



รายงานวิจัยฉบับสมบูรณ์

โครงการ

พฤติกรรมเซมิคลาสสิกของอนุภาคในสนามศักย์ที่นิยามโดย $(Q-xy)^2$

The Semiclassical Behaviour of the Particle in the $(Q-xy)^2$ Potential

โดย

ผศ.ดร.กฤษณะเดช เจริญสุธาสินี

สำนักวิชาวิทยาศาสตร์ มหาวิทยาลัยวลัยลักษณ์

กันยายน 2545



รายงานวิจัยฉบับสมบูรณ์

โครงการ

พฤติกรรมเซมิคลาสสิกของอนุภาคในสนามศักย์ที่นิยามโดย $(Q-xy)^2$

The Semiclassical Behaviour of the Particle in the $(Q-xy)^2$ Potential

โดย

ผศ.ดร.กฤษณะเดช เจริญสุธาสินี
สำนักวิชาวิทยาศาสตร์ มหาวิทยาลัยวลัยลักษณ์

กันยายน 2545

สำนักงานกองทุนสนับสนุนการวิจัย (สกว.)
ชั้น 14 อาคาร เอส เอ็ม เทวาเจอร์
เลขที่ 979 17-21 ถนนพหลโยธิน แขวงสามเสนใน
เขตจตุจักร กรุงเทพมหานคร 10400
โทร 298-0455 โทรสาร 298-0476
Home page : <http://www.trf.or.th>
E-mail : trf-info@trf.or.th



วันที่.....	3 ส.ย. 2546
เลขทะเบียน.....	00008
	PDF
เลขเรียกหนังสือ.....	40
	8

C062

รายงานวิจัยฉบับสมบูรณ์

โครงการ

พฤติกรรมเซมิคลาสสิกของอนุภาคในสนามศักย์ที่นิยามโดย $(Q-xy)^2$

The Semiclassical Behaviour of the Particle in the $(Q-xy)^2$ Potential

ผู้วิจัย

ผศ.ดร.กฤษณะเดช เจริญสุธาสินี

สำนักวิชาวิทยาศาสตร์ มหาวิทยาลัยวลัยลักษณ์

สนับสนุนโดยสำนักงานกองทุนสนับสนุนการวิจัย

(ความเห็นในรายงานฉบับนี้เป็นของผู้วิจัย สกว.ไม่จำเป็นต้องเห็นด้วยเสมอไป)

บทคัดย่อ

(รหัสโครงการ) PDF/62/2540
(ชื่อโครงการ) พฤติกรรมเซมิลาสสิกของอนุภาคในสนามศักย์ที่นิยามโดย $(Q-xy)^2$
ชื่อนักวิจัย ผศ.ดร.กฤษณะเดช เจริญสุธาสินี
สำนักวิชาวิทยาศาสตร์ มหาวิทยาลัยวลัยลักษณ์
อ.ท่าศาลา จ.นครศรีธรรมราช
Email address: jkrisana@wu.ac.th
ระยะเวลาโครงการ 3 ปี
ชื่อนักวิจัยที่ปรึกษา Prof.Dr. George Rowlands
Department of Physics, Warwick University, Coventry CV4 7AL
United Kingdom

โครงการนี้มีวัตถุประสงค์

เพื่อคำนวณหาโครงสร้างมหภาคของระนาบเฟสของอนุภาคที่อยู่ในสนามศักย์ $(Q-xy)^2$ จะสนใจระนาบเฟสที่ประกอบด้วยวงโคจรมีคาบที่มีคาบสั้นและคาบไม่ยาวมากนักเนื่องจากวงโคจรเหล่านี้มีผลที่สำคัญอย่างมากกับการคำนวณระดับพลังงาน (Energy Eigenvalues) ซึ่งจะเป็นการวิจัยต่อไป

Keywords: Chaos, Periodic Orbit Calculation, Periodic Orbit Family

Abstract

Project Code : PDF/62/2540
Project Title : The Semiclassical Behaviour of the Particle in the $(Q-xy)^2$ Potential
Investigator : Asst.Prof.Dr. Krisanadej Jaroensutasinee
Institute of Science, Walailak University
Tasala, NakhonSiThammarat
Email address: jkrisana@wu.ac.th

Project Period 3 years

ชื่อนักวิจัยที่ปรึกษา Prof.Dr. George Rowlands
Department of Physics, Warwick University, Coventry CV4 7AL
United Kingdom

Objectives of this project

To calculate macroscopic structure of the phase space of the dynamic of a single particle in $(Q-xy)^2$ potential. We consider the phase space which composes of orbits of short period. This knowledge is important when computing semiclassical energy eigenvalues which is the object of further study.

Keywords: Chaos, Periodic Orbit Calculation, Periodic Orbit Family

พฤติกรรมเซมิคลาสสิกของอนุภาคในสนามศักย์ที่นิยามโดย $(Q-xy)^2$

The Semiclassical Behaviour of the Particle in the $(Q-xy)^2$ Potential

1 เนื้อหางานวิจัย

1.1 บทนำ

1.1.1 ความสำคัญและที่มาของปัญหาที่ทำการวิจัย

ความสำเร็จในการศึกษาระบบต่างๆ ทางฟิสิกส์ในธรรมชาติเกิดจากการสร้างแบบจำลองทางฟิสิกส์ของระบบนั้นๆ แบบจำลองเหล่านี้ช่วยให้เกิดความเข้าใจ และโดยอาศัยความสามารถในการทำนายความเป็นไปในอนาคตของระบบจากแบบจำลองนั้นทำให้เกิดความสามารถที่จะควบคุม ความสามารถในการทำนายและความสามารถที่จะควบคุมธรรมชาติรอบๆตัวเราเหล่านี้ได้ทำให้มนุษย์มีความสามารถที่จะสร้างสิ่งประดิษฐ์และเทคโนโลยีต่างๆ รอบตัวเราที่ช่วยยกมาตรฐานและอำนวยความสะดวกต่อการดำรงชีวิต และเป็นที่มาของเทคโนโลยีที่ใช้กันอยู่ในปัจจุบัน อย่างไรก็ตามขณะนี้เราทราบว่ามีแง่มุมหนึ่งของธรรมชาติ ซึ่งรู้จักกันดีในชื่อ “ส่วนที่ไม่เป็นเชิงเส้น” ของธรรมชาติ แง่มุมนี้ของธรรมชาติมักจะถูก “ละเลย” จาก การทำให้แบบจำลองง่าย (Simplification of the model) การละเลยนี้พบว่าแบบจำลองที่ได้จะสูญเสียคุณสมบัติและพฤติกรรมที่สำคัญๆ ไป แต่เมื่อเราไม่ตัดส่วนไม่เป็นเชิงเส้นออกจากตัวโมเดล (จากการศึกษาโดยใช้คอมพิวเตอร์ประสิทธิภาพสูงในช่วง 10-20 ปีที่ผ่านมา) พบว่าระบบมีความสลับซับซ้อนที่น่าสนใจมากและบรรจุพฤติกรรมมากมายที่เคยถูกละเลย สิ่งนี้ทำลายความสามารถในการทำนายแบบเก่า (Unpredictability) และการควบคุมทำได้ยากขึ้น หรือที่รู้จักกันในชื่อ “พฤติกรรมแบบเคออส” (ความซับซ้อนไร้ระเบียบสามารถเกิดจากองค์ประกอบภายนอกที่รบกวนระบบได้ ซึ่งเราไม่สนใจในที่นี้) ฟิสิกส์ของระบบที่ไม่เป็นเชิงเส้นและเคออสได้กลายเป็นสาขาใหญ่ และแทรกตัวเข้าไปในหลายๆ สาขาวิชานอกจากฟิสิกส์ มีกิจกรรมการวิจัยทั่วโลก มีการตั้งศูนย์วิจัยเฉพาะเรื่องนี้ขึ้นทั่วโลก ตัวอย่างที่สำคัญมากคือ Santa Fe Institute ที่สละการวิจัยทั้งหมดให้กับศาสตร์ของความซับซ้อน (Complex systems) นักวิจัยทั่วโลกเริ่มเข้าถึงปรัชญาที่เกิดจากวิชาเคออสที่กล่าวว่า ความสลับซับซ้อนไม่จำเป็นต้องมาจากความซับซ้อน มันสามารถเกิดจากโครงสร้างง่ายๆ ได้

จากการค้นพบปรัชญานี้ก่อให้เกิดการถกเถียงและศึกษากันอย่างกว้างขวางถึงสิ่งที่เป็นรากฐานของวิชาฟิสิกส์ (และวิทยาศาสตร์) นักวิทยาศาสตร์ทั่วโลกได้ให้ความสนใจศึกษาระบบทางควอนตัมที่ในระดับคลาสสิกเป็นเคออสอย่างมากมาย หนึ่งในสิ่งที่ศึกษาคือความไม่สอดคล้องกันระหว่างกลศาสตร์ควอนตัมกับกลศาสตร์คลาสสิก สำหรับระบบที่มีการแสดงพฤติกรรมเคออสในระดับคลาสสิกควอนตัมเคออสคือการศึกษาถึงความเกี่ยวพันเพื่อจะแก้ความขัดแย้งนี้ การวิจัยจะมุ่งเน้นไปที่ระดับเซมิคลาสสิกที่เป็นรอยต่อของทั้งสองระดับ ปัจจุบันเริ่มมีความเข้าใจสิ่งที่เกิดขึ้นในบริเวณเชื่อมโยง

นี้สำหรับบางระบบที่ได้ถูกศึกษาแล้ว ตัวอย่างเช่น ระบบของอะตอมที่ถูกกระตุ้นไปอยู่ที่สภาวะเลขควอนตัมสูงๆและอยู่ภายในสนามแม่เหล็กความเข้มสูง เป็นต้น

ในการศึกษาหรือทำความเข้าใจระบบพลวัตแบบไม่เชิงเส้นนี้ต้องอาศัยความรู้เกี่ยวกับโครงสร้างของระนาบเฟส ที่ประกอบด้วยวงโคจรที่มีคาบ ทั้งเสถียรและไม่เสถียร รวมไปถึงวงโคจรที่มีคาบเทียม (Quasiperiodicity) การหาวงโคจรเหล่านี้ไม่ใช่เรื่องง่าย วงโคจรเหล่านี้ดูเหมือนว่าจะมีอยู่โดยไม่มีเหตุผล นักคณิตศาสตร์และนักวิจัยได้พยายามสร้างเครื่องมือสำหรับหาวงโคจรเหล่านี้ ยังไม่มีผู้ใดค้นพบวิธีมาตรฐาน โครงสร้างมหภาค (Global Structure) เป็นสิ่งที่ต้องการอย่างมากแต่คำนวณได้ยากในการนำไปใช้ การหาวิธีใหม่ๆในการสร้างระนาบเฟสมหภาคนี้เป็นวัตถุประสงค์หนึ่งของโครงการวิจัยนี้

1.1.2 วัตถุประสงค์ของโครงการ

- เพื่อประยุกต์หรือคิดค้นวิธีการต่างๆ เพื่อหาวงโคจรที่มีคาบ ทั้งที่มีเสถียรภาพและไม่มีเสถียรภาพ สำหรับอนุภาคในสนามศักย์ $(Q-xy)^2$
- เพื่อศึกษาโครงสร้างมหภาค (Global Structure) ของระนาบเฟสภาพตัดขวางของปองคาเรของพลศาสตร์ของอนุภาคในสนามศักย์ $(Q-xy)^2$ ที่ประกอบด้วย วงโคจรมีคาบ โครงสร้าง มหภาคของระนาบเฟสนี้สามารถนำไปสู่ความเข้าใจในพลศาสตร์ของอนุภาคทั้งหมด
- เพื่อศึกษาพฤติกรรมของอนุภาคในระบบนี้ในระดับเซมิคลาสสิก ซึ่งเกี่ยวข้องกับปัญหาที่สำคัญๆในสาขาการวิจัยควอนตัมเคออส เช่น ความสมเหตุสมผลของหลักความสอดคล้อง

1.2 วิธีการทดลอง

1.2.1 ระเบียบวิธีวิจัย

การศึกษาเพื่อสร้างโครงสร้างมหภาค (Global Structure) ของระนาบเฟสภาพตัดขวางของปองคาเรของระบบที่สนใจ ด้วยการใช้คอมพิวเตอร์คำนวณซึ่งผสมผสานวิธีที่ใช้ในระดับคลาสสิกที่เคยใช้มาแล้ว และวิธีตรง (Direct Function Definition) ซึ่งกำลังพัฒนาขึ้นมาใหม่

นอกจากนี้อาจจะมีการประยุกต์วิธีทางเซมิคลาสสิก เช่น ใช้การคำนวณของ Husimi, และการสร้างภาพตัดขวางของฟังก์ชันคลื่นตามวิธีการศึกษาของ Heller เป็นต้น อาจจะต้องนำวิธีของการใส่สัญลักษณ์ (Symbolic Coding) มาใช้ถ้าจำเป็น การวิเคราะห์ผลที่ได้จะบอกถึงพฤติกรรมรวมของอนุภาคในระบบนี้ทั้งในระดับเซมิคลาสสิกและในระดับคลาสสิก

1.2.2 ขอบเขตของการวิจัย

การศึกษาจะพยายามคำนวณหาโครงสร้างมหภาคของระนาบเฟสของอนุภาคที่อยู่ในสนามศักย์ $(Q-xy)^2$ จะสนใจระนาบเฟสที่ประกอบด้วยวงโคจรมีคาบที่มีคาบสั้นและคาบไม่ยาวมากนักเนื่องจากวง

โคจรเหล่านี้มีผลที่สำคัญอย่างมากกับการคำนวณระดับพลังงาน (Energy Eigenvalues) ซึ่งจะเป็นการวิจัยต่อไป

1.2.3 อุปกรณ์ที่ใช้ในการวิจัย

จัดหาในโครงการวิจัย

- ประกอบเครื่องคอมพิวเตอร์ Pentium 200 MHz 2 CPU, 256 MB RAM, 4 GB Harddisk พร้อมระบบ Backup ข้อมูล
- ระบบปฏิบัติการ Windows NT 4.0, Windows 2000, Linux Redhat

ได้มาภายหลังปี 2544

- Mathematica
- Parallel Computing Toolkit
- WAC16P4 Cluster

1.3 ผลการทดลอง

ได้ทำการวิเคราะห์และคำนวณ ทั้งพัฒนาวิธีการต่างๆ โดยสามารถศึกษารายละเอียดได้จากบทความวิจัยต่างๆ ที่แนบมาด้วย

