ภาคผนวก

ภาคผนวก ก.

แบบเสนอโครงการวิจัย (Research project)

สำนักงานคณะกรรมการวิจัยแห่งชาติ แบบ ว-1ด

(ฉบับปรับปรุงปี พ.ศ. 2551)

แบบเสนอโครงการวิจัย (Research project)

ประกอบการเสนอของบประมาณ ประจำปีงบประมาณ พ.ศ. 2552 ตามมติคณะรัฐมนตรี

ชื่อโครงการวิจัย

(ภาษาไทย) **การพัฒนาระบบการคำนวณแบบขนานสำหรับการคำนวณงานทางด้านวิทยาศาสตร์**

(ภาษาอังกฤษ) **Development of Parallel Computing System for Scientific Computing**

ชื่อแผนงานวิจัย (กรณีเป็นโครงการวิจัยภายใต้แผนงานวิจัย)

(ภาษาไทย) .....

(ภาษาอังกฤษ) .....

<u>ส่วน ก</u> : ลักษณะแผนงานวิจัย

☑ โครงการวิจัยใหม่

○ โครงการวิจัยต่อเนื่องระยะเวลา....ปี ปีนี้เป็นปีที่..... รหัสแผนงานวิจัย .....

I ระบุความสอดคล้องของโครงการวิจัยกับยุทธศาสตร์การพัฒนาประเทศตามแผนพัฒนาเศรษฐกิจและสังคมแห่งชาติ ฉบับที่ 10 (พ.ศ. 2550-2554) (กรุณาระบุความสอดคล้องเพียง 1 ยุทธศาสตร์ ที่มีความสอดคล้องมากที่สุด โดยโปรดดูรายละเอียดในผนวก 2)

ยุทธศาสตร์ ...**ยุทธศาสตร์การปรับโครงสร้างเศรษฐกิจให้สมดุลและยั่งยืน**...

- ระบุความสำคัญกับเรื่องที่สอดคล้องมากที่สุดในยุทธศาสตร์นั้น ๆ (โปรดดูรายละเอียดในผนวก 2)

.....**การสนับสนุนให้เกิดการแข่งขันที่เป็นธรรม และการกระจายผลประโยชน์จากการพัฒนาอย่างเป็นธรรม**.....

II ระบุความสอดคล้องของโครงการวิจัยกับนโยบายและยุทธศาสตร์การวิจัยของชาติ (พ.ศ. 2551-2553) (กรุณาระบุความสอดคล้องเพียง 1 ยุทธศาสตร์ 1 กลยุทธ์ และ 1 แผนงานวิจัยที่มีความสอดคล้องมากที่สุด โดยโปรดดูรายละเอียดในผนวก 3)

- ยุทธศาสตร์การวิจัย **การสร้างศักยภาพและความสามารถเพื่อการพัฒนาทางเศรษฐกิจ**.....
- กลยุทธ์การวิจัยที่ 7 **การเพิ่มสมรรถนะและขีดความสามารถในการแข่งขันของประเทศด้านเทคโนโลยีการสนเทศและการสื่อสาร**.....

- แผนงานวิจัยที่ ..1. **การวิจัยเกี่ยวกับการเพิ่มสมรรถนะและพัฒนาศักยภาพขีดความสามารถทางเทคโนโลยีสารสนเทศและการสื่อสาร**

III ระบุความสอดคล้องของโครงการวิจัยกับกลุ่มเรื่องที่ควรวิจัยเร่งด่วนตามนโยบายและยุทธศาสตร์การวิจัยของชาติ (พ.ศ. 2551-2553) (โปรดดูรายละเอียดในผนวก 3)

- กลุ่มเรื่อง ...**เทคโนโลยีใหม่และเทคโนโลยีที่สำคัญเพื่ออุตสาหกรรม**...

IV ระบุความสอดคล้องของโครงการวิจัยกับนโยบายรัฐบาล (กรุณาระบุความสอดคล้องเพียง 1 หัวข้อที่มีความสอดคล้องมากที่สุดโดยโปรดดูรายละเอียดในผนวก 4)

- นโยบายระยะการบริหารราชการ 4 ปี ของรัฐบาล

นโยบาย **การศึกษา**

<u>ส่วน ข</u> : องค์ประกอบในการจัดทำโครงการวิจัย

1. ผู้รับผิดชอบ หน่วยงานหลักและหน่วยงานสนับสนุน

1.1 คณะผู้วิจัย

ชื่อ-สกุล	บทบาทของนักวิจัย	สัดส่วนที่ทำการวิจัย
1.1.1 ดร. สอาด ม่วงจันทร์ (ผู้ร่วมงานวิจัย)	**1. พัฒนาโครงการวิจัย** **2. ดำเนินการในทุกขั้นตอนของวิธีการดำเนินการวิจัย** **3. เขียนรายงานการวิจัย**	**100 %**

1.2 หน่วยงานหลักที่รับผิดชอบงานวิจัย

สาขาวิชาคณิตศาสตร์และสถิติ คณะวิทยาศาสตร์และเทคโนโลยี
มหาวิทยาลัยราชภัฏสกลนคร เลขที่ 680 หมู่ 11 ถนนนิตโย ตำบลธาตุเชิงชุม
อำเภอเมือง จังหวัดสกลนคร 47000

1.3 หน่วยงานที่สนับสนุนงานวิจัย

มหาวิทยาลัยราชภัฏสกลนคร เลขที่ 680 หมู่ 11 ถนนนิตโย
ตำบลธาตุเชิงชุม อำเภอเมือง จังหวัดสกลนคร 47000

2. ประเภทของการวิจัย

การพัฒนาทดลอง (Experimental Development)

3. สาขาวิชาการและกลุ่มวิชาที่ทำการวิจัย

สาขาวิทยาศาสตร์กายภาพและคณิตศาสตร์

4. คำสำคัญ (keywords) ของโครงการวิจัย

Parallel computing; Theory of computation; System of computer; Message-passing interface; Parallel programming; Parallel numerical method

5. ความสำคัญและที่มาของปัญหาที่ทำการวิจัย

เนื่องจากในปัจจุบัน งานวิจัยทางด้านการคำนวณ ไม่ว่าจะเป็น การคำนวณตัวเลขทางคณิตศาสตร์(Computational Mathematics) เคมีคำนวณ(Computational Chemistry) ฟิสิกส์คำนวณ (Computational Physics) และกลศาสตร์ของไหล(Fluid dynamics) เป็นต้น ล้วนแล้วแต่ต้องใช้เวลาในการคำนวณเป็นเวลานานมาก บางงานวิจัย ในการทดสอบย่อยอาจจะต้องใช้เวลาถึง 5 ชั่วโมง หรือเป็นอาทิตย์ถึงจะทราบผลการคำนวณ ซึ่งอาจจะแก้ปัญหานี้ได้โดยการคำนวณบนเครื่องคอมพิวเตอร์สมรรถนะสูง(Super Computer) แต่ราคาของเครื่องและค่าบำรุงรักษาก็สูงมาก และด้วยวิทยาการการคำนวณทางคณิตศาสตร์แบบใหม่(การคำนวณแบบขนาน) ประกอบกับเทคโนโลยีทางคอมพิวเตอร์ที่พัฒนาอย่างรวดเร็ว ทำให้หลายๆหน่วยงานวิจัยทั้งในและต่างประเทศ ได้สร้างระบบการคำนวณคอมพิวเตอร์แบบขนานขึ้นจากเครื่องคอมพิวเตอร์ส่วนบุคคลซึ่งมีราคาต่ำกว่ามากเมื่อเทียบกับเครื่องคอมพิวเตอร์สมรรถนะสูง เพื่อใช้เป็นเครื่องมือในการทำงานวิจัยทางด้านการคำนวณ การสร้างขั้นตอนวิธีคิดแบบใหม่ทางคณิตศาสตร์ และการเรียนการสอนการคำนวณทางคณิตศาสตร์แบบใหม่นี้

ดังนั้นทางผู้วิจัยได้เล็งเห็นว่า ระบบการคำนวณแบบขนานนี้จะเป็นประโยชน์อย่างมากต่อมหาวิทยาลัยของเราและสถานศึกษาใกล้เคียง ที่จะใช้เป็นเครื่องมือช่วยสร้างงานวิจัยทางการคำนวณ และใช้ในด้านการเรียนการสอนคณิตศาสตร์ การสร้างขั้นตอนวิธีคำนวณแบบขนานในด้านวิทยาศาสตร์และวิศวกรรมศาสตร์

6. วัตถุประสงค์ของโครงการวิจัย

6.1 เพื่อสร้างและพัฒนาระบบการคำนวณแบบขนาน

6.2 เพื่อใช้ในการเรียนการสอนในรายวิชาทางด้านการคำนวณ

6.3 เพื่อรองรับงานวิจัยทางด้านการสร้างตัวแบบและการจำลองสถานการณ์ทางวิทยาศาสตร์และคณิตศาสตร์ เช่น งานคำนวณทางคณิตศาสตร์ เคมี ฟิสิกส์ เป็นต้น

6.4 เพื่อศึกษาและคิดค้นขั้นตอนวิธีคำนวณแบบขนานใหม่ๆ

7. ขอบเขตของโครงการวิจัย

7.1 ระบบจะดำเนินการภายใต้ระบบปฏิบัติการยูนิกต์

7.2 คอมไพเลอร์จะใช้ MPICH 2.0, Sun MPI

7.3 ภาษาโปรแกรมคอมพิวเตอร์จะใช้ ภาษาซีและภาษาฟอร์แทรน 90

8. ทฤษฎี สมมุติฐาน (ถ้ามี) และกรอบแนวความคิดของโครงการวิจัย

สำหรับแนวทางการวิจัยโครงการพัฒนาระบบการคำนวณแบบขนานสำหรับการคำนวณงานทางด้านวิทยาศาสตร์นี้ จะเริ่มจากการศึกษาและคัดเลือกระบบปฏิบัติการคอมพิวเตอร์ที่เป็นโอเพนซอส (Open Source) ที่เหมาะสม ซึ่งโดยส่วนใหญ่จะเป็นระบบปฏิบัติการที่มีต้นกำเนิดมาจากยูนิกต์ ซึ่งในแต่ละระบบปฏิบัติการจะมีจุดเด่นจุดด้อยที่แตกต่างกัน ทั้งนี้ก็ขึ้นอยู่กับเทคโนโลยีด้านฮาร์ดแวร์ด้วย จากนั้นต้องทำตามติดตั้งระบบปฏิบัติการและระบบการคำนวณแบบขนาน ซึ่งไม่ใช่เรื่องง่าย แต่ต้องทำการทดสอบหลายๆ ครั้งเพื่อให้ได้ระบบที่มีเสถียรภาพ ซึ่งในการทดสอบเสถียรภาพของระบบสามารถที่จะใช้ ซอฟแวร์มาตรฐานที่ใช้กันทั่วไป หรือสามารถเขียนซอฟแวร์ทางคณิตศาสตร์เพื่อตรวจสอบระบบได้ เมื่อได้ระบบที่มีเสถียรภาพแล้ว ต่อไปก็จะต้องพัฒนาขั้นตอนวิธีการคำนวณแบบขนานสำหรับปัญหาทางด้านวิทยาศาสตร์และคณิตศาสตร์ เพื่อนำมาประมวลผลบนระบบการคำนวณแบบขนาน เพื่อวัดประสิทธิภาพ แก้ไข ปรับปรุงขั้นตอนวิธีแบบใหม่ๆและระบบ ซึ่งเหล่านี้เองจะเป็นประโยชน์อย่างมากต่อการพัฒนางานวิจัยทางด้านการคำนวณ ซึ่งถือเป็นการวิจัยแนวใหม่ทางด้านวิทยาศาสตร์และคณิตศาสตร์

9. การทบทวนวรรณกรรม/สารสนเทศ (information) ที่เกี่ยวข้อง

สำหรับองค์ความรู้ที่เกี่ยวข้องกับงานวิจัยประกอบด้วย 3 ส่วนที่สำคัญคือ

9.1 ระบบปฏิบัติการคอมพิวเตอร์และระบบเครือข่าย

9.2 ทฤษฎีการคำนวณแบบขนาน (Parallel Computing Theory)

9.3 การคิดค้นและออกแบบขั้นตอนวิธีการคำนวณใหม่ๆ

10. เอกสารอ้างอิงของโครงการวิจัย

10.1 The world's TOP500 supercomputers http://www.top500.org/

10.2 Kumar V., Grama A., Gupta A. and Karypis G. Introduction to Parallel computing, The Benjamin/Cummings, CA.

10.3 Burden R. and Faires J.D. Numerical analysis, Brooks/Cole, CA.

10.4 Wilkinson B. and Allen M. Parallel programming, Pearson Prentice Hall, NJ.

10.5 Michael J. Q., Parallel computing: theory and practice, McGraw-Hill, NY.

10.6 Timothy G. M., Parallel computing in computational chemistry, American Chemical Society, USA

10.7 Freeman T.L., Phillips C., Parallel numerical algorithms, Prentice Hall, New York.

10.8 Valduriez P., Parallel processing and data management, Chapman & Hall, London.

10.9 Hojjat A. and Osama K., Parallel processing in structural engineering, Elsevier Applied Science, New York.

10.10 Seyed H. R., Parallel processing and parallel algorithms : theory and

computation, Springer,New York.

10.11 Chandy K. M., and Jayadev M., Parallel program design : a foundation, Addison-Wesley Pub. Co., USA.

10.12 Michael H. C., Parallel programming : a new approach, Silicon Press, NJ.

10.13 Barry W. and Michael A., Parallel programming : techniques and applications using networked workstations and parallel computers, Pearson/Prentice Hall, NJ.

10.14 Michael J. Q., Parallel programming in C with MPI and OpenMP, McGraw-Hill Higher Education, Boston.

10.15 Hord R. M., Parallel supercomputing in MIMD architectures, CRC Press, Boca Raton, FL.

10.16 Selim G. A. and Kelly A. L., Parallel computational geometry, Prentice Hall, Englewood Cliffs, NJ.

10.17 Fijany A. and Bejczy A., Parallel computation systems for robotics : algorithms and architecturesWorld Scientific, Singapore.

10.18 David E. C., Jaswinder P. S.and Anoop G., Parallel computer architecture : a hardware/software approach, Morgan Kaufmann Publishers, San Francisco.

10.19 Saratchandran P., Sundarajan N, and Shou K. F., Parallel implementations of backpropagation neural networks on transputers : a study of training set parallelism, World Scientific, Singapore.

