD2D4 covD3D1 covD3D2
      covD3D3 covD3D4 covD4D1 covD4D2 covD4D3 covD4D4 cov

end

% To compose each sub-domain into whole domain
load('CD1D1.mat', 'covD1D1')          % Loading variables covD1D1 from file CD1D1.mat
load('CD1D2.mat', 'covD1D2')          % Loading variables covD1D2 from file CD1D2.mat
load('CD1D3.mat', 'covD1D3')          % Loading variables covD1D3 from file CD1D3.mat
load('CD1D4.mat', 'covD1D4')          % Loading variables covD1D4 from file CD1D4.mat

load('CD2D1.mat', 'covD2D1')          % Loading variables covD2D1 from file CD2D1.mat
load('CD2D2.mat', 'covD2D2')          % Loading variables covD2D2 from file CD2D2.mat
load('CD2D3.mat', 'covD2D3')          % Loading variables covD2D3 from file CD2D3.mat
load('CD2D4.mat', 'covD2D4')          % Loading variables covD2D4 from file CD2D4.mat

load('CD3D1.mat', 'covD3D1')          % Loading variables covD3D1 from file CD3D1.mat
load('CD3D2.mat', 'covD3D2')          % Loading variables covD3D2 from file CD3D2.mat
load('CD3D3.mat', 'covD3D3')          % Loading variables covD3D3 from file CD3D3.mat
load('CD3D4.mat', 'covD3D4')          % Loading variables covD3D4 from file CD3D4.mat

load('CD4D1.mat', 'covD4D1')          % Loading variables covD4D1 from file CD4D1.mat
load('CD4D2.mat', 'covD4D2')          % Loading variables covD4D2 from file CD4D2.mat
load('CD4D3.mat', 'covD4D3')          % Loading variables covD4D3 from file CD4D3.mat
```

```
load('CD4D4.mat', 'covD4D4')                % Loading variables covD4D4 from file CD4D4.mat

 for i=1:n
    for j=1:n
       for a=1:n
            for b=1:n

                 if     (i>=1&i<=N)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=1&b<=N)
                        cov(i,j,a,b)=covD1D1(i,j,a,b);
                 elseif (i>=1&i<=N)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=N&b<=n)
                        cov(i,j,a,b)=covD1D2(i,j,a,b);
                 elseif (i>=1&i<=N)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=1&b<=N)
                        cov(i,j,a,b)=covD1D3(i,j,a,b);
                 elseif (i>=1&i<=N)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=N&b<=n)
                        cov(i,j,a,b)=covD1D4(i,j,a,b);
                 elseif (i>=1&i<=N)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=1&b<=N)
                        cov(i,j,a,b)=covD2D1(i,j,a,b);
                 elseif (i>=1&i<=N)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=N&b<=n)
                        cov(i,j,a,b)=covD2D2(i,j,a,b);
                 elseif (i>=1&i<=N)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=1&b<=N)
                        cov(i,j,a,b)=covD2D3(i,j,a,b);
                 elseif (i>=1&i<=N)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=N&b<=n)
                        cov(i,j,a,b)=covD2D4(i,j,a,b);
                 elseif (i>=N&i<=n)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=1&b<=N)
                        cov(i,j,a,b)=covD3D1(i,j,a,b);
                 elseif (i>=N&i<=n)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=N&b<=n)
                        cov(i,j,a,b)=covD3D2(i,j,a,b);
                 elseif (i>=N&i<=n)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=1&b<=N)
                        cov(i,j,a,b)=covD3D3(i,j,a,b);
```

```
                elseif  (i>=N&i<=n)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=N&b<=n)
                        cov(i,j,a,b)=covD3D4(i,j,a,b);
                elseif  (i>=N&i<=n)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=1&b<=N)
                        cov(i,j,a,b)=covD4D1(i,j,a,b);
                elseif  (i>=N&i<=n)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=N&b<=n)
                        cov(i,j,a,b)=covD4D2(i,j,a,b);
                elseif  (i>=N&i<=n)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=1&b<=N)
                        cov(i,j,a,b)=covD4D3(i,j,a,b);
                elseif  (i>=N&i<=n)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=N&b<=n)
                        cov(i,j,a,b)=covD4D4(i,j,a,b);
                end

            end
        end
    end
 end
clear covD1D2 covD1D3 covD1D4 covD2D1 covD2D2 covD2D3 covD2D4 covD3D1 covD3D2
      covD3D3 covD3D4 covD4D1 covD4D2 covD4D3 covD4D4
stop=datestr(now);                  % Marking ending time of calculation
t1=toc;
tot=sec2hms(t1);

% ---------------------FINDNIG AN ERROR OF THE MODEL ----------------------------
%    EXACT SOLUTION
     for i=1:n
          for j=1:n
             for a=1:n
                for b=1:n

                x1=(j-2)*delta;    y1=(i-2)*delta;    x2=(b-2)*delta;    y2=(a-2)*delta;
```

```
                cov_exact(i,j,a,b) =exactV(x1,y1,x2,y2,c1,c2,c3,c4,c5,c6,c7,c8,k1,k2,k3,k4);
                end
            end
        end
    end

    %   To generate symmetry covariance of an exact solution
    for i=1:n
        for j=1:n
            for a=1:n
                for b=1:n
                    cov_exact(i,j,a,b)=(cov_exact(i,j,a,b)+cov_exact(a,b,i,j))/2;
                    cov_exact(a,b,i,j)=cov_exact(i,j,a,b);
                end
            end
        end
    end

app=cov(3:n-2,3:n-2,3:n-2,3:n-2);
exact=cov_exact(3:n-2,3:n-2,3:n-2,3:n-2);
abs_error=(abs(app-exact)*100)./exact;
error=reshape(abs_error,1,(n-4)^4);
min_error=min(error);
max_error=max(error);
average_error=mean(error);

% ----------To save an output into file cov_dom_app2D.txt ---------------------------------
fid=fopen('cov_dom_app2D.txt','a');
fprintf(fid,'------------------------------------------------------------------------\n');
fprintf(fid,'n= %-5.2f\n',n);
```

```
fprintf(fid,'Stopping Tolerance= %2.10f\n',stt);
fprintf(fid,'Iteration= %-5.2f\n',iteration);
fprintf(fid,'\n');

 % Showing solution
for i=[3,n-2]
   for j=[3,n-2]
      for a=[3,n-2]
         for b=[3,n-2]
            x1=(j-2)*delta;    y1=(i-2)*delta;   x2=(b-2)*delta;    y2=(a-2)*delta;
            fprintf(fid,'cov[t1(%-2.8f, %-2.8f), t2(%-2.8f, %-2.8f)]=
                                                              %-3.5f\n',x1,y1,x2,y2,cov(i,j,a,b));
         end
      end
   end
end

fprintf(fid,'------------------------------\n');
fprintf(fid,'Starting time:\n');
                fprintf(fid,'%s\n',start);
fprintf(fid,'Ending time:\n');
                fprintf(fid,'%s\n',stop);
fprintf(fid,'\n');
fprintf(fid,'Total time:  %f  \n',tot);
fprintf(fid,'------------------------------\n');
fprintf(fid,'\n');
fprintf(fid,'Min Error:   %-3.6f  percent\n',min_error);
fprintf(fid,'Max Error:   %-3.6f  percent\n',max_error);
fprintf(fid,'Avg Error:   %-3.6f  percent\n',average_error);
fprintf(fid,'-------------------------------------------------------------------\n');
```

```
fclose(fid)

% To delete file that save variable D1D1-D4D4 from disk
delete CD1D1.mat CD1D2.mat CD1D3.mat CD1D4.mat CD2D1.mat CD2D2.mat CD2D3.mat
CD2D4.mat CD3D1.mat CD3D2.mat CD3D3.mat CD3D4.mat CD4D1.mat CD4D2.mat
CD4D3.mat CD4D4.mat Ccov.mat
```

4. โปรแกรมประมาณฟังก์ชันความแปรปรวนร่วมสำหรับสมการลาปลาสในปัญหา 3 มิติ (กรณีประยุกต์ใช้ระเบียบวิธีแบ่งส่วนโดเมน)

```
function cov_approx=cov_dom_app3D(xxxx)

% From Laplace's equation in 3D an exact solution can be shown in the
% following form
%   cov[t1(x1,y1,z1), t2(x2,y2,z2)] =  ( c1*exp(sqrt(k2)*x1)+ c2*exp(-sqrt(k2)*x1) )*
%                                      ( c3*exp(sqrt(k3)*y1) + c4*exp(-sqrt(k3)*y1) ) *
%                                      ( c5*exp(sqrt(k4)*z1) + c6*exp(-sqrt(k4)*z1) )*
%                                      ( c7*sin(sqrt(k5)*x2) + c8*cos(sqrt(k5)*x2) )*
%                                      ( c9*exp(sqrt(k6)*y2) + c10*exp(-sqrt(k6)*y2) )*
%                                      ( c11*exp(sqrt(k7)*z2) + c12*exp(-sqrt(k7)*z2) );
 start=datestr(now);                 % Marking starting time of calculation
tic
% ------------------------ INPUT VARIABLES --------------------------------
% Setting any constants in exact solution
c1=1; c2=1; c3=1; c4=1; c5=1; c6=1; c7=1; c8=1; c9=1; c10=1; c11=1; c12=1;
k1=3; k2=1; k3=1; k4=k1-k2-k3; k5=2; k6=1; k7=k5-k6; % where k1>k2+k3 and k5>k6

%Setting n-by-n array of whole domain
n=7;                % In order to equalise grid number in each sub-domain,n must be odd number
N=ceil(n/2);        % Setting NxN array of each sub-domain
% Setting length(L) of X and Y axis of whole domain
L=0.1;
LD=L/2;             % Setting length(LD)of X and Y axis of each sub-domain
delta=L/(n-3);
stt=0.0000005;    % Setting stopping tolerance
% Setting weighting-factor of Relaxation method
WF=1.2;             % 0 < WF < 2
```

```
 % ------------------- FINDING COVARIANCE IN EACH SUB-DOMAIN ---------------
exactV=inline('  ( c1*exp(sqrt(k2)*x1)+ c2*exp(-sqrt(k2)*x1) )*
                    ( c3*exp(sqrt(k3)*y1) + c4*exp(-sqrt(k3)*y1) ) *
                    ( c5*exp(sqrt(k4)*z1) + c6*exp(-sqrt(k4)*z1) )*
                    ( c7*sin(sqrt(k5)*x2) + c8*cos(sqrt(k5)*x2) )*
                    ( c9*exp(sqrt(k6)*y2) + c10*exp(-sqrt(k6)*y2) )*
                    ( c11*exp(sqrt(k7)*z2) + c12*exp(-sqrt(k7)*z2) )',
                    'x1','y1','z1','x2','y2','z2','c1','c2','c3','c4','c5','c6','c7','c8','c9','c10','c11','c12','k1',
                    'k2','k3','k4','k5','k6','k7');


%       /--------/--------/|
%      /  D1    /   D2   / |
%     /--------/--------/| |
%    /   D3   /   D4   / |/
%   /--------/--------/| /
%   |        |        |/
%   |--------|-------- |
% Using exact solution in finding boundary conditions
for i=1:n
   for j=1:n
      for k=1:n
         for a=1:n
            for b=1:n
               for c=1:n
            x1=(j-2)*delta; y1=(i-2)*delta; z1=(k-2)*delta;
            x2=(b-2)*delta; y2=(a-2)*delta; z2=(c-2)*delta;


            % Setting innitial guess value for inner grids
                  cov(i,j,k,a,b,c)=100;
         if i==1|j==1|k==1|a==1|b==1|c==1|i==n|j==n|k==n|a==n|b==n|c==n
                  cov(i,j,k,a,b,c)=0;
```