1.4 บทวิจารณ์

1.4.1 แผนการดำเนินงานตลอดโครงการ และผล (output) ที่จะได้

1.4.2 แผนการดำเนินงาน

ระยะเวลาของโครงการ 3 ปี

แผนงานวิจัยและขั้นตอนวิธีการวิจัย

1. การสำรวจและเก็บรวบรวม ติดตาม บทความและผลงานที่ทำโดยนักวิจัยทั่วโลก
2. การติดตั้งคอมพิวเตอร์ ระบบปฏิบัติการ และซอฟต์แวร์ที่จำเป็น ขั้นตอนนี้เป็นขั้นตอนที่สำคัญมากที่สุดขั้นตอนหนึ่ง และเป็นขั้นตอนที่ใช้เวลามาก เนื่องจากการวิจัยเป็นการคำนวณที่ซับซ้อนและใช้เวลามาก
3. การเตรียมระบบพิมพ์ ซึ่งจะใช้โปรแกรมมาตรฐานสำหรับการตีพิมพ์ในวารสารนานาชาติ คือระบบ TeX LaTeX และ RevTeX ขณะนี้ได้ทำสำเร็จแล้วในระดับหนึ่ง แต่ยังคงมีการทดสอบต่อไปถึงการรวมภาพกราฟิก
4. โปรแกรมเครื่องคอมพิวเตอร์เพื่อให้ทำการคำนวณหาโครงสร้างของวงโคจรที่มีคาบทั้งแบบที่เสถียรและแบบที่ไม่เสถียรโดยใช้ระบบเฟสแบบภาพตัดของปองคาเร
5. พัฒนาริขตรง (Direct Function Definition) ให้ใช้งานได้ใน 2 มิติแล้วนำมาทดสอบคำนวณ

6. โปรแกรมเครื่องคอมพิวเตอร์เพื่อให้ทำการคำนวณหาโครงสร้างของวงโคจรที่มีคาบทั้งแบบที่เสถียรและแบบที่ไม่เสถียรโดยใช้วิธีทางเซมิคลาสสิก เช่นวิธีของ Husimi, วิธีของ Heller และวิธีของ Wigner
 7. โปรแกรมเครื่องคอมพิวเตอร์เพื่อให้ทำการคำนวณหาโครงสร้างของวงโคจรที่มีคาบทั้งแบบที่เสถียรและแบบที่ไม่เสถียรโดยใช้วิธีของการใส่สัญลักษณ์ (Symbolic Coding) ถ้าจำเป็น
 8. ทดสอบกับระบบที่ได้ศึกษามาแล้ว เช่น สนามคึกย์แบบ x^2+x^4 (Takahashi 1989)
 9. นำมาประยุกต์กับสนามคึกย์ $(Q-xy)^2$
 10. การวิเคราะห์ และแปลความหมายของผลที่ได้ทั้งหมด
 11. การนำเอาผลที่ได้ไปใช้ในการสัมมนา หรือแสดงโปสเตอร์ในการประชุมวิชาการระดับในประเทศ และการเตรียมบทความสำหรับตีพิมพ์ในวารสารภายในประเทศ
 12. การเตรียมบทความเพื่อไปตีพิมพ์ในวารสารนานาชาติ และการสัมมนาหรือแสดงโปสเตอร์ในการประชุมวิชาการองค์ความรู้ใหม่ที่ได้
- 1.4.3 ผลงานหรือกิจกรรมที่ทำได้อ้างอิง
- 1.4.3.1 ติดตั้งระบบคอมพิวเตอร์ Cluster
- ติดตั้งและทดสอบระบบปฏิบัติการ Linux
 - ติดตั้ง Windows Terminal Edition
 - ทำการแปลง Code มาอยู่ในภาษา Java เพื่อให้โปรแกรมเชื่อมโยงกันเป็นแบบ Cluster
 - พัฒนาวิธีตรง (Direct Function Definition Method – DirectD Method) สำหรับปัญหา $1/2(Q-xy)^2$ เพื่อคำนวณวงโคจรที่มีคาบ
 - พัฒนาเทคนิคเพื่อแก้ปัญหาการใช้เวลาในการคำนวณสำหรับกรณีที่ Magnetic Moment มีค่าน้อยๆ
 - ติดตั้งและทดสอบ Windows 2000 Advanced Server
 - ติดตั้งโปรแกรม Matlab และ Mathematica
 - สั่งซื้อ Parallel Computing Toolkit
 - Upgrade Mathematica จากรุ่น 3.0 ไปรุ่น 4.1 เพื่อเพิ่มความเร็วในการคำนวณเชิงตัวเลข
 - ได้ทำการแปลง Code มาอยู่ในภาษา Java เพื่อให้โปรแกรมเชื่อมโยงกันเป็นแบบ Cluster
 - ติดตั้งเครือข่าย Network ให้เครื่องคอมพิวเตอร์ติดต่อกันได้
 - เตรียมระบบสำหรับการทำงานแบบ Parallel
 - ติดตั้งระบบคลัสเตอร์ เพราะทางมหาวิทยาลัยพึงจะจัดหาครุภัณฑ์ให้
 - ทำการทดสอบ Parallel Computing Toolkit เพื่อนำมาใช้ในการคำนวณหาวงโคจรมีคาบโดยวิธี DirectD

- ได้งบประมาณจากทางมหาวิทยาลัยเพื่อจัดหาระบบคลัสเตอร์ประสิทธิภาพสูง ขณะนี้อยู่ในขั้นตอนการติดตั้งระบบคลัสเตอร์และโปรแกรมคำนวณขนาน
- กำลังติดต่อกับทาง Wolfram เพื่อจัดหาซอฟต์แวร์ *Mathematica* รวมทั้งพวก Toolkit ต่างๆ ซึ่งถ้าได้มาก็จะทำให้การคำนวณเป็นไปได้ง่ายและสะดวกขึ้น เพราะมี Parallel Toolkit ที่สามารถนำไปใช้งานได้ทันที

1.4.3.2 การวิจัย

- ทดสอบวิธีการ DirectD กับ Logistic Map และได้สร้างโครงสร้างที่ซ่อนอยู่ ซึ่งสามารถอธิบายกระบวนการ Bifurcation 2 แบบที่เกิดขึ้นใน Logistic Map ได้
- ทดสอบวิธีการ DirectD กับ Standard Map และค้นพบวิธีการสร้างภาพของ Homoclinic Tangle อย่างรวดเร็วได้ ทำให้เข้าใจกระบวนการ Mixing ที่เกิดใน Map นี้ได้เป็นอย่างดี อย่างไรก็ตามก็ตีการคำนวณตำแหน่ง Periodic Orbit โดยใช้วิธีนี้ต้องอาศัยการทำ Optimization และในปัญหานี้เทคนิคการทำ Optimization เป็นเรื่องที่ยากมากเนื่องจากโครงสร้าง Fractal ที่เกิดขึ้นโดยปกติทำให้กระบวนการวิธีค้นหาแบบปกติ เช่น Simplex Searching ล้มเหลว
- เขียนโปรแกรมสำหรับการคำนวณ Poincare Map สำหรับปัญหา $1/2(Q-xy)^2$ ซึ่ง Map นี้จะเป็นรากฐานที่สำคัญในการคำนวณหาวงโคจรมีคาบโดยวิธี DirectD โดยใช้ Mathematica ซึ่งยาก เพราะถ้าเขียนไม่ถูกวิธีจะทำงานช้ามาก
- พัฒนาเทคนิคเพื่อแก้ปัญหาการใช้เวลานานในการคำนวณสำหรับกรณีที่มี Magnetic Moment มีค่าน้อยๆ โดยได้ตีพิมพ์เนื้อหาในส่วนนี้แล้วใน Journal of Plasma Physics
- พัฒนาวิธีการ Multiple Scale Perturbation Analysis เพื่อสร้าง Map สำหรับทุกค่า Q อันจะนำไปใช้ในการศึกษาพฤติกรรม Quantum ของระบบต่อไป

1.4.3.3 สํารวจบทความและรวบรวมผลงานวิจัยที่เกี่ยวข้อง

- บอกรับเป็นสมาชิกวารสาร Chaos, Phys. Rev. E, Nonlinearity, American Journal of Physics และ Computing in Science and Engineering โดยสมัครเป็นสมาชิกสมาคม IEEE, APS, AIP, IOP
- จัดซื้อหนังสือเกี่ยวกับการดูแลและโครงสร้างของ Windows 2000 และภาษา Java เพื่อที่จะได้ใช้งานระบบปฏิบัติการนี้ให้มีประสิทธิภาพสูงสุด
- ได้ทำสำเนาเอกสารที่เกี่ยวข้องกับงานวิจัย โดยเฉพาะด้าน Cluster และ Parallel Computing

1.4.3.4 การเขียนบทความเพื่อเผยแพร่ผลงานวิจัย

- เขียนบทความเพื่อตีเผยแพร่ในการประชุมต่างๆ เช่น Annual National Symposium on Computational Science and Engineering ครั้งที่ 1-6, การประชุม NCSEC 2001, การประชุมวทท.และที่สกว.จัดขึ้น โดยสามารถศึกษารายละเอียดได้จากบทความวิจัยที่แนบมาด้วย

1.4.3.5 ทบทวนงานวิจัยของนักวิจัยอื่นที่ได้ตีพิมพ์

- Computing in Science and Engineering March/April 2001 ตีพิมพ์บทความเปรียบเทียบการทำงานของ Java และ ภาษาอื่นๆ เช่น C, Fortran พบว่าในบาง Platform Java มีการทำงานที่เร็วกว่า โดยเฉพาะ Platform Intel, เป็นการยืนยันว่าการที่ทางโครงการตัดสินใจเปลี่ยนไปใช้ Java เป็นการตัดสินใจที่ถูกต้อง นอกจากนี้ Mathematica 4.1 มี Jlink ที่ออกแบบมาเพื่อเชื่อมโยง Java เข้ากับ Mathematica
- งานวิจัยที่ได้ตีพิมพ์ไปแล้วมักจะทำการคำนวณโดยใช้เครื่องคอมพิวเตอร์ระดับ Supercomputer เช่นในการคำนวณระดับพลังงานของ Billiard ที่ต้องใช้การแก้สมการหาค่า Eigenvalue ของปัญหา Finite Elements หรือการคำนวณโดยวิธี Monodromy Matrix แต่ไม่มีงานใดที่ซ้ำหรือใกล้เคียงกับงานที่ทำอยู่มากนัก

1.4.4 ผลที่ได้เมื่อจบโครงการ

- วิธีการคำนวณโครงสร้างมหภาคที่สามารถนำไปใช้กับระบบไม่เชิงเส้นอื่นๆ โดยได้ทดสอบกับ Logistic Map และ Standard Map
- ได้โครงสร้างมหภาคของระนาบเฟสของระบบที่ศึกษาแต่ยังไม่ครอบคลุมทั้งเส้น
- ได้ฐานข้อมูลของงานวิจัยในสาขานี้ ซึ่งจะก่อให้เกิดโมเมนตัมให้งานวิจัยเกี่ยวกับ เคออส และควอนตัมเคออส กระทำได้ต่อไป
- ได้ฐานข้อมูลของวิธีการคำนวณเชิงตัวเลขเพื่อหาวงโคจรที่มีคาบที่ประกอบขึ้นเป็นโครงสร้างแบบมหภาคของระนาบเฟส
- การสร้าง Poincare Section โดยระบบคลัสเตอร์ที่ทำงานแบบขนานโดยใช้วิธีการโปรแกรมแบบฟังก์ชัน

1.4.5 ความเห็นและข้อเสนอแนะ

ปัญหาอุปสรรคสำคัญของโครงการนี้คือ

- ความไม่พร้อมของบุคลากรในประเทศที่เชี่ยวชาญในการติดตั้งระบบคอมพิวเตอร์ ทำให้นักวิจัยต้องลงไปดูในรายละเอียดเองทุกเรื่อง สิ้นเปลืองพลังงานและเวลามาก แทนที่จะได้พุ่งเป้าไปทำงานวิจัย
- ความไม่พร้อมของเครื่องคอมพิวเตอร์และอุปกรณ์ต่อพ่วงต่างๆ เพราะเราไม่สามารถจัดหาได้ทุกอย่างในระยะเวลาที่กำหนด ระบบคลัสเตอร์เป็นระบบซับซ้อนที่ต้องการงานวิจัยดูแลอีกโครงการหนึ่ง
- ความไม่พร้อมของซอฟต์แวร์และระบบปฏิบัติการ ในที่นี้ระบบปฏิบัติการที่ต้องใช้ในงานวิจัยจะต้องมีเสถียรภาพและมีประสิทธิภาพพอที่ต้องทำงานเป็นเวลานานๆ ซึ่งไม่ใช่เรื่องง่ายที่จะติดตั้งระบบปฏิบัติการให้สมบูรณ์ได้ในระดับนี้ และซอฟต์แวร์ที่ต้องใช้ก็ขึ้นกับระบบปฏิบัติการมาก นอกจากนี้ โดยมากซอฟต์แวร์ที่ใช้กันในงานวิจัยฟิสิกส์เชิงคำนวณมักมี

ราคาแพงมากเช่น IDL สำหรับ Windows NT ปัจจุบันมีราคาสูงมากถึงประมาณ 60,000-70,000 บาท

- ความจำเป็นที่ต้องเรียนรู้ระบบคอมพิวเตอร์ใหม่สำหรับผู้ทำวิจัย เนื่องจากระบบคอมพิวเตอร์ที่จัดเตรียมได้ไม่มีทางเหมือนกับระบบคอมพิวเตอร์ที่ผู้ทำวิจัยเคยใช้ในการศึกษาที่ประเทศอังกฤษซึ่งมีความพร้อมสูงมาก ความพร้อมนี้เกิดขึ้นได้จากต้นทุนที่มหาวิทยาลัย Warwick ได้จ่ายไปอย่างมากมายและเป็นเวลานานในการจัดสรรซึ่ง ระบบฮาร์ดแวร์ ระบบซอฟต์แวร์ และ ที่สำคัญที่สุด บุคลากรที่ดูแลระบบการทำงานทั้งหมดของระบบ
- เนื่องจากการเปลี่ยนแปลงสถานที่ทำงานจาก ภาควิชาฟิสิกส์ คณะวิทยาศาสตร์ มหาวิทยาลัยมหิดล มาเป็น สำนักวิชาวิทยาศาสตร์ มหาวิทยาลัยวลัยลักษณ์ ทำให้แผนการศึกษาวิจัยและพัฒนาที่จะใช้เครื่องคอมพิวเตอร์ที่ห้องปฏิบัติการคอมพิวเตอร์เพื่อการวิจัย ที่ภาควิชาฟิสิกส์ และคณะวิทยาศาสตร์ มหาวิทยาลัยมหิดล ต้องเปลี่ยนแปลงไป โดยขณะนี้ผู้วิจัยสามารถใช้เครื่องคอมพิวเตอร์สมรรถนะสูงของหลักสูตรบัณฑิตศึกษาวิทยาศาสตร์เชิงคำนวณ มหาวิทยาลัยวลัยลักษณ์ที่เพิ่งได้รับมา กำลังอยู่ในการติดตั้งให้เป็นระบบคลัสเตอร์อยู่ ซึ่งถ้าเสร็จสิ้นก็จะทดลองคำนวณเพิ่มเติมต่อไป
- เครื่องคอมพิวเตอร์คอมพิวเตอร์หลักที่ใช้งาน เป็นเครื่องที่ประกอบขึ้นเองจากงบประมาณด้านวัสดุ มีความเร็วไม่มากและหน่วยความจำไม่สูงนัก แต่ก็พอใช้งานได้ โปรแกรมหรือซอฟต์แวร์ที่ใช้คือ Mathematica ซึ่งเป็นรุ่นที่มหาวิทยาลัยซื้อไว้ มีการพัฒนาระบบเก็บและสำรองข้อมูล นอกจากนี้มีการใช้ซอฟต์แวร์เถื่อนบางตัวในการทำงาน แต่คาดว่า ไม่มีปัญหาลิขสิทธิ์เพราะเป็นรุ่นเก่าไปหมดแล้ว
- ที่มหาวิทยาลัยวลัยลักษณ์มีปัญหาการจ่ายกระแสไฟฟ้า ทำให้ไฟฟ้าดับเป็นเวลานาน และบ่อยมากที่จะมีไฟฟ้าดับเป็นเวลา 1 วัน ทำให้ Program ที่ทำงานอยู่ต้องหยุด และเริ่มใหม่ ไฟฟ้าดับนานมากทำให้ระบบ UPS ที่มีอยู่รองรับไม่ไหว อาจจะต้องหา UPS ที่มีกำลังสูงขึ้น แต่ราคาแพงมาก นอกจากนี้ยังมีปัญหาไฟฟ้ากระชากทำให้อุปกรณ์คอมพิวเตอร์เสียหาย
- โครงการวิจัยหลังปริญญาเอก ของสกว. นับเป็นโครงการนำร่องที่ดีมาก อย่างไรก็ตาม เป็นที่น่าเสียดายว่า ปรัชญาของทุนที่ใช้ชื่อ postdoc นั้น ไม่เป็นไปตามชื่อ postdoc ในโครงการนี้ สาเหตุหลักคือ นักวิจัยไม่สามารถทุ่มเทเวลาทำงานวิจัยนี้ 100 % สำหรับในต่างประเทศทุน postdoc เป็นลักษณะของการจ้างทำวิจัยของนักวิจัย ฉะนั้นนักวิจัย postdoc จะมีเวลาทำงานนี้ได้เต็มเวลา ในขณะที่ นักวิจัยในโครงการนี้ของสกว.จะต้องทำการสอน งานบริการ วิชาการ งานเป็นที่ปรึกษาและการวิจัยอื่นๆ ที่มหาวิทยาลัย assign ให้ การที่โครงการจะคาดหวังว่า นักวิจัยจะทุ่มเทเวลา 100% ดังเช่น postdoc จริงๆ นั้นเป็นไปได้เลย ทางออกหนึ่งที่เห็นจากตัวอย่างๆ อื่นที่นักวิจัยในโครงการนี้ทำกันในภายหลังก็คือ การจ้างนักศึกษาหรือผู้ช่วยวิจัยในการช่วยงาน สิ่งนี้จะช่วยให้โอกาสที่โครงการนี้จะสำเร็จสูงมาก เพราะถ้าไม่จ้างหรือไม่มีผู้ช่วยวิจัยเลย เมื่อนักวิจัยหยุดทำงานวิจัย งานก็จะหยุด 100% ทำ