10.20 Freeman T.L. and Phillips C., Parallel numerical algorithms, Prentice Hall, NY.

11. ประโยชน์ที่คาดว่าจะได้รับ เช่น การเผยแพร่ในวารสาร จดสิทธิบัตร ฯลฯ และหน่วยงานที่นำผลการวิจัยไปใช้ประโยชน์

11.1 ระบบการคำนวณแบบขนาน

11.2 งานวิจัยที่ต้องใช้ระบบคำนวณแบบขนาน ทั้งทางด้าน คณิตศาสตร์และวิทยาศาสตร์

11.3 การเรียนการสอนในรายวิชาทางคณิตศาสตร์ประยุกต์

หน่วยงานที่นำผลการวิจัยไปใช้ประโยชน์

ทุกสถาบันการศึกษาทั้งที่กำลังปฏิรูปการเรียนการสอนที่มุ่งจะพัฒนาขั้นตอนวิธีการคำนวณแนวใหม่ทั้ง ทางคณิตศาสตร์ เคมีคำนวณ ฟิสิกส์คำนวณ และทางวิศวกรรมศาสตร์ เพื่อให้มีประสิทธิภาพมากยิ่งขึ้น

12. แผนการถ่ายทอดเทคโนโลยีหรือผลการวิจัยสู่กลุ่มเป้าหมาย

12.1 พัฒนาระบบการคำนวณแบบขนานให้มีเสถียรภาพ

12.2 ทดสอบระบบการคำนวณแบบขนาน

12.3 จัดอบรมการออกแบบและเขียนโปรแกรมการคำนวณแบบขนานให้กับกลุ่มเป้าหมาย

12.4 กลุ่มเป้าหมายพัฒนาขั้นตอนการคำนวณขั้นสูงและประมวลผลบนระบบ

12.5 จัดทำเอกสารเผยแพร่งานวิจัย

13. วิธีการดำเนินการวิจัย และสถานที่ทำการทดลอง/เก็บข้อมูล

13.1 ศึกษาระบบปฏิบัติการคอมพิวเตอร์ที่มีเสถียรภาพเพื่อใช้พัฒนาระบบ

13.2 จัดหาคอมพิวเตอร์และอุปกรณ์เชื่อมต่อระบบเครือข่าย

13.3 ติดตั้งระบบปฏิบัติการและระบบเครือข่ายคอมพิวเตอร์

13.4 ติดตั้งระบบการคำนวณแบบขนาน และซอฟต์แวร์ทางคณิตศาสตร์

13.5 ทดสอบระบบด้วยซอฟต์แวร์ทดสอบ เพื่อปรับปรุงและตรวจสอบความถูกต้อง ความเสถียรภาพของระบบ

13.6 สร้างซอฟแวร์ทางคณิตศาสตร์ที่ต้องใช้ระบบการคำนวณแบบขนาน

13.7 ตรวจสอบและรายงานผลประสิทธิภาพของระบบ

สถานที่ทำการทดลอง: สาขาวิชาคณิตศาสตร์และสถิติ และ ศูนย์คอมพิวเตอร์

14. ระยะเวลาทำการวิจัย และแผนการดำเนินงานตลอดโครงการวิจัย (ให้ระบุขั้นตอนอย่างละเอียด)

ระยะเวลาทำการวิจัย และแผนการดำเนินงานตลอดโครงการวิจัย

แผนการดำเนินงาน	ระยะเวลา											
	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.
1. ศึกษาระบบปฏิบัติการคอมพิวเตอร์ที่มีเสถียรภาพเพื่อใช้พัฒนาระบบ												
2. จัดหาคอมพิวเตอร์และอุปกรณ์เชื่อมต่อระบบเครือข่าย												
3.ติดตั้งระบบปฏิบัติการและระบบเครือข่ายคอมพิวเตอร์												
4. ติดตั้งระบบการคำนวณแบบขนาน และซอฟต์แวร์ทางคณิตศาสตร์												
5. ทดสอบระบบด้วยซอฟต์แวร์ทดสอบ เพื่อปรับปรุงและตรวจสอบความถูกต้องความเสถียรภาพของระบบ												
6. สร้างซอฟแวร์ทาง												

คณิตศาสตร์ที่ต้องใช้ระบบการคำนวณแบบขนาน												
7. ตรวจสอบและรายงานผลประสิทธิภาพของระบบ												
8. ถ่ายทอดและเผยแพร่ผลงานวิจัย												

15. ปัจจัยที่เอื้อต่อการวิจัย (อุปกรณ์การวิจัย, โครงสร้างพื้นฐาน ฯลฯ) ระบุเฉพาะปัจจัยที่ต้องการเพิ่มเติม

15.1 หมายเลข IP จริง จำนวน 1 หมายเลข

15.2. เครื่องพริ้นเตอร์

15.3 ห้อง สำหรับติดตั้งระบบคอมพิวเตอร์ ที่ศูนย์คอมพิวเตอร์

15.4 เครื่องเข้าหัวสายยูทีพี

16. งบประมาณของโครงการวิจัย

รายละเอียดงบประมาณการวิจัยของข้อเสนอการวิจัย จำแนกตามงบประเภทต่าง ๆ

รายการ	จำนวนเงิน (บาท)
1. งบบุคลากร	**80,000**
ค่าจ้างผู้ช่วยนักวิจัย 2 คน 10 เดือน เดือนละ 4,000 บาท	80,000
2. งบดำเนินการ	**328,800**
2.1 ค่าตอบแทน ใช้สอยและวัสดุ	**322,800**
2.1.1 ค่าตอบแทน	**70,880**
- ค่าตอบแทนนักวิจัย 1 คน	40,880
-ค่าตอบแทนผู้เชี่ยวชาญ	30,000
2. 1.2 ค่าใช้สอย	**29,900**
- ค่าเบี้ยเลี้ยงไปราชการ 10 วันๆละ 1 คนๆละ 210 บาท	21,00
- ค่าที่พักของผู้เชี่ยวชาญ 2 ครั้งๆละ 2 คืนๆละ 1,200 บาท	4,800
- ค่าเดินทางของผู้เชี่ยวชาญ 2 ครั้งๆละ 2,500 บาท	5,000
- ค่าจ้างจัดทำเล่มรายงานการวิจัยเพื่อการเผยแพร่ 60 เล่มๆละ 300 บาท	18,000
2.1.3 ค่าวัสดุ	**33,020**

- วัสดุสำนักงาน	13,020
- หนังสือและตำราที่จำเป็นต้องใช้	20,000
2.1.4 ค่าครุภัณฑ์	**189,000**
- คอมพิวเตอร์ส่วนบุคคล จำนวน 5 เครื่อง (เครื่องละ 35,800)	179,000
- อุปกรณ์เชื่อมต่อระบบเครือข่าย	10,000
2.2 **ค่าสาธารณูปโภค**	**6,000**
- ค่าโทรศัพท์ติดต่อประสานงาน 12 เดือนๆละ 500 บาท	6,000
รวมงบประมาณ	**408,800**
3. **ค่าอื่น ๆ หัก 8 % ให้มหาวิทยาลัย**	**32,704**
รวมงบประมาณทั้งหมดที่เสนอขอ	**441,504**

17. ผลสำเร็จและความคุ้มค่าของการวิจัยที่คาดว่าจะได้รับ

17.1 ผลสำเร็จของงานวิจัย

1. เป็นการพัฒนาระบบคำนวณแบบขนานเพื่อรองรับกับขั้นตอนวิธีคำนวณแบบใหม่ทางวิทยาศาสตร์และคณิตศาสตร์
2. ได้งานวิจัยตีพิมพ์ระดับชาติ/นานาชาติ 1 เรื่อง
3. ได้เสนอผลงานในที่ประชุมวิชาการระดับชาติ 1 เรื่อง
4. ได้ระบบการคำนวณสำหรับการวิจัย การเรียนการสอนด้านการออกแบบและเขียนโปรแกรมคำนวณทางคณิตศาสตร์

17.2 หน่วยงานที่จะนำไปใช้ประโยชน์

ทุกสถาบันการศึกษาทั้งที่กำลังปฏิรูปการเรียนการสอนที่เน้นผู้เรียนเป็นศูนย์กลางและที่มีการสอนโดยเน้นผู้เรียนเป็นศูนย์กลาง

20. ลงลายมือชื่อ หัวหน้าโครงการวิจัย พร้อมวัน เดือน ปี

ลงชื่อ.....หัวหน้าโครงการ

(อ. ดร. สอาด ม่วงจันทร์)

วันที่ 28 กุมภาพันธ์ 2552

ส่วน ค : ประวัติคณะผู้วิจัย

1. ดร. สอาด ม่วงจันทร์

Dr. Sa-at Muangchan

1.1 เลขหมายบัตรประจำตัวประชาชน **3460900261260**

1.2 ตำแหน่งปัจจุบัน **อาจารย์ ระดับ 5**

1.3 หน่วยงานและสถานที่อยู่ที่ติดต่อได้สะดวก พร้อมหมายเลขโทรศัพท์ โทรสาร และไปรษณีย์อิเล็กทรอนิกส์ (e-mail)

สาขาวิชาคณิตศาสตร์และสถิติ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏสกลนคร เลขที่ 680 หมู่ 11 ถนนนิตโย ตำบลธาตุเชิงชุม อำเภอเมือง จังหวัดสกลนคร

47000

E-mail: msaat@hotmail.com

1.4 ประวัติการศึกษา

1. วิทยาศาสตรบัณฑิต (คณิตศาสตร์)

ภาควิชาคณิตศาสตร์ คณะ วิทยาศาสตร์ มหาวิทยาลัยขอนแก่น (2541)

2. วิทยาศาสตรมหาบัณฑิต (วิทยาการคณนา)

ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย (2544)

3. วิทยาศาสตรดุษฎีบัณฑิต (คณิตศาสตร์ประยุกต์)

สาขาคณิตศาสตร์ สำนักวิชาวิทยาศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี (2550)

1.5 สาขาวิชาการที่มีความชำนาญพิเศษ (แตกต่างจากวุฒิการศึกษา) ระบุสาขาวิชาการ

-วิทยาการคอมพิวเตอร์

1.6 ประสบการณ์ที่เกี่ยวข้องกับการบริหารงานวิจัยทั้งภายในและภายนอกประเทศ โดยระบุสถานภาพในการทำการวิจัยว่าเป็นผู้อำนวยการแผนงานวิจัย หัวหน้าโครงการวิจัย หรือผู้ร่วมวิจัยในแต่ละข้อเสนอการวิจัย

-2543 ผู้ช่วยวิจัยเกี่ยวกับเรื่อง Thai Optical Character Recognition (Thai OCR)

ภาคผนวก ข.

ปัญหาทดสอบระบบ

เรื่องพลศาสตร์ของร่องรอยการไหลแบบปั่นป่วนในของไหลที่เป็นชั้นๆ

(TURBULENT WAKE DYNAMICS IN LINEAR STRATIFIED FLUID)

Parallel numerical simulation of turbulent wake dynamics in linear stratified fluid

Sa-at Muangchan [1, C], N. P. Moshkin[2]
[1] *School of Mathematics, Suranaree University of Technology, Thailand*
[2] *School of Mathematics, Suranaree University of Technology, Thailand*
[C] **E-mail**: msaat@hotmail.com; **Tel.** 083-7965996

ABSTRACT

The aim of this research is to apply a parallel computation technique of functional and domain decompositions to a numerical simulation of turbulent wake behind an axisymmetric body in stratified fluid. The numerical results have been compared with the experimental data for momentumless and drag turbulent wakes in homogeneous and stratified fluid. We developed a theoretical estimate of speedup under assumption of ideal computational process. We studied the dependence of speedup on the latency and bandwidth of parallel computer systems.

Keywords: turbulent wake, parallel computing, functional and domain decomposition, splitting method.

INTRODUCTION

Research on stratified turbulent wakes has both meteorological (flow over islands, mountains or sea-mountains) and naval applications (submerged bodies). Turbulent wake behind towed and self-propelled body in the homogeneous and stratified fluid has been considered in a number of theoretical, experimental and computational works [1-6]. Based on a three-dimensional parabolized system of averaged equation for motion, continuity, incompressibility, turbulent energy, and the rate of dissipation a numerical modeling of turbulent wake dynamics has been conducted in [1, 2, and 4]. To predict the average characteristics of turbulent wake, it can take a long time on a serial computer. So the objectives of recent research are to develop and analyze parallel algorithms that improve efficiently the computational time of numerical models considered in [1, 2, and 4]. Both functional and domain decompositions techniques are used to develop parallel algorithm in the present study.

COMPUTATIONAL DETAILS

There are several numerical models of turbulent wake dynamics in stratified fluid that can successfully be applied [1-6] to predict decay of turbulence wake behind towed and self-propelled bodies. However, computer implementation of these models on single CPU computer is still very time consuming even these models developed on the based of RANS approach. These models constitute of more than dozen partial differential equations which coupled average characteristics of the turbulent flow. According to the problem nature and the characteristic of numerical models, both domain and functional decomposition techniques can be used to develop the parallel algorithm for these class of the problems.

Parallel functional decomposition

The algorithm for solving the problem of turbulent wake dynamics in stratified fluid reduced to the successive integration of the transport equations system to find averaged velocity components, turbulent energy, dissipation rate, components of the Reynolds stress tensor and so on. Suppose we have M jobs J_i, $i = 1, 2, ..., M$ which are executed in sequential order as shown in Figure 1.

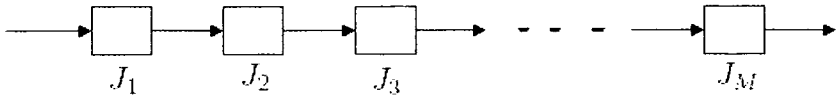

Figure 1. A sequential process.

Let t_i be the run time of job J_i for $i = 1,2,...,M$. Hence, the run time of a sequential program T_s is $T_s = \sum_{i=1}^{M} t_i$. To utilize the functional decomposition technique, let us assume that the sequential processes can be executed as k groups of disjoint tasks as depicted in Figure 2.

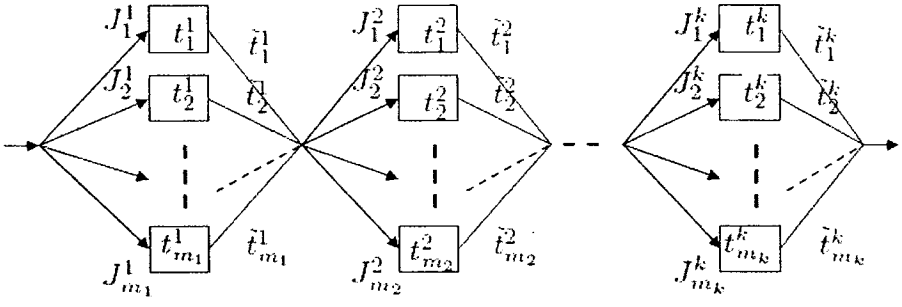

Figure 2. A parallel functional decomposition process.

where $\sum_{j=1}^{k} m_j = M$, t_i^j are the run time of job J_i^j and $\tilde{t}_i^j$ are the time of communications for $i = 1,2,...,m_j$ and $j = 1,2,...,k$. Let P be a number of processors and $P \le \max\{m_j\}_{i=1,..,k}$. The estimate of parallel run time T_P is $T_p = \sum_{j=1}^{k}\left(T^j + T_{comm}^j\right)$, where T^j and T_{comm}^j are the time for execution and the time of communications for j-th group $j = 1,2,...,k$, respectively. So we have

$$T^j \approx \left\lceil \frac{m_j}{P} \right\rceil \max_{1 \le i \le m_j} t_i^j \quad \text{and} \quad T_{comm}^j \approx \sum_{i=1}^{m_j}(P-1)\tilde{t}_i^j .$$

Since, $\tilde{t}_i^j = t_l + \frac{MessageSize_i^j}{BandWidth}$ where t_l is the latency time of parallel network and $MessageSize_i^j$ is the message size of job J_i^j for $j = 1,2,...,k$; $i = 1,2,...,m_j$. Therefore, an estimate speedup is

$$S = \frac{\sum_{i=1}^{M} t_i}{\sum_{j=1}^{k}\left(\left\lceil \frac{m_j}{P} \right\rceil \max_{1 \le i \le m_j} t_i^j + \sum_{i=1}^{m_j}(P-1) \times \left(t_l + \frac{MessageSize_i^j}{Bandwidth}\right)\right)}.$$

For example, let us assume that we have k groups where each group has m jobs and each job uses t seconds for the execution time and the message size of the data for all jobs are the same, $MessageSize_i^j = MessageSize$. So, the estimate speedup is

$$S \approx \frac{m \times t \times k}{k \times \left(\left\lceil \frac{m}{P} \right\rceil \times t + m \times (P-1) \times \left(t_l + \frac{MessageSize}{Bandwidth}\right)\right)}.$$

If the term of communication time is zero, the speedup will be about P times of the sequential algorithm which corresponds to the ideal speedup.