```
    % Setting boundary conditions by using exat solution
elseif (j==2&b==2)                                  % 1) left&left
    cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                             k1,k2,k3,k4,k5,k6,k7);
elseif (j==2&b==n-1)|(j==n-1&b==2)                  % 2) left&right or right&left
    cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                             k1,k2,k3,k4,k5,k6,k7);
elseif (j==2&a==n-1)|(i==n-1&b==2)                  % 3) left&front or front&left
    cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                             k1,k2,k3,k4,k5,k6,k7);
elseif (j==2&a==2)|(i==2&b==2)                      % 4) left&rear or rear&left
    cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                             k1,k2,k3,k4,k5,k6,k7);
elseif (j==2&c==n-1)|(k==n-1&b==2)                  % 5) left&upper or upper&left
    cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                             k1,k2,k3,k4,k5,k6,k7);
elseif (j==2&c==2)|(k==2&b==2)                      % 6) left&lower or lower&left
    cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                             k1,k2,k3,k4,k5,k6,k7);
elseif (j==n-1&b==n-1)                              % 7) right&right
    cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                             k1,k2,k3,k4,k5,k6,k7);
elseif (j==n-1&a==n-1)|(i==n-1&b==n-1)              % 8) right&front or front&right
    cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                             k1,k2,k3,k4,k5,k6,k7);
elseif (j==n-1&a==2)|(i==2&b==n-1)                  % 9) right&rear or rear&right
    cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                             k1,k2,k3,k4,k5,k6,k7);
```

```
elseif (j==n-1&c==n-1)|(k==n-1&b==n-1)          % 10) right&upper or upper&right
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                            k1,k2,k3,k4,k5,k6,k7);
elseif (j==n-1&c==2)|(k==2&b==n-1)              % 11) right&lower or lower&right
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                            k1,k2,k3,k4,k5,k6,k7);
elseif (i==n-1&a==n-1)                          % 12) front&front
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                            k1,k2,k3,k4,k5,k6,k7);
elseif (i==n-1&a==2)|(i==2&a==n-1)              % 13) front&rear or rear&front
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                            k1,k2,k3,k4,k5,k6,k7);
elseif (i==n-1&c==n-1)|(k==n-1&a==n-1)          % 14) front&upper or upper&front
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                            k1,k2,k3,k4,k5,k6,k7);
elseif (i==n-1&c==2)|(k==2&a==n-1)              % 15) front&lower or lower&front
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                            k1,k2,k3,k4,k5,k6,k7);
elseif (i==2&a==2)                              % 16) rear&rear
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                            k1,k2,k3,k4,k5,k6,k7);
elseif (i==2&c==n-1)|(k==n-1&a==2)              % 17) rear&upper or upper&rear
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                            k1,k2,k3,k4,k5,k6,k7);
elseif (i==2&c==2)|(k==2&a==2)                  % 18) rear&lower or lower&rear
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                            k1,k2,k3,k4,k5,k6,k7);
elseif (k==n-1&c==n-1)                          % 19) upper&upper
   cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                            k1,k2,k3,k4,k5,k6,k7);
```

```
        elseif (k==n-1&c==2)|(k==2&c==n-1)                % 20) upper&lower or lower&upper
           cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                                    k1,k2,k3,k4,k5,k6,k7);
        elseif (k==2&c==2)                                % 21) lower&lower
           cov(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,
                                    k1,k2,k3,k4,k5,k6,k7);
        end


            end
          end
        end
      end
   end
end

% To generate the symmetry covariance at boundaries of rectangular box
for i=1:n
   for j=1:n
      for k=1:n
         for a=1:n
            for b=1:n
               for c=1:n
                  if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                     (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
                     (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
                     (j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
                     (j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
                     (i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-
                     1&c==2)|(k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|
                     (i==2&c==2)|(k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|
```

```
                (k==2&c==n-1)|(k==2&c==2)
                 cov(i,j,k,a,b,c)=( cov(i,j,k,a,b,c)+ cov(a,b,c,i,j,k) )/2.0;
                 cov(a,b,c,i,j,k)=cov(i,j,k,a,b,c);
              end
            end
          end
        end
      end
    end
end

% To Divide whole domain into 4 sub-domain
for i=1:n
   for j=1:n
      for k=1:n
         for a=1:n
            for b=1:n
               for c=1:n

                  if     (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
               covD1D1(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
                  end
                  if  (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
               covD1D2(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
                  end
                  if  (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
               covD1D3(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
                  end
                  if  (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
               covD1D4(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
```

```
    end
    if (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
covD2D1(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
covD2D2(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
covD2D3(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
covD2D4(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
covD3D1(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
covD3D2(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
covD3D3(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
covD3D4(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
covD4D1(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
covD4D2(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
```

```
                end
                if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
            covD4D3(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
                end
                if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
            covD4D4(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
                end
            end
          end
        end
      end
   end
end
save('CD1D1.mat', 'covD1D1')              % Saving variables covD1D1 into file CD1D1.mat
save('CD1D2.mat', 'covD1D2')              % Saving variables covD1D2 into file CD1D2.mat
save('CD1D3.mat', 'covD1D3')              % Saving variables covD1D3 into file CD1D3.mat
save('CD1D4.mat', 'covD1D4')              % Saving variables covD1D4 into file CD1D4.mat
save('CD2D1.mat', 'covD2D1')              % Saving variables covD2D1 into file CD2D1.mat
save('CD2D2.mat', 'covD2D2')              % Saving variables covD2D2 into file CD2D2.mat
save('CD2D3.mat', 'covD2D3')              % Saving variables covD2D3 into file CD2D3.mat
save('CD2D4.mat', 'covD2D4')              % Saving variables covD2D4 into file CD2D4.mat
save('CD3D1.mat', 'covD3D1')              % Saving variables covD3D1 into file CD3D1.mat
save('CD3D2.mat', 'covD3D2')              % Saving variables covD3D2 into file CD3D2.mat
save('CD3D3.mat', 'covD3D3')              % Saving variables covD3D3 into file CD3D3.mat
save('CD3D4.mat', 'covD3D4')              % Saving variables covD3D4 into file CD3D4.mat
save('CD4D1.mat', 'covD4D1')              % Saving variables covD4D1 into file CD4D1.mat
save('CD4D2.mat', 'covD4D2')              % Saving variables covD4D2 into file CD4D2.mat
save('CD4D3.mat', 'covD4D3')              % Saving variables covD4D3 into file CD4D3.mat
save('CD4D4.mat', 'covD4D4')              % Saving variables covD4D4 into file CD4D4.mat
```

```
save('Ccov.mat', 'cov')                % Saving variables cov into file Ccov.mat

% To clear variables in each sub-domain from the memory
clear covD1D1 covD1D2 covD1D3 covD1D4 covD2D1 covD2D2 covD2D3 covD2D4 covD3D1
covD3D2 covD3D3 covD3D4 covD4D1 covD4D2 covD4D3 covD4D4 cov
% ----------To calculate covariance of each sub-domain----------
err=stt+1;
iteration=0;
while err>stt
        % Calculate covariance between inner grid of each subdomain
        % 1) D1D1
        load ('CD1D1.mat', 'covD1D1')
        load ('Ccov.mat', 'cov')
        for i=2:N+1
           for j=2:N+1
              for k=2:n-1
                 for a=2:N+1
                    for b=2:N+1
                       for c=2:n-1

covD1D1(i,j,k,a,b,c)=( covD1D1(i+1,j,k,a+1,b,c)+ covD1D1(i-1,j,k,a+1,b,c)+
covD1D1(i+1,j,k,a-1,b,c)+ covD1D1(i-1,j,k,a-1,b,c)+covD1D1(i+1,j,k,a,b+1,c)+
covD1D1(i-1,j,k,a,b+1,c)+ covD1D1(i+1,j,k,a,b-1,c)+ covD1D1(i-1,j,k,a,b-1,c) +
covD1D1(i+1,j,k,a,b,c+1)+ covD1D1(i-1,j,k,a,b,c+1)+ covD1D1(i+1,j,k,a,b,c-1)+
covD1D1(i-1,j,k,a,b,c-1)+covD1D1(i,j+1,k,a+1,b,c)+ covD1D1(i,j-1,k,a+1,b,c)+
covD1D1(i,j+1,k,a-1,b,c)+ covD1D1(i,j-1,k,a-1,b,c) +covD1D1(i,j+1,k,a,b+1,c)+
covD1D1(i,j-1,k,a,b+1,c)+ covD1D1(i,j+1,k,a,b-1,c)+ covD1D1(i,j-1,k,a,b-1,c)+
covD1D1(i,j+1,k,a,b,c+1)+ covD1D1(i,j-1,k,a,b,c+1)+ covD1D1(i,j+1,k,a,b,c-1)+
covD1D1(i,j-1,k,a,b,c-1)+covD1D1(i,j,k+1,a+1,b,c)+ covD1D1(i,j,k-1,a+1,b,c)+
covD1D1(i,j,k+1,a-1,b,c)+ covD1D1(i,j,k-1,a-1,b,c)+covD1D1(i,j,k+1,a,b+1,c)+
```

```
covD1D1(i,j,k-1,a,b+1,c)+ covD1D1(i,j,k+1,a,b-1,c)+ covD1D1(i,j,k-1,a,b-1,c)+
covD1D1(i,j,k+1,a,b,c+1)+ covD1D1(i,j,k-1,a,b,c+1)+ covD1D1(i,j,k+1,a,b,c-1)+
covD1D1(i,j,k-1,a,b,c-1)-6*covD1D1(i+1,j,k,a,b,c) -6*covD1D1(i-1,j,k,a,b,c) -
6*covD1D1(i,j+1,k,a,b,c) -6*covD1D1(i,j-1,k,a,b,c)-6*covD1D1(i,j,k+1,a,b,c) –
6*covD1D1(i,j,k-1,a,b,c) -6*covD1D1(i,j,k,a+1,b,c) -6*covD1D1(i,j,k,a-1,b,c)-
6*covD1D1(i,j,k,a,b+1,c) -6*covD1D1(i,j,k,a,b-1,c) -6*covD1D1(i,j,k,a,b,c+1) -
6*covD1D1(i,j,k,a,b,c-1) )/(-36);
                % Unchange boundary conditions
                if   (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                    (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
                    (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
                    (j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
                    (j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
                    (i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
                    (k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
                    (k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
                  covD1D1(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
                end
                     end
                    end
                  end
                end
              end
            end
save ('CD1D1.mat', 'covD1D1')
save ('Ccov.mat', 'cov')
clear covD1D1 cov

load ('CD1D2.mat', 'covD1D2')
load ('Ccov.mat', 'cov')
```

```
% 2) D1D2
for i=2:N+1
   for j=2:N+1
      for k=2:n-1
         for a=2:N+1
            for b=N-1:n-1
               for c=2:n-1