ให้ไม่เกิดโมเมนต์ที่ต่อเนื่อง และในโครงการนี้ไม่ได้ตั้งบไว้เพื่อจ้างเนื่องจากเท่าที่ทราบมาในช่วงเขียนข้อเสนอโครงการว่าโครงการนี้ไม่สนับสนุนให้มีการจ้างงานผู้ช่วยวิจัย ดังนั้นจึงไม่ได้ตั้งบหรือวางแผนการทำงานไว้ในส่วนนี้

1.5 เอกสารอ้างอิง

1.5.1 ผลงานวิจัยที่เกี่ยวข้อง (literature review) และเอกสารอ้างอิง

ผลงานวิจัยที่เกี่ยวข้อง

ดังที่กล่าวมาแล้ว ในการศึกษาหรือทำความเข้าใจระบบพลวัตแบบไม่เชิงเส้นนี้ต้องอาศัยความรู้เกี่ยวกับโครงสร้างของระนาบเฟส ที่ประกอบด้วยวงโคจรที่มีคาบ ทั้งเสถียรและไม่เสถียร รวมไปถึงวงโคจรที่มีคาบเทียม (Quasiperiodicity) โดยปกติการหาวงโคจรเหล่านี้ทำได้โดยการหาจุดตัดแกนนอนของฟังก์ชันที่นิยามจากหลาย ๆ ตัวแปรที่อธิบายระนาบเฟส วิธีเชิงคำนวณที่มักใช้คือวิธี Newton Raphson ซึ่งต้องการการเดาจุดเริ่มต้นที่ดี (Good initial guess) ในการที่จะประกันการค้นพบวงโคจรดังกล่าว (Jaroensutasinee and Rowlands 1993 และเอกสารอ้างอิงอื่นที่ระบุไว้) วิธีการนี้เปรียบได้กับงมเข็มในมหาสมุทร ในกระบวนการนี้เราเพียงแค่ว่าโครงสร้างของระนาบเฟสเพียงบริเวณเล็กๆ เท่านั้น โครงสร้างของระนาบเฟสมหภาค (Global Structure) เป็นสิ่งที่ต้องการอย่างมากในการนำไปใช้ นี่เป็นวัตถุประสงค์หนึ่งของโครงการวิจัยนี้

เนื่องจากพฤติกรรมแบบเคออสนี้เกิดจากความไม่เชิงเส้นของระบบเอง เราจะต้องได้พฤติกรรมนี้ถ้าเราศึกษาระบบเดียวกันนี้ในระดับควอนตัม แต่เราจะพบความขัดกันอย่างทันที เนื่องด้วยความเป็นเชิงเส้นของทฤษฎีกลศาสตร์ควอนตัมประกันการไม่เกิดขึ้นของการตีจากแบบเอกโปเนนเชียล (Exponential deviation) หรือความไวต่อสถานะเริ่มต้น (Sensitivity to initial conditions) ได้มีการทบทวนงานวิจัยถึงความขัดแย้งกันอย่างกว้างขวาง แต่ก็ยังหาข้อสรุปที่แน่ชัดไม่ได้ (Ford and Mantica 1992) ผลการวิจัยในส่วนนี้จะกระทบโดยตรงกับหลักความสอดคล้อง (Correspondence Principle) ซึ่งเคยเชื่อกันว่าเป็นหลักเดียวที่เชื่อมโลกมหภาค (Classical World) และโลกระดับควอนตัม (Quantum World) เข้าด้วยกัน ความเข้าใจในการเปลี่ยนแปลงจากของระบบจากระดับคลาสสิกไปเป็นระดับควอนตัมเป็นสิ่งสำคัญอย่างมากต่อพัฒนาการของวิชาฟิสิกส์

สนามศักย์ $(Q-xy)^2$ เป็นสนามศักย์ที่มีรูปไม่ซับซ้อน Q เป็นค่าคงที่ที่มากกว่า 0 และ x กับ y ก็คือโคออร์ดิเนตในระบบ Rectangular ได้มีการศึกษาระบบนี้อย่างมากมายตั้งแต่ปี ค.ศ. 1970 จนถึงปัจจุบัน เนื่องจากมันสามารถช่วยให้เข้าใจถึงปัญหาของการสร้างเตาปฏิกรณ์ปรมาณูแบบฟิวชันแบบใช้สนามแม่เหล็กกักพลาสมา ยุโรป และสหรัฐอเมริกา รวมทั้งญี่ปุ่น ได้ทำการวิจัยเรื่องนี้มาเป็นเวลากว่า 20 ปี ใช้งบประมาณมหาศาลเพื่อสร้างเตาปฏิกรณ์นี้แต่ก็ยังคงไม่ประสบความสำเร็จ เตาปฏิกรณ์ยังไม่สามารถให้พลังงานออกมามากกว่าพลังงานที่ใส่เข้าไป ปัจจุบันเราทราบว่าสาเหตุหนึ่งของความล้มเหลวก็คือเกิดปรากฏการณ์เคออสที่หมายถึงเกิดสนามแม่เหล็กที่ไม่เสถียรในเตาปฏิกรณ์ปรมาณู และเกิดการรั่วของอนุภาคมีประจุที่กระจายตัวออกไปทั่วระนาบเฟสก่อให้เกิดการสูญเสียพลาสมาที่กักไว้ (Jaroensutasinee 1994)

นอกจากสนามศักย์ $(Q-xy)^2$ จะมีการประยุกต์ในการศึกษาเตาปฏิกรณ์ปรมาณูแบบฟิวชันแบบใช้สนามแม่เหล็กกักพลาสมาแล้ว เรายังสามารถพบสนามศักย์นี้ในระบบอื่นๆอีกมากมาย อีกทั้งยังมีการศึกษาสนามศักย์นี้ในกรณีพิเศษ $(Q=0) \ x^2 y^2$ ในแง่เซมิคลาสสิกอีกด้วย (Dahlqvist and Russberg 1991)

การศึกษาในระดับคลาสสิกได้กระทำอย่างละเอียดในวิทยานิพนธ์ของผู้วิจัย แต่ด้วยข้อจำกัดทางเครื่องมือทั้งในแง่ทฤษฎีที่มีอยู่ และความสามารถของเครื่องคอมพิวเตอร์ที่มีในขณะนั้น ทำให้โครงสร้างระนาบเฟสมหภาคของระบบนี้ยังเป็นปริศนา ปัจจุบันผู้เสนอโครงการมีความมั่นใจว่าความก้าวหน้าของการวิจัยควอนตัมเคออสก่อให้เกิดเครื่องมือใหม่ๆ ที่สามารถนำมาพัฒนา ปรับเปลี่ยนและประยุกต์ใช้กับระบบนี้อันก่อให้เกิดวิธีการใหม่ในการสร้างภาพระนาบเฟสมหภาคของระบบที่ต้องการได้

1.5.2 เอกสารอ้างอิงที่สำคัญ

- Allegrini P, Bonci L, Grigolini P, Mannella R, Roncaglia R and Vitali D 1995 Phys. Rev. Lett. 74 (8) p 1484
- Berry M V 1987 Proc. R. Soc. Lond. A 412 pp 183-198
- Biswas D, Azam M, Lawande QV and Lawande SV 1991 J. Phys. A 25 pp L297-L301.
- Casati G 1995 Physica D 86 p 220
- Crespi B, Perez G and Chang S J 1993 Phys. Rev. E 47 (2) pp 986-991
- Dahlqvist P and Russberg G 1991 J. Phys. A 24 pp 4763-4778
- Eckhardt B 1993 Phys. Lett. A 172 pp 411-415
- Ford J and Mantica G 1992 Am.J.Phys. 60 p 1086
- Fromhold T M, Eaves L, Sheard F W, Foster T J, Leadbeater M L and Main P C 1994 Physica B 201 pp 367-373
- Grobged D, Pollak E and Zakrzewski J 1992 Physica D 56 pp 368-380
- Gutzwiller MC 1990 Chaos in Classical and Quantum Mechanics (Berlin: Springer)
- Heller EJ 1991 Les Houches, Session LII Chaos and quantum physics (Giannoni MJ, Voros A and Zinn-Justin J eds., 1989) pp 547-663.
- Jaroensutasinee K and Rowlands G 1993 Physics Computing'92 (Singapore: World Scientific) p 355
- Jaroensutasinee K and Rowlands G 1994 J. Phys. A 27 p 1163
- Jaroensutasinee K 1994 Thesis (Submitted to the University of Warwick)
- Jensen R V 1992 Nature 355 (23 January 1992)

- Kim J W and Su W P 1994 Phys. Rev. A 50 (2 part A) pp 1019-1023
- Leboeuf P, Kurchan J, Feingold M and Arovas D P 1990 Phys. Rev. Lett. 65 (25) pp 3076-3079
- Nakamura K 1993 Quantum Chaos (Cambridge: CUP)
- Reichl L E 1992 The transition to chaos (New York: Springer)
- Sieber M and Steiner F 1991 Phys. Rev. Lett. 67 (15) pp 1941-1944
- Takahashi K 1989 Prog. Theo. Phys. Suppl. 98 (Saito N and Aizawa Y, Eds.) p 109
- Zelevinsky V G 1994 Nuclear Phys. A 570 (1-2) pp C 411-C 420

2 ผลงานจากโครงการวิจัยที่ได้รับทุนจาก สกว. และที่เกี่ยวข้อง

1. K. Jaroensutasinee and G. Rowlands 'A New Method for Cycle Calculation' Proceedings of The First National Computational Science and Engineering Symposium. Bangkok, Thailand. 20th -21st March 1997, pp. 32-35.
2. K. Jaroensutasinee and G. Rowlands 'Prime Cycles in the Logistic Map' Proceedings of The Second National Computational Science and Engineering Symposium. Bangkok, Thailand. 26th -27th March 1998, pp.174-181.
3. K. Jaroensutasinee and G. Rowlands 'Homoclinic Tangle Visualization' Proceedings of The Third National Computational Science and Engineering Symposium. Bangkok, Thailand. 24th -26th March 1999, pp. 75-83.
4. K. Jaroensutasinee and G. Rowlands 2000. 'Charged-particle orbits near a magnetic null point' Journal of Plasma Physics 64 (3) 255-262.
5. K. Jaroensutasinee and G. Rowlands 'Visualization of Particle Motion in $(Q-xy)^2$ Potential' Proceedings of The Fifth National Computational Science and Engineering Symposium. Bangkok, Thailand. 19th -20th June 2001, pp. 91-97.
6. K. Jaroensutasinee and G. Rowlands 'Q dependence of global particle behavior in $(Q-xy)^2$ potential' 27th Congress on Science and Technology of Thailand. Bangkok, Thailand. 16th -18th Oct, 2001, pp. 381.
7. K. Jaroensutasinee, Proceedings of the 5th National Computer Science and Engineering Conference, Nov.7th -9th, 2001, Chiang Mai, Thailand, p 429.
8. K. Jaroensutasinee and G. Rowlands, to be appeared in the proceedings of the 6th Annual National Symposium on Computational Science and Engineering, Apr.3rd -5th, 2002, Walailak University, Nakhon Si Thammarat, Thailand.

9. K. Jaroensutasinee and G. Rowlands, to be appeared in the proceedings of International Conference on Computational Mathematics and Modeling, May 22-24, 2002, Bangkok, Thailand.
10. K. Jaroensutasinee. Parallel Construction of Poincaré Surface of Section Method on WAC16P4 Cluster, to be appeared in the proceedings of the 6th National Computer Science and Engineering Conference, Nov., 2002, Thailand.

3 รายงานการเงินพร้อมสำเนาสมุดบัญชีเงินฝาก

ดังปรากฏในเอกสารแนบที่ส่งมาด้วย

A New Method for Cycle Calculation

Dr. Krisanadej Jaroensutasinee

Department of Physics, Faculty of Science,

Salaya Centre, Mahidol University, NakronPratom 73170 Thailand

email: sckjr@mahidol.ac.th WWW: <http://www.sc.mahidol.ac.th>

Abstract

Cycles or periodic solutions play a very important role in the modelling of nature using differential or difference equations. These special orbits have the nature to exist without any other indicators, hence making them extremely hard to search for, especially the unstable ones. We have developed a novel method to compute both stable and unstable cycles numerically. This new method is equipped with a special searching technique that is similar to simulated annealing. This special technique allows us to probe as many cycles with a given period as the computational limits impose. For small cycles, we expect to discover all of them. The method has been applied to the well-known logistic map and the results are shown here.

1 Periodic Orbits and Their Importance

Knowledge of "Periodic orbits" (POs) in a chaotic system is the most important key to understanding chaos in such a system. Many researchers (e.g. Wintgen D 1988 or Dahlqvist and Russberg 1991) are now using periodic orbits to compute semiclassical eigenvalues of classically chaotic systems. Other researchers use POs to determine the fractal dimension of a complicated strange attractor (e.g. Parker and Chua 1989). The diffusion rate in the standard map is also found to connect with POs (Eckhardt 1993).