Parallel Domain decomposition

Let us consider a grid of size $N \times M$ and assume that one node of grid requires t seconds for computation. So, for all nodes of grid is required $(N \times M) \times t$ seconds. Let P be the number of processors and each processor works with $(N \times M)/P$ nodes of grid. So, each processor is required $((N \times M)/P) \times t$ seconds and $P \times (P-1)$ times for updating data blocks. Therefore, speedup is

$$S = \frac{(N \times M) \times t}{\frac{(N \times M) \times t}{P} + P \times (P-1) \times \left(t_l + \frac{MessageSize}{Bandwidth} \right)}.$$

Numerical models to simulate turbulent wake include many transport equations which are solved by implicit scheme of splitting over spatial variables [1]. We applied the idea of the domain decomposition to solve the parabolic problem by using the splitting scheme [7]. Let us consider the following parabolic equation

$$\begin{cases} u_t = \frac{\partial}{\partial x}\left(A \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y}\left(B \frac{\partial u}{\partial y} \right) + F \quad (x, y) \in \Omega \subset \Re^2, \; t \in [0, T], \\ u(x, y, 0) = u_0(x, y, 0) \qquad (x,y) \in \Omega, \\ u(x, y, t) = 0 \qquad \text{on } \partial\Omega \times [0, T]. \end{cases}$$

The splitting scheme is

$$\begin{cases} \frac{u^{n+1/2} - u^n}{\Delta t} = \frac{\partial}{\partial x}\left(A \frac{\partial u^{n+1/2}}{\partial x} \right) + F, \\ \frac{u^{n+1} - u^{n+1/2}}{\Delta t} = \frac{\partial}{\partial y}\left(B \frac{\partial u^{n+1}}{\partial y} \right). \end{cases}$$

Thus, each above equation takes into account only one direction. Diffusive terms are approximated using central differences. For the first step of the splitting scheme, we can rewrite as

$$-a_i u_{i-1,j}^{n+1/2} + c_i u_{i,j}^{n+1/2} - b_i u_{i+1,j}^{n+1/2} = f_{i,j}, \; i = 1, \ldots, N; \; j = 1, \ldots, M. \qquad (1)$$

Therefore, at each fixed j we can use the elimination method for the three-point equations [7]. We can simply parallelize computation in (1) by assigning *M/P* equations to each processor. Processor K will solve equations of system (1) for $j = (K-1) \times M/P, \ldots, K \times M/P$. With this parallelization strategy, the coefficient matrix needs to be distributed row-wise as demonstrated in Figure 3 a).

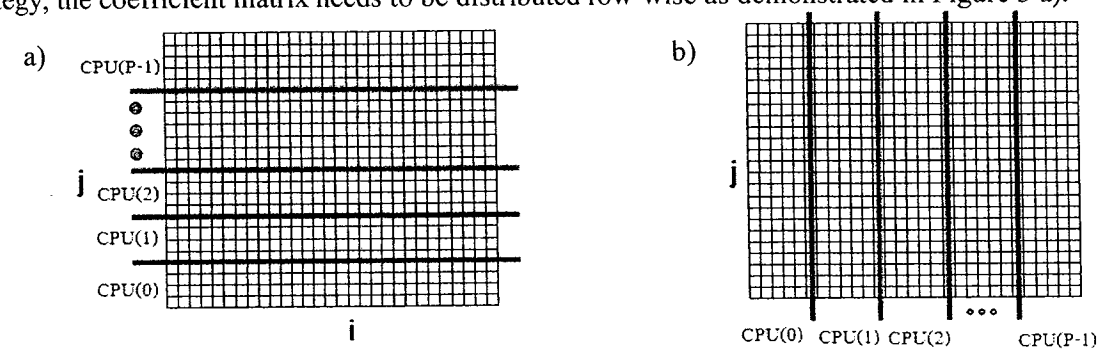

Figure 3. The sketch of domain decomposition process in direction of y-axis and x-axis.

For the second step of the splitting scheme (1), we have

$$-\tilde{a}_i u^n_{i,j-1} + \tilde{c}_i u^n_{i,j} - \tilde{b}_i u^n_{i,j+1} = \tilde{f}_{i,j}, i = 1,\ldots,N;\ j = 1,\ldots,M. \qquad (2)$$

At fixed i we can use the elimination method. Let us assign N/P equations to each processor. Processor K will solve equations of system (2) for $i = (K-1)\times N/P,\ldots,K\times N/P$. The coefficient matrix in (2) needs to be distributed column-wise as demonstrated in Figure 3 b). We should compute in parallel on every strip in the direction of x-axis for the first step and on every strip in the direction of y-axis for the second step. Consequently, we need to redistribute the data between the row-splitting and column-splitting or vice versa.

RESULTS AND DISCUSSION

To describe the far turbulent wake behind a towed body in a linear stratified medium a hierarchy of semi-empirical turbulent models is used in [1]. The most complex model comprises differential equations for transport of normal Reynolds stresses. Totally, this model consists of 12 differential equations. These equations are solved using splitting techniques [7]. In numerical algorithm this model has 14 jobs, which executed sequentially. For more details due to their bulkiness, it is referenced to [1]. We chose this model to implement the ideas of functional and domain decomposition given above. The analysis of numerical model shows that it can be represented in form of 5 independent groups of tasks. In the notations of Figure 1 we have $m_1 = 3$, $m_2 = 1$, $m_3 = 2$, $m_4 = 7$, $m_5 = 1$. Jobs in each group can be executed independently. First we utilized idea of functional decomposition. In the second approach we used idea of domain decomposition to each of 14 jobs solved in sequential sequence. It means that each job perform splitting scheme by parallel algorithm with all available CPU. The numerical results using the parallel codes have been compared with the model using the sequential code to guarantee the correctness of the parallel algorithm. In Figure 4 a) and b) the axial values of the turbulent energy and the longitudinal velocity component defect in drag turbulent wake calculated by sequential algorithm [1], parallel functional decomposition algorithm, and parallel domain decomposition algorithm are compared with Lin and Pao's experimental data and Hassid's computational results [5,6]. Computational grids in these runs consist of 200 nodes in vertical direction and 350 nodes in horizontal direction.

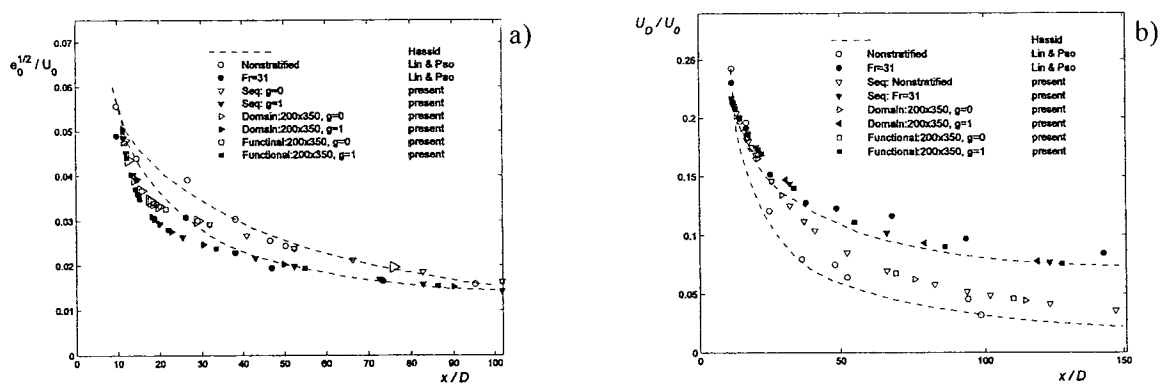

Figure 4. Comparison of the axial values of the turbulent energy, $e_0(x)$, a) and the longitudinal velocity component defect, $U_{d0}(x)$, b) in the wake behind a towed body calculated by sequential and parallel algorithms with Lin and Pao's experimental data and Hassid's computational results.

All the considered algorithms yield a satisfactory agreement with experimental data. The other main characteristics of the wake (horizontal and vertical size of the wake in homogeneous and stratified fluid, the intensities of turbulent fluctuations of the longitudinal and vertical velocity and density components, and etc.) are differed not more than 1-2%.

Since the computational and communication pattern are the same per one time step, we only consider the run time of 999 steps to measure speedup. The run time of the sequential code is used as

a measure of the run time on one processor. In this study, the run time starts after generation an initial state. Wall-clock time is used to record the run time. The wall clock time [8, 9] is used to represent the total run time since it includes the idle time and communication time [8, 9]. For numerical experiments, the sequential and parallel algorithms compiled and run using two clusters of Solaris (Cluster-I) and Linux (SUT-HPCC) operating systems which are located at School of Mathematics and High Performance Computing Center of Suranaree University of Technology. Cluster-I based on PC sever with 10 processors of Opteron (2 cores) 1.6 GHz, 8 GB of RAM, Intel (4 cores) 2.0 GHz, 8 GB of RAM, and two machines of AMD (2 cores) 2.4 GHz, 2 GB of RAM. SUT-HPCC based on 7 machines of 2×Quad cores Xeon 2.33 GHz, 8 GB of RAM and 7 machines of 2×Dual cores Xeon 3.0 GHz, 4 GB of RAM. On both of clusters, we used Sun FORTRAN compiler of Sun Studio 11 with MPI 2.0 library. Latency time and bandwidth was estimated by using LATENCY_BANDWIDTH program [10] and was 5.2E−5 sec, 118MB/sec for Cluster-I, and 5.9E−5 sec, 260MB/sec for SUT-HPCC. The execution times are shown in Table 1 and Table 3 for Cluster-I and in Table 2 and Table 4 for SUT-HPCC, respectively.

The performance results for both clusters in terms of speedup characteristics are shown in Figure 4 and 5 and timing for all these performance results are tabulated in Table 1 and 2. These figures give the detailed statistics of the parallel code on 1, 2, 4, 7 and 8 processors for Cluster-I and SUT-HPCC.

Table 1. Results of parallel functional decomposition running time on Cluster-I (sec.).

NCPUs	size=400x750	speedup	Size=600x1150	speedup	size=800x1550	speedup
1	3006.91	-	6565.46	-	12565.32	-
2	2247.96	1.34	4695.33	1.39	8285.68	1.51
4	1578.49	1.90	3407.09	1.93	6138.72	2.05
7	1626.40	1.85	3315.16	1.98	5636.48	2.23

Table 2. Results of parallel functional decomposition running time on SUT-HPCC (sec.).

NCPUs	size=400x750	speedup	size=600x1150	speedup	size=800x1550	speedup
1	2850.92	-	6329.26	-	11659.74	-
2	2043.86	1.39	4437.54	1.42	7581.24	1.53
4	1421.72	2.01	3217.46	1.96	5642.07	2.06
7	1464.28	1.94	3049.15	2.07	4979.82	2.34

Table 3. Results of parallel domain decomposition running time on Cluster-I (sec.).

NCPUs	size=400x750	speedup	size=600x1150	speedup	size=800x1550	speedup
1	3006.91	-	6565.46	-	12565.32	-
2	1646.69	1.82	3533.53	1.85	6578.76	1.91
4	1038.44	2.89	2263.19	2.90	4297.38	2.92
7	861.19	3.49	1863.42	3.52	3491.74	3.59
8	744.74	4.03	1621.35	4.05	3064.02	4.10

Table 4. Results of parallel domain decomposition running time on SUT-HPCC (sec.).

NCPUs	size=400x750	speedup	size=600x1150	speedup	size=800x1550	speedup
1	2850.92	-	6329.26	-	11659.74	-
2	1542.24	1.84	3342.54	1.89	6034.58	1.93
4	978.51	2.91	2168.43	2.92	3986.46	2.92
7	764.51	3.72	1685.44	3.75	3098.47	3.76
8	684.37	4.16	1502.58	4.21	. 2761.82	4.22

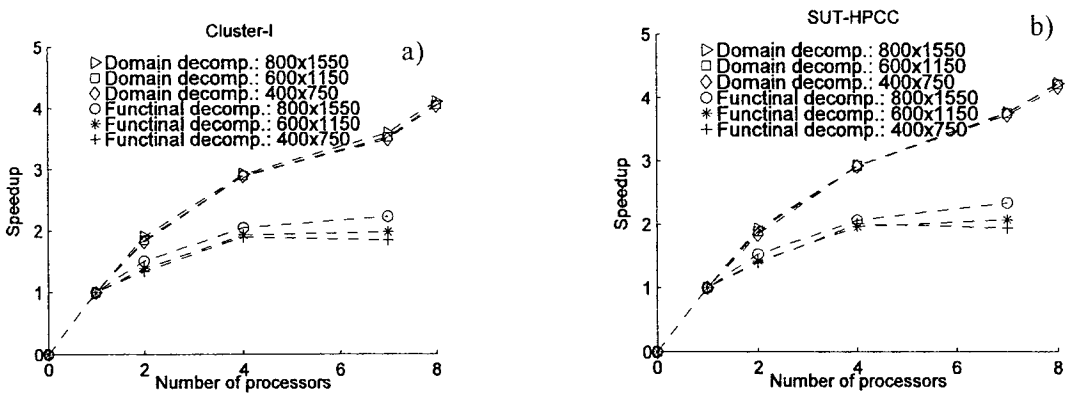

Figure 5. Speedup results on Cluster-I, a), and SUT-HPCC b).

Figure 5 a) and b) show the speedup results of parallel functional and domain decompositions on the different grid size of Cluster-I and SUT-HPCC, respectively. The performance results for parallel runs for both clusters depend on grid size. The speedup of parallel functional decomposition increases almost linearly up to four processors after that it starts to deviate away from the perfect speedup, while the speedup of parallel domain decomposition increases linearly up to eight processors.

CONCLUSION

We have developed the parallel algorithms based on the functional and domain decompositions. Also theoretical estimate of speedup is represented. Developed parallel algorithms have been successfully applied to the numerical simulation of turbulent wake dynamics behind towed body in linear stratified fluid. Experimental speedup is in a good agreement with the theoretical estimate.

LITERATURE CITED

1. Chernykh G.G, Fomina A.V., Moshkin N.P., *Russ. J. Numer. Anal. Math.*, 2006, 21(5), 395–424.
2. Chernykh G.G., Voropayeva O.F., *Computers and Fluids*, 1999, 28 (3), 281–306.
3. Meunier P., Spedding G.R., *J. Fluid Mech.*, 2006, 552, 229–256.
4. Chernykh G.G., Moshkin N.P., Voropayeva O.F., *Proc. of 1st Int. Conf. on CFD.* Kyoto, Japan, 2000, 455–460.
5. Hassid S., *J. Hydronautics.*, 1980, 14(1), 25–32.
6. Lin J.T., Pao Y.H., *Rev. Fluid Mech.*, 1979, 11, 317–336.
7. Yanenko N.N., *The Method of Fractional Steps. The Solution of Multidimensional Problems of Mathematical Physics Variable*, Springer–Verlag, New York-Heidelberg-Berlin, 1971.
8. W. Sch¨onauer. *Scientific supercomputing architecture and use of shared and distributed memory parallel computing.* Self-edition by Willi Sch¨onauer, Germany, 2000.
9. V. Kumar, A. Grama, A. Gupta and G. Karypis. *Introduction to parallel computing.* The Benjamin/Cummings Publishing Company, Inc, 1994.
10. O. M. Nielsen., *Graduate Course in Scientific Computing*, http://datamining.anu.edu.au/ole/, 2003.