covD1D2(i,j,k,a,b,c)=( covD1D2(i+1,j,k,a+1,b,c)+ covD1D2(i-1,j,k,a+1,b,c)+
covD1D2(i+1,j,k,a-1,b,c)+ covD1D2(i-1,j,k,a-1,b,c)+covD1D2(i+1,j,k,a,b+1,c)+
covD1D2(i-1,j,k,a,b+1,c)+ covD1D2(i+1,j,k,a,b-1,c)+ covD1D2(i-1,j,k,a,b-1,c) +
covD1D2(i+1,j,k,a,b,c+1)+ covD1D2(i-1,j,k,a,b,c+1)+ covD1D2(i+1,j,k,a,b,c-1)+
covD1D2(i-1,j,k,a,b,c-1)+covD1D2(i,j+1,k,a+1,b,c)+ covD1D2(i,j-1,k,a+1,b,c)+
covD1D2(i,j+1,k,a-1,b,c)+ covD1D2(i,j-1,k,a-1,b,c) +covD1D2(i,j+1,k,a,b+1,c)+
covD1D2(i,j-1,k,a,b+1,c)+ covD1D2(i,j+1,k,a,b-1,c)+ covD1D2(i,j-1,k,a,b-1,c)+
covD1D2(i,j+1,k,a,b,c+1)+ covD1D2(i,j-1,k,a,b,c+1)+ covD1D2(i,j+1,k,a,b,c-1)+
covD1D2(i,j-1,k,a,b,c-1)+covD1D2(i,j,k+1,a+1,b,c)+ covD1D2(i,j,k-1,a+1,b,c)+
covD1D2(i,j,k+1,a-1,b,c)+ covD1D2(i,j,k-1,a-1,b,c)+covD1D2(i,j,k+1,a,b+1,c)+
covD1D2(i,j,k-1,a,b+1,c)+ covD1D2(i,j,k+1,a,b-1,c)+ covD1D2(i,j,k-1,a,b-1,c)+
covD1D2(i,j,k+1,a,b,c+1)+ covD1D2(i,j,k-1,a,b,c+1)+ covD1D2(i,j,k+1,a,b,c-1)+
covD1D2(i,j,k-1,a,b,c-1)-6*covD1D2(i+1,j,k,a,b,c) -6*covD1D2(i-1,j,k,a,b,c) -
6*covD1D2(i,j+1,k,a,b,c) -6*covD1D2(i,j-1,k,a,b,c)-6*covD1D2(i,j,k+1,a,b,c) –
6*covD1D2(i,j,k-1,a,b,c) -6*covD1D2(i,j,k,a+1,b,c) -6*covD1D2(i,j,k,a-1,b,c)-
6*covD1D2(i,j,k,a,b+1,c) -6*covD1D2(i,j,k,a,b-1,c) -6*covD1D2(i,j,k,a,b,c+1) -
6*covD1D2(i,j,k,a,b,c-1) )/(-36);

                  % Unchange boundary conditions
                 if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                    (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
                    (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
```

```
(j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
(j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
(i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
(k==2&a==n-1)| (i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
(k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
covD1D2(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
end
end
end
end
end
end
end
save ('CD1D2.mat', 'covD1D2')
save ('Ccov.mat', 'cov')
clear covD1D2 cov

load ('CD1D3.mat', 'covD1D3')
load ('Ccov.mat', 'cov')
% 3) D1D3
for i=2:N+1
   for j=2:N+1
      for k=2:n-1
         for a=N-1:n-1
            for b=2:N+1
               for c=2:n-1

covD1D3(i,j,k,a,b,c)=( covD1D3(i+1,j,k,a+1,b,c)+ covD1D3(i-1,j,k,a+1,b,c)+
covD1D3(i+1,j,k,a-1,b,c)+ covD1D3(i-1,j,k,a-1,b,c)+covD1D3(i+1,j,k,a,b+1,c)+
```

```
covD1D3(i-1,j,k,a,b+1,c)+ covD1D3(i+1,j,k,a,b-1,c)+ covD1D3(i-1,j,k,a,b-1,c) +
covD1D3(i+1,j,k,a,b,c+1)+ covD1D3(i-1,j,k,a,b,c+1)+ covD1D3(i+1,j,k,a,b,c-1)+
covD1D3(i-1,j,k,a,b,c-1)+covD1D3(i,j+1,k,a+1,b,c)+ covD1D3(i,j-1,k,a+1,b,c)+
covD1D3(i,j+1,k,a-1,b,c)+ covD1D3(i,j-1,k,a-1,b,c) +covD1D3(i,j+1,k,a,b+1,c)+
covD1D3(i,j-1,k,a,b+1,c)+ covD1D3(i,j+1,k,a,b-1,c)+ covD1D3(i,j-1,k,a,b-1,c)+
covD1D3(i,j+1,k,a,b,c+1)+ covD1D3(i,j-1,k,a,b,c+1)+ covD1D3(i,j+1,k,a,b,c-1)+
covD1D3(i,j-1,k,a,b,c-1)+covD1D3(i,j,k+1,a+1,b,c)+ covD1D3(i,j,k-1,a+1,b,c)+
covD1D3(i,j,k+1,a-1,b,c)+ covD1D3(i,j,k-1,a-1,b,c)+covD1D3(i,j,k+1,a,b+1,c)+
covD1D3(i,j,k-1,a,b+1,c)+ covD1D3(i,j,k+1,a,b-1,c)+ covD1D3(i,j,k-1,a,b-1,c)+
covD1D3(i,j,k+1,a,b,c+1)+ covD1D3(i,j,k-1,a,b,c+1)+ covD1D3(i,j,k+1,a,b,c-1)+
covD1D3(i,j,k-1,a,b,c-1)-6*covD1D3(i+1,j,k,a,b,c) -6*covD1D3(i-1,j,k,a,b,c) -
6*covD1D3(i,j+1,k,a,b,c) -6*covD1D3(i,j-1,k,a,b,c)-6*covD1D3(i,j,k+1,a,b,c) –
6*covD1D3(i,j,k-1,a,b,c) -6*covD1D3(i,j,k,a+1,b,c) -6*covD1D3(i,j,k,a-1,b,c)-
6*covD1D3(i,j,k,a,b+1,c) -6*covD1D3(i,j,k,a,b-1,c) -6*covD1D3(i,j,k,a,b,c+1) -
6*covD1D3(i,j,k,a,b,c-1) )/(-36);

                % Unchange boundary conditions
              if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                 (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
                 (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
                 (j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
                 (j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
                 (i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
                 (k==2&a==n-1)| (i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
                 (k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
                covD1D3(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
              end
                   end
                 end
              end
```

```
        end
    end
end
save ('CD1D3.mat', 'covD1D3')
save ('Ccov.mat', 'cov')
clear covD1D3 cov

load ('CD1D4.mat', 'covD1D4')
load ('Ccov.mat', 'cov')
%4) D1D4
for i=2:N+1
    for j=2:N+1
        for k=2:n-1
            for a=N-1:n-1
                for b=N-1:n-1
                    for c=2:n-1

covD1D4(i,j,k,a,b,c)=( covD1D4(i+1,j,k,a+1,b,c)+ covD1D4(i-1,j,k,a+1,b,c)+
covD1D4(i+1,j,k,a-1,b,c)+ covD1D4(i-1,j,k,a-1,b,c)+covD1D4(i+1,j,k,a,b+1,c)+
covD1D4(i-1,j,k,a,b+1,c)+ covD1D4(i+1,j,k,a,b-1,c)+ covD1D4(i-1,j,k,a,b-1,c) +
covD1D4(i+1,j,k,a,b,c+1)+ covD1D4(i-1,j,k,a,b,c+1)+ covD1D4(i+1,j,k,a,b,c-1)+
covD1D4(i-1,j,k,a,b,c-1)+covD1D4(i,j+1,k,a+1,b,c)+ covD1D4(i,j-1,k,a+1,b,c)+
covD1D4(i,j+1,k,a-1,b,c)+ covD1D4(i,j-1,k,a-1,b,c) +covD1D4(i,j+1,k,a,b+1,c)+
covD1D4(i,j-1,k,a,b+1,c)+ covD1D4(i,j+1,k,a,b-1,c)+ covD1D4(i,j-1,k,a,b-1,c)+
covD1D4(i,j+1,k,a,b,c+1)+ covD1D4(i,j-1,k,a,b,c+1)+ covD1D4(i,j+1,k,a,b,c-1)+
covD1D4(i,j-1,k,a,b,c-1)+covD1D4(i,j,k+1,a+1,b,c)+ covD1D4(i,j,k-1,a+1,b,c)+
covD1D4(i,j,k+1,a-1,b,c)+ covD1D4(i,j,k-1,a-1,b,c)+covD1D4(i,j,k+1,a,b+1,c)+
covD1D4(i,j,k-1,a,b+1,c)+ covD1D4(i,j,k+1,a,b-1,c)+ covD1D4(i,j,k-1,a,b-1,c)+
covD1D4(i,j,k+1,a,b,c+1)+ covD1D4(i,j,k-1,a,b,c+1)+ covD1D4(i,j,k+1,a,b,c-1)+
```

```
covD1D4(i,j,k-1,a,b,c-1)-6*covD1D4(i+1,j,k,a,b,c) -6*covD1D4(i-1,j,k,a,b,c) -
6*covD1D4(i,j+1,k,a,b,c) -6*covD1D4(i,j-1,k,a,b,c)-6*covD1D4(i,j,k+1,a,b,c) –
6*covD1D4(i,j,k-1,a,b,c) -6*covD1D4(i,j,k,a+1,b,c) -6*covD1D4(i,j,k,a-1,b,c)-
6*covD1D4(i,j,k,a,b+1,c) -6*covD1D4(i,j,k,a,b-1,c) -6*covD1D4(i,j,k,a,b,c+1) -
6*covD1D4(i,j,k,a,b,c-1) )/(-36);


            % Unchange boundary conditions
           if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
              (j==2&a==2)| (i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
              (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
              (j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
              (j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
              (i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
              (k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
              (k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
            covD1D4(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
           end
                end
               end
             end
          end
       end
end
save ('CD1D4.mat', 'covD1D4')
save ('Ccov.mat', 'cov')
clear covD1D4 cov

load ('CD2D1.mat', 'covD2D1')
load ('Ccov.mat', 'cov')
```

```
%5) D2D1
for i=2:N+1
   for j=N-1:n-1
      for k=2:n-1
         for a=2:N+1
            for b=2:N+1
               for c=2:n-1

covD2D1(i,j,k,a,b,c)=( covD2D1(i+1,j,k,a+1,b,c)+ covD2D1(i-1,j,k,a+1,b,c)+
covD2D1(i+1,j,k,a-1,b,c)+ covD2D1(i-1,j,k,a-1,b,c)+covD2D1(i+1,j,k,a,b+1,c)+
covD2D1(i-1,j,k,a,b+1,c)+ covD2D1(i+1,j,k,a,b-1,c)+ covD2D1(i-1,j,k,a,b-1,c)
+covD2D1(i+1,j,k,a,b,c+1)+ covD2D1(i-1,j,k,a,b,c+1)+ covD2D1(i+1,j,k,a,b,c-1)+
covD2D1(i-1,j,k,a,b,c-1)+covD2D1(i,j+1,k,a+1,b,c)+ covD2D1(i,j-1,k,a+1,b,c)+
covD2D1(i,j+1,k,a-1,b,c)+ covD2D1(i,j-1,k,a-1,b,c) +covD2D1(i,j+1,k,a,b+1,c)+
covD2D1(i,j-1,k,a,b+1,c)+ covD2D1(i,j+1,k,a,b-1,c)+ covD2D1(i,j-1,k,a,b-1,c)+
covD2D1(i,j+1,k,a,b,c+1)+ covD2D1(i,j-1,k,a,b,c+1)+ covD2D1(i,j+1,k,a,b,c-1)+
covD2D1(i,j-1,k,a,b,c-1)+covD2D1(i,j,k+1,a+1,b,c)+ covD2D1(i,j,k-1,a+1,b,c)+
covD2D1(i,j,k+1,a-1,b,c)+ covD2D1(i,j,k-1,a-1,b,c)+covD2D1(i,j,k+1,a,b+1,c)+
covD2D1(i,j,k-1,a,b+1,c)+ covD2D1(i,j,k+1,a,b-1,c)+ covD2D1(i,j,k-1,a,b-1,c)+
covD2D1(i,j,k+1,a,b,c+1)+ covD2D1(i,j,k-1,a,b,c+1)+ covD2D1(i,j,k+1,a,b,c-1)+
covD2D1(i,j,k-1,a,b,c-1)-6*covD2D1(i+1,j,k,a,b,c) -6*covD2D1(i-1,j,k,a,b,c) -
6*covD2D1(i,j+1,k,a,b,c) -6*covD2D1(i,j-1,k,a,b,c)-6*covD2D1(i,j,k+1,a,b,c) –
6*covD2D1(i,j,k-1,a,b,c) -6*covD2D1(i,j,k,a+1,b,c) -6*covD2D1(i,j,k,a-1,b,c)-
6*covD2D1(i,j,k,a,b+1,c) -6*covD2D1(i,j,k,a,b-1,c) -6*covD2D1(i,j,k,a,b,c+1) -
6*covD2D1(i,j,k,a,b,c-1) )/(-36);