It is known that although a system under consideration is chaotic, its POs are regular and attainable. POs can be classified into 2 groups, stable and unstable. It is easier to calculate stable POs numerically, and sometimes analytically, than to calculate unstable POs. Nevertheless, stable POs are much less used in chaotic systems since one definition of chaos is that almost no stable POs exist.

To calculate unstable POs is not a simple task. A number of methods for computing these UPOs have been invented but which one is the generally best method is still arguable. Moreover, the methods are normally invented to study some specific systems. We present here a novel method that can be applied generally to maps and the Poincaré surface of section. The method can be used to compute both stable and unstable POs. We have applied it to study the logistic map which is a 1-d map. The study gives us many interesting results which are to be presented later on in this paper.

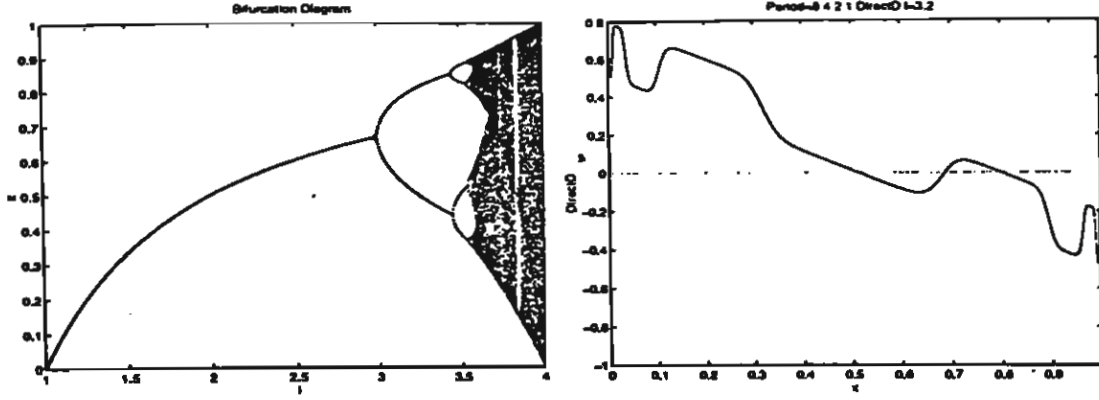


Figure 1: (a) Bifurcation Diagram of Logistic Map. (b) An example of D_{Def} plot for the Logistic map.

2 The new method

We define a function D_{Def} , which we call the DirectD function, as:

$$D_{Def}(\lambda, x, n) = F_\lambda^n(x) - x. \quad (1)$$

where n is the period of POs, F represents the mapping function, λ is the control parameter, x is the state variable, and $F_\lambda^n(x) = F_\lambda(\dots n \text{ times } \dots(x))$.

Obviously, POs result from the condition that simply requires the function to be zero. This makes it easy to visualise the problem which is one of the two convenient points of this DirectD approach. The other point is that stable or unstable POs appear to be the same in this aspect. So, the task is now just to search for zeros of the DirectD function.

One can assume the task is simple because it is possible to employ the zeros finding algorithm such as the Gauss-Newton method that converges to the solution very quickly but needs a very good initial guess. However, technical problems can prevent one from obtaining a meaningful result easily. Technical problems are, for example, traps caused by the local minima and numerical overflow due to high values of gradient around UPOs.

There are a number of approaches that can be applied to increase the degree of convergence of the Gauss-Newton method such as the *damped*-Newton method (Dennis and Schnabel 1983) and some authors use the recurrence theorem to find good initial guesses for the routine. In general it is found that computing costs can be very high, not only due to these technical problems, but also as a result of a dense set of POs in one particular region of interest.

We resort to the method of simulated annealing to solve the local minimum problem. By slowly cooling the system down, we assume that the local minima can be avoided. Then, we take the points as initial guesses for further refinement in which we employ the Gauss-Newton algorithm. Roughly speaking, we choose points at random over the domain of interest, calculate the total energy which is defined by the sum of the square of D_{Def} and let them cool down by rearranging the configuration so that the total energy reduces slowly. Then we take all the points as initial guesses for Gauss-Newton methods in which the refinements take place. The combination of these two numerical methods and the DirectD definition to calculate POs is the new method we propose here. We call it the

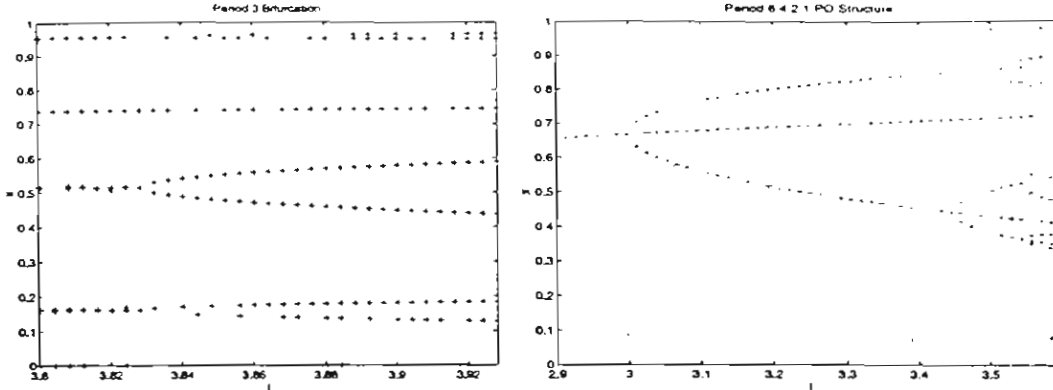


Figure 2: (a) The hidden bifurcation of period-3 cycles. (b) Full Bifurcation Diagram for Period 8 4 2 1.

DirectD method.

3 Application to Logistic Map

The logistic map is a 1-dimensional map that has a rich structure due to the presence of nonlinearity. It also has a bifurcation route (Fig. 1a) to chaos. The most important thing is that it is very simple, and is hence widely used to demonstrate period doubling, n -period oscillation, universality, and chaos. One of the mapping equations has the form:

$$x_{n+1} = \lambda x_n(1 - x_n) \quad (2)$$

where x is the state variable ranging from 0 to 1, λ is the control parameter taking the domain from 1 to 4, and n is the index of iteration.

For the logistic map, we have

$$D_{Def}(\lambda, x, n) = F_\lambda^n(x) - x, \quad (3)$$

where $F_\lambda(x) = \lambda x(1 - x)$ and n is the period of POs of interest. Fig. 1b gives an idea of what the DirectD function looks like for $\lambda = 3.2$ and $n = 8$. Note that when we define $n = 8$, the function effectively includes the periods 4, 2, and 1 as well. We have used the method to calculate period 3 and period 1 POs for λ s around the well-known Period-3 structure, and found that this cycle bifurcated further into stable period-3 and unstable period-3 orbits. We don't usually see this additional branch in the bifurcation diagram (Fig. 1a). The reason is that one of the branches that corresponds to the just-bifurcated cycle is unstable, and therefore missing from the bifurcation diagram. The result is shown graphically in Fig. 2a. The full bifurcation diagram (including unstable cycles - Fig. 2b) can also be obtained by the method.

4 Conclusions

For the logistic map, we perform a simulation by choosing a uniform distribution of 50000 points over the domain 0 to 1 and then iterating the map on each point

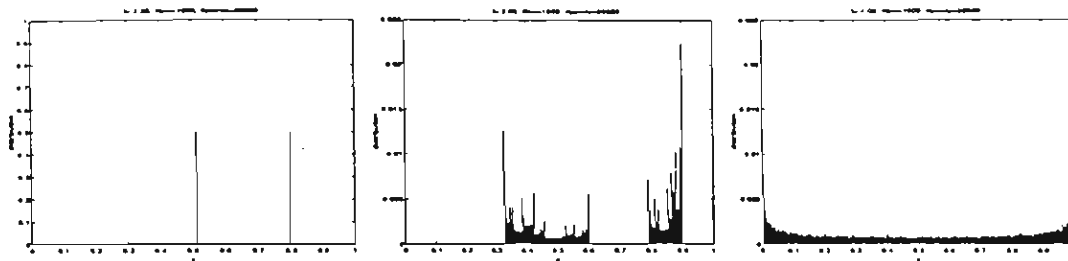


Figure 3: (a) Distribution function obtained from simulation of 50000 points at $\lambda = 3.25$ where the cycle-2 dominates. (b) at $\lambda = 3.6$ where chaos is present. (c) at $\lambda = 4.0$ where chaos completely dominates and no stable orbits are present.

1000 times. A histogram with 1000 bins of these points is then obtained. At $\lambda = 2.5$ it is found that the histogram is dominated by the only stable cycle. This domination of POs in the histogram can be seen more clearly in Fig. 3a at $\lambda = 3.25$ in which the period-two cycles and the single unstable period-one cycle have a great influence. The situation when we have the mixing of chaos and stable cycles is shown in Fig. 3b. Notice the shallow curve in the middle of two sharp peaks. Fig. 3c gives the picture when all stable cycles disappear.

Analytical equations for the determination of properties of POs after knowing their positions will be the subject of future investigation. Analytical formulas that link the first derivative to the Lyapunov number are also waiting to be found. One can also speed up the method by employing a different approach in the simulated annealing part. Generalisation of this method to apply to two-dimensional problems such as the standard map and the 2D Poincaré surface of section for the flow of 2D Hamiltonian system is another very interesting avenue for further work.

Acknowledgements

This work is supported by the Faculty of Science, Mahidol University and the Development and Promotion of Science and Technology Talents project (DPST).

References

- Dahlqvist P and Russberg G 1991 *J.Phys.* A **24** pp 4763-4778.
- Dennis J E and Schnabel R 1983 *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Prentice-Hall, Eaglewood Cliffs, NJ).
- Eckhardt B 1993 *Phys.Lett.* A **172** pp 411-415.
- Parker TS and Chua LO 1989 *Practical Numerical Algorithms for Chaotic Systems* (Springer-Verlag, New York).
- Wintgen D 1988 *Phys.Rev.Lett.* **61** (16) pp 1803-1806.

Prime Cycles in the Logistic Map

K. Jaroensutasinee¹ and G. Rowlands²

¹Dept. Of Physics, Faculty of Science, Mahidol University, Rama VI Road,
Bangkok 10400 Thailand

and

²Dept. Of Physics, University of Warwick, Coventry, CV4 7AL England

Abstract

Generating the Bifurcation diagrams of the Logistic map is an easy computational task. The mapping equation is very simple and the algorithm requires only the power of a desktop computer. Nevertheless, the diagram is found to be very complicated. It is fractal and contains a number of interesting structures apart from the well-known bifurcation route to chaos. One example of such structures is the birth of the period 3 cycle which appears immediately after chaos. For this reason this structure can be clearly noticed in the diagram. There are other structures of other periods too. But these structures are not obvious - they are hidden in the diagram. By employing the DirectD method, these structures can be calculated easily and so they are revealed. The prime cycles are chosen for detailed calculation. The structures found by the method are very interesting. For no obvious reason, they appear in the diagram, bifurcate, and disappear in the bifurcation diagram. The strength of the method is that specific periods can be chosen and it can also detect the unstable cycles. These unstable cycles are needed to connect up the structures.

1. Introduction

1.1 The Logistic Map

It is now widely known that simple mathematical models can possess very complicated behavior. May [8] was one of the pioneers to point this out to the scientific society. The difference equation of the form:

$$x_{n+1} = \lambda x_n (1 - x_n) \quad (1)$$

was the first to be used to illustrate this point where x is the state variable having the domain from 0 to 1, λ is the control parameter taking the domain from 1 to 4, and n is the index of iteration. This equation is known as "the Logistic equation". This equation can be used to model various situations ranging from physics to biology. It also illustrates many of the phenomena found in realistic models of physics [12].

Mathematically speaking, the Logistic map is a 1-dimensional map that has the rich feature of nonlinearity. It has a bifurcation route to chaos. The most important thing is because of its simplicity, it is widely used to demonstrate period doubling, n -period oscillation, universality, and chaos. It contains many bifurcation sequences and has been explored in detail since its discovery [5,8].

1.2 Importance of cycles

It is known that although a system under consideration is chaotic, its POs are regular and attainable. Therefore, knowledge of "Periodic orbits" (POs) in a chaotic system is the most important key to understanding chaos in such a system. POs can be classified into 2 groups, stable and unstable. It is easier to calculate stable POs numerically, and sometimes analytically, than to calculate unstable POs. However, stable POs are much less used in chaotic systems since one definition of chaos is that no stable POs exist.

Many researchers (e.g. [1,13]) are now using periodic orbits to compute semiclassical eigenvalues of classically chaotic systems. Other researchers use POs to determine the fractal dimension of a complicated strange attractor [10]. Diffusion rate in the standard map is also found to connect with POs [4]. Cycles are the key structure of bifurcation diagram (fig.1 and fig.4). For the Logistic map, special consideration was given to cycles. Very fine details of cycles of many period were given in May [8].

1.3 Prime cycles

Prime numbers have always been special. It is widely noted that the period 3 emerges out of chaos in the bifurcation diagram. Other prime period cycles such as period 5 are noted to exist but buried in the chaotic bands of the diagram, therefore it is hard to observe those cycles. In our work, we have revealed these structures and we discovered their convergence, by numerical means, to the value of the control parameter 3.6786. In the following, the first hundred prime numbers are presented.

2	3	5	7	11	13	17	19	23	29
31	37	41	43	47	53	59	61	67	71
73	79	83	89	97	101	103	107	109	113
127	131	137	139	149	151	157	163	167	173
179	181	191	193	197	199	211	223	227	229
233	239	241	251	257	263	269	271	277	281
283	293	307	311	313	317	331	337	347	349
353	359	367	373	379	383	389	397	401	409
419	421	431	433	439	443	449	457	461	463
467	479	487	491	499	503	509	521	523	541

Table 1 The first hundred prime numbers.

2. Cycle calculation

2.1 Survey of the methods

Up to now, there are a number of numerical techniques for calculating cycles or periodic solutions for a system of ordinary differential equations and also for a map. Each technique has its own advantages. The simplest method is to let the system execute until it reaches the cycle. This method is called the "Brute-force method"[10] and just like many other techniques, it has some advantages despite its simplicity. This method is easy to code in a programming language and it is relatively general because it can locate many different types of cycles (equilibrium point, cycles of period one or more). For the Logistic Map, cycles are the results of the iteration of the map after having the transient removed by ignoring the first

few hundred iterations. Nevertheless, the method has many problems. It obviously cannot be used for conservative Hamiltonian systems since for such systems the state of the systems will never reach an asymptotic state and can go quasiperiodic. Next, the method is slow for lightly damped systems. Furthermore, in many cases it is not possible to say that the system has reached an asymptotic state. Most importantly, the method can only locate stable cycles.

More sophisticated methods turn the problem of locating their cycles into a boundary value problem (BVP). This method is natural and is extensively utilised in bifurcation studies of dynamical systems (see for example [2,6]). The condition of a cycle for the BVP is:

$$\mathbf{x}(T) = \mathbf{x}(0) \quad (2)$$

There are two standard methods for solving two point boundary value problems[12]: the shooting method and the relaxation method. They both, however, are unsuitable for chaotic systems in general since the orbits are bound to be complicated, highly oscillating, which requires more time and more grid points.