ภาคผนวก ค.

งานวิจัยอื่นที่ได้ประมวลผลบนระบบ

Applied Mathematical Sciences, Vol. 7, 2013, no. 16, 765 - 772

A New Algorithm of Numerical Integration by 3-Points Parabolic Regression Technique

S. Muangchan[1], P. Prasertsang[2], P. Mukwachi[3] and S. Sompong[4]

[1,3,4]Department of Mathematics and Statistics
Faculty of Science and Technology
Sakon Nakhon Rajabhat University
Sakon Nakhon 47000, Thailand
[1] msaat@hotmail.com
[3] gpongphan@gmail.com
[4] s_sanpinij@yahoo.com

[2]Department of Sciences, Faculty of Science and Engineering
Kasetsart University, Chalermprakiat Sakon Nakhon Province Campus
Sakon Nakhon 47000, Thailand
[2] kai_patty@hotmail.com

Abstract

The aim of this research is to proposed a new computational algorithm of numerical integration by using 3-points parabolic regression technique. In numerical results, we performed the algorithm on Matlab program and compared the error with Trapezoidal and Simpson's rules.

Keywords: Numerical Integration, Parabolic Regression

1 Introduction

The definite integrals are arise in many different areas and for evaluating definite integrals, anyone can use the Fundamental Theorem of Calculus ([4], [5], [6]). However, it can not always be applied: there are some functions which do not have an antiderivative which can be expressed in terms of familiar functions such as polynomials, exponentials and trigonometric functions. But, we can approximated these definite integrals by using the techniques of numerical integral such as Riemann sum, Trapezoidal, Simpson's rule or other variants of Simpson([1],[3],[2],[4], [5], [6]).

Let f be defined on $[A, B]$, and let $P = \{x_0, x_1, ..., x_n\}$ be a regular partition of $[A, B]$ which subdividing the interval into n subintervals each of length

$h = (B - A)/n$. Then we can use the following formulas to approximate the definite integral:

Riemann sum:

$$\int_A^B f(x)\,dx \approx \sum_{k=1}^{n} f(t_k)h \tag{1}$$

where $t_k (k = 1, 2, ..., n)$ is an arbitrary number in $[x_{k-1}, x_k]$.

Trapezoidal Rule:

$$\int_A^B f(x)\,dx \approx \frac{B-A}{2n}\left(f(x_0) + 2f(x_1) + 2f(x_2) + \ldots + 2f(x_{n-1}) + f(x_n)\right). \tag{2}$$

Simpson's Rule:

$$\int_A^B f(x)\,dx \approx \tag{3}$$

$$\frac{B-A}{3n}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)].$$

where n is an even number.

From the above formulas, it is obvious that the sum appearing in the Trapezoidal Rule is a genuine Riemann sum and yields an approximation that is usually superior to those obtained with the more ordinary Riemann sums. Although, the formula of Simpson's Rule is generally produces more accurate estimates than Riemann sums and Trapezoidal Rule, but the n subinterval must large enough. In this paper, we constructed a new computational algorithm of numerical integration which produces more accuracy with a small number of subintervals.

2 Main Results

Let f be defined on $[A, B]$, and let $P = \{x_0, x_1, ..., x_n\}$ be a regular partition of $[A, B]$ which subdividing the interval into n subintervals each of length $h = (B - A)/n$.

For each $k (k = 1, 2, \ldots, n)$, we consider a subinterval $[x_{k-1}, x_k]$. Let $f^*(x) = ax^2 + bx + c$ be an approximately function of $f(x)$ on $[x_{k-1}, x_k]$ where a, b and c are constants. Set $x_1 = x_{k-1}, x_2 = x_k$ and $x_3 = \frac{x_k + x_{k-1}}{2}$, then we have:

$$\begin{array}{rllll} f(x_{k-1}) & = f_1^* & = f^*(x_1) & = ax_1^2 + bx_1 + c \\ f(x_k) & = f_2^* & = f^*(x_2) & = ax_2^2 + bx_2 + c \\ f(\frac{x_k + x_{k-1}}{2}) & = f_3^* & = f^*(x_3) & = ax_3^2 + bx_3 + c. \end{array} \tag{4}$$

By the parabolic regression, we use the data of these three points to fits the graph of $f(x)$ on $[x_{k-1}, x_k]$. Hence, to find the coefficients a, b and c in equation (4), we can state as the following:
Firstly, we summarize the above system as the follows:

$$\begin{aligned} a\sum_{i=1}^{3} x_i^2 + b\sum_{i=1}^{3} x_i + 3c &= \sum_{i=1}^{3} f_i^* \\ a\sum_{i=1}^{3} x_i^3 + b\sum_{i=1}^{3} x_i^2 + \sum_{i=1}^{3} x_i &= \sum_{i=1}^{3} x_i f_i^* \\ a\sum_{i=1}^{3} x_i^4 + b\sum_{i=1}^{3} x_i^3 + \sum_{i=1}^{3} x_i^2 &= \sum_{i=1}^{3} x_i^2 f_i^* \end{aligned} \tag{5}$$

For convenience, we can rewrite the system (5) as below:

$$\begin{bmatrix} 1 & \beta & \alpha \\ \xi & 1 & \beta \\ \delta & \xi & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} \tag{6}$$

where

$$\beta = \frac{\sum_{i=1}^{3} x_i}{\sum_{i=1}^{3} x_i^2}, \quad \alpha = \frac{3}{\sum_{i=1}^{3} x_i^2}, \quad \xi = \frac{\sum_{i=1}^{3} x_i^3}{\sum_{i=1}^{3} x_i^2}, \quad \delta = \frac{\sum_{i=1}^{3} x_i^4}{\sum_{i=1}^{3} x_i^2}, \quad \varepsilon_1 = \frac{\sum_{i=1}^{3} f_i^*}{\sum_{i=1}^{3} x_i^2},$$

$$\varepsilon_2 = \frac{\sum_{i=1}^{3} x_i f_i^*}{\sum_{i=1}^{3} x_i^2} \quad \text{and} \quad \varepsilon_3 = \frac{\sum_{i=1}^{3} x_i^2 f_i^*}{\sum_{i=1}^{3} x_i^2}.$$

Secondly, we use the row eliminations for the first and second rows. Hence, we have

$$\begin{bmatrix} 1-\alpha\delta & \beta-\alpha\xi & 0 \\ \xi-\beta\delta & 1-\beta\xi & 0 \\ \delta & \xi & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \varepsilon_1-\alpha\varepsilon_3 \\ \varepsilon_2-\beta\varepsilon_3 \\ \varepsilon_3 \end{bmatrix}. \tag{7}$$

From (7), we can consider the following system of two equations

$$(1-\alpha\delta)\,a + (\beta-\alpha\xi)\,b = \varepsilon_1 - \alpha\varepsilon_3$$

$$(\xi-\beta\delta)\,a + (1-\beta\xi)\,b = \varepsilon_2 - \beta\varepsilon_3.$$

Thus, we have

$$b = \frac{(\xi - \beta\delta)(\varepsilon_1 - \alpha\varepsilon_3) - (1 - \alpha\delta)(\varepsilon_2 - \beta\varepsilon_3)}{(\xi - \beta\delta)(\beta - \alpha\xi) - (1 - \alpha\delta)(1 - \beta\xi)} \tag{8}$$

$$a = \frac{(\varepsilon_1 - \alpha\varepsilon_3) - (\beta - \alpha\xi)\, b}{1 - \alpha\delta} \tag{9}$$

where $\alpha(\xi^2 - \delta) + \beta(\delta - 2\xi) \neq -1$ and $\alpha\delta \neq 1$. From the third row of system (7), we have

$$c = \varepsilon_3 - \delta a - \xi b \tag{10}$$

Finally, we can approximate the definite integral of $f(x)$ on $[x_{k-1}, x_k]$ as follows:

$$\int_{x_{k-1}}^{x_k} f(x)dx \approx \int_{x_{k-1}}^{x_k} (ax^2 + bx + c)\, dx = (\frac{ax^3}{3} + \frac{bx^2}{2} + cx)\Big|_{x_{k-1}}^{x_k}. \tag{11}$$

Then, the numerical approximation of the definite integral are obtained from the summation value of all subintervals $[x_0, x_1], [x_1, x_2], ..., [x_{n-1}, x_n]$. So, we have

$$\int_{x_0}^{x_n} f(x)dx \approx \sum_{i=1}^{n} (\frac{a_i x^3}{3} + \frac{b_i x^2}{2} + c_i x)\Big|_{x_{i-1}}^{x_i} \tag{12}$$

where a_i, b_i and c_i are the constants in the subinterval $[x_{i-1}, x_i]$.

Therefore, we can state the algorithm for finding the numerical integration as follows:

Three-Points Parabolic Algorithm (TPA):

Step 1: Set $x_i = x_0 + ih$; $(i = 1, 2, \ldots, n)$ where $x_0 = A, x_n = B$ and $h = (B - A)/n$.

Step 2: For $i = 1, 2, \ldots, n$

Step 2.1: Set $x_1 = x_{i-1}, x_2 = x_i$ and $x_3 = \frac{x_i + x_{i-1}}{2}$,

$$f_1^* = f(x_{i-1}), f_2^* = f(x_i), f_3^* = f(\frac{x_i + x_{i-1}}{2}).$$

Step 2.2: Compute the value of $\beta, \alpha, \xi, \delta, \varepsilon_1, \varepsilon_2$ and ε_3 as in Equation (6).

Step 2.3: Compute the value of b, a and c from Equation (8), (9) and (10), respectively.

Step 2.4: Compute the numerical approximation of the definite integral in subinterval $[x_{i-1}, x_i]$:

$$\Delta_i = (\frac{ax^3}{3} + \frac{bx^2}{2} + cx)|_{x_{i-1}}^{x_i}. \tag{13}$$

Step 3: Compute the numerical approximation of the definite integral in the interval $[x_0, x_n]$ from:

$$\int_{x_0}^{x_n} f(x)dx \approx \sum_{i=1}^{n} \Delta_i.$$

From Equation (12), if the values of $a_i = a$, $b_i = b$ and $c_i = c$ for all i where a, b and c are constants. Then we have the following theorem.

Theorem 2.1. *Let f be defined on $[A, B]$, and $P = \{x_0, x_1, ..., x_n\}$ be an even regular partition of $[A, B]$ which a step length $h = (B-A)/n$ and $x_i = x_0 + ih$. If $a_i = a_j$, $b_i = b_j$ and $c_i = c_j$ for all $i, j = 0, 1, 2, \ldots, n$ then*

$$\sum_{i=1}^{n}(\frac{a_ix^3}{3} + \frac{b_ix^2}{2} + c_ix)|_{x_{i-1}}^{x_i} = \frac{B-A}{3n}[y_0 + 4y_1 + 2y_2 + 4y_3 + \ldots + 2y_{n-2} + 4y_{n-1} + y_n]. \tag{14}$$

Proof: By the parabolic regression and let $a_i = a$, $b_i = b$ and $c_i = c$ for all i and j. So the curve passes through the points. Then we have:

$$y_i = a(x_0 + ih)^2 + b(x_0 + ih) + c.$$

Since n is even, we have

$$\sum_{i=1}^{n}(\frac{a_ix^3}{3} + \frac{b_ix^2}{2} + c_ix)|_{x_{i-1}}^{x_i} =$$

$$\sum_{i=1}^{n}(\frac{ax^3}{3} + \frac{bx^2}{2} + cx)|_{x_{i-1}}^{x_i} = \frac{a}{3}(x_n^3 - x_0^3) + \frac{b}{2}(x_n^2 - x_0^2) + c(x_n - x_0).$$

We can now evaluate the coefficients $a_i = a$, $b_i = b$, $c_i = c$ and $x_n = x_0 + nh$ and the above equation in terms of y_i. Then, we have

$$\frac{a}{3}(x_n^3 - x_0^3) + \frac{b}{2}(x_n^2 - x_0^2) + c(x_n - x_0) =$$

$$\frac{B-A}{3n}[y_0 + 4y_1 + 2y_2 + 4y_3 + \ldots + 2y_{n-2} + 4y_{n-1} + y_n].$$

Theorem 2.2. *Let f be defined on $[A, B]$, and $P = \{x_0, x_1, ..., x_n\}$ be a regular partition of $[A, B]$ which a step length $h = (B - A)/n$ and $x_i = x_0 + ih$. If $|f^{(4)}(\mu)| \leq M$ for $\mu \in [x_i, x_{c_{i+1}}]$ and $x_{c_{i+1}} = x_i + \frac{3h}{2}$. Then*

$$|E_n| \leq \frac{M(B-A)^5}{2880n^5}$$

Proof: We firstly determine the error term from the remainder of Lagrange polynomial $P(x)$ with $n = 3$ over the interval $[x_i, x_{i+1}]$ as in the following equation

$$E_n = \frac{f^{(4)}(\mu)}{4!}(x - x_i)(x - x_{c_i})(x - x_{i+1})(x - x_{c_{i+1}})$$

where $\mu \in [x_i, x_{c_{i+1}}]$, $x_c = x_i + \frac{h}{2}$ and $x_{c_{i+1}} = x_i + \frac{3h}{2}$.

By intergrating the above equation and let $x = x_i + ht, t \in (0, 1)$, then $dx = hdt$. So, the absolute error can rewrite as:

$$\begin{aligned}
|E_n| &= \left| \int_{x_i}^{x_{i+1}} \frac{f^{(4)}(\mu)}{4!}(x - x_i)(x - x_c)(x - x_{i+1})(x - x_{c+1})dx \right| \\
&= \left| \int_0^1 hth(t - \frac{1}{2})h(t - 1)h(t - \frac{3}{2})hdt \right| \\
&\leq \frac{Mh^5}{24} \left| \int_0^1 (t - \frac{1}{2})(t - 1)(t - \frac{3}{2})dt \right| \\
&= \frac{Mh^5}{2880}.
\end{aligned}$$

3 Numerical Results

In this section, we tested the algorithm with the specific examples ([1], [2], [3], [4]) on Matlab program and compared the results with Trapezoidal and Simpson's rule. The tested nonlinear functions of the definite integration are below:

$$\begin{aligned}
f_1(x) &= \frac{1}{1+x^2}; \text{on } [0, 1], & f_2(x) &= \frac{\sin(x)}{x}; \text{on } [0, 0.8], \\
f_3(x) &= e^{-x^2}; \text{on } [0, 1], & f_4(x) &= 3e^{-x}\sin(x^2) + 1; \text{on } [0, 3], \\
f_5(x) &= 2 + \sin(2\sqrt{x}); \text{on } [1, 6]
\end{aligned}$$

where the correct values, to eight decimals, are 0.78539816, 0.77209579, 0.74682413, 3.83086839 and 8.18347292, respectively.

The results of the algorithms are shown in the following tables with $n = 4$ and $n = 8$.