                  % Unchange boundary conditions
                  if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                     (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2) |(j==2&c==2)|
                     (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
```

```
(j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
(j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
(i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
(k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)| (i==2&c==2)|
(k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
covD2D1(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
end
end
end
end
end
end
end
save ('CD2D1.mat', 'covD2D1')
save ('Ccov.mat', 'cov')
clear covD2D1 cov

load ('CD2D2.mat', 'covD2D2')
load ('Ccov.mat', 'cov')
%6) D2D2
for i=2:N+1
for j=N-1:n-1
for k=2:n-1
for a=2:N+1
for b=N-1:n-1
for c=2:n-1

covD2D2(i,j,k,a,b,c)=( covD2D2(i+1,j,k,a+1,b,c)+ covD2D2(i-1,j,k,a+1,b,c)+
covD2D2(i+1,j,k,a-1,b,c)+ covD2D2(i-1,j,k,a-1,b,c)+covD2D2(i+1,j,k,a,b+1,c)+
```

```
covD2D2(i-1,j,k,a,b+1,c)+ covD2D2(i+1,j,k,a,b-1,c)+ covD2D2(i-1,j,k,a,b-1,c) +
covD2D2(i+1,j,k,a,b,c+1)+ covD2D2(i-1,j,k,a,b,c+1)+ covD2D2(i+1,j,k,a,b,c-1)+
covD2D2(i-1,j,k,a,b,c-1)+covD2D2(i,j+1,k,a+1,b,c)+ covD2D2(i,j-1,k,a+1,b,c)+
covD2D2(i,j+1,k,a-1,b,c)+ covD2D2(i,j-1,k,a-1,b,c) +covD2D2(i,j+1,k,a,b+1,c)+
covD2D2(i,j-1,k,a,b+1,c)+ covD2D2(i,j+1,k,a,b-1,c)+ covD2D2(i,j-1,k,a,b-1,c)+
covD2D2(i,j+1,k,a,b,c+1)+ covD2D2(i,j-1,k,a,b,c+1)+ covD2D2(i,j+1,k,a,b,c-1)+
covD2D2(i,j-1,k,a,b,c-1)+covD2D2(i,j,k+1,a+1,b,c)+ covD2D2(i,j,k-1,a+1,b,c)+
covD2D2(i,j,k+1,a-1,b,c)+ covD2D2(i,j,k-1,a-1,b,c)+covD2D2(i,j,k+1,a,b+1,c)+
covD2D2(i,j,k-1,a,b+1,c)+ covD2D2(i,j,k+1,a,b-1,c)+ covD2D2(i,j,k-1,a,b-1,c)+
covD2D2(i,j,k+1,a,b,c+1)+ covD2D2(i,j,k-1,a,b,c+1)+ covD2D2(i,j,k+1,a,b,c-1)+
covD2D2(i,j,k-1,a,b,c-1)-6*covD2D2(i+1,j,k,a,b,c) -6*covD2D2(i-1,j,k,a,b,c) -
6*covD2D2(i,j+1,k,a,b,c) -6*covD2D2(i,j-1,k,a,b,c)-6*covD2D2(i,j,k+1,a,b,c) -
6*covD2D2(i,j,k-1,a,b,c) -6*covD2D2(i,j,k,a+1,b,c) -6*covD2D2(i,j,k,a-1,b,c)-
6*covD2D2(i,j,k,a,b+1,c) -6*covD2D2(i,j,k,a,b-1,c) -6*covD2D2(i,j,k,a,b,c+1) -
6*covD2D2(i,j,k,a,b,c-1) )/(-36);

                % Unchange boundary conditions
               if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|
                   (i==n-1&b==2)|(j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|
                   (j==2&c==2)|(k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|
                   (i==n-1&b==n-1)| (j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|
                   (k==n-1&b==n-1)|(j==n-1&c==2)| (k==2&b==n-1)|(i==n-1&a==n-1)|
                   (i==n-1&a==2)|(i==2&a==n-1)|(i==n-1&c==n-1)| (k==n-1&a==n-1)|
                   (i==n-1&c==2)|(k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|
                   (i==2&c==2)|(k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|
                   (k==2&c==2)
                 covD2D2(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
               end
                    end
                   end
```

```
            end
        end
    end
end
save ('CD2D2.mat', 'covD2D2')
save ('Ccov.mat', 'cov')
clear covD2D2 cov

load ('CD2D3.mat', 'covD2D3')
load ('Ccov.mat', 'cov')
%7) D2D3
for i=2:N+1
    for j=N-1:n-1
        for k=2:n-1
            for a=N-1:n-1
                for b=2:N+1
                    for c=2:n-1

 covD2D3(i,j,k,a,b,c)=( covD2D3(i+1,j,k,a+1,b,c)+ covD2D3(i-1,j,k,a+1,b,c)+
covD2D3(i+1,j,k,a-1,b,c)+ covD2D3(i-1,j,k,a-1,b,c)+covD2D3(i+1,j,k,a,b+1,c)+
covD2D3(i-1,j,k,a,b+1,c)+ covD2D3(i+1,j,k,a,b-1,c)+ covD2D3(i-1,j,k,a,b-1,c)
+covD2D3(i+1,j,k,a,b,c+1)+ covD2D3(i-1,j,k,a,b,c+1)+ covD2D3(i+1,j,k,a,b,c-1)+
covD2D3(i-1,j,k,a,b,c-1)+covD2D3(i,j+1,k,a+1,b,c)+ covD2D3(i,j-1,k,a+1,b,c)+
covD2D3(i,j+1,k,a-1,b,c)+ covD2D3(i,j-1,k,a-1,b,c) +covD2D3(i,j+1,k,a,b+1,c)+
covD2D3(i,j-1,k,a,b+1,c)+ covD2D3(i,j+1,k,a,b-1,c)+ covD2D3(i,j-1,k,a,b-1,c)+
covD2D3(i,j+1,k,a,b,c+1)+ covD2D3(i,j-1,k,a,b,c+1)+ covD2D3(i,j+1,k,a,b,c-1)+
covD2D3(i,j-1,k,a,b,c-1)+covD2D3(i,j,k+1,a+1,b,c)+ covD2D3(i,j,k-1,a+1,b,c)+
covD2D3(i,j,k+1,a-1,b,c)+ covD2D3(i,j,k-1,a-1,b,c)+covD2D3(i,j,k+1,a,b+1,c)+
covD2D3(i,j,k-1,a,b+1,c)+ covD2D3(i,j,k+1,a,b-1,c)+ covD2D3(i,j,k-1,a,b-1,c)+
covD2D3(i,j,k+1,a,b,c+1)+ covD2D3(i,j,k-1,a,b,c+1)+ covD2D3(i,j,k+1,a,b,c-1)+
```

```
covD2D3(i,j,k-1,a,b,c-1)-6*covD2D3(i+1,j,k,a,b,c) -6*covD2D3(i-1,j,k,a,b,c) -
6*covD2D3(i,j+1,k,a,b,c) -6*covD2D3(i,j-1,k,a,b,c)-6*covD2D3(i,j,k+1,a,b,c) –
6*covD2D3(i,j,k-1,a,b,c) -6*covD2D3(i,j,k,a+1,b,c) -6*covD2D3(i,j,k,a-1,b,c)-
6*covD2D3(i,j,k,a,b+1,c) -6*covD2D3(i,j,k,a,b-1,c) -6*covD2D3(i,j,k,a,b,c+1) -
6*covD2D3(i,j,k,a,b,c-1) )/(-36);

                % Unchange boundary conditions
              if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                  (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
                  (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
                  (j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
                  (j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
                  (i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
                  (k==2&a==n-1)| (i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
                  (k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
               covD2D3(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
              end
                    end
                   end
                 end
               end
             end
          end
save ('CD2D3.mat', 'covD2D3')
save ('Ccov.mat', 'cov')
clear covD2D3 cov

load ('CD2D4.mat', 'covD2D4')
load ('Ccov.mat', 'cov')
%8) D2D4
```

```
for i=2:N+1
    for j=N-1:n-1
        for k=2:n-1
            for a=N-1:n-1
                for b=N-1:n-1
                    for c=2:n-1

covD2D4(i,j,k,a,b,c)=( covD2D4(i+1,j,k,a+1,b,c)+ covD2D4(i-1,j,k,a+1,b,c)+
covD2D4(i+1,j,k,a-1,b,c)+ covD2D4(i-1,j,k,a-1,b,c)+covD2D4(i+1,j,k,a,b+1,c)+
covD2D4(i-1,j,k,a,b+1,c)+ covD2D4(i+1,j,k,a,b-1,c)+ covD2D4(i-1,j,k,a,b-1,c)
+covD2D4(i+1,j,k,a,b,c+1)+ covD2D4(i-1,j,k,a,b,c+1)+ covD2D4(i+1,j,k,a,b,c-1)+
covD2D4(i-1,j,k,a,b,c-1)+covD2D4(i,j+1,k,a+1,b,c)+ covD2D4(i,j-1,k,a+1,b,c)+
covD2D4(i,j+1,k,a-1,b,c)+ covD2D4(i,j-1,k,a-1,b,c) +covD2D4(i,j+1,k,a,b+1,c)+
covD2D4(i,j-1,k,a,b+1,c)+ covD2D4(i,j+1,k,a,b-1,c)+ covD2D4(i,j-1,k,a,b-1,c)+
covD2D4(i,j+1,k,a,b,c+1)+ covD2D4(i,j-1,k,a,b,c+1)+ covD2D4(i,j+1,k,a,b,c-1)+
covD2D4(i,j-1,k,a,b,c-1)+covD2D4(i,j,k+1,a+1,b,c)+ covD2D4(i,j,k-1,a+1,b,c)+
covD2D4(i,j,k+1,a-1,b,c)+ covD2D4(i,j,k-1,a-1,b,c)+covD2D4(i,j,k+1,a,b+1,c)+
covD2D4(i,j,k-1,a,b+1,c)+ covD2D4(i,j,k+1,a,b-1,c)+ covD2D4(i,j,k-1,a,b-1,c)+
covD2D4(i,j,k+1,a,b,c+1)+ covD2D4(i,j,k-1,a,b,c+1)+ covD2D4(i,j,k+1,a,b,c-1)+
covD2D4(i,j,k-1,a,b,c-1)-6*covD2D4(i+1,j,k,a,b,c) -6*covD2D4(i-1,j,k,a,b,c) -
6*covD2D4(i,j+1,k,a,b,c) -6*covD2D4(i,j-1,k,a,b,c)-6*covD2D4(i,j,k+1,a,b,c) –
6*covD2D4(i,j,k-1,a,b,c) -6*covD2D4(i,j,k,a+1,b,c) -6*covD2D4(i,j,k,a-1,b,c)-
6*covD2D4(i,j,k,a,b+1,c) -6*covD2D4(i,j,k,a,b-1,c) -6*covD2D4(i,j,k,a,b,c+1) -
6*covD2D4(i,j,k,a,b,c-1) )/(-36);