2.2 DirectD method and Prime cycles

Calculation of unstable POs is more difficult. A method called DirectD method was developed [7]. This method can be applied generally to maps and Poincare surfaces of section. In the present work, this method was used to compute both stable and unstable POs of the Logistic map. For the Logistic map, a function D_{def} , which we call the DirectD function, can be defined by

$$D_{def}(\lambda, x, n) = F_{\lambda}^n(x) - x \quad (3)$$

where n is the period of POs, F represents the mapping function, λ is the control parameter and x is the state variable, and $F_{\lambda}^n(x) = F_{\lambda}(\dots n \text{ times } \dots(x))$. For the Logistic map $F_{\lambda}(x) = \lambda x(1 - x)$. POs result from the condition that simply requires the function to be zero. Fig.2 gives an idea of what the DirectD function looks like for $\lambda = 3.2$ and $n = 8$.

2.3 What are enclosed in Period n DirectD calculation?

In order to calculate unstable cycles, there is a price to pay. The DirectD function has one major drawback. It cannot separate period 4 from period 8, nor cannot separate period 5 from 10. To determine the period of the output from the method is not difficult. Making the drawback into a slight inconvenience. We simply need to generate the possible period sequence and test the output one by one from the lowest to the highest period, n . It is also possible to test their stability property at the same time.

The other solution is simple. We just concentrate our calculation on prime cycles, since one drawback of this method is that it finds all the lower period cycles that can be fractions of the given period. By focusing on primes, we then rule out this problem for the calculation.

3. Cycles

3.1 Existence of period 3

It can be seen from the bifurcation diagram (Fig.1) that for λ in the range 3.8284 to 3.8495 there is a stable period 3 solution. What is not seen is the other period 3 solution that exists but it is unstable. Even though they both seem to originate from the same origin, their behavior is completely different. The stable one undergoes period doubling, just like another Feigenbaum sequence, while the unstable just continues to exist for the rest of λ . This result is obtained analytically and shown in Drazin [3], but the more complete result is shown here in fig.3. The visibility of this period 3 cycle leads to the discovery of other prime and interger period cycles that are buried in the bifurcation diagram.

3.2 Low period prime cycles and the hidden skeleton

Fig.4 is the structure of POs period 8 4 2 1 that is hidden in the bifurcation diagram. The birth of some structures can be clearly noticed and these are regular structures in chaotic region. May [8] mentioned that the birth is produced by the tangent bifurcation process while the birth of 2^n cycles in the main bifurcation sequence before the critical value of the controlling parameter (λ_∞) is caused by the pitchfork bifurcation process.

The appearance of period 3 right after the Chaos at λ around 3.8284 is very appealing. It makes one wonder if this situation happens for the other prime cycles, or does it just happens for this special cycle. By using our method we can reveal this prime structures at ease.

The results for other prime cycles show that they exist (but undergo unstable) to λ equals to 4. Thus, this can be treated as numerically proof of the complete chaotic state at this value of λ where all cycles exist but unstable:

The more intriguing result is when we overlap the prime cycles on the same plot starting from low period that is period 3, 5, 7 and so on. We found geometrically that the sequence of λ when the first time these cycles exist should converge to a special value of λ . By calculating these numerical values of λ for each cycles from the period from 3 upward, we found that these λ s converge to 3.6786. Surprisingly this value of λ is reported in May [8]. It is the first λ where the first odd period cycle appears. Together with our result, we conclude that the first odd cycle must have a very high period (infinity?). Numerical values of these λ s are shown in Table 2 and plotted in Fig.6.

period	the first appearing λ
3	3.828258
5	3.738068
7	3.701481
11	3.681572
13	3.679700
17	3.678679

Table 2 First appearing values of λ for the prime cycles.

4. Conclusion

To sum up, we have explored prime cycles structures in the bifurcation diagram of the Logistic map. Some of the results accidentally confirm the first value of λ where the first odd cycle exist which was reported by May [8]. Further work includes (1) calculation of Feigenbaum numbers using the DirectD method and (2) exploration of the fine structure in the bifurcation diagram more in detail. Then, the result can be compared with the result reported in Metropolis, Stein and Stein [9].

Acknowledgement

This work was supported in part by the post-doctoral grant of Thailand Research Fund (TRF). One of authors (KJ) would like to thank Dr.M.A.Allen for fruitful discussion.

5. References

- [1] Dahlqvist P and Russberg G 1991 *J.Phys. A* **24** pp 4763-4778.
- [2] Deuffhard P 1984 *BIT* **24** p 456.
- [3] Drazin PG 1992 *Nonlinear Systems* (CUP: Cambridge).
- [4] Eckhardt B 1993 *Phys.Lett. A* **172** pp 411-415.
- [5] Feigenbaum M J 1980 *Los Alamos Sci.* **1** pp 4-27.
- [6] Holodniok M and Kubicek M 1984 *J.Comp.Phys.* **55** p 254.
- [7] Jaroensutasinee K 1997 *A New Method for Cycle Calculation*, Proceeding of the 1st National Computational Science and Engineering Symposium, Bangkok, Thailand, March 1997, pp 32-35.
- [8] May R 1976 *Nature* **251** pp 459-67.
- [9] Metropolis N, Stein ML, and Stein JM 1973 *J. Combinatorial Theory* **15** (A) pp 25-44.
- [10] Parker TS and Chua LO 1989 *Practical Numerical Algorithms for Chaotic Systems* (Springer-Verlag, New York).
- [11] Press WH, Flannery BP, Teukolsky SA and Vetterling WT 1991 *Numerical Recipes in C* (Cambridge: CUP).
- [12] Rowlands G 1990 *Non-linear phenomena in science and engineering* (Ellis Horwood series in physics and its applications: England)
- [13] Wintgen D 1988 *Phys.Rev.Lett.* **61** (16) pp 1803-1806.

Figure captions

1. Bifurcation diagram of the Logistic map.
2. An example of D_{Def} plot for the Logistic map for period 8 and at $\lambda = 3.2$.
3. The hidden bifurcation of period-3 cycles.
4. Structure of Period 8 4 2 1.
5. Structure of Period 1 3 5 7. (a) Period 3 (b) Period 5 and (c) Period 7.
6. Convergence of the sequence of the first apperence λ of some prime cycles.

To calculate period Do the histogram
and check no. of nonzero bin.

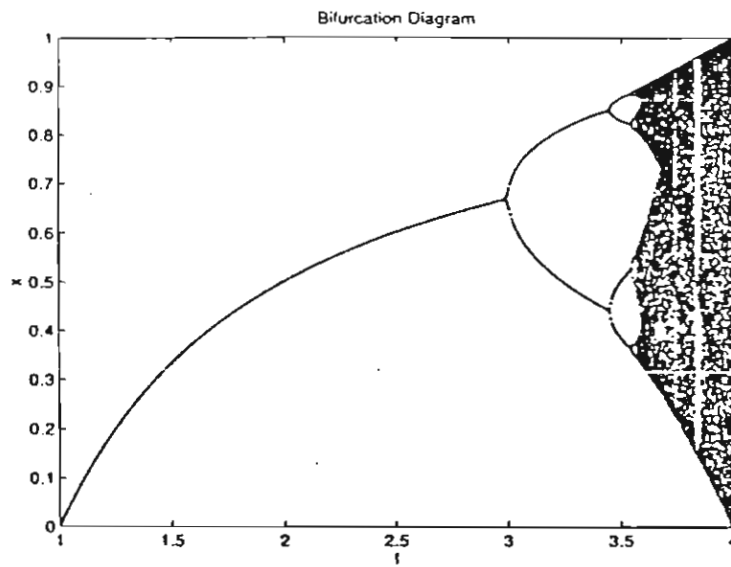


Fig. 1

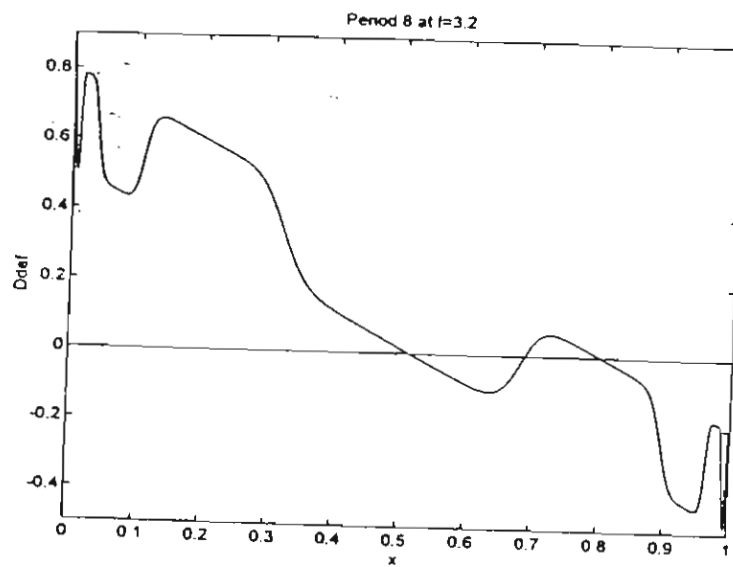


Fig. 2

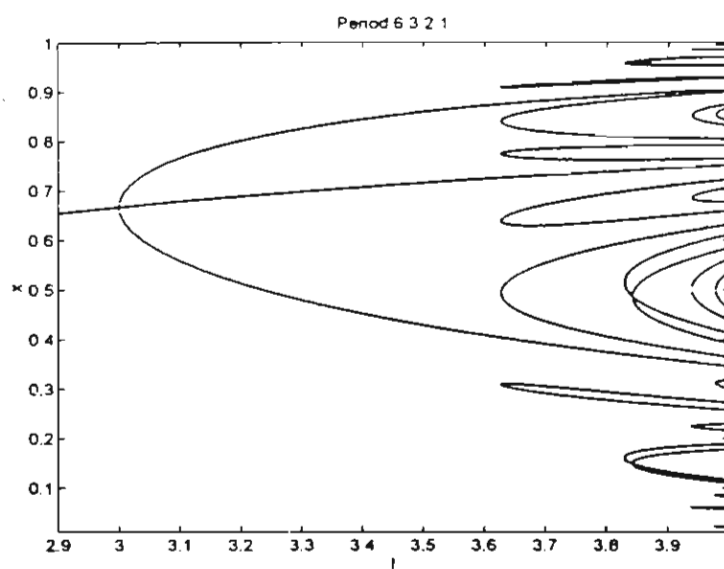


Fig. 3

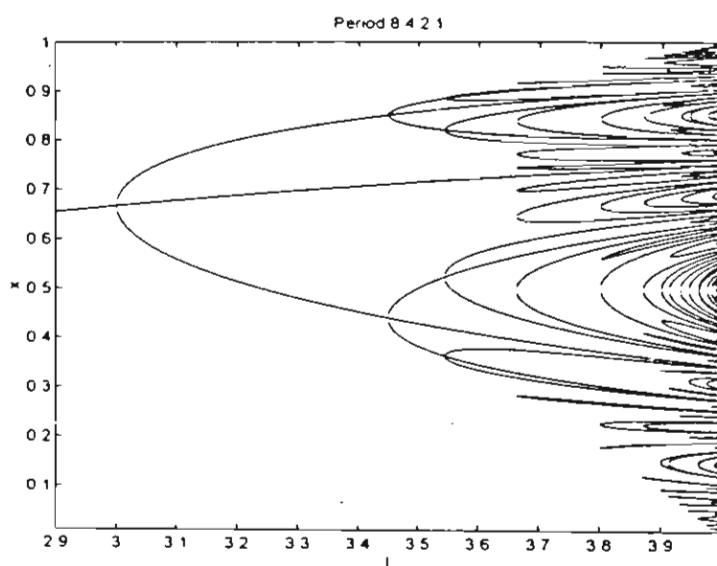


Fig. 4

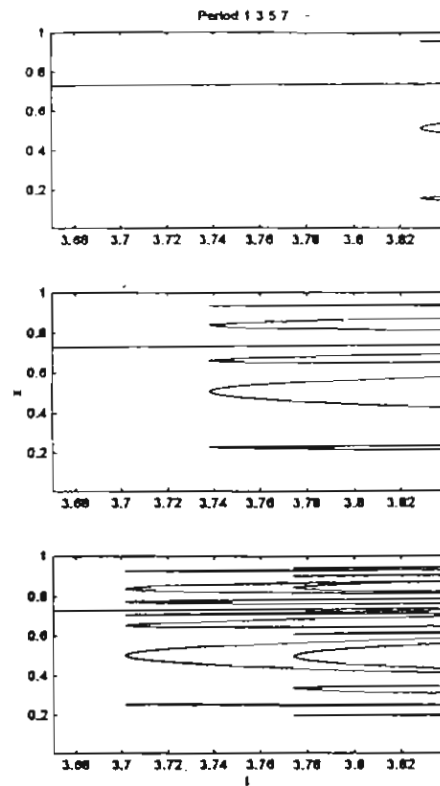


Fig. 5

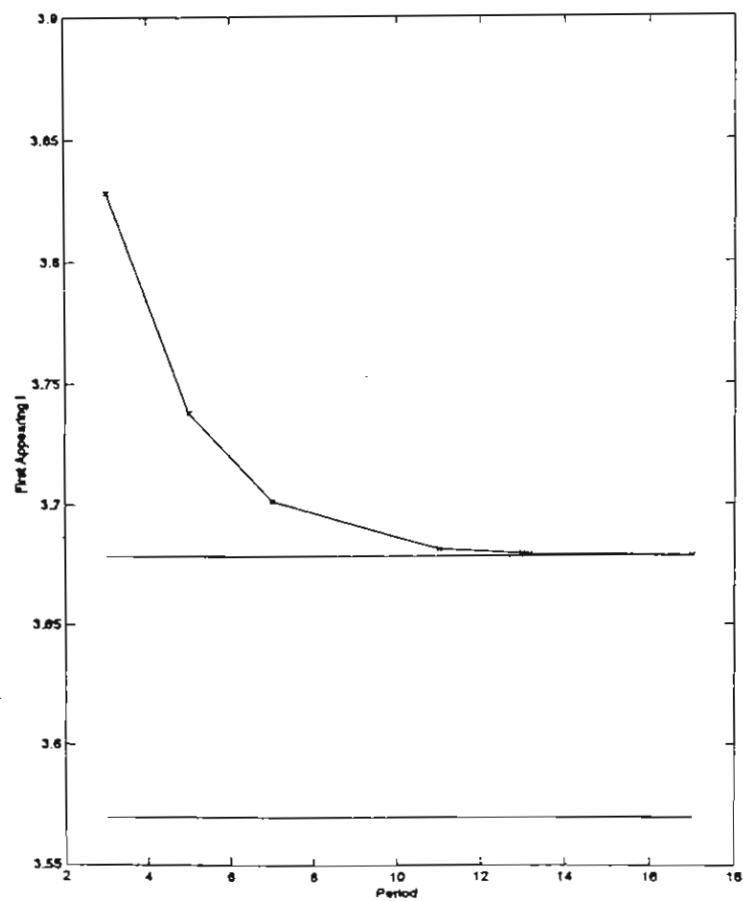


Fig. 6

Homoclinic Tangle Visualisation

Krisanadej Jaroensutasinee,
School of Science, Walailak University
222 Tasala NakronSiThammarat 80160 Thailand
jkrisana@praduu2.wu.ac.th

Abstract

It is proved mathematically that Homoclinic Tangle (HT) is the source of chaos in nonlinear Hamiltonian systems and area conservative maps. The complicated and intricate cutting of stable manifold and unstable manifold causes this. It is possible to make HT visible in some systems such as Henon Map by continuously iterating the map in which we choose many initial conditions that lie on the desired stable manifold. This method works well except it requires long time to locate the selected stable manifold. Here a method called DirectD is applied to the Standard Map (SM) and with slight improvement, this method can make HT very clearly visible. More over, global behaviour of the map at any value of K can be seen with this same method. Periodic orbits structure is also visible. This is possible because both stable and unstable periodic orbits are computable by the method. Interesting plots of the method which show interesting global behaviour of SM are presented.