Table 1: Numerical Results with $n = 4$.

n=4	Trapezoidal rule		Simpson's rule		TPA	
	Results	Error	Results	Error	Results	Error
$f_1(x)$	0.78279412	2.6×10^{-2}	0.78539216	6.0×10^{-6}	0.78539813	3.3×10^{-8}
$f_2(x)$	0.77126207	8.3×10^{-4}	0.77209701	1.2×10^{-6}	0.77209577	1.5×10^{-8}
$f_3(x)$	0.74298410	3.8×10^{-3}	0.74685538	3.1×10^{-5}	0.74682612	1.9×10^{-6}
$f_4(x)$	3.75775474	7.3×10^{-2}	3.73453363	9.6×10^{-2}	3.83953267	8.7×10^{-3}
$f_5(x)$	8.25197667	6.8×10^{-2}	8.17044256	1.3×10^{-2}	8.18240542	1.1×10^{-3}

Table 2: Numerical Results with $n = 8$.

n=8	Trapezoidal rule		Simpson's rule		TPA	
	Results	Error	Results	Error	Results	Error
$f_1(x)$	0.78474712	6.5×10^{-4}	0.78539813	3.3×10^{-8}	0.78539816	3.3×10^{-9}
$f_2(x)$	0.77188734	2.0×10^{-4}	0.77209577	1.5×10^{-8}	0.77209579	9.5×10^{-9}
$f_3(x)$	0.74586561	9.5×10^{-4}	0.74682612	1.9×10^{-6}	0.74682426	1.2×10^{-7}
$f_4(x)$	3.81908819	1.1×10^{-2}	3.83953267	8.7×10^{-3}	3.83124826	3.7×10^{-4}
$f_5(x)$	8.19979823	1.6×10^{-2}	8.18240542	1.1×10^{-3}	8.18340328	6.9×10^{-5}

4 Conclusion and Remarks

From the numerical results, our algorithm have more accuracy than both of Trapezoidal and Simpson's rule with the small value of number of subinterval. However, our computational algorithm have to solve the equations system of three variables which more complicate in computation than others.

References

[1] G. Dalquist and A. Björck,*Numerical Methods in Scientific Computing*, SIAM. 1(2008).

[2] J. H. Mathews and K. K. Fink, *Numerical Methods using Matlab*, 4th Edition, (2004).

[3] J. E. Flaherty, *Numerical Integration*, http://www.cs.rpi.edu/ flaherje/pdf/fea6.pdf.

[4] J. H. Mathews,*Modules for Numerical Methods and Numerical Analysis*, http://math.fullerton.edu/mathews/n2003/NumericalUndergradMod.html.

[5] P. K. Kythe and M. R. Schaferkotter,*Methods of Numerical Integration : Second Edition*, Chapman and Hall/CRC (2005).

[6] S. J. Farlow and G. M. Haggard, *Calculus and It's Application*, McGraw-Hill (1990).

Received: October, 2012

บันทึกข้อความ

ส่วนราชการ สาขาวิชาคณิตศาสตร์และสถิติ คณะวิทยาศาสตร์และเทคโนโลยี โทร. ๒๑๙

ที่. **วันที่** ๒๕ มีนาคม ๒๕๕๖

เรื่อง ขอใช้เครื่อง Parallel.snru.ac.th

เรียน ดร. สอาด ม่วงจันทร์

ด้วยข้าพเจ้าอาจารย์ศรีจันทร์ ทานะขันธ์ ตำแหน่งอาจารย์ สังกัดสาขาวิชาคณิตศาสตร์และสถิติ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏสกลนคร กำลังทำวิจัยเรื่อง A New Algorithm of Modified Bisection Method for Nonlinear Equation ซึ่งมีความจำเป็นที่จะขอเข้าใช้เครื่องเพื่อรันโปรแกรมของงานวิจัย เพื่อตรวจสอบความถูกต้องและความสอดคล้องของผลที่ได้กับทฤษฎี ดังนั้นข้าพเจ้าจึงขอความอนุเคราะห์เข้าใช้เครื่องเพื่อประมวลผลการคำนวณ

จึงเรียนมาเพื่อโปรดพิจารณา

(อาจารย์ศรีจันทร์ ทานะขันธ์)

อาจารย์ประจำสาขาวิชาคณิตศาสตร์และสถิติ

Applied Mathematical Sciences, Vol. 7, 2013, no. 123, 6107 - 6114
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ams.2013.39531

A New Algorithm of Modified Bisection Method for Nonlinear Equation

S. Tanakan

Department of Mathematics and Statistics
Faculty of Science and Technology
Sakon Nakhon Rajabhat University
Sakon Nakhon 47000, Thailand
srijan220406@gmail.com

Abstract

The aim of this research is to proposed a new computational algorithm of modified bisection method for nonlinear equation. In numerical results, we performed the algorithm on GNU Octavce(version 3.6.1) and compared the error and number of iterations with bisection and Newton methods.

Keywords: Nonlinear equation, Bisection method, Newton method

1 Introduction

The problem of finding an approximation to the root of an nonlinear equation can be found in many fields of sciences and engineering ([2], [3], [4]). A root-finding algorithm is a numerical method, or algorithm, for finding a value x such that $f(x) = 0$, for a given function f. Such an x is called a root of the function f.

The simplest root-finding algorithm is *the bisection method.* It works when f is a continuous function and it requires previous knowledge of two initial guesses, a and b, such that $f(a)$ and $f(b)$ have opposite signs, then find the midpoint of $[a, b]$, and then decide whether the root lies on $[a, (a + b)/2]$ or $[(a+b)/2, b]$. Repeat until the interval is sufficiently small. By the intermediate

value theorem ([2], [3], [4],[6]) implies that a number x^* exists in (a, b) with $f(x^*) = 0$. Although the bisection method is reliable, but it converges slowly, gaining one bit of accuracy with each iteration.

Theorem 1.1. ***(Intermediate Value Theorem)****: Given a continuous real-valued function $f(x)$ defined on an interval $[a, b]$, then if y is a point between the values of $f(a)$ and $f(b)$, then there exists a point r such that $y = f(r)$.*

The *Newton method* is much more efficient than the bisection method. However, the Newton method requires the calculation of the derivative of a function at the reference point, which is not always easy. Furthermore, the tangent line often shoots wildly and might occasionally be trapped in a loop ([2], [3], [4],[8]). Newton's method may not converge if started too far away from a root. However, when it does converge, it is faster than the bisection method, and is usually quadratic.

However, Newton's Method fails to produce a solution, if there is no solution to be found or the initial solution is not good enough for the method. In the practice, the initial solution is really important for the Newton method. Some initial solutions will lead to the exact numerical solution. But some initial solutions can make the method diverges ([2], [4], [7], [8]).

Hence, in this paper, we construct and propose a new algorithm of modified bisection algorithm for solve nonlinear equation. In numerical results, we tested the algorithm and compared the number of iterative and numerical error with the bisection and Newton methods.

2 Main Results

Let f be a continuous function and defined on $[a, b]$ which $f(a) \cdot f(b) < 0$. Firstly, we set $a_1 = a$ and $b_1 = b$. For an integer $k \geq 1$, By the bisection method, we have $c_k = (a_k + b_k)/2$. Next, we consider a new subinterval (a_k^*, b_k^*) by

$$(a_k^*, b_k^*) = \begin{cases} (a_k, c_k) & , \text{ if } f(a_k)f(c_k) < 0 \\ (c_k, b_k) & , \text{ if } f(c_k)f(b_k) < 0 \end{cases} \tag{1}$$

Then, we can find the equation of straight line from the points $(a_k^*, f(a_k^*))$ and $(b_k^*, f(b_k^*))$ as follows:

$$y = mx + c \tag{2}$$

where

$$m = \frac{f(b_k^*) - f(a_k^*)}{b_k^* - a_k^*}$$

and

$$c = f(b_k^*) - m \cdot b_k^* \quad \text{or} \quad c = f(a_k^*) - m \cdot a_k^*.$$

Hence, the x-intercept of the straight line is at a point $x_k = -\frac{c}{m}$. That is

$$x_k = b_k^* - f(b_k^*) \cdot \frac{b_k^* - a_k^*}{f(b_k^*) - f(a_k^*)}$$

or

$$x_k = a_k^* - f(a_k^*) \cdot \frac{b_k^* - a_k^*}{f(b_k^*) - f(a_k^*)}.$$

Finally, we choose the new subinterval for the next iteration as follows:

$$(a_{k+1}, b_{k+1}) = \begin{cases} (a_k^*, x_k) & , \text{ if } f(a_k^*)f(x_k) < 0 \\ (x_k, b_k^*) & , \text{ if } f(x_k)f(b_k^*)) < 0. \end{cases} \tag{3}$$

The process is continued until the interval is sufficiently small or the approximate solution is sufficiently close to the exact solution.

Therefore, we can state the algorithm for finding a solution of nonlinear equation $f(x) = 0$ on an interval $[a, b]$ as follows:

Modified Bisection Algorithm (MBA):

Step 1: Set $k = 1$, tolerance$(TOL) \approx 0.1 \times 10^{-7}$
and $a_k = a$, $b_k = b$; where $f(a)f(b) < 0$.

Step 2: Compute $c = (a_k + b_k)/2$.

Step 3: Compute for a subinterval (a_k^*, b_k^*) by

$$(a_k^*, b_k^*) = \begin{cases} (a_k, c_k) & , \text{ if } f(a_k)f(c_k) < 0 \\ (c_k, b_k) & , \text{ if } f(c_k)f(b_k) < 0 \end{cases}$$

Step 4: Compute for $x_k = -\frac{c}{m}$, where $m = \frac{f(b_k^*) - f(a_k^*)}{b_k^* - a_k^*}$ and $c = f(b_k^*) - m \cdot b_k^*$
or $c = f(a_k^*) - m \cdot a_k^*$.

Step 5: IF $|f(x_k)| < TOL$, then STOP program
ELSE

$$(a_{k+1}, b_{k+1}) = \begin{cases} (a_k^*, x_k) & , \text{ if } f(a_k^*)f(x_k) < 0 \\ (x_k, b_k^*) & , \text{ if } f(x_k)f(b_k^*)) < 0. \end{cases}$$

and set $k = k + 1$, GOTO step 2.

From the algorithm of modified bisection method, we have the following theorems.

Theorem 2.1. *Let f be a continuous function and defined on $[a,b]$ which $f(a)\cdot f(b)<0$. The modified bisection method generates a sequence $\{x_n\}_{n=1}^{\infty}$ with*

$$a_k < x_k < b_k, \qquad for \quad k \geq 1. \tag{4}$$

Proof: Since $f(a)\cdot f(b)<0$, hence we separate to two cases:

Case 1: $f(a_k)<0$ and $f(b_k)>0$.
Consider a subinterval (a_k^*, b_k^*) in equation (1):
(i.) If $f(a_k)f(c_k)<0$ then we have $a_k^*=a_k$, $b_k^*=c_k$ and $f(b_k^*)>0$. So, we have

$$f(b_k^*)\cdot\frac{b_k^*-a_k^*}{f(b_k^*)-f(a_k^*)}>0$$

then

$$x_k=b_k^*-f(b_k^*)\cdot\frac{b_k^*-a_k^*}{f(b_k^*)-f(a_k^*)}<b_k^*<b_k.$$

Since,

$$f(a_k^*)\cdot\frac{b_k^*-a_k^*}{f(b_k^*)-f(a_k^*)}<0$$

then, we have that

$$x_k=a_k^*-f(b_k^*)\cdot\frac{b_k^*-a_k^*}{f(b_k^*)-f(a_k^*)}>a_k^*=a_k.$$

Hence,

$$a_k<x_k<b_k.$$

(ii.) If $f(c_k)f(b_k)<0$, then we have $a_k^*=c_k$, $b_k^*=b_k$ and $f(a_k^*)<0$.
The proof is similarly.

Case 2: $f(a_k)<0$ and $f(b_k)>0$.
This proof is rather similar to the above.

□

Theorem 2.2. *If a_n and b_n are satistfy equation (3) then*

$$b_n - a_n \leq \frac{b-a}{2^n}, \quad for \ n \geq 1$$

where $b_1 = b, a_1 = a$.

Proof: It's easy to prove by using a mathematical induction..

□

Theorem 2.3. *Let f be a continuous function and defined on $[a,b]$ which $f(a) \cdot f(b) < 0$. The modified bisection method generates a sequence $\{x_n\}_{n=1}^{\infty}$ approximating a zero x^* of f with*

$$|x_n - x^*| < \frac{b-a}{2^{n+1}}, \quad where \quad n \geq 1. \tag{5}$$

Proof: From the algorithm, for any $n \geq 1$ we have two cases as follows:

Case 1: If $f(a_n)f(c_n) < 0$, then $a_n < x_n < c_n < b_n$ and $a_n < x^* < c_n < b_n$. So, we have

$$x_n - b_n < x_n - c_n < x_n - x^* < x_n - a_n. \tag{6}$$

Since $x_n < c_n$, then $x_n - a_n < c_n - a_n = \frac{b_n - a_n}{2}$. Since $a_n < x_n$. So, we have that $a_n - c_n < x_n - c_n$. That is

$$-\frac{(b_n - a_n)}{2} < x_n - c_n < x_n - x^* < \frac{b_n - a_n}{2}.$$

Therfore, from theorem 2.2, we have

$$|x_n - x^*| < \frac{b_n - a_n}{2} = \frac{b-a}{2^{n+1}}.$$

Case 2: If $f(c_n)f(b_n) < 0$, then $a_n < c_n < x_n < b_n$ and $a_n < c_n < x^* < b_n$. Hence, we have

$$x_n - b_n < x_n - x^* < x_n - c_n < x_n - a_n. \tag{7}$$

Since $c_n < x_n$ and equation (7), then

$$c_n - b_n < x_n - b_n$$

or

$$-\frac{(b_n - a_n)}{2} < x_n - b_n < x_n - x^*. \tag{8}$$

Since $x_n < b_n$ and $c_n < x^*$, then we have $x_n - x^* < b_n - x^*$ and $b_n - x^* < b_n - c_n$. Hence, we have

$$x_n - x^* < b_n - c_n = \frac{b_n - a_n}{2}. \tag{9}$$

Therefore, from equations (7), (8) and (9) we have that

$$-\frac{(b_n - a_n)}{2} < x_n - x^* < \frac{b_n - a_n}{2}.$$

Hence, from theorem 2.2

$$|x_n - x^*| < \frac{b_n - a_n}{2} = \frac{b - a}{2^{n+1}}. \tag{10}$$

The proof is completed.

□

3 Numerical Results

In this section, we tested the algorithms with the specific examples ([1], [5], [7], [8]) on GNU Octavce(version 3.6.1) program. For the accuracy, we use tolerance error(TOL) less than 0.1×10^{-7}. The tested nonlinear equations are below:

$f_1(x) = e^{-x} + \cos(x)$ on $[-2, 2]$,
$f_2(x) = 10xe^{-x^2} - 1$ on $[-1, 1]$,
$f_3(x) = x^3 - 2x + 2$ on $[-3, 3]$,
$f_4(x) = x^2 - e^x - 3x + 2$ on $[-2, 2]$,
$f_5(x) = \sin(x) - \frac{x}{2}$ on $[-3, 5]$,
$f_6(x) = \sin(x) - x^5 + x^3 - 1$ on $[-5, 2]$.

The results of the algorithms are shown in the following tables:
In experimental results, we set the initial solutions for Newton method by $x_0 = (a + b)/2$ and choose one point in the interval.

4 Conclusion and Remarks

From the numerical results, a modified bisection algorithms can reduced the number of iterations which less than the iteration number of the bisection method and nearby to the iteration number of Newton method while the error are less than the tolerance. However, if we choose the initial solution $x_0 = 0$ for $f_3(x)$ and $x_0 = 1$ for $f_6(x)$, then the Newton's method fails to produce a solution.

In the future work, we may apply these algorithms to find another solutions on the interval or one may extend the algorithm for the system of nonlinear equation.