                    % Unchange boundary conditions
                if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                    (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
                    (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
```

```
(j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
(j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
(i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
(k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
(k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
covD2D4(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
end
end
end
end
end
end
save ('CD2D4.mat', 'covD2D4')
save ('Ccov.mat', 'cov')
clear covD2D4 cov

load ('CD3D1.mat', 'covD3D1')
load ('Ccov.mat', 'cov')
%9) D3D1
for i=N-1:n-1
for j=2:N+1
for k=2:n-1
for a=2:N+1
for b=2:N+1
for c=2:n-1

covD3D1(i,j,k,a,b,c)=( covD3D1(i+1,j,k,a+1,b,c)+ covD3D1(i-1,j,k,a+1,b,c)+
covD3D1(i+1,j,k,a-1,b,c)+ covD3D1(i-1,j,k,a-1,b,c)+covD3D1(i+1,j,k,a,b+1,c)+
```

```
covD3D1(i-1,j,k,a,b+1,c)+ covD3D1(i+1,j,k,a,b-1,c)+ covD3D1(i-1,j,k,a,b-1,c) +
covD3D1(i+1,j,k,a,b,c+1)+ covD3D1(i-1,j,k,a,b,c+1)+ covD3D1(i+1,j,k,a,b,c-1)+
covD3D1(i-1,j,k,a,b,c-1)+covD3D1(i,j+1,k,a+1,b,c)+ covD3D1(i,j-1,k,a+1,b,c)+
covD3D1(i,j+1,k,a-1,b,c)+ covD3D1(i,j-1,k,a-1,b,c) +covD3D1(i,j+1,k,a,b+1,c)+
covD3D1(i,j-1,k,a,b+1,c)+ covD3D1(i,j+1,k,a,b-1,c)+ covD3D1(i,j-1,k,a,b-1,c)+
covD3D1(i,j+1,k,a,b,c+1)+ covD3D1(i,j-1,k,a,b,c+1)+ covD3D1(i,j+1,k,a,b,c-1)+
covD3D1(i,j-1,k,a,b,c-1)+covD3D1(i,j,k+1,a+1,b,c)+ covD3D1(i,j,k-1,a+1,b,c)+
covD3D1(i,j,k+1,a-1,b,c)+ covD3D1(i,j,k-1,a-1,b,c)+covD3D1(i,j,k+1,a,b+1,c)+
covD3D1(i,j,k-1,a,b+1,c)+ covD3D1(i,j,k+1,a,b-1,c)+ covD3D1(i,j,k-1,a,b-1,c)+
covD3D1(i,j,k+1,a,b,c+1)+ covD3D1(i,j,k-1,a,b,c+1)+ covD3D1(i,j,k+1,a,b,c-1)+
covD3D1(i,j,k-1,a,b,c-1)-6*covD3D1(i+1,j,k,a,b,c) -6*covD3D1(i-1,j,k,a,b,c) -
6*covD3D1(i,j+1,k,a,b,c) -6*covD3D1(i,j-1,k,a,b,c)-6*covD3D1(i,j,k+1,a,b,c) –
6*covD3D1(i,j,k-1,a,b,c) -6*covD3D1(i,j,k,a+1,b,c) -6*covD3D1(i,j,k,a-1,b,c)-
6*covD3D1(i,j,k,a,b+1,c) -6*covD3D1(i,j,k,a,b-1,c) -6*covD3D1(i,j,k,a,b,c+1) -
6*covD3D1(i,j,k,a,b,c-1) )/(-36);

                % Unchange boundary conditions
              if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                 (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
                 (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
                 (j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
                 (j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
                 (i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
                 (k==2&a==n-1) |(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
                 (k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
               covD3D1(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
              end
                   end
                  end
               end
```

```
        end
    end
end
save ('CD3D1.mat', 'covD3D1')
save ('Ccov.mat', 'cov')
clear covD3D1 cov

load ('CD3D2.mat', 'covD3D2')
load ('Ccov.mat', 'cov')
%10) D3D2
for i=N-1:n-1
    for j=2:N+1
        for k=2:n-1
            for a=2:N+1
                for b=N-1:n-1
                    for c=2:n-1

 covD3D2(i,j,k,a,b,c)=( covD3D2(i+1,j,k,a+1,b,c)+ covD3D2(i-1,j,k,a+1,b,c)+
covD3D2(i+1,j,k,a-1,b,c)+ covD3D2(i-1,j,k,a-1,b,c)+covD3D2(i+1,j,k,a,b+1,c)+
covD3D2(i-1,j,k,a,b+1,c)+ covD3D2(i+1,j,k,a,b-1,c)+ covD3D2(i-1,j,k,a,b-1,c) +
covD3D2(i+1,j,k,a,b,c+1)+ covD3D2(i-1,j,k,a,b,c+1)+ covD3D2(i+1,j,k,a,b,c-1)+
covD3D2(i-1,j,k,a,b,c-1)+covD3D2(i,j+1,k,a+1,b,c)+ covD3D2(i,j-1,k,a+1,b,c)+
covD3D2(i,j+1,k,a-1,b,c)+ covD3D2(i,j-1,k,a-1,b,c) +covD3D2(i,j+1,k,a,b+1,c)+
covD3D2(i,j-1,k,a,b+1,c)+ covD3D2(i,j+1,k,a,b-1,c)+ covD3D2(i,j-1,k,a,b-1,c)+
covD3D2(i,j+1,k,a,b,c+1)+ covD3D2(i,j-1,k,a,b,c+1)+ covD3D2(i,j+1,k,a,b,c-1)+
covD3D2(i,j-1,k,a,b,c-1)+covD3D2(i,j,k+1,a+1,b,c)+ covD3D2(i,j,k-1,a+1,b,c)+
covD3D2(i,j,k+1,a-1,b,c)+ covD3D2(i,j,k-1,a-1,b,c)+covD3D2(i,j,k+1,a,b+1,c)+
covD3D2(i,j,k-1,a,b+1,c)+ covD3D2(i,j,k+1,a,b-1,c)+ covD3D2(i,j,k-1,a,b-1,c)+
covD3D2(i,j,k+1,a,b,c+1)+ covD3D2(i,j,k-1,a,b,c+1)+ covD3D2(i,j,k+1,a,b,c-1)+
```

```
covD3D2(i,j,k-1,a,b,c-1)-6*covD3D2(i+1,j,k,a,b,c) -6*covD3D2(i-1,j,k,a,b,c) -
6*covD3D2(i,j+1,k,a,b,c) -6*covD3D2(i,j-1,k,a,b,c)-6*covD3D2(i,j,k+1,a,b,c) –
6*covD3D2(i,j,k-1,a,b,c) -6*covD3D2(i,j,k,a+1,b,c) -6*covD3D2(i,j,k,a-1,b,c)-
6*covD3D2(i,j,k,a,b+1,c) -6*covD3D2(i,j,k,a,b-1,c) -6*covD3D2(i,j,k,a,b,c+1) -
6*covD3D2(i,j,k,a,b,c-1) )/(-36);

                % Unchange boundary conditions
              if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                 (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
                 (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
                 (j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
                 (j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
                 (i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
                 (k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
                 (k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
               covD3D2(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
              end
                   end
                  end
                end
              end
            end
          end
        end
        save ('CD3D2.mat', 'covD3D2')
        save ('Ccov.mat', 'cov')
        clear covD3D2 cov

        load ('CD3D3.mat', 'covD3D3')
        load ('Ccov.mat', 'cov')
        %11) D3D3
```

```
for i=N-1:n-1
    for j=2:N+1
        for k=2:n-1
            for a=N-1:n-1
                for b=2:N+1
                    for c=2:n-1

 covD3D3(i,j,k,a,b,c)=( covD3D3(i+1,j,k,a+1,b,c)+ covD3D3(i-1,j,k,a+1,b,c)+
covD3D3(i+1,j,k,a-1,b,c)+ covD3D3(i-1,j,k,a-1,b,c)+covD3D3(i+1,j,k,a,b+1,c)+
covD3D3(i-1,j,k,a,b+1,c)+ covD3D3(i+1,j,k,a,b-1,c)+ covD3D3(i-1,j,k,a,b-1,c) +
covD3D3(i+1,j,k,a,b,c+1)+ covD3D3(i-1,j,k,a,b,c+1)+ covD3D3(i+1,j,k,a,b,c-1)+
covD3D3(i-1,j,k,a,b,c-1)+covD3D3(i,j+1,k,a+1,b,c)+ covD3D3(i,j-1,k,a+1,b,c)+
covD3D3(i,j+1,k,a-1,b,c)+ covD3D3(i,j-1,k,a-1,b,c) +covD3D3(i,j+1,k,a,b+1,c)+
covD3D3(i,j-1,k,a,b+1,c)+ covD3D3(i,j+1,k,a,b-1,c)+ covD3D3(i,j-1,k,a,b-1,c)+
covD3D3(i,j+1,k,a,b,c+1)+ covD3D3(i,j-1,k,a,b,c+1)+ covD3D3(i,j+1,k,a,b,c-1)+
covD3D3(i,j-1,k,a,b,c-1)+covD3D3(i,j,k+1,a+1,b,c)+ covD3D3(i,j,k-1,a+1,b,c)+
covD3D3(i,j,k+1,a-1,b,c)+ covD3D3(i,j,k-1,a-1,b,c)+covD3D3(i,j,k+1,a,b+1,c)+
covD3D3(i,j,k-1,a,b+1,c)+ covD3D3(i,j,k+1,a,b-1,c)+ covD3D3(i,j,k-1,a,b-1,c)+
covD3D3(i,j,k+1,a,b,c+1)+ covD3D3(i,j,k-1,a,b,c+1)+ covD3D3(i,j,k+1,a,b,c-1)+
covD3D3(i,j,k-1,a,b,c-1)-6*covD3D3(i+1,j,k,a,b,c) -6*covD3D3(i-1,j,k,a,b,c) -
6*covD3D3(i,j+1,k,a,b,c) -6*covD3D3(i,j-1,k,a,b,c)-6*covD3D3(i,j,k+1,a,b,c) –
6*covD3D3(i,j,k-1,a,b,c) -6*covD3D3(i,j,k,a+1,b,c) -6*covD3D3(i,j,k,a-1,b,c)-
6*covD3D3(i,j,k,a,b+1,c) -6*covD3D3(i,j,k,a,b-1,c) -6*covD3D3(i,j,k,a,b,c+1) -
6*covD3D3(i,j,k,a,b,c-1) )/(-36);

                    % Unchange boundary conditions
                if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                    (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
                    (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
                    (j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
```

```
                (j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
                (i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
                (k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
                (k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
              covD3D3(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
            end
                end
              end
            end
          end
        end
    end
save ('CD3D3.mat', 'covD3D3')
save ('Ccov.mat', 'cov')
clear covD3D3 cov

load ('CD3D4.mat', 'covD3D4')
load ('Ccov.mat', 'cov')
%12) D3D4
for i=N-1:n-1
    for j=2:N+1
        for k=2:n-1
            for a=N-1:n-1
                for b=N-1:n-1
                    for c=2:n-1

covD3D4(i,j,k,a,b,c)=( covD3D4(i+1,j,k,a+1,b,c)+ covD3D4(i-1,j,k,a+1,b,c)+
covD3D4(i+1,j,k,a-1,b,c)+ covD3D4(i-1,j,k,a-1,b,c)+covD3D4(i+1,j,k,a,b+1,c)+
covD3D4(i-1,j,k,a,b+1,c)+ covD3D4(i+1,j,k,a,b-1,c)+ covD3D4(i-1,j,k,a,b-1,c) +
covD3D4(i+1,j,k,a,b,c+1)+ covD3D4(i-1,j,k,a,b,c+1)+ covD3D4(i+1,j,k,a,b,c-1)+
```