1. Introduction

Understanding "global" behaviour of chaotic Hamiltonian systems is certainly not an easy task, especially for autonomous or time independent systems because it is very difficult to locate the positions of nonlinear resonance in the phase space. By using the DirectD method, I can compute the behaviour in the phase space for the systems that have the dimension of the phase space equal to 2 (these include the cases of 2d Poincare Surface of Section).

2. Improving DirectD for 2 dimensional Map

The method of DirectD is applicable effectively for the calculation of periodic orbits (POs) in 1d dimensional systems [1,2]. Even though there is some references to methods similar to this method (see for example in [3]) but here I have improved the calculation in sense that it can help us to visualise the systems in a global view. The means we do not need to locate just one particular orbit but we calculate this function for the whole domain. In this work, I calculate this function for Standard Map (SM) in the all-possible domain of interest at any value of K and then study the results. And unexpectedly, this function gives the global picture of the dynamical behaviour of the system automatically.

2.1 DirectD Method of the Standard Map

The method begins with the definition of D_{def} which I called the DirectD function as the following :

$$D_{def}(K, q, p, n) = (Q_K^n(q, p) - q)^2 + (P_K^n(q, p) - p)^2$$

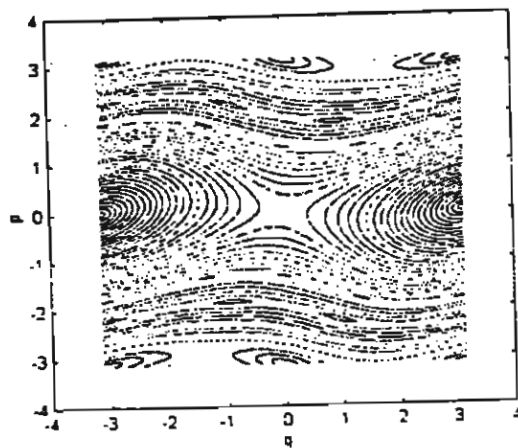
where n is the period of the orbit of interest. Q_n^n is the variable q after n iteration by the mapping function of the Standard Map. Similarly, P_n^n is the variable p after the same number of iteration with the same mapping functions. K is the control parameter of the SM and by definition POs can be found by asserting the condition that requires this function to vanish. For SM the mapping functions are:

$$p' = p + K \sin(q)$$

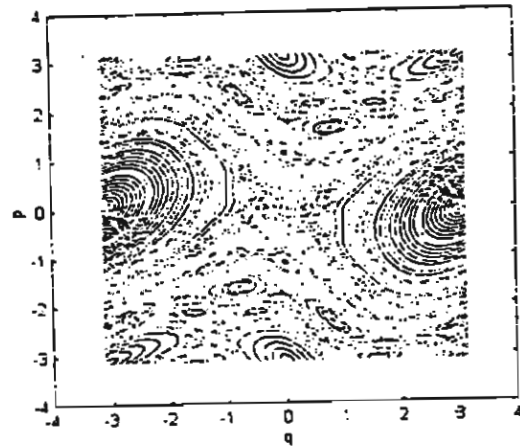
$$q' = (q + p') \bmod 2\pi$$

3. Results

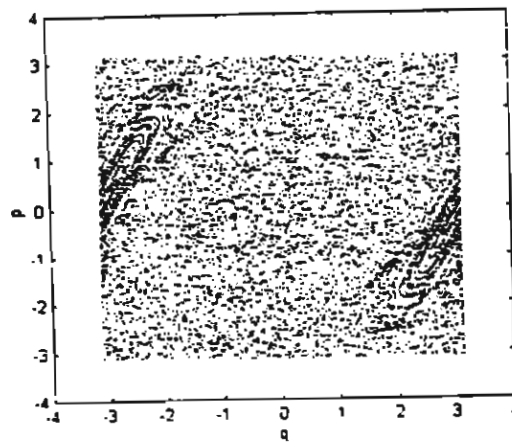
Results are presenting in the following figures:



$K = 0.5, n = 150, m = 100$



$K = 1.0, n = 150, m = 100$



$K = 4.0, n = 70, m = 100$

Fig. 1 The Standard Map.

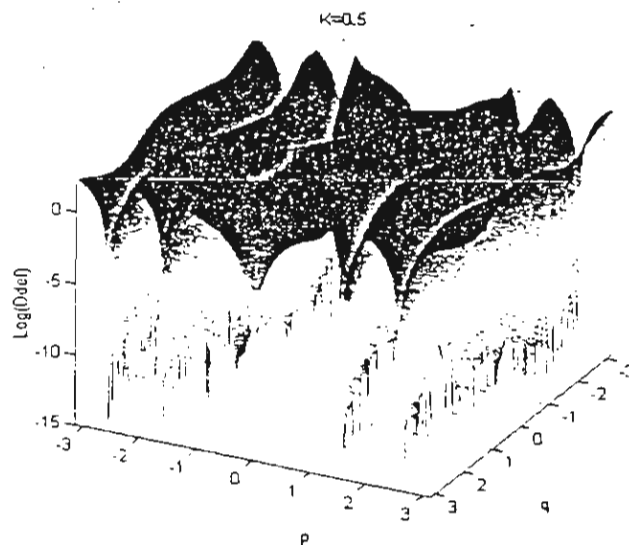


Fig. 2 DirectD function for SM at $K = 0.5$ and $n = 5$.

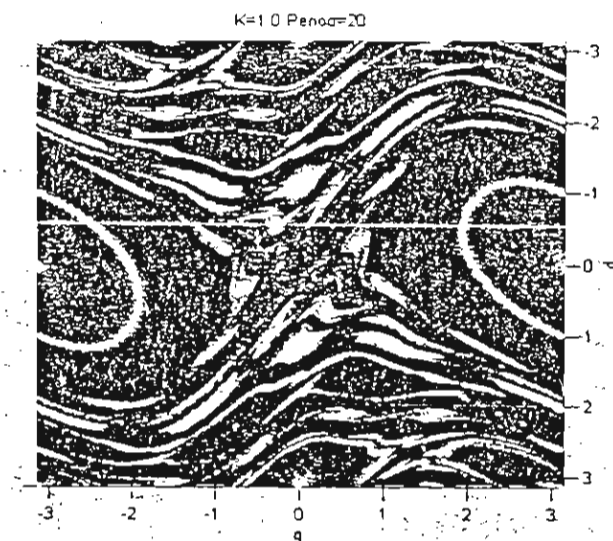
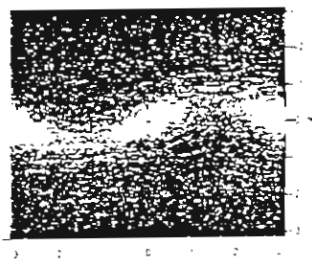


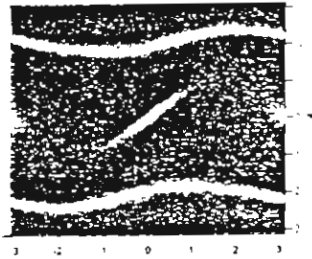
Fig. 3 $K = 1.0$ and $n = 20$.



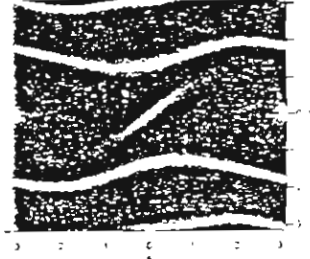
Period = 1



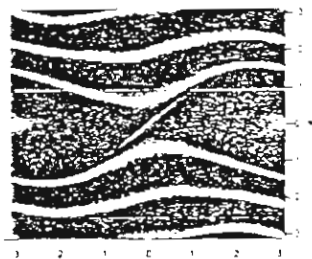
Period = 2



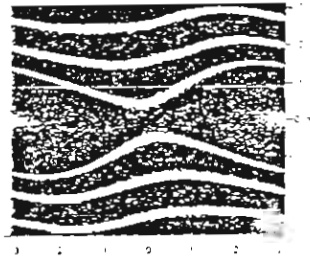
Period = 3



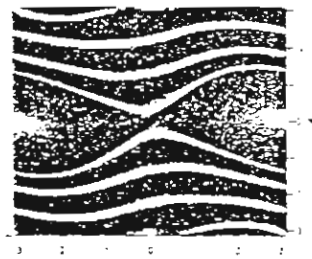
Period = 5



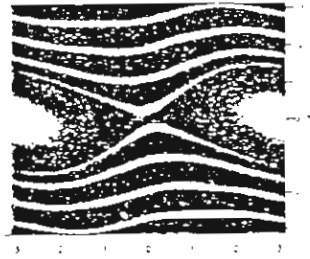
Period = 6



Period = 7

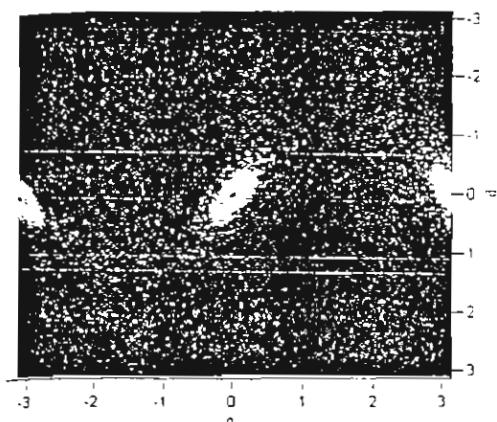


Period = 8

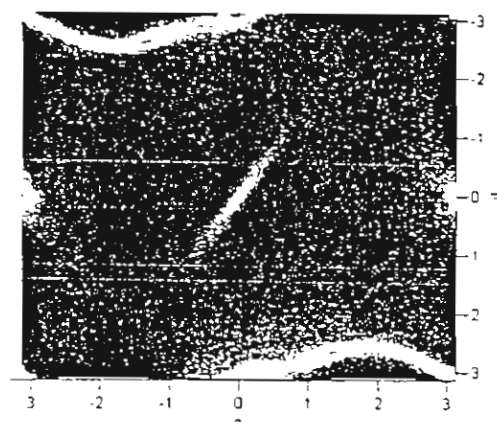


Period = 9

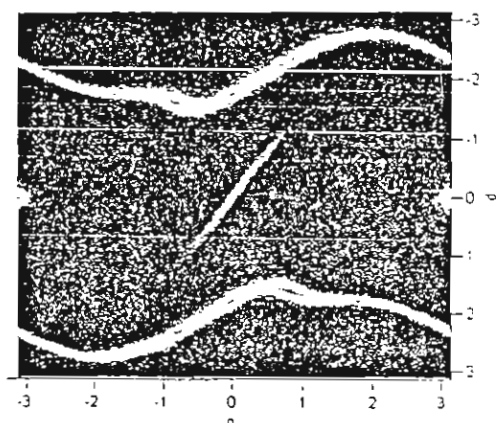
Fig. 4 DirectD functions for $K = 0.5$.



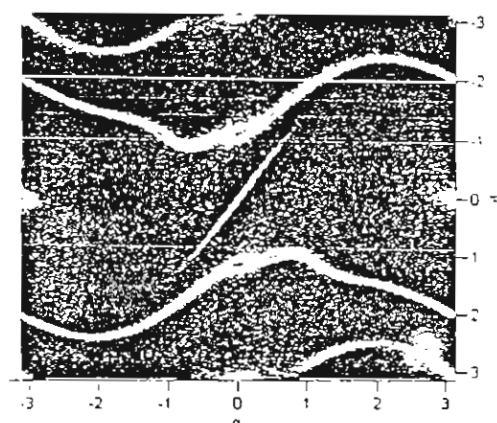
Period = 1



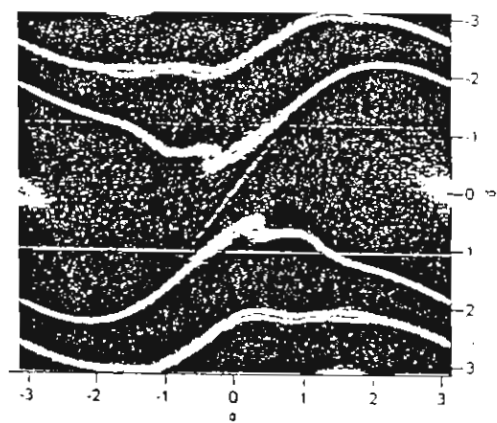
Period = 2



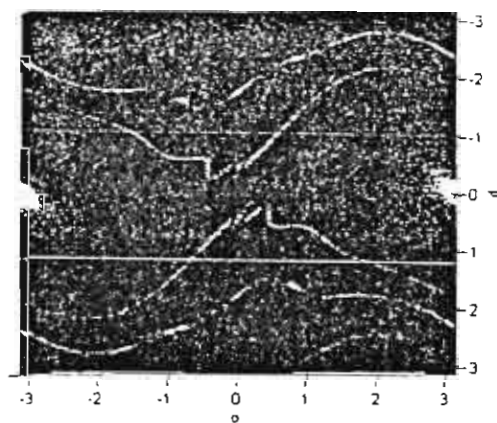
Period = 3



Period = 4



Period = 5



Period = 6

Fig. 5 DirectD functions for $K = 1.0$.