Table 1: Numerical results of bisection, MBA and Newton methods.

Problem	Algorithms	Interval	No. iterations	Solution
$f_1(x)$	Bisection	[-2,2]	18	1.74613953
	MBA	[-2,2]	5	1.74613954
	Newton	$x_0 = 0$	4	1.74613953
		$x_0 = -1$	5	1.74613953
$f_2(x)$	Bisection	[-1,1]	26	0.10102585
	MBA	[-1,1]	5	0.10102585
	Newton	$x_0 = 0$	3	0.10102585
		$x_0 = 1$	4	0.10102585
$f_3(x)$	Bisection	[-3,3]	22	-1.76929235
	MBA	[-3,3]	9	-1.76929235
	Newton	$x_0 = 0$	-	-
		$x_0 = -1$	7	-1.76929235
$f_4(x)$	Bisection	[-2,2]	28	0.25753029
	MBA	[-2,2]	4	0.25753029
	Newton	$x_0 = 0$	3	0.25753029
		$x_0 = -1$	4	0.25753029
$f_5(x)$	Bisection	[-3,5]	29	1.89549427
	MBA	[-3,5]	8	1.89549427
	Newton	$x_0 = 1$	13	1.89549427
		$x_0 = 3$	5	1.89549427
$f_6(x)$	Bisection	[-5,2]	32	-1.34557313
	MBA	[-5,2]	5	-1.34557313
	Newton	$x_0 = -1.5$	4	-1.34557313
		$x_0 = 1$	-	-

References

[1] A. Cordero and *at el.*, *Efficient three-step iterative methods with sixth order convergence for nonlinear equations*, Numer. Algor., **53** (2010), 485–495.

[2] G. Dalquist and A. Bjorck,*Numerical Methods in Scientific Computing*, SIAM. **1**(2008).

[3] J. H. Mathews and K. K. Fink, *Numerical Methods using Matlab*, 4th Edition, (2004).

[4] J. H. Mathews,*Modules for Numerical Methods and Numerical Analysis*, http://math.fullerton.edu/mathews/n2003/NumericalUndergradMod.html.

[5] R. Thukral, *Introduction to Fifth-order Iterative Method for Solving Nonlinear Equations*, Inter. J. of Computational and Applied Mathematics, **4** (2)(2009), 135-140.

[6] S. J. Farlow and G. M. Haggard, *Calculus and It's Application*, McGraw-Hill (1990).

[7] S. Muangchan and S. Sompong, *Technique of Initial Solution Finding for the Newton Method*, Int. J. Contemp. Math. Sciences, Vol. 6, 2011, no. 41, 2037-2043.

[8] Wikipedia: The Free Encyclopedia. Wikimedia Foundation Inc. *Newton's method*, Available from http://http://en.wikipedia.org/wiki/Newton's_method, Retrieved 5 September 2013.

Received: September 27, 2013

ภาคผนวก ง.

เอกสารเพิ่มเติมกรณีที่นำไปใช้
ในการเรียนการสอนวิชาการวิเคราะห์เชิงตัวเลข

มหาวิทยาลัยราชภัฏสกลนคร

กลุ่มโปรแกรมวิชาคณิตศาสตร์และสถิติ **คณะวิทยาศาสตร์และเทคโนโลยี**

รายวิชา การวิเคราะห์เชิงตัวเลข (Numerical Analysis)

จำนวน 3 (2-2-5) หน่วยกิต (จำนวนคาบ/สัปดาห์) ระดับปริญญาตรี

1. คำอธิบายรายวิชา

การวิเคราะห์ความคลาดเคลื่อน ผลเฉลยของสมการแบบไม่เชิงเส้น ผลเฉลยของระบบสมการเชิงเส้น การประมาณค่าในช่วง การประมาณค่ากำลังสองน้อยที่สุด อนุพันธ์และการอินทิเกรตเชิงตัวเลข ผลเฉลยเชิงตัวเลขของสมการเชิงอนุพันธ์

2. จุดประสงค์รายวิชา

2.1 เมื่อนักศึกษาศึกษาการคำนวณเชิงตัวเลข แล้ว นักศึกษาสามารถนำความรู้เรื่องแนวคิดเบื้องต้น ไปเป็นพื้นฐานในการแก้ปัญหาอื่นที่เกี่ยวข้องได้

2.2 นำความรู้เรื่องความผิดพลาดของกระบวนการคำนวณเชิงตัวเลข ไปเป็นพื้นฐานในการแก้ปัญหาคณิตศาสตร์และปัญหาอื่นที่เกี่ยวข้องได้

2.3 นำความรู้เรื่องวิธีการประมาณค่าในช่วงและช่วงข้อมูล รากสมการที่ไม่เป็นเชิงเส้นไปเป็นพื้นฐานในการแก้ปัญหาคณิตศาสตร์และปัญหาอื่นที่เกี่ยวข้องได้

2.4 นำความรู้เรื่องอินทิกรัลและค่าอนุพันธ์เชิงตัวเลข ระบบสมการเชิงเส้นไปใช้ในการสร้างและแก้โจทย์ปัญหาทางคณิตศาสตร์คอมพิวเตอร์ได้

2.5 นำความรู้เรื่องเกี่ยวกับการผลเฉลยเชิงตัวเลขของสมการเชิงอนุพันธ์เพื่อแก้ปัญหาอื่นที่เกี่ยวข้องได้

3. อาจารย์ผู้สอน

ดร. สอาด ม่วงจันทร์ วุฒิการศึกษา วท.บ. (คณิตศาสตร์)

วท.ม. (Computational Science)

วท.ด. (Applied Mathematics)

E-mail: msaat@hotmail.com **ห้องทำงาน:** ตึก7 ชั้น 1 สาขาวิชาคณิตศาสตร์และสถิติ

4. **เนื้อหาหลัก**

การวิเคราะห์ความคลาดเคลื่อน ผลเฉลยของสมการแบบไม่เชิงเส้น ผลเฉลยของระบบสมการเชิงเส้น การประมาณค่าในช่วง การประมาณค่ากำลังสองน้อยที่สุด อนุพันธ์และอินทิเกรตเชิงตัวเลข เสริมกรณีศึกษาการเขียนโปรแกรมคำนวณแบบขนาน ผลเฉลยเชิงตัวเลขของสมการเชิงอนุพันธ์ และจัดทำโครงงาน

5. **การวัดผลประเมินผล**

5.1 การวัดผล กำหนดคะแนน **100 คะแนน**

1. คะแนนระหว่างภาค *60 คะแนน*

(ก) เวลาเรียน กำหนดอย่างน้อยต้องมาเรียน 80% 10 คะแนน

(ข) ส่งงาน หรือแบบฝึกหัด 20 คะแนน

(ค) สอบวัดผลระหว่างภาค 30 คะแนน

2. คะแนนสอบปลายภาค *40 คะแนน*

5.2 การประเมินผล

คะแนนระหว่าง	80-100 คะแนน	ได้เกรด A
คะแนนระหว่าง	75-79 คะแนน	ได้เกรด B+
คะแนนระหว่าง	70-74 คะแนน	ได้เกรด B
คะแนนระหว่าง	65-69 คะแนน	ได้เกรด C+
คะแนนระหว่าง	60-64 คะแนน	ได้เกรด C
คะแนนระหว่าง	50-59 คะแนน	ได้เกรด D+
คะแนนระหว่าง	40-49 คะแนน	ได้เกรด D
คะแนนระหว่าง	0-39 คะแนน	ได้เกรด F

```
        tmp(i,j) = a(i,j)
      ENDDO
    ELSE
      DO i = 1, n
        tmp(i,j) = 0.0
      ENDDO
    ENDIF
  ENDDO
  CALL MPI_ALLREDUCE(tmp, a, n*n, MPI_REAL, MPI_SUM,
 &                   MPI_COMM_WORLD, ierr)
```

This code uses the technique of superposition (Chapter 3.5.5, "Superposition" on page 78).

4.4 SOR Method

The following program solves a two-dimensional Laplace equation using the successive over-relaxation (SOR) method. The outermost loop is for the convergence of `x()`, and the inner loops regarding `i` and `j` are for updating the value of `x()`. The convergence is accelerated with the over-relaxation parameter omega.

```
      PROGRAM sor
      PARAMETER (mmax = 6, nmax = 9)
      PARAMETER (m = mmax - 1, n = nmax - 1)
      REAL x(0:mmax, 0:nmax)
      ...
      DO k = 1, 300
        err1 = 0.0
        DO j = 1, n
          DO i = 1, m
            temp = 0.25 * (x(i,j-1) + x(i-1,j)
     &                   + x(i+1,j) + x(i,j+1)) - x(i,j)
            x(i,j) = x(i,j) + omega * temp
            IF (abs(temp) > err1) err1 = abs(temp)
          ENDDO
        ENDDO
        IF (err1 <= eps) exit
      ENDDO
      ...
      END
```

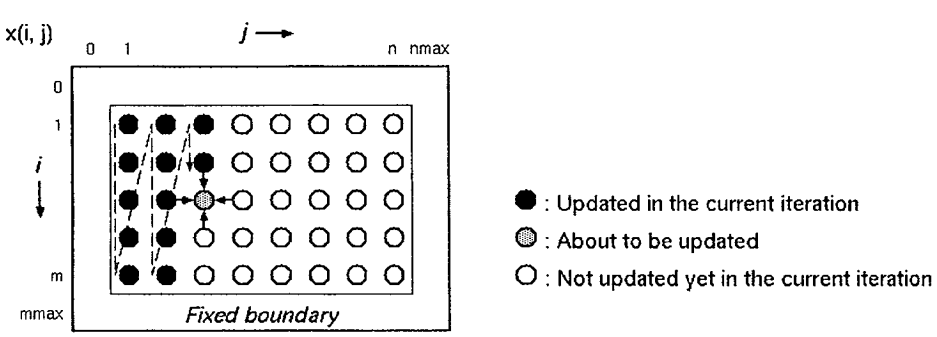

Figure 104. SOR Method: Serial Run

Figure 104 on page 120 shows data dependence of the SOR program. In updating `x(i,j)`, its four neighbors are referenced. However, due to the loop structure, when `x(i,j)` is updated, `x(i,j-1)` and `x(i-1,j)` are already updated in that iteration, whereas `x(i+1,j)` and `x(i,j+1)` are not. Therefore, there is a strict restriction in the order of update, which is shown in dashed arrows in the figure.

You could use the pipeline method (Chapter 3.5.6, "The Pipeline Method" on page 79) to parallelize this program, but, usually, an alternative method called red-black SOR is used.

4.4.1 Red-Black SOR Method

In the red-black SOR, the order of computation is modified from the original SOR method, which means the steps required for convergence may change and the result may be different within the precision that is imposed by the variable eps. First, each matrix element is conceptually colored with red or black as follows.

If `i+j` is even, `x(i,j)` is red. (Shown as circles in Figure 105)

If `i+j` is odd, `x(i,j)` is black. (Shown as boxes in Figure 105)

The red-black SOR updates red elements and black elements alternately. Note that the four neighbors of a red element are all black, and vice versa. So, red (black) elements can be updated independently of other red (black) elements, and, thereby, the red-black SOR can be efficiently parallelized.

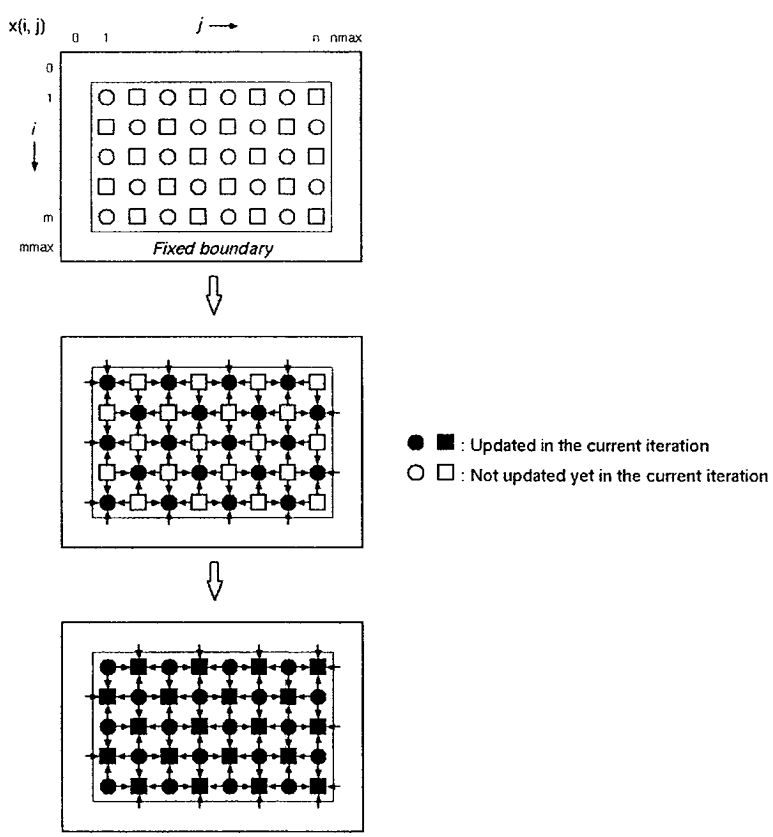

Figure 105. Red-Black SOR Method

The program of the red-black SOR method is as follows.

```
PROGRAM red_black
PARAMETER (mmax = 6, nmax = 9)
PARAMETER (m = mmax - 1, n = nmax - 1)
REAL x(0:mmax, 0:nmax)
...
DO k = 1, 300
  err1 = 0.0
  DO j = 1, n
    DO i = MOD(j+1,2) + 1, m, 2
      temp = 0.25 * (x(i,j-1) + x(i-1,j)
&                  + x(i+1,j) + x(i,j+1)) - x(i,j)
      x(i,j) = x(i,j) + omega * temp
      IF (abs(temp) > err1) err1 = abs(temp)
    ENDDO
  ENDDO
  DO j = 1, n
    DO i = MOD(j,2) + 1, m, 2
      temp = 0.25 * (x(i,j-1) + x(i-1,j)
&                  + x(i+1,j) + x(i,j+1)) - x(i,j)
      x(i,j) = x(i,j) + omega * temp
      IF (abs(temp) > err1) err1 = abs(temp)
    ENDDO
  ENDDO
   IF (err1 <= eps) exit
ENDDO
...
END
```

Figure 106 on page 123 illustrates how the red-black SOR works with three processes. Matrix `x()` is block-distributed by columns. First, each process packs black (shown as boxes in the figure) boundary elements to buffers, sends them to adjacent processes, and red elements (circles) are updated using black elements (boxes). Next, each process sends updated red boundary elements to adjacent processes, and black elements are updated using red elements. These steps repeat until the matrix converges.

For convenience, a module is used in the parallel program.

```
MODULE para
INCLUDE 'mpif.h'
PARAMETER (mmax = 6, nmax = 9)
PARAMETER (m = mmax - 1, n = nmax - 1)
REAL x(0:mmax, 0:nmax)
REAL bufs1(mmax), bufr1(mmax)
REAL bufs2(mmax), bufr2(mmax)
INTEGER istatus(MPI_STATUS_SIZE)
INTEGER nprocs, myrank, jsta, jend, inext, iprev
END
```

Arrays `bufs1()` and `bufr1()` are the send and receive buffers for the left neighbor process (`rank=iprev`), and arrays `bufs2()` and `bufr2()` are for the right neighbor process (`rank=inext`).