```
covD3D4(i-1,j,k,a,b,c-1)+covD3D4(i,j+1,k,a+1,b,c)+ covD3D4(i,j-1,k,a+1,b,c)+
covD3D4(i,j+1,k,a-1,b,c)+ covD3D4(i,j-1,k,a-1,b,c) +covD3D4(i,j+1,k,a,b+1,c)+
covD3D4(i,j-1,k,a,b+1,c)+ covD3D4(i,j+1,k,a,b-1,c)+ covD3D4(i,j-1,k,a,b-1,c)+
covD3D4(i,j+1,k,a,b,c+1)+ covD3D4(i,j-1,k,a,b,c+1)+ covD3D4(i,j+1,k,a,b,c-1)+
covD3D4(i,j-1,k,a,b,c-1)+covD3D4(i,j,k+1,a+1,b,c)+ covD3D4(i,j,k-1,a+1,b,c)+
covD3D4(i,j,k+1,a-1,b,c)+ covD3D4(i,j,k-1,a-1,b,c)+covD3D4(i,j,k+1,a,b+1,c)+
covD3D4(i,j,k-1,a,b+1,c)+ covD3D4(i,j,k+1,a,b-1,c)+ covD3D4(i,j,k-1,a,b-1,c)+
covD3D4(i,j,k+1,a,b,c+1)+ covD3D4(i,j,k-1,a,b,c+1)+ covD3D4(i,j,k+1,a,b,c-1)+
covD3D4(i,j,k-1,a,b,c-1)-6*covD3D4(i+1,j,k,a,b,c) -6*covD3D4(i-1,j,k,a,b,c) -
6*covD3D4(i,j+1,k,a,b,c) -6*covD3D4(i,j-1,k,a,b,c)-6*covD3D4(i,j,k+1,a,b,c) –
6*covD3D4(i,j,k-1,a,b,c) -6*covD3D4(i,j,k,a+1,b,c) -6*covD3D4(i,j,k,a-1,b,c)-
6*covD3D4(i,j,k,a,b+1,c) -6*covD3D4(i,j,k,a,b-1,c) -6*covD3D4(i,j,k,a,b,c+1) -
6*covD3D4(i,j,k,a,b,c-1) )/(-36);

                % Unchange boundary conditions
              if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                 (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
                 (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
                 (j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
                 (j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
                 (i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
                 (k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
                 (k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
                covD3D4(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
               end
                   end
                  end
                end
              end
            end
```

```
end
save ('CD3D4.mat', 'covD3D4')
save ('Ccov.mat', 'cov')
clear covD3D4 cov

load ('CD4D1.mat', 'covD4D1')
load ('Ccov.mat', 'cov')
%13) D4D1
for i=N-1:n-1
   for j=N-1:n-1
      for k=2:n-1
         for a=2:N+1
            for b=2:N+1
               for c=2:n-1

covD4D1(i,j,k,a,b,c)=( covD4D1(i+1,j,k,a+1,b,c)+ covD4D1(i-1,j,k,a+1,b,c)+
covD4D1(i+1,j,k,a-1,b,c)+ covD4D1(i-1,j,k,a-1,b,c)+covD4D1(i+1,j,k,a,b+1,c)+
covD4D1(i-1,j,k,a,b+1,c)+ covD4D1(i+1,j,k,a,b-1,c)+ covD4D1(i-1,j,k,a,b-1,c) +
covD4D1(i+1,j,k,a,b,c+1)+ covD4D1(i-1,j,k,a,b,c+1)+ covD4D1(i+1,j,k,a,b,c-1)+
covD4D1(i-1,j,k,a,b,c-1)+covD4D1(i,j+1,k,a+1,b,c)+ covD4D1(i,j-1,k,a+1,b,c)+
covD4D1(i,j+1,k,a-1,b,c)+ covD4D1(i,j-1,k,a-1,b,c) +covD4D1(i,j+1,k,a,b+1,c)+
covD4D1(i,j-1,k,a,b+1,c)+ covD4D1(i,j+1,k,a,b-1,c)+ covD4D1(i,j-1,k,a,b-1,c)+
covD4D1(i,j+1,k,a,b,c+1)+ covD4D1(i,j-1,k,a,b,c+1)+ covD4D1(i,j+1,k,a,b,c-1)+
covD4D1(i,j-1,k,a,b,c-1)+covD4D1(i,j,k+1,a+1,b,c)+ covD4D1(i,j,k-1,a+1,b,c)+
covD4D1(i,j,k+1,a-1,b,c)+ covD4D1(i,j,k-1,a-1,b,c)+covD4D1(i,j,k+1,a,b+1,c)+
covD4D1(i,j,k-1,a,b+1,c)+ covD4D1(i,j,k+1,a,b-1,c)+ covD4D1(i,j,k-1,a,b-1,c)+
covD4D1(i,j,k+1,a,b,c+1)+ covD4D1(i,j,k-1,a,b,c+1)+ covD4D1(i,j,k+1,a,b,c-1)+
covD4D1(i,j,k-1,a,b,c-1)-6*covD4D1(i+1,j,k,a,b,c) -6*covD4D1(i-1,j,k,a,b,c) -
6*covD4D1(i,j+1,k,a,b,c) -6*covD4D1(i,j-1,k,a,b,c)-6*covD4D1(i,j,k+1,a,b,c) –
```

```
6*covD4D1(i,j,k-1,a,b,c) -6*covD4D1(i,j,k,a+1,b,c) -6*covD4D1(i,j,k,a-1,b,c)-
6*covD4D1(i,j,k,a,b+1,c) -6*covD4D1(i,j,k,a,b-1,c) -6*covD4D1(i,j,k,a,b,c+1) -
6*covD4D1(i,j,k,a,b,c-1) )/(-36);

                % Unchange boundary conditions
              if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                 (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
                 (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
                 (j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
                 (j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
                 (i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
                 (k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
                 (k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
               covD4D1(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
              end
                  end
                 end
               end
             end
           end
        end
save ('CD4D1.mat', 'covD4D1')
save ('Ccov.mat', 'cov')
clear covD4D1 cov

load ('CD4D2.mat', 'covD4D2')
load ('Ccov.mat', 'cov')
%14) D4D2
for i=N-1:n-1
   for j=N-1:n-1
```

```
for k=2:n-1
   for a=2:N+1
      for b=N-1:n-1
         for c=2:n-1

 covD4D2(i,j,k,a,b,c)=( covD4D2(i+1,j,k,a+1,b,c)+ covD4D2(i-1,j,k,a+1,b,c)+
covD4D2(i+1,j,k,a-1,b,c)+ covD4D2(i-1,j,k,a-1,b,c)+covD4D2(i+1,j,k,a,b+1,c)+
covD4D2(i-1,j,k,a,b+1,c)+ covD4D2(i+1,j,k,a,b-1,c)+ covD4D2(i-1,j,k,a,b-1,c) +
covD4D2(i+1,j,k,a,b,c+1)+ covD4D2(i-1,j,k,a,b,c+1)+ covD4D2(i+1,j,k,a,b,c-1)+
covD4D2(i-1,j,k,a,b,c-1)+covD4D2(i,j+1,k,a+1,b,c)+ covD4D2(i,j-1,k,a+1,b,c)+
covD4D2(i,j+1,k,a-1,b,c)+ covD4D2(i,j-1,k,a-1,b,c) +covD4D2(i,j+1,k,a,b+1,c)+
covD4D2(i,j-1,k,a,b+1,c)+ covD4D2(i,j+1,k,a,b-1,c)+ covD4D2(i,j-1,k,a,b-1,c)+
covD4D2(i,j+1,k,a,b,c+1)+ covD4D2(i,j-1,k,a,b,c+1)+ covD4D2(i,j+1,k,a,b,c-1)+
covD4D2(i,j-1,k,a,b,c-1)+covD4D2(i,j,k+1,a+1,b,c)+ covD4D2(i,j,k-1,a+1,b,c)+
covD4D2(i,j,k+1,a-1,b,c)+ covD4D2(i,j,k-1,a-1,b,c)+covD4D2(i,j,k+1,a,b+1,c)+
covD4D2(i,j,k-1,a,b+1,c)+ covD4D2(i,j,k+1,a,b-1,c)+ covD4D2(i,j,k-1,a,b-1,c)+
covD4D2(i,j,k+1,a,b,c+1)+ covD4D2(i,j,k-1,a,b,c+1)+ covD4D2(i,j,k+1,a,b,c-1)+
covD4D2(i,j,k-1,a,b,c-1)-6*covD4D2(i+1,j,k,a,b,c) -6*covD4D2(i-1,j,k,a,b,c) -
6*covD4D2(i,j+1,k,a,b,c) -6*covD4D2(i,j-1,k,a,b,c)-6*covD4D2(i,j,k+1,a,b,c) –
6*covD4D2(i,j,k-1,a,b,c) -6*covD4D2(i,j,k,a+1,b,c) -6*covD4D2(i,j,k,a-1,b,c)-
6*covD4D2(i,j,k,a,b+1,c) -6*covD4D2(i,j,k,a,b-1,c) -6*covD4D2(i,j,k,a,b,c+1) -
6*covD4D2(i,j,k,a,b,c-1) )/(-36);

            % Unchange boundary conditions
         if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
            (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
            (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
            (j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
            (j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
```

```
                (i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
                (k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
                (k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
              covD4D2(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
            end
                end
              end
            end
          end
        end
        save ('CD4D2.mat', 'covD4D2')
        save ('Ccov.mat', 'cov')
        clear covD4D2 cov

        load ('CD4D3.mat', 'covD4D3')
        load ('Ccov.mat', 'cov')
        %15) D4D3
        for i=N-1:n-1
          for j=N-1:n-1
            for k=2:n-1
              for a=N-1:n-1
                for b=2:N+1
                  for c=2:n-1

covD4D3(i,j,k,a,b,c)=( covD4D3(i+1,j,k,a+1,b,c)+ covD4D3(i-1,j,k,a+1,b,c)+
covD4D3(i+1,j,k,a-1,b,c)+ covD4D3(i-1,j,k,a-1,b,c)+covD4D3(i+1,j,k,a,b+1,c)+
covD4D3(i-1,j,k,a,b+1,c)+ covD4D3(i+1,j,k,a,b-1,c)+ covD4D3(i-1,j,k,a,b-1,c) +
covD4D3(i+1,j,k,a,b,c+1)+ covD4D3(i-1,j,k,a,b,c+1)+ covD4D3(i+1,j,k,a,b,c-1)+
```

```
covD4D3(i-1,j,k,a,b,c-1)+covD4D3(i,j+1,k,a+1,b,c)+ covD4D3(i,j-1,k,a+1,b,c)+
covD4D3(i,j+1,k,a-1,b,c)+ covD4D3(i,j-1,k,a-1,b,c) +covD4D3(i,j+1,k,a,b+1,c)+
covD4D3(i,j-1,k,a,b+1,c)+ covD4D3(i,j+1,k,a,b-1,c)+ covD4D3(i,j-1,k,a,b-1,c)+
covD4D3(i,j+1,k,a,b,c+1)+ covD4D3(i,j-1,k,a,b,c+1)+ covD4D3(i,j+1,k,a,b,c-1)+
covD4D3(i,j-1,k,a,b,c-1)+covD4D3(i,j,k+1,a+1,b,c)+ covD4D3(i,j,k-1,a+1,b,c)+
covD4D3(i,j,k+1,a-1,b,c)+ covD4D3(i,j,k-1,a-1,b,c)+covD4D3(i,j,k+1,a,b+1,c)+
covD4D3(i,j,k-1,a,b+1,c)+ covD4D3(i,j,k+1,a,b-1,c)+ covD4D3(i,j,k-1,a,b-1,c)+
covD4D3(i,j,k+1,a,b,c+1)+ covD4D3(i,j,k-1,a,b,c+1)+ covD4D3(i,j,k+1,a,b,c-1)+
covD4D3(i,j,k-1,a,b,c-1)-6*covD4D3(i+1,j,k,a,b,c) -6*covD4D3(i-1,j,k,a,b,c) -
6*covD4D3(i,j+1,k,a,b,c) -6*covD4D3(i,j-1,k,a,b,c)-6*covD4D3(i,j,k+1,a,b,c) –
6*covD4D3(i,j,k-1,a,b,c) -6*covD4D3(i,j,k,a+1,b,c) -6*covD4D3(i,j,k,a-1,b,c)-
6*covD4D3(i,j,k,a,b+1,c) -6*covD4D3(i,j,k,a,b-1,c) -6*covD4D3(i,j,k,a,b,c+1) -
6*covD4D3(i,j,k,a,b,c-1) )/(-36);