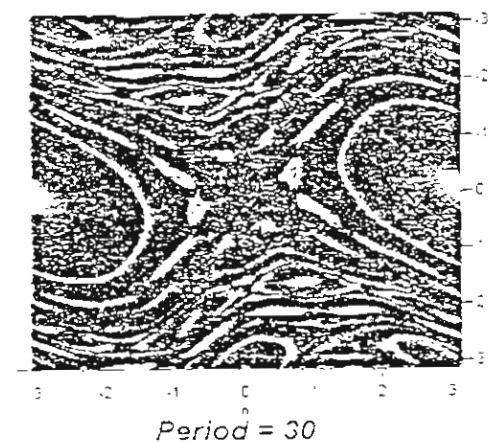
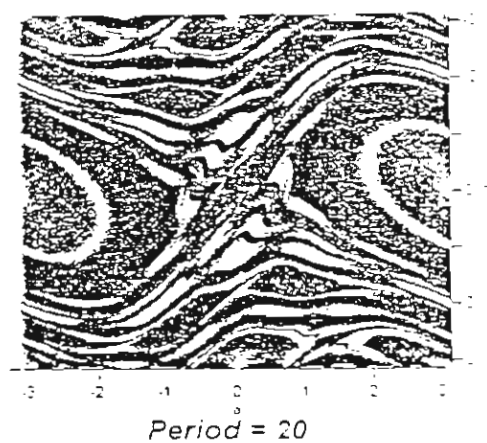
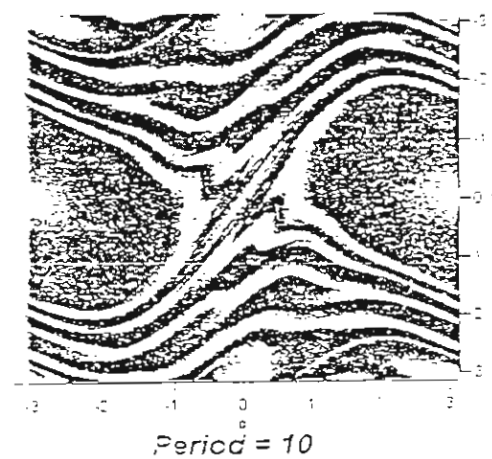
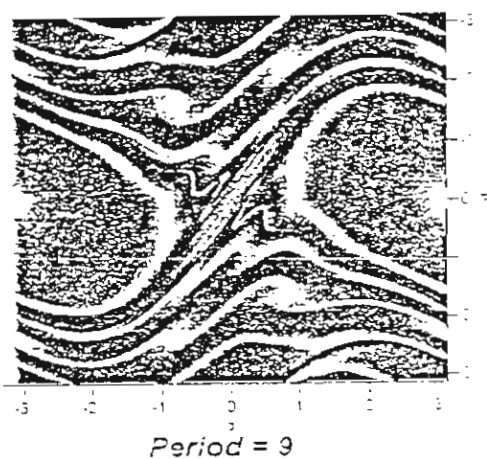
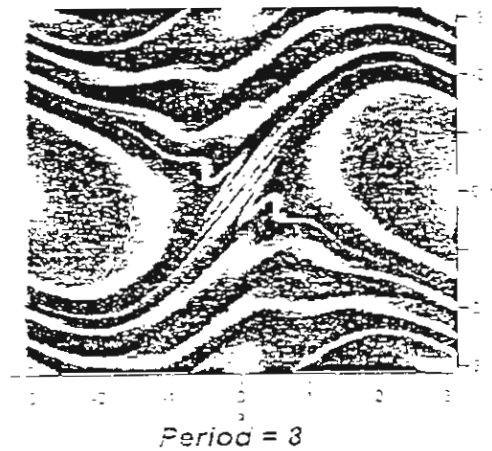
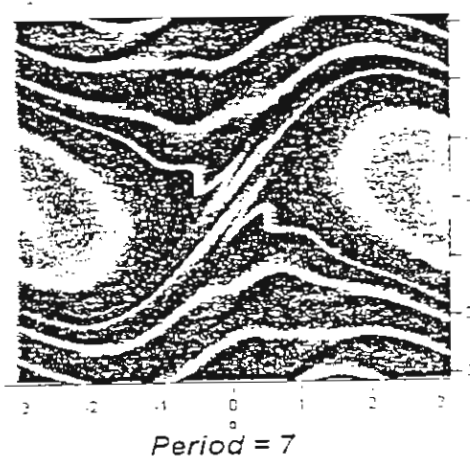
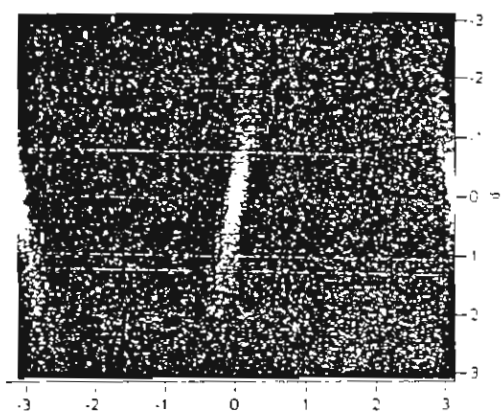
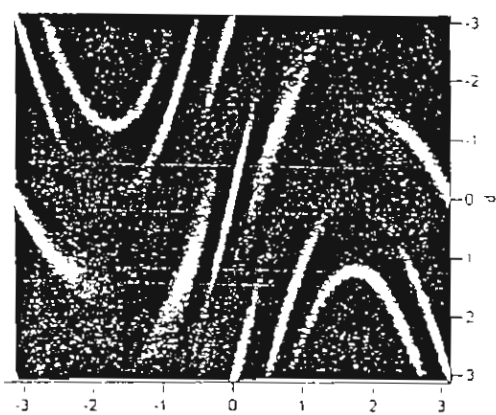


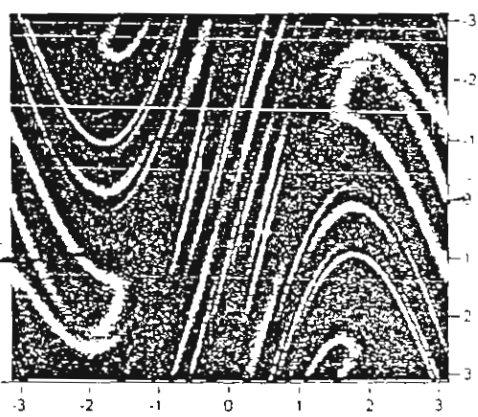
Fig. 5 DirectD function for $K = 1.0$



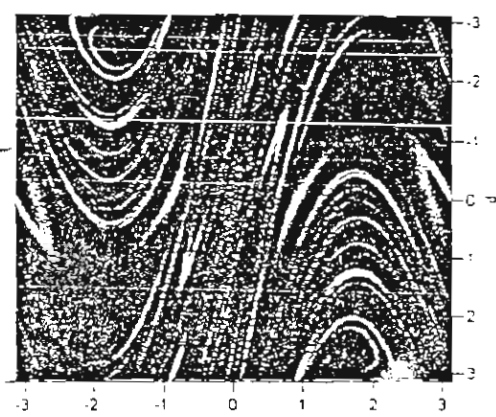
Period = 1



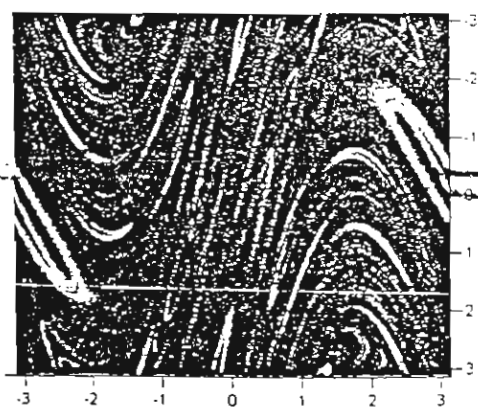
Period = 2



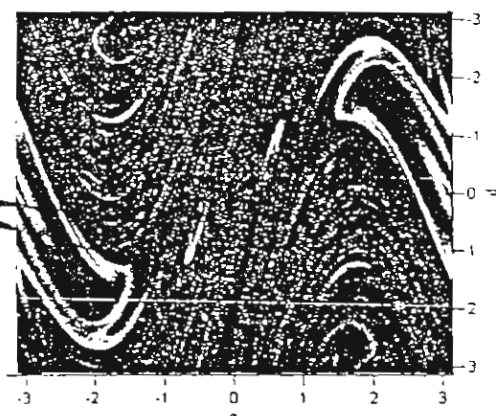
Period = 3



Period = 4



Period = 5



Period = 6

Fig. 7 DirectD functions for $K = 4.0$

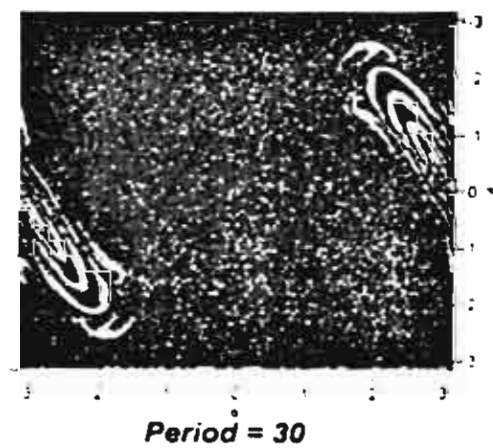
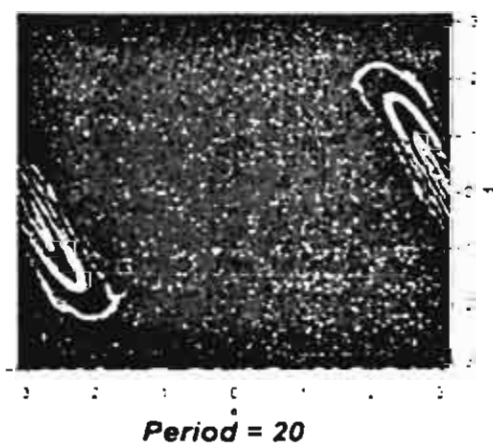
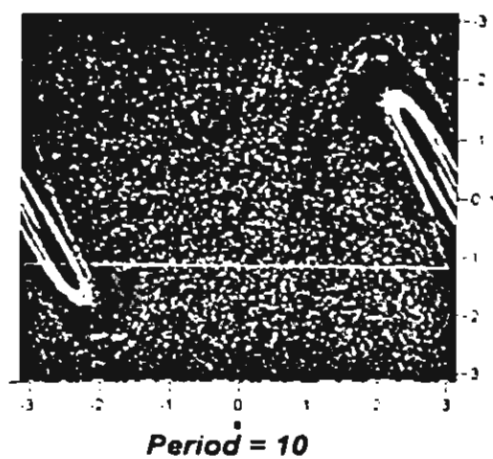
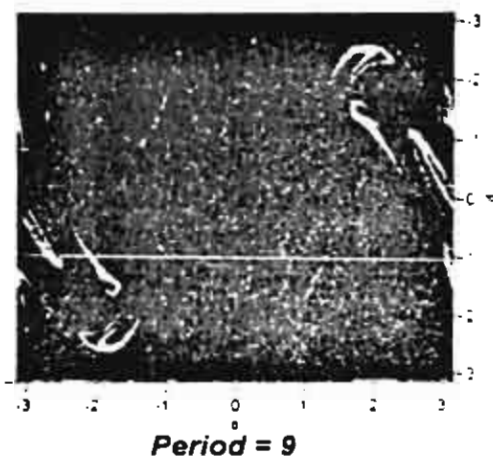
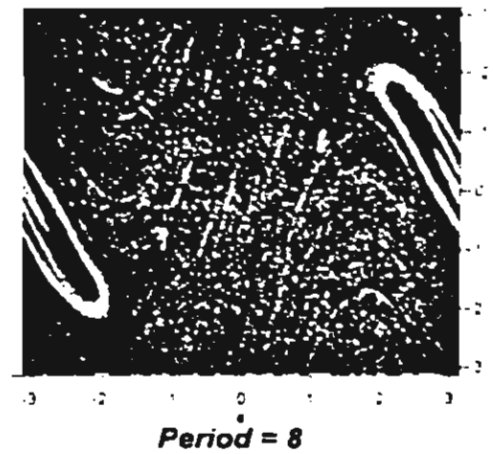
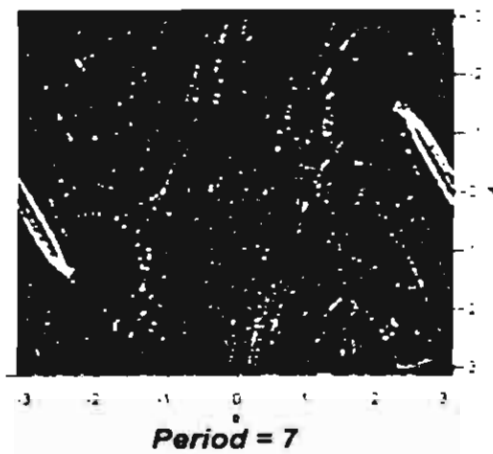


Fig. 8 DirectD function for $K = 4.0$

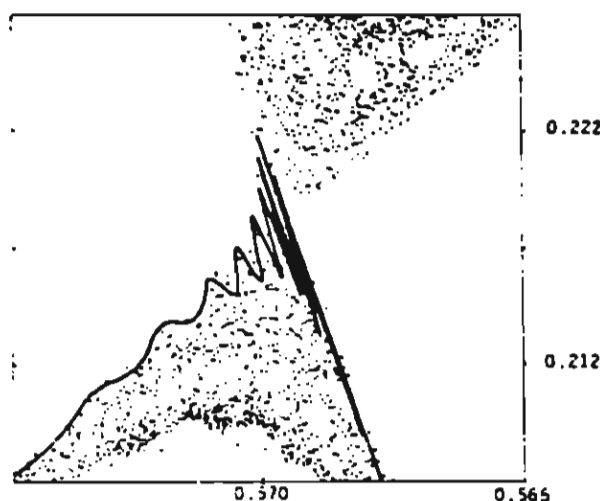


Fig 9. Homoclinic Tangle that appears in [4]

4. Conclusions

I did the calculation for various values of K and found that periodic points can be seen clearly in the plots. As we change K , we found that when K is increased to 1 where chaotic behaviour can be found in some regions. The plots show the intricate structures of the cutting between stable manifolds and unstable ones very distinctively. These structures are known as Homoclinic Tangle (HT) as shown in figure 3. In this figure one could see HTs in many other regions in the plot but these HTs appear in smaller (fractal) sizes such as in the region when q has the value between 1 to 2 and p is between 2 to 3. Another region is when q is between -2 to -1 and p is -3 to -2. [4] has a plot showing orbits that lie on stable manifold to give the clue how HT should be appeared. However, the DirectD method visualises HT a lot more clearly.

And for stable cycles one could see a hollow smooth region in which the centre is the exact location of the POs. Cutting the hole quasi-periodic orbits can be recovered. In conclusion, this method visually gives global dynamical behaviour of the system.

Acknowledgement

This study is supported by the post doctoral grant of Thailand Research Fund (TRF).

References

- [1] Jaroensutasinee K 1997 *A New Method for Cycle Calculation* The proceeding of the 1st Annual National Symposium on Computational Science and Engineering, NSTDA, Bangkok pp 32-35.
- [2] Jaroensutasinee K and Rowlands G 1998 *Prime Cycles in the Logistic Map*, The proceeding of the 2nd Annual National Symposium on Computational Science and Engineering, NSTDA, Bangkok pp 174-181.
- [3] Parker TS and Chua LO 1989 *Practical Numerical Algorithms for Chaotic Systems* (Springer-Verlag, New York).
- [4] Helleman RHG 1980 *Fundamental Problems in Statistical Mechanics 5* (ed Cohen EGD) pp 165-233.

Charged-particle orbits near a magnetic null point

K. JAROENSUTASINEE* and G. ROWLANDS

Department of Physics, University of Warwick, Coventry CV4 7AL, UK

(Received 10 January 2000)

Abstract. An approximate analytical expression is obtained for the orbits of a charged particle moving in a cusp magnetic field. The particle orbits pass close to or through a region of zero magnetic field before being reflected in regions where the magnetic field is strong. Comparison with numerically evaluated orbits shows that the analytical formula is surprisingly good and captures all the main features of the particle motion. A map describing the long-time behaviour of such orbits is obtained.