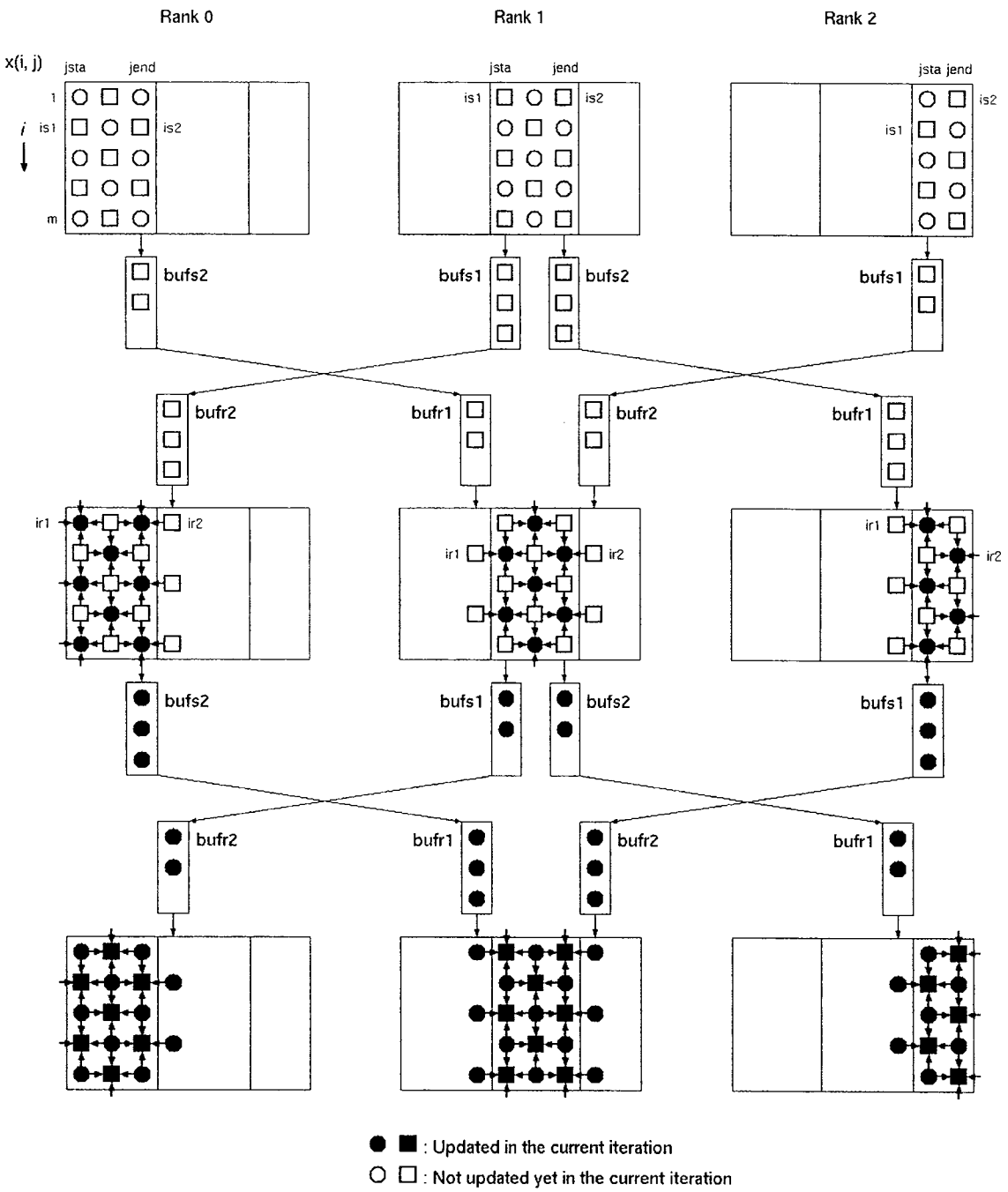

Figure 106. Red-Black SOR Method: Parallel Run

The following is the main program.

```
PROGRAM main
USE para
CALL MPI_INIT(ierr)
CALL MPI_COMM_SIZE(MPI_COMM_WORLD, nprocs, ierr)
CALL MPI_COMM_RANK(MPI_COMM_WORLD, myrank, ierr)
CALL para_range(1, n, nprocs, myrank, jsta, jend)
```

```
      inext = myrank + 1
      iprev = myrank - 1
      IF (inext == nprocs) inext = MPI_PROC_NULL
      IF (iprev == -1)     iprev = MPI_PROC_NULL
      ...
      DO k = 1, 300
        err1 = 0.0
        CALL shift(0)
        DO j = jsta, jend
          DO i = MOD(j+1,2) + 1, m, 2
            temp = 0.25 * (x(i,j-1) + x(i-1,j)
     &                   + x(i+1,j) + x(i,j+1)) - x(i,j)
            x(i,j) = x(i,j) + omega * temp
            IF (abs(temp) > err1) err1 = abs(temp)
          ENDDO
        ENDDO
        CALL shift(1)
        DO j = jsta, jend
          DO i = MOD(j,2) + 1, m, 2
            temp = 0.25 * (x(i,j-1) + x(i-1,j)
     &                   + x(i+1,j) + x(i,j+1)) - x(i,j)
            x(i,j) = x(i,j) + omega * temp
            IF (abs(temp) > err1) err1 = abs(temp)
          ENDDO
        ENDDO
        CALL MPI_ALLREDUCE(err1, err2, 1, MPI_REAL, MPI_MAX,
     &                     MPI_COMM_WORLD, ierr)
        err1 = err2
        IF (err1 <= eps) exit
      ENDDO
      ...
      CALL MPI_FINALIZE(ierr)
      END
```

Subroutine `shift(iflg)` takes care of all the data transmissions: `iflg=0` is for black elements (boxes) and `iflg=1` is for red elements (circles).

```
      SUBROUTINE shift(iflg)
      USE para
      is1 = MOD(jsta + iflg, 2) + 1
      is2 = MOD(jend + iflg, 2) + 1
      ir1 = 3 - is1
      ir2 = 3 - is2
      IF (myrank /= 0) THEN
        icnt1 = 0
        DO i = is1, m, 2
          icnt1 = icnt1 + 1
          bufs1(icnt1) = x(i,jsta)
        ENDDO
      ENDIF
      IF (myrank /= nprocs - 1) THEN
        icnt2 = 0
        DO i = is2, m, 2
          icnt2 = icnt2 + 1
          bufs2(icnt2) = x(i,jend)
        ENDDO
      ENDIF
      CALL MPI_ISEND(bufs1,icnt1,MPI_REAL,iprev,1,MPI_COMM_WORLD,ireqs1,ierr)
      CALL MPI_ISEND(bufs2,icnt2,MPI_REAL,inext,1,MPI_COMM_WORLD,ireqs2,ierr)
      CALL MPI_IRECV(bufr1,mmax, MPI_REAL,iprev,1,MPI_COMM_WORLD,ireqr1,ierr)
      CALL MPI_IRECV(bufr2,mmax, MPI_REAL,inext,1,MPI_COMM_WORLD,ireqr2,ierr)
      CALL MPI_WAIT(ireqs1, istatus, ierr)
      CALL MPI_WAIT(ireqs2, istatus, ierr)
      CALL MPI_WAIT(ireqr1, istatus, ierr)
      CALL MPI_WAIT(ireqr2, istatus, ierr)
      IF (myrank /= 0) THEN
        icnt = 0
        DO i = ir1, m, 2
          icnt = icnt + 1
          x(i,jsta-1) = bufr1(icnt)
        ENDDO
      ENDIF
      IF (myrank /= nprocs - 1) THEN
        icnt = 0
        DO i = ir2, m, 2
          icnt = icnt + 1
          x(i,jend+1) = bufr2(icnt)
```

```
  ENDDO
ENDIF
END
```

4.4.2 Zebra SOR Method

If you don't plan to distribute the matrix in both dimensions, besides red-black SOR, consider the option of coloring columns alternately using two colors, for example, black and white. First, update all the white elements, and then update the black elements, which is called zebra SOR method in this publication.

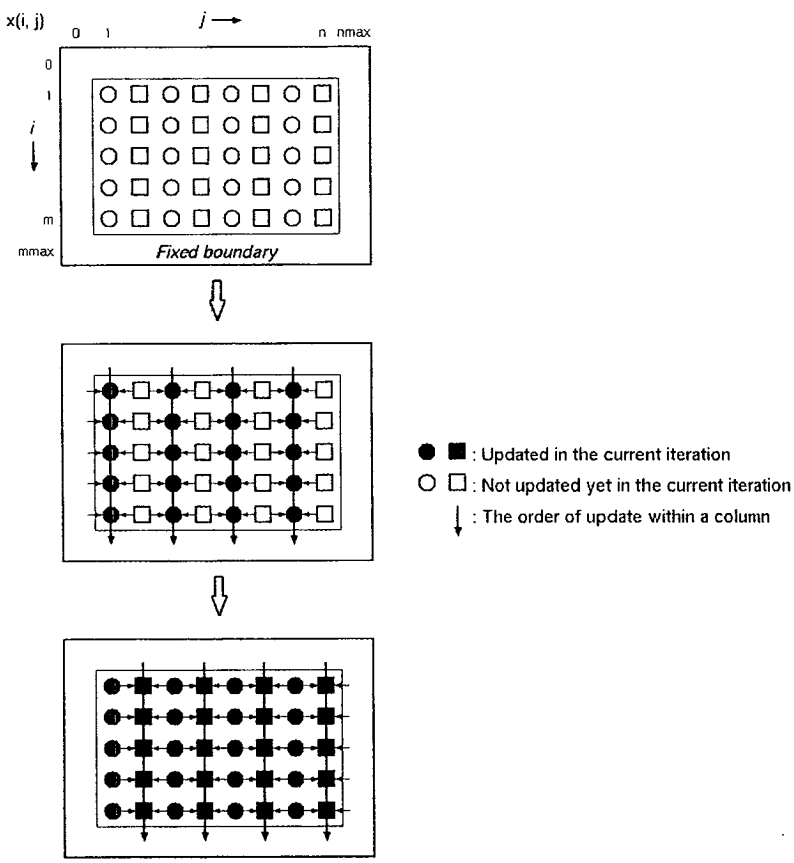

Figure 107. Zebra SOR Method

Figure 107 shows how the computation proceeds in the zebra SOR method. The following is the serial program of zebra SOR.

```
PROGRAM zebra
PARAMETER (mmax = ..., nmax = ...)
PARAMETER (m = mmax - 1, n = nmax - 1)
DIMENSION x(0:mmax, 0:nmax)
...
DO k = 1, 300
  err1 = 0.0
  DO j = 1, n, 2
    DO i = 1, m
      Update x(i,j) and err1
    ENDDO
```

```
      ENDDO
      DO j = 2, n, 2
        DO i = 1, m
          Update x(i,j) and err1
        ENDDO
      ENDDO
      IF (err1 <= eps) EXIT
    ENDDO
    ...
    END
```

When updating elements in a column, you have to maintain the order of the elements. That is, you have to update elements from top to bottom in the figure. In other words, there is a flow dependence within a column. Remember that in red-black SOR, elements of the same color can be updated in any order. In that sense, the minimum block for parallelization is a matrix element in red-black SOR. In zebra SOR, the minimum block for parallelization is a column of elements: as long as the order of update within a column is kept unchanged, you can update the columns in any order. This property enables parallelization.

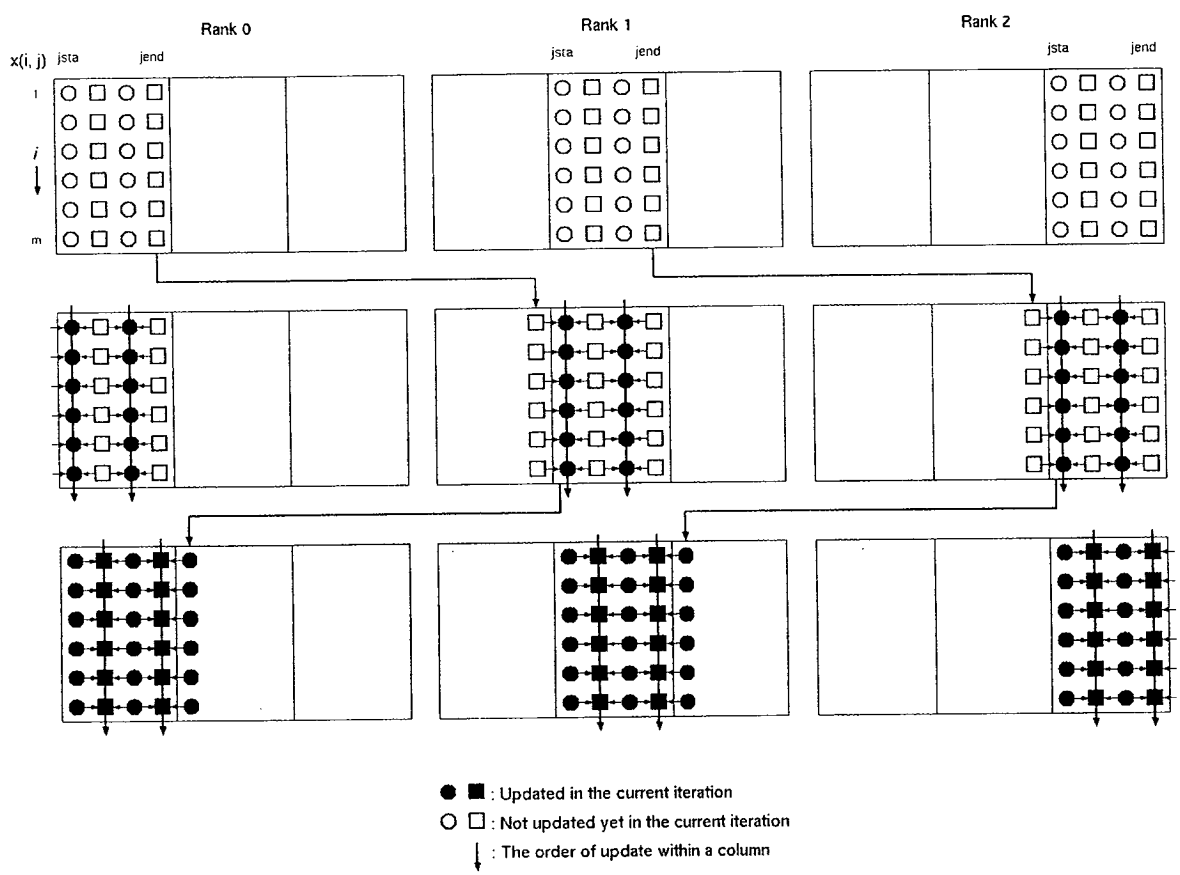

Figure 108. Zebra SOR Method: Parallel Run

The Figure 108 shows how the matrix elements are updated in the parallelized zebra SOR method. The matrix elements are assigned to processes in column-wise block distribution. To make the data transmission uniform among

processes, distribute columns so that the same pattern appears on the boundary between processes. Stated in another way, `jend-jsta+1` should be even in all the processes, except possibly for the last process.

```
      PROGRAM zebrap
      INCLUDE 'mpif.h'
      PARAMETER (mmax = ..., nmax = ...)
      PARAMETER (m = mmax - 1, n = nmax - 1)
      DIMENSION x(0:mmax, 0:nmax)
      INTEGER istatus(MPI_STATUS_SIZE)
      CALL MPI_INIT(ierr)
      CALL MPI_COMM_SIZE(MPI_COMM_WORLD, nprocs, ierr)
      CALL MPI_COMM_RANK(MPI_COMM_WORLD, myrank, ierr)
      CALL para_range(1, (n + 1)/2, nprocs, myrank, jsta, jend)
      jsta = jsta * 2 - 1
      jend = MIN(n, jend * 2)
      inext = myrank + 1
      iprev = myrank - 1
      IF (inext == nprocs) inext = MPI_PROC_NULL
      IF (iprev == -1)     iprev = MPI_PROC_NULL
      ...
      DO k = 1, 300
        err1 = 0.0
        CALL MPI_ISEND(x(1,jend), m, MPI_REAL,
     &       inext, 1, MPI_COMM_WORLD, ireqs, ierr)
        CALL MPI_IRECV(x(1,jsta-1), m, MPI_REAL,
     &       iprev, 1, MPI_COMM_WORLD, ireqr, ierr)
        CALL MPI_WAIT(ireqs, istatus, ierr)
        CALL MPI_WAIT(ireqr, istatus, ierr)
        DO j = jsta, jend, 2
          DO i = 1, m
            Update x(i,j) and err1
          ENDDO
        ENDDO
        CALL MPI_ISEND(x(1,jsta), m, MPI_REAL,
     &       iprev, 1, MPI_COMM_WORLD, ireqs, ierr)
        CALL MPI_IRECV(x(1,jend+1), m, MPI_REAL,
     &       inext, 1, MPI_COMM_WORLD, ireqr, ierr)
        CALL MPI_WAIT(ireqs, istatus, ierr)
        CALL MPI_WAIT(ireqr, istatus, ierr)
        DO j = jsta + 1, jend, 2
          DO i = 1, m
            Update x(i,j) and err1
          ENDDO
        ENDDO
        CALL MPI_ALLREDUCE(err1, err2, 1, MPI_REAL,
     &                     MPI_MAX, MPI_COMM_WORLD, ierr)
        IF (err2 <= eps) EXIT
      ENDDO
      ...
      CALL MPI_FINALIZE(ierr)
      END
```

Note that the data transmission is simple compared with red-black SOR. Since red-black SOR and zebra SOR use different orders in updating matrix elements, the number of time steps needed for convergence may be different. Choose the best algorithm in terms of the running time: there can be other variations of coloring, or ordering, of elements.