                % Unchange boundary conditions
              if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                 (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
                 (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
                 (j==n-1&a==2)|(i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
                 (j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
                 (i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
                 (k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
                 (k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
                covD4D3(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
              end
                   end
                 end
               end
             end
           end
```

```
end
save ('CD4D3.mat', 'covD4D3')
save ('Ccov.mat', 'cov')
clear covD4D3 cov

load ('CD4D4.mat', 'covD4D4')
load ('Ccov.mat', 'cov')
%16) D4D4
for i=N-1:n-1
   for j=N-1:n-1
      for k=2:n-1
         for a=N-1:n-1
            for b=N-1:n-1
               for c=2:n-1

covD4D4(i,j,k,a,b,c)=( covD4D4(i+1,j,k,a+1,b,c)+ covD4D4(i-1,j,k,a+1,b,c)+
covD4D4(i+1,j,k,a-1,b,c)+ covD4D4(i-1,j,k,a-1,b,c)+covD4D4(i+1,j,k,a,b+1,c)+
covD4D4(i-1,j,k,a,b+1,c)+ covD4D4(i+1,j,k,a,b-1,c)+ covD4D4(i-1,j,k,a,b-1,c) +
covD4D4(i+1,j,k,a,b,c+1)+ covD4D4(i-1,j,k,a,b,c+1)+ covD4D4(i+1,j,k,a,b,c-1)+
covD4D4(i-1,j,k,a,b,c-1)+covD4D4(i,j+1,k,a+1,b,c)+ covD4D4(i,j-1,k,a+1,b,c)+
covD4D4(i,j+1,k,a-1,b,c)+ covD4D4(i,j-1,k,a-1,b,c) +covD4D4(i,j+1,k,a,b+1,c)+
covD4D4(i,j-1,k,a,b+1,c)+ covD4D4(i,j+1,k,a,b-1,c)+ covD4D4(i,j-1,k,a,b-1,c)+
covD4D4(i,j+1,k,a,b,c+1)+ covD4D4(i,j-1,k,a,b,c+1)+ covD4D4(i,j+1,k,a,b,c-1)+
 covD4D4(i,j-1,k,a,b,c-1)+covD4D4(i,j,k+1,a+1,b,c)+ covD4D4(i,j,k-1,a+1,b,c)+
covD4D4(i,j,k+1,a-1,b,c)+ covD4D4(i,j,k-1,a-1,b,c)+covD4D4(i,j,k+1,a,b+1,c)+
 covD4D4(i,j,k-1,a,b+1,c)+ covD4D4(i,j,k+1,a,b-1,c)+ covD4D4(i,j,k-1,a,b-1,c)+
covD4D4(i,j,k+1,a,b,c+1)+ covD4D4(i,j,k-1,a,b,c+1)+ covD4D4(i,j,k+1,a,b,c-1)+
covD4D4(i,j,k-1,a,b,c-1)-6*covD4D4(i+1,j,k,a,b,c) -6*covD4D4(i-1,j,k,a,b,c) -
6*covD4D4(i,j+1,k,a,b,c) -6*covD4D4(i,j-1,k,a,b,c)-6*covD4D4(i,j,k+1,a,b,c) -6*covD4D4(i,j,k-
```

```
1,a,b,c) -6*covD4D4(i,j,k,a+1,b,c) -6*covD4D4(i,j,k,a-1,b,c)-6*covD4D4(i,j,k,a,b+1,c) -
6*covD4D4(i,j,k,a,b-1,c) -6*covD4D4(i,j,k,a,b,c+1) -6*covD4D4(i,j,k,a,b,c-1) )/(-36);

                % Unchange boundary conditions
               if (j==2&b==2)|(j==2&b==n-1)|(j==n-1&b==2) |(j==2&a==n-1)|(i==n-1&b==2)|
                  (j==2&a==2)|(i==2&b==2)|(j==2&c==n-1)|(k==n-1&b==2)|(j==2&c==2)|
                  (k==2&b==2)|(j==n-1&b==n-1)|(j==n-1&a==n-1)|(i==n-1&b==n-1)|
                  (j==n-1&a==2)| (i==2&b==n-1)|(j==n-1&c==n-1)|(k==n-1&b==n-1)|
                  (j==n-1&c==2)|(k==2&b==n-1)|(i==n-1&a==n-1)|(i==n-1&a==2)|
                  (i==2&a==n-1)|(i==n-1&c==n-1)|(k==n-1&a==n-1)|(i==n-1&c==2)|
                  (k==2&a==n-1)|(i==2&a==2)|(i==2&c==n-1)|(k==n-1&a==2)|(i==2&c==2)|
                  (k==2&a==2)|(k==n-1&c==n-1)|(k==n-1&c==2)|(k==2&c==n-1)|(k==2&c==2)
                covD4D4(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
               end
                    end
                end
               end
              end
            end
          end
        save ('CD4D4.mat', 'covD4D4')
        save ('Ccov.mat', 'cov')
        clear covD4D4 cov


%       Composing each sub-domain into whole domain to check convergent condition
 load('CD1D1.mat', 'covD1D1')              % Loading variables covD1D1 from file CD1D1.mat
load('CD1D2.mat', 'covD1D2')               % Loading variables covD1D2 from file CD1D2.mat
load('CD1D3.mat', 'covD1D3')               % Loading variables covD1D3 from file CD1D3.mat
load('CD1D4.mat', 'covD1D4')               % Loading variables covD1D4 from file CD1D4.mat
```

```
load('CD2D1.mat', 'covD2D1')            % Loading variables covD2D1 from file CD2D1.mat
load('CD2D2.mat', 'covD2D2')            % Loading variables covD2D2 from file CD2D2.mat
load('CD2D3.mat', 'covD2D3')            % Loading variables covD2D3 from file CD2D3.mat
load('CD2D4.mat', 'covD2D4')            % Loading variables covD2D4 from file CD2D4.mat
load('CD3D1.mat', 'covD3D1')            % Loading variables covD3D1 from file CD3D1.mat
load('CD3D2.mat', 'covD3D2')            % Loading variables covD3D2 from file CD3D2.mat
load('CD3D3.mat', 'covD3D3')            % Loading variables covD3D3 from file CD3D3.mat
load('CD3D4.mat', 'covD3D4')            % Loading variables covD3D4 from file CD3D4.mat
load('CD4D1.mat', 'covD4D1')            % Loading variables covD4D1 from file CD4D1.mat
load('CD4D2.mat', 'covD4D2')            % Loading variables covD4D2 from file CD4D2.mat
load('CD4D3.mat', 'covD4D3')            % Loading variables covD4D3 from file CD4D3.mat
load('CD4D4.mat', 'covD4D4')            % Loading variables covD4D4 from file CD4D4.mat
load ('Ccov.mat', 'cov')

err=0;
 for i=2:n-1
    for j=2:n-1
       for k=2:n-1
          for a=2:n-1
             for b=2:n-1
                for c=2:n-1

                   if     (i>=1&i<=N)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=1&b<=N)
                cov_new(i,j,k,a,b,c)=covD1D1(i,j,k,a,b,c);
                   elseif  (i>=1&i<=N)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=N&b<=n)
                cov_new(i,j,k,a,b,c)=covD1D2(i,j,k,a,b,c);
                   elseif  (i>=1&i<=N)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=1&b<=N)
                cov_new(i,j,k,a,b,c)=covD1D3(i,j,k,a,b,c);
                   elseif  (i>=1&i<=N)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=N&b<=n)
```

```
cov_new(i,j,k,a,b,c)=covD1D4(i,j,k,a,b,c);
   elseif (i>=1&i<=N)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=1&b<=N)
cov_new(i,j,k,a,b,c)=covD2D1(i,j,k,a,b,c);
   elseif (i>=1&i<=N)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=N&b<=n)
cov_new(i,j,k,a,b,c)=covD2D2(i,j,k,a,b,c);
   elseif (i>=1&i<=N)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=1&b<=N)
cov_new(i,j,k,a,b,c)=covD2D3(i,j,k,a,b,c);
   elseif (i>=1&i<=N)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=N&b<=n)
cov_new(i,j,k,a,b,c)=covD2D4(i,j,k,a,b,c);
   elseif (i>=N&i<=n)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=1&b<=N)
cov_new(i,j,k,a,b,c)=covD3D1(i,j,k,a,b,c);
   elseif (i>=N&i<=n)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=N&b<=n)
cov_new(i,j,k,a,b,c)=covD3D2(i,j,k,a,b,c);
   elseif (i>=N&i<=n)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=1&b<=N)
cov_new(i,j,k,a,b,c)=covD3D3(i,j,k,a,b,c);
   elseif (i>=N&i<=n)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=N&b<=n)
cov_new(i,j,k,a,b,c)=covD3D4(i,j,k,a,b,c);
   elseif (i>=N&i<=n)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=1&b<=N)
cov_new(i,j,k,a,b,c)=covD4D1(i,j,k,a,b,c);
   elseif (i>=N&i<=n)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=N&b<=n)
cov_new(i,j,k,a,b,c)=covD4D2(i,j,k,a,b,c);
   elseif (i>=N&i<=n)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=1&b<=N)
cov_new(i,j,k,a,b,c)=covD4D3(i,j,k,a,b,c);
   elseif (i>=N&i<=n)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=N&b<=n)
cov_new(i,j,k,a,b,c)=covD4D4(i,j,k,a,b,c);
    end
```

```
                % Ussing Successive over-relaxation method
                err=( (abs(cov_new(i,j,k,a,b,c)-cov(i,j,k,a,b,c))*100)/cov_new(i,j,k,a,b,c) )+err;
                cov(i,j,k,a,b,c)=WF*cov_new(i,j,k,a,b,c)+(1-WF)*cov(i,j,k,a,b,c);
                end
              end
            end
          end
      end
end
iteration=iteration+1;

  % To Update covariance of each sub-domain
for i=2:n-1
   for j=2:n-1
      for k=2:n-1
         for a=2:n-1
            for b=2:n-1
               for c=2:n-1

                  if     (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
               covD1D1(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
                  end
                  if  (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
               covD1D2(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
                  end
                  if  (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
               covD1D3(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
                  end
                  if  (i>=1&i<=N+2)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
               covD1D4(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
```

```
    end
    if (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
covD2D1(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
covD2D2(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
covD2D3(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=1&i<=N+2)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
covD2D4(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
covD3D1(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
covD3D2(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
covD3D3(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=N-2&i<=n)&(j>=1&j<=N+2)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
covD3D4(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
    if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=1&b<=N+2)
covD4D1(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
    end
```