The zebra SOR can be regarded as having the same dependence of one-dimensional red-black SOR. The red-black coloring in a lower dimension can be applied to higher dimensions at the cost of the restriction as to how the arrays may be distributed. In the zebra SOR method presented in this section, it is not allowed to divide the matrix in row-wise distribution.

In solving three-dimensional SOR, there also can be various ways of coloring elements. Suppose that a matrix element $x(i, j, k)$ is updated using its six neighbors, $x(i \pm 1, j, k)$, $x(i, j \pm 1, k)$, and $x(i, j, k \pm 1)$ and that you use two colors. The following shows three examples of coloring.

If `i+j+k=1 (mod 2)`, color `x(i,j,k)` red; otherwise color it black

If `j+k=1 (mod 2)`, color `x(i,j,k)` red; otherwise color it black

If `k=1 (mod 2)`, color `x(i,j,k)` red; otherwise color it black

Again, you should decide how to color elements based on the performance of the parallel program and sometimes the amount of work for parallelization.

4.4.3 Four-Color SOR Method

In the following program, eight neighbors are involved in updating the value of `x(i,j)`. Red-black SOR does not work for this case.

```
      PROGRAM main
      PARAMETER (mmax = ..., nmax = ...)
      PARAMETER (m = mmax - 1, n = nmax - 1)
      DIMENSION x(0:mmax, 0:nmax)
      ...
      DO k = 1, 300
        err1 = 0.0
        DO j = 1, n
          DO i = 1, m
            temp = 0.125 * (x(i,  j-1) + x(i-1,j)
     &                    + x(i+1,j)   + x(i,  j+1)
     &                    + x(i-1,j-1) + x(i+1,j-1)
     &                    + x(i-1,j+1) + x(i+1,j+1))
     &              - x(i,j)
            x(i,j) = x(i,j) + omega * temp
            IF (abs(temp) > err1) err1 = abs(temp)
          ENDDO
        ENDDO
      IF (err1 <= eps) EXIT
      ENDDO
      ...
      END
```

You can parallelize the above program using zebra SOR method, but another method, four-color SOR, is presented in this section.

Four-color SOR is a generalization of red-black SOR, which colors `x(i,j)` with one of the four colors depending on whether `i` is even/odd and `j` is even/odd, and updates `x(i,j)` one color at a time. In the following program, "*Update x(i,j) and err1*" represents the italic statements in the original program.

```
      PROGRAM fourcolor
      PARAMETER (mmax = ..., nmax = ...)
      PARAMETER (m = mmax - 1, n = nmax - 1)
      DIMENSION x(0:mmax, 0:nmax)
      ...
      DO k = 1, 300
        err1 = 0.0
        DO j = 1, n, 2
          DO i = 1, m, 2
            Update x(i,j) and err1
          ENDDO
        ENDDO
        DO j = 1, n, 2
          DO i = 2, m, 2
```

```
        Update x(i,j) and err1
      ENDDO
    ENDDO
    DO j = 2, n, 2
      DO i = 1, m, 2
        Update x(i,j) and err1
      ENDDO
    ENDDO
    DO j = 2, n, 2
      DO i = 2, m, 2
        Update x(i,j) and err1
      ENDDO
    ENDDO
    IF (err1 <= eps) EXIT
  ENDDO
  ...
  END
```

Figure 109 shows one iteration of the four-color SOR method.

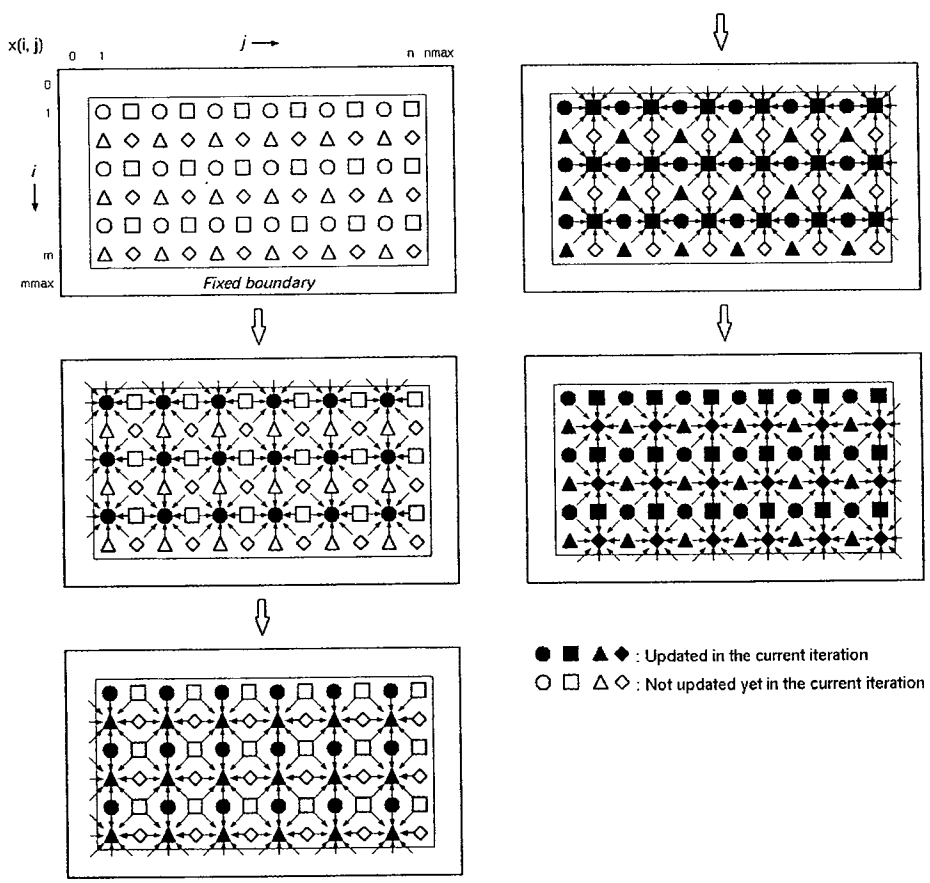

Figure 109. Four-Color SOR Method

Suppose you parallelize the four-color SOR program using column-wise block distribution. For the ease of programming, the number of columns (excluding the fixed boundary elements) assigned to processes should be an even integer

except for process nprocs-1. The behavior of the parallelized program is shown in Figure 110 on page 130.

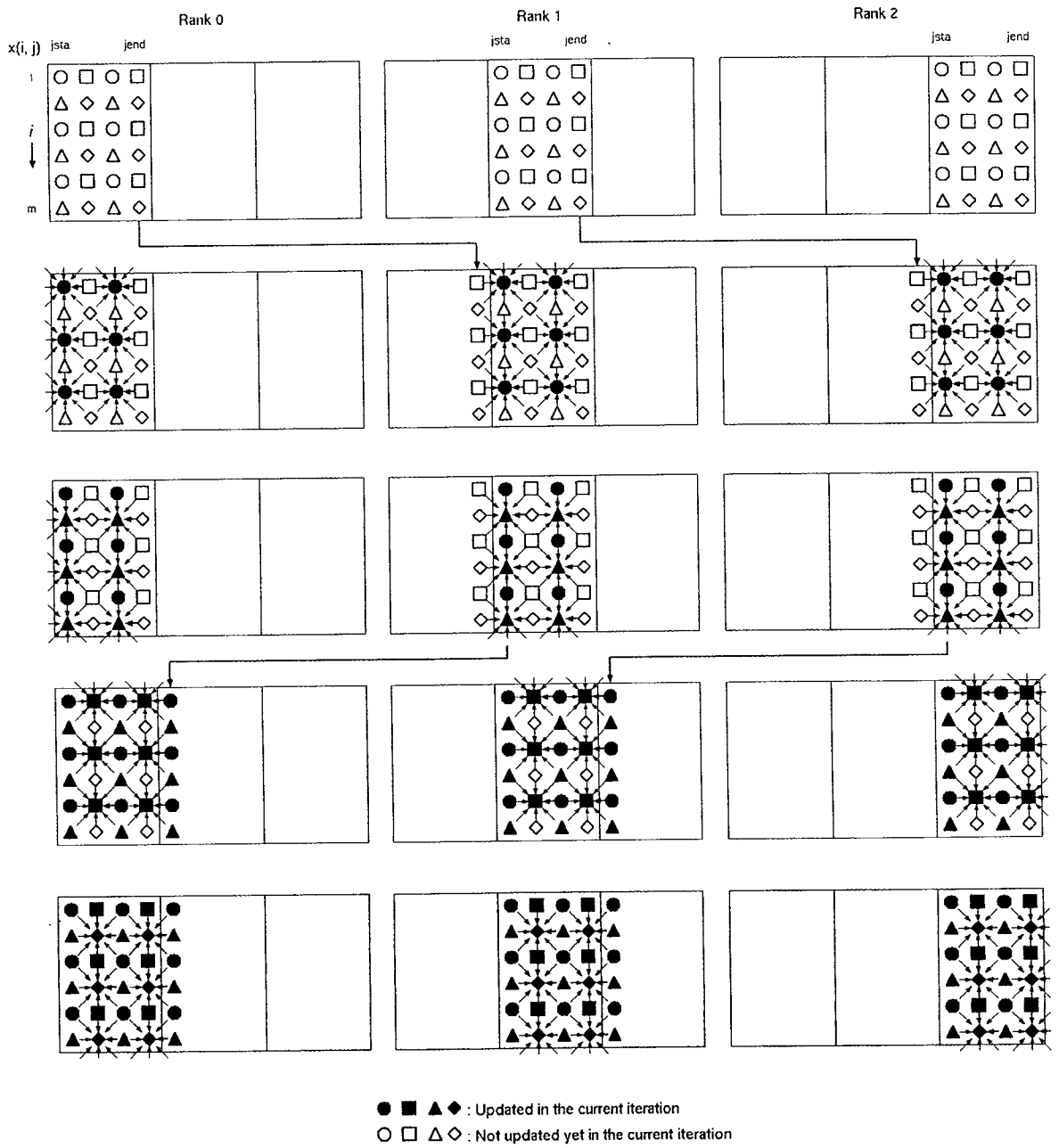

Figure 110. Four-Color SOR Method: Parallel Run

The difference from the red-black SOR method is that data transmissions take place in one direction at a time and that the whole column is transmitted instead of every other element packed into a working array. The following is the parallelized code.

```
PROGRAM fourcolorp
```

```
INCLUDE 'mpif.h'
PARAMETER (mmax = ..., nmax = ...)
PARAMETER (m = mmax - 1, n = nmax - 1)
DIMENSION x(0:mmax, 0:nmax)
INTEGER istatus(MPI_STATUS_SIZE)
CALL MPI_INIT(ierr)
CALL MPI_COMM_SIZE(MPI_COMM_WORLD, nprocs, ierr)
CALL MPI_COMM_RANK(MPI_COMM_WORLD, myrank, ierr)
CALL para_range(1, (n + 1)/2, nprocs, myrank, jsta, jend)
jsta = jsta * 2 - 1
jend = MIN(n, jend * 2)
inext = myrank + 1
iprev = myrank - 1
IF (inext == nprocs) inext = MPI_PROC_NULL
IF (iprev == -1)     iprev = MPI_PROC_NULL
...
DO k = 1, 300
  err1 = 0.0
  CALL MPI_ISEND(x(1,jend), m, MPI_REAL,
&       inext, 1, MPI_COMM_WORLD, ireqs, ierr)
  CALL MPI_IRECV(x(1,jsta-1), m, MPI_REAL,
&       iprev, 1, MPI_COMM_WORLD, ireqr, ierr)
  CALL MPI_WAIT(ireqs, istatus, ierr)
  CALL MPI_WAIT(ireqr, istatus, ierr)
  DO j = jsta, jend, 2
    DO i = 1, m, 2
      Update x(i,j) and err1
    ENDDO
  ENDDO
  DO j = jsta, jend, 2
    DO i = 2, m, 2
      Update x(i,j) and err1
    ENDDO
  ENDDO
  CALL MPI_ISEND(x(1,jsta), m, MPI_REAL,
&       iprev, 1, MPI_COMM_WORLD, ireqs, ierr)
  CALL MPI_IRECV(x(1,jend+1), m, MPI_REAL,
&       inext, 1, MPI_COMM_WORLD, ireqr, ierr)
  CALL MPI_WAIT(ireqs, istatus, ierr)
  CALL MPI_WAIT(ireqr, istatus, ierr)
  DO j = jsta + 1, jend, 2
    DO i = 1, m, 2
      Update x(i,j) and err1
    ENDDO
  ENDDO
  DO j = jsta + 1, jend, 2
    DO i = 2, m, 2
      Update x(i,j) and err1
    ENDDO
  ENDDO
  CALL MPI_ALLREDUCE(err1, err2, 1, MPI_REAL,
&                    MPI_MAX, MPI_COMM_WORLD, ierr)
  IF (err2 <= eps) EXIT
ENDDO
...
CALL MPI_FINALIZE(ierr)
END
```

In spite of the more complex dependence compared to the original program, the parallelized four-color SOR is simpler than the parallelized red-black SOR.

4.5 Monte Carlo Method

As an example of the Monte Carlo method, a random walk in two-dimensions is considered. Issues about random number generation is the subject of this section. The following program simulates a random walk of 100,000 particles in 10 steps, and outputs the distribution of the distances that particles have traveled.

```
PROGRAM main
PARAMETER (n = 100000)
INTEGER itotal(0:9)
```