```
            if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=1&a<=N+2)&(b>=N-2&b<=n)
          covD4D2(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
            end
            if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=1&b<=N+2)
          covD4D3(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
            end
            if (i>=N-2&i<=n)&(j>=N-2&j<=n)&(a>=N-2&a<=n)&(b>=N-2&b<=n)
          covD4D4(i,j,k,a,b,c)=cov(i,j,k,a,b,c);
            end
          end
        end
      end
    end
  end
end

save('CD1D1.mat', 'covD1D1')          % Saving variables covD1D1 into file CD1D1.mat
save('CD1D2.mat', 'covD1D2')          % Saving variables covD1D2 into file CD1D2.mat
save('CD1D3.mat', 'covD1D3')          % Saving variables covD1D3 into file CD1D3.mat
save('CD1D4.mat', 'covD1D4')          % Saving variables covD1D4 into file CD1D4.mat
save('CD2D1.mat', 'covD2D1')          % Saving variables covD2D1 into file CD2D1.mat
save('CD2D2.mat', 'covD2D2')          % Saving variables covD2D2 into file CD2D2.mat
save('CD2D3.mat', 'covD2D3')          % Saving variables covD2D3 into file CD2D3.mat
save('CD2D4.mat', 'covD2D4')          % Saving variables covD2D4 into file CD2D4.mat
save('CD3D1.mat', 'covD3D1')          % Saving variables covD3D1 into file CD3D1.mat
save('CD3D2.mat', 'covD3D2')          % Saving variables covD3D2 into file CD3D2.mat
save('CD3D3.mat', 'covD3D3')          % Saving variables covD3D3 into file CD3D3.mat
save('CD3D4.mat', 'covD3D4')          % Saving variables covD3D4 into file CD3D4.mat
save('CD4D1.mat', 'covD4D1')          % Saving variables covD4D1 into file CD4D1.mat
save('CD4D2.mat', 'covD4D2')          % Saving variables covD4D2 into file CD4D2.mat
```

```
save('CD4D3.mat', 'covD4D3')            % Saving variables covD4D3 into file CD4D3.mat
save('CD4D4.mat', 'covD4D4')            % Saving variables covD4D4 into file CD4D4.mat
save('Ccov.mat', 'cov')                 % Saving variables cov into file Ccov.mat
clear covD1D2 covD1D3 covD1D4 covD2D1 covD2D2 covD2D3 covD2D4 covD3D1 covD3D2
covD3D3 covD3D4 covD4D1 covD4D2 covD4D3 covD4D4 cov
end

% To compose each sub-domain into whole domain
load('CD1D1.mat', 'covD1D1')            % Loading variables covD1D1 from file CD1D1.mat
load('CD1D2.mat', 'covD1D2')            % Loading variables covD1D2 from file CD1D2.mat
load('CD1D3.mat', 'covD1D3')            % Loading variables covD1D3 from file CD1D3.mat
load('CD1D4.mat', 'covD1D4')            % Loading variables covD1D4 from file CD1D4.mat
load('CD2D1.mat', 'covD2D1')            % Loading variables covD2D1 from file CD2D1.mat
load('CD2D2.mat', 'covD2D2')            % Loading variables covD2D2 from file CD2D2.mat
load('CD2D3.mat', 'covD2D3')            % Loading variables covD2D3 from file CD2D3.mat
load('CD2D4.mat', 'covD2D4')            % Loading variables covD2D4 from file CD2D4.mat
load('CD3D1.mat', 'covD3D1')            % Loading variables covD3D1 from file CD3D1.mat
load('CD3D2.mat', 'covD3D2')             % Loading variables covD3D2 from file CD3D2.mat
load('CD3D3.mat', 'covD3D3')            % Loading variables covD3D3 from file CD3D3.mat
load('CD3D4.mat', 'covD3D4')            % Loading variables covD3D4 from file CD3D4.mat
load('CD4D1.mat', 'covD4D1')            % Loading variables covD4D1 from file CD4D1.mat
load('CD4D2.mat', 'covD4D2')            % Loading variables covD4D2 from file CD4D2.mat
load('CD4D3.mat', 'covD4D3')            % Loading variables covD4D3 from file CD4D3.mat
load('CD4D4.mat', 'covD4D4')            % Loading variables covD4D4 from file CD4D4.mat

 for i=1:n
    for j=1:n
       for k=1:n
          for a=1:n
             for b=1:n
```

```
for c=1:n

   if     (i>=1&i<=N)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=1&b<=N)
cov(i,j,k,a,b,c)=covD1D1(i,j,k,a,b,c);
   elseif  (i>=1&i<=N)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=N&b<=n)
cov(i,j,k,a,b,c)=covD1D2(i,j,k,a,b,c);
   elseif  (i>=1&i<=N)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=1&b<=N)
cov(i,j,k,a,b,c)=covD1D3(i,j,k,a,b,c);
   elseif  (i>=1&i<=N)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=N&b<=n)
cov(i,j,k,a,b,c)=covD1D4(i,j,k,a,b,c);

   elseif  (i>=1&i<=N)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=1&b<=N)
cov(i,j,k,a,b,c)=covD2D1(i,j,k,a,b,c);
   elseif  (i>=1&i<=N)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=N&b<=n)
cov(i,j,k,a,b,c)=covD2D2(i,j,k,a,b,c);
   elseif  (i>=1&i<=N)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=1&b<=N)
cov(i,j,k,a,b,c)=covD2D3(i,j,k,a,b,c);
   elseif  (i>=1&i<=N)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=N&b<=n)
cov(i,j,k,a,b,c)=covD2D4(i,j,k,a,b,c);

   elseif  (i>=N&i<=n)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=1&b<=N)
cov(i,j,k,a,b,c)=covD3D1(i,j,k,a,b,c);
   elseif  (i>=N&i<=n)&(j>=1&j<=N)&(a>=1&a<=N)&(b>=N&b<=n)
cov(i,j,k,a,b,c)=covD3D2(i,j,k,a,b,c);
   elseif  (i>=N&i<=n)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=1&b<=N)
cov(i,j,k,a,b,c)=covD3D3(i,j,k,a,b,c);
   elseif  (i>=N&i<=n)&(j>=1&j<=N)&(a>=N&a<=n)&(b>=N&b<=n)
cov(i,j,k,a,b,c)=covD3D4(i,j,k,a,b,c);
```

```
                elseif  (i>=N&i<=n)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=1&b<=N)
            cov(i,j,k,a,b,c)=covD4D1(i,j,k,a,b,c);
                elseif  (i>=N&i<=n)&(j>=N&j<=n)&(a>=1&a<=N)&(b>=N&b<=n)
            cov(i,j,k,a,b,c)=covD4D2(i,j,k,a,b,c);
                elseif  (i>=N&i<=n)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=1&b<=N)
            cov(i,j,k,a,b,c)=covD4D3(i,j,k,a,b,c);
                elseif  (i>=N&i<=n)&(j>=N&j<=n)&(a>=N&a<=n)&(b>=N&b<=n)
            cov(i,j,k,a,b,c)=covD4D4(i,j,k,a,b,c);
                end
            end
          end
        end
      end
    end
  end

clear covD1D2 covD1D3 covD1D4 covD2D1 covD2D2 covD2D3 covD2D4 covD3D1 covD3D2
covD3D3 covD3D4 covD4D1 covD4D2 covD4D3 covD4D4
stop=datestr(now);                  % Marking ending time of calculation
t1=toc;
tot=sec2hms(t1);
% ---------------------FINDNIG AN ERROR OF THE MODEL ----------------------------
%    EXACT SOLUTION
      for i=1:n
          for j=1:n
             for k=1:n
                for a=1:n
                   for b=1:n
                      for c=1:n
            x1=(j-2)*delta; y1=(i-2)*delta; z1=(k-2)*delta;
```

```
            x2=(b-2)*delta; y2=(a-2)*delta; z2=(c-2)*delta;
            cov_exact(i,j,k,a,b,c) =exactV(x1,y1,z1,x2,y2,z2,c1,c2,c3,c4,c5,c6,c7,c8,c9,
                                           c10,c11,c12,k1,k2,k3,k4,k5,k6,k7);
                end
              end
            end
          end
        end
    end

    %   To generate symmetry covariance of an exact solution
    for i=1:n
      for j=1:n
        for k=1:n
          for a=1:n
            for b=1:n
              for c=1:n
                cov_exact(i,j,k,a,b,c)=( cov_exact(i,j,k,a,b,c)+ cov_exact(a,b,c,i,j,k) )/2.0;
                cov_exact(a,b,c,i,j,k)=cov_exact(i,j,k,a,b,c);
              end
            end
          end
        end
      end
    end

app=cov(3:n-2,3:n-2,3:n-2,3:n-2,3:n-2,3:n-2);
exact=cov_exact(3:n-2,3:n-2,3:n-2,3:n-2,3:n-2,3:n-2);
abs_error=(abs(app-exact)*100)./exact;
```

```
error=reshape(abs_error,1,(n-4)^6);
min_error=min(error);
max_error=max(error);
average_error=mean(error);

% ------------------To save an output into file cov_dom_app3D.txt ----------------------------------
fid=fopen('cov_dom_app3D.txt','a');
fprintf(fid,'-------------------------------------------------------------------------\n');
fprintf(fid,'n= %-5.2f\n',n);
fprintf(fid,'Stopping Tolerance= %2.10f\n',stt);
fprintf(fid,'Iteration= %-5.2f\n',iteration);
fprintf(fid,'\n');
 % Showing solution
for i=[3,n-2]
   for j=[3,n-2]
      for k=[3,n-2]
         for a=[3,n-2]
            for b=[3,n-2]
               for c=[3,n-2]
                  x1=(j-2)*delta; y1=(i-2)*delta; z1=(k-2)*delta;
                  x2=(b-2)*delta; y2=(a-2)*delta; z2=(c-2)*delta;
            fprintf(fid,'cov[t1(%-2.8f, %-2.8f, %-2.8f), t2(%-2.8f, %-2.8f, %-2.8f)]= %-
                                 3.5f\n',x1,y1,z1,x2,y2,z2,cov(i,j,k,a,b,c));
               end
            end
         end
      end
   end
end
fprintf(fid,'-----------------------------\n');
```

```
fprintf(fid,'\n');
fprintf(fid,'Starting time:\n');
                fprintf(fid,'%s\n',start);
fprintf(fid,'Ending time:\n');
                fprintf(fid,'%s\n',stop);
fprintf(fid,'-----------------------------\n');
fprintf(fid,'\n');
fprintf(fid,'Min Error:   %-3.6f  percent\n',min_error);
fprintf(fid,'Max Error:   %-3.6f  percent\n',max_error);
fprintf(fid,'Avg Error:   %-3.6f  percent\n',average_error);
fprintf(fid,'Total time:  %f ',tot);
% ------------------------------------------------------------------------
fprintf(fid,'\n------------------------------------------------------------------------');
fclose(fid)
% To delete file that save variable D1D1-D4D4 from disk
delete CD1D1.mat CD1D2.mat CD1D3.mat CD1D4.mat CD2D1.mat CD2D2.mat CD2D3.mat
CD2D4.mat CD3D1.mat CD3D2.mat CD3D3.mat CD3D4.mat CD4D1.mat CD4D2.mat
CD4D3.mat CD4D4.mat Ccov.mat
```

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	นายณัฐพงศ์ มูลผลึก
วัน เดือน ปี ที่เกิด	วันที่ 11 มีนาคม 2526
สถานที่เกิด	ร้อยเอ็ด
ประวัติการศึกษา	ปริญญาวิทยาศาสตรบัณฑิต (สาขาฟิสิกส์) จุฬาลงกรณ์มหาวิทยาลัย
ตำแหน่งหน้าที่การงานปัจจุบัน	
สถานที่ทำงานปัจจุบัน	
ผลงานดีเด่นและรางวัลทางวิชาการ	
ทุนการศึกษาที่ได้รับ	