The motion of charged particles in spatially varying magnetic fields has received a great amount of attention because of its relevance to plasma fusion devices, particle accelerators and astrophysics. Even in the simplest cases, the motion is complicated and is now known to be an example of chaos. One simplifying assumption, which is good when the ratio of the Larmor radius to a scale length describing the spatial variation of the magnetic field, ϵ , is small, is that the so-called adiabatic invariant μ is a constant. This immediately leads to an explanation of charged-particle containment in the Van Allen radiation belts and in magnetic mirror fusion devices. For larger values of ϵ , it has been found that the adiabatic invariant undergoes jumps $\Delta\mu$ where μ changes rapidly in just a few Larmor periods in special regions of symmetry, but otherwise μ is to all intents and purposes constant. The jumps are such that $\Delta\mu \propto \exp(-1/\epsilon)$. For a specific calculation of $\Delta\mu$ for a wide range of magnetic field configurations, see for example Cohen et al. (1978). In this case, the long-time behaviour of particles can be understood in terms of a map relating the values (μ_n, θ_n) before a jump to the values $(\mu_{n+1}, \theta_{n+1})$ after a jump. Here θ is an angle specifying the Larmor phase of the particle. It is found that, to a reasonable approximation (terms of order $\exp(-2/\epsilon)$ being neglected), that one can write

$$\mu_{n+1} = \mu_n + \Delta\mu \cos \theta_n,$$

where, of course, $\Delta\mu$ is a function of μ_n . In many applications, it is sufficient to restrict attention to changes in μ that are small, so that one may linearize the value of μ_n about a chosen mean. Then the above equation reduces to

$$\delta\mu_{n+1} = \delta\mu_n + K \cos \theta_n, \quad (1)$$

and since the original equations of motion were Hamiltonian, the equation for θ variation can be obtained by insisting that the Jacobian is unity. This gives

$$\theta_{n+1} = \theta_n + \delta\mu_{n+1}. \quad (2)$$

* Present address: Walailak University, Thailand.

In the above, K is a constant whose value is determined by the field configuration and energy of the particle.

The above map $(\delta\mu, \theta)$ is the Chirikov map, and is used to study the long-time behaviour of nearly adiabatic particles in spatially varying magnetic fields. For sufficiently small values of $\Delta\mu$, it is found that the particle motion is such that μ changes periodically about a constant value (superadiabatic). For larger values, the motion can become chaotic; and for sufficiently large values, the motion of the charged particle can be understood in terms of a diffusion in momentum space with diffusion coefficient proportional to $\exp(-1/\varepsilon)$. Numerous examples of this type of behaviour have now been studied in detail, and are described in the book by Lichtenberg and Lieberman (1983).

It must be stressed that the direct numerical solution of the particle-orbit equations becomes prohibitively expensive in machine time because one has to follow the particle around its Larmor orbit, whereas it is the motion of the guiding centre that is really needed. Adiabatic and weakly non-adiabatic theory overcome this problem by essentially introducing a suitable averaging procedure to remove the fast motion associated with motion about the Larmor orbit.

However, the whole theory is totally inadequate if, during its motion, a particle can move in a region where the field strength is small or even zero. An example of such a field is the two-dimensional cusp described by the vector potential $\mathbf{A} = xy\mathbf{k}$, where $\mathbf{k}$ is a unit vector along the z axis. For such a field, $\mathbf{B} = (x, -y, 0)$, and the motion of a charged particle in this field is governed by the reduced Hamiltonian (Jaroensutasinee and Rowlands, 1994)

$$H = \frac{1}{2}[\dot{x}^2 + \dot{y}^2 + (Q - xy)^2], \quad (3)$$

where Q is a constant proportional to the z component of the momentum and $\dot{x} \equiv dx/dt$. An immediate consequence of the constancy of H (which in the following we normalize to $\frac{1}{2}$) is that the particle motion is confined to regions between the curves $y = (Q \pm 1)/x$. Thus, for $Q > 1$, the particle is excluded from the origin, the position of the zero of the magnetic field. For $Q \gg 1$, adiabatic theory applies, and the value of the jump $\Delta\mu$ was given some time ago by Howard (1971). A typical orbit is shown in Fig. 1(d).

For $Q < 1$, the origin is no longer excluded, and particle orbits may pass through or close to the zero-magnetic-field region. Some typical orbits are shown in Fig. 1. A subset of these orbits (Figs 1a, b) are such that they remain close to the x axis, and it is for this type of orbit that we now develop a novel analytical approach. The case $Q = 0$ is developed in detail, although the method is applicable to all $Q < 1$.

The exact equations of motion are simply

$$\frac{d^2x}{dt^2} = -xy^2, \quad \frac{d^2y}{dt^2} = -x^2y,$$

whilst the adiabatic invariant μ (the ratio of the perpendicular kinetic energy to the magnitude of the magnetic field) is given by

$$\mu = \frac{1}{(x^2 + y^2)^{3/2}} [(\dot{x}x + \dot{y}y)^2 + x^2y^2(x^2 + y^2)]. \quad (4)$$

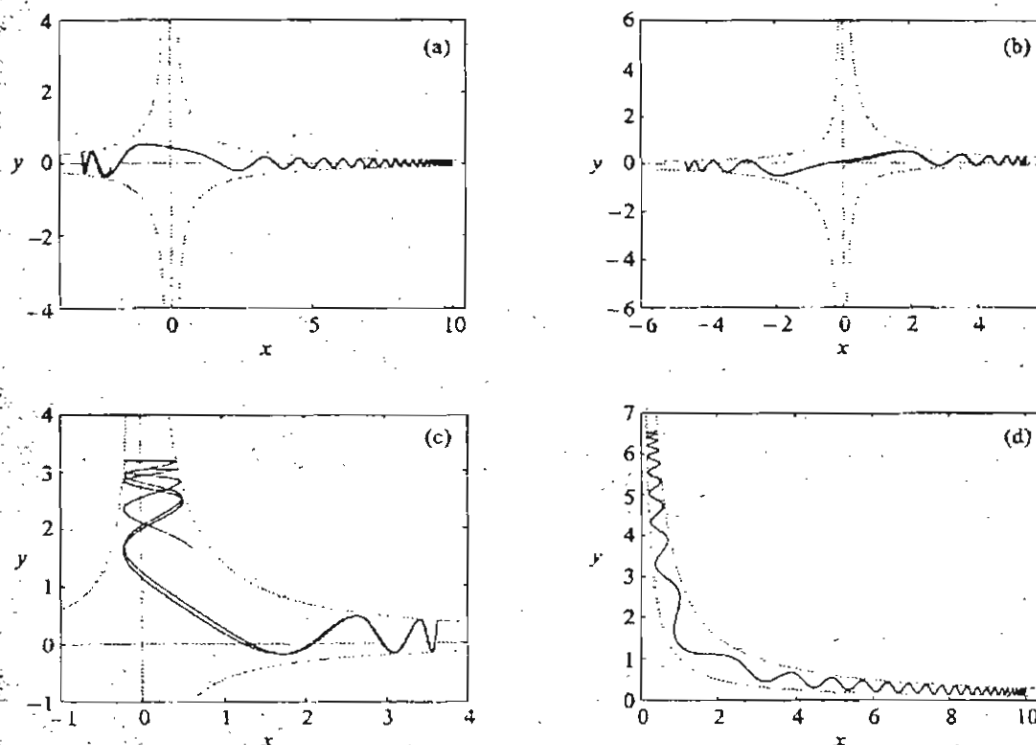


Figure 1. Typical orbits in the linear cusp field: (a) $Q = 0$; (b) 0.4; (c) 0.4; (d) Dotted lines are $y = (Q \pm 1)/x$.

For the type of orbit under discussion, we write $y = g(x)$ and take g to be a monotone function of x . Then, using the equations of motion, we find that g satisfies the following differential equation:

$$\frac{d^2 g}{dx^2} = \frac{1}{1 - x^2 g^2} [xg(1 + g'^2)(gg' - x)], \quad (5)$$

where $g' = dg/dx$.

It is apparent from Figs 1(a, b) that the orbits that pass close to the region where the magnetic field is zero are such that the value of y is small for a range of values of x . Thus we treat both g and x as small, and linearize the above equation in g about $x = 0$ to give $d^2 g/dx^2 = -x^2 g$, whose solution is

$$y = A \sqrt{x} J_{1/4}(\frac{1}{2} x^2) + B \sqrt{x} J_{-1/4}(\frac{1}{2} x^2), \quad (6)$$

where A and B are constants and $J_{\pm 1/4}$ are Bessel functions (Nip et al. 1994). However, such a solution for all x is totally inappropriate, since it excludes the possibility of the particle being reflected in regions of high magnetic field (large x), which is clearly the case as shown in Fig. 1. On the other hand, it is in such regions that adiabatic theory is valid. Thus, in the following, we obtain an approximate analytical expression for the whole orbit of a particle by combining the above form, valid for small x , with an expression, valid for large x , obtained by assuming constancy of the adiabatic invariant.

With $y = g(x)$, so that $\dot{y} = g'(x)\dot{x}$, one can use the expression for the Hamiltonian given by (1) to obtain an expression for $\dot{x}$ and $\dot{y}$ in terms of x , y and

$g'(x)$. This is then substituted into the expression for μ , as given by (4), to give the exact relation

$$(\mu - xg^2)x^3 = \frac{(1 - x^2g^2)(g + xg')^2}{1 + g'^2} \quad (7)$$

The turning point of a trajectory ($x = \bar{x}$) is where $g = 0$ and $g' = \infty$. The above equation gives $\mu\bar{x} = 1$, so the turning point is uniquely specified by the value of μ . If x_n is the position of the n th zero of g , the above equation can be rearranged to give

$$g'^2(x_n) = \frac{x_n}{\bar{x} - x_n}$$

On the other hand, if x_m is the position of m th zero of $g'(x)$, then, neglecting 1 and $g(x_m)$ compared with x_m , (7) reduces to

$$g(x_m) = \sqrt{\frac{\mu}{x_m}}$$

This is a good approximation because, as seen from Fig. 1, the particles move deep into the regions where the magnetic field is large where y is small.

We now use these values of g and g' , obtained at specific points, to fit $y(x)$ for all large values of x to a form suggested by the numerical results shown in Fig. 1, namely

$$y(x) = D(x) \cos \left[\int_0^x \lambda(x') dx' + \phi \right], \quad (8)$$

where $D(x)$ and λ are to be taken as slowly varying functions of x . To obtain expressions for $D(x)$ and $\lambda(x)$, we equate the value of $y'(x)$ calculated at values of x' where $y(x) = 0$, namely $\lambda(x)D(x)$, with the value obtained above at the discrete values x_n . Thus

$$[\lambda(x)D(x)]^2 = \frac{x}{\bar{x} - x}$$

Similarly, at the zeros of $y'(x)$, we obtain the relation $D(x) = \sqrt{\mu(x)}$, where in this case we have neglected terms of order $1/x^{3/2}$. Thus

$$\lambda(x) = \sqrt{\mu(\bar{x} - x)}$$

We now have an explicit solution for $y(x)$ in terms of the two constants μ and ϕ . To obtain a solution for all x , we combine the features of the solution for small x , as given by (6), with that for large x , as given by (8), and propose a solution for all x of the form

$$y \equiv g(x) = A \sqrt{\frac{2h(x)}{x}} J_{1/4}(h(x)) + B \sqrt{\frac{2h(x)}{x}} J_{-1/4}(h(x)), \quad (9)$$

where

$$h(x) = \int_0^x dx \lambda(x) = \frac{2[2\bar{x}^{3/2} - 2(2\bar{x} + x)\sqrt{\bar{x} + x}]}{3\sqrt{\mu}} \quad (10)$$

For small x , where $h(x) = \frac{1}{2}x^2$, this reduces to (4), whilst comparison of this form in the asymptotic region where x and $h(x)$ are much larger than unity, with (8), gives

$$A = -\sqrt{\frac{1}{2}\pi\mu} \sin(\phi + \frac{1}{8}\pi), \quad B = \sqrt{\frac{1}{2}\pi\mu} \cos(\phi - \frac{1}{8}\pi).$$

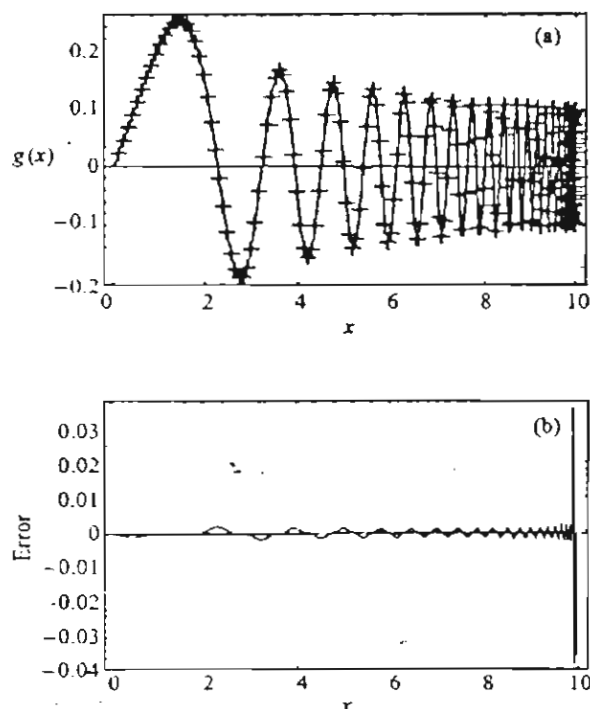


Figure 2. (a) Comparison of a numerically generated orbit with the analytic one. The crosses indicate numerical results, while the solid line shows the analytical results. (b) Relative error of the analytical orbit from the numerically obtained one.

These last two equations can be rearranged to give

$$A^2 + B^2 + \sqrt{2} AB = \frac{\pi\mu}{4} \equiv \frac{\pi}{4x} \quad (11)$$

and

$$\tan \phi = \frac{B/A + \alpha}{1 + (B/A)\alpha}, \quad (12)$$

where $\alpha = \tan(\frac{3}{8}\pi)$. Thus the analytical expression for the orbit, as given by (9), agrees with adiabatic theory for large x and the exact solution in the region of small x and small y . The quantities A and B are obtained from initial conditions, as expressed for example by the values of y and $\dot{y}(t)$ at $x=0$, whilst the value for μ (or x) is obtained in terms of A and B using (11). The form for y , as given by (9), has similar features to a WKB solution, but here the solution is of a nonlinear problem.

The equations of motion for x and y have been integrated numerically, using a symplectic integrating routine (Rowlands 1991), starting at $t=0$ with $x=\mu=0$, $\dot{x}(0)=p$ and $\dot{y}(0)^2=1-p^2$. Such an orbit corresponds to the case

$$B=0, \quad A^2 = 2\Gamma\left(\frac{5}{4}\right)^2 \frac{1-p^2}{p^2}, \quad \phi = -\frac{3}{8}\pi, \quad \mu = \frac{1}{x} = \frac{4A^2}{\pi}.$$

The integrator conserved energy to one part in 10^{-6} . An orbit so obtained is shown in Fig. 2, where it is also compared with an orbit obtained from the

analytical expression given by (9). The value of p is chosen to give a value of $\bar{x} = 10$, and it is this value that is used in (10) to uniquely determine $h(x)$. The agreement is surprisingly good over the whole range of possible values of x .

Equation (9) specifies an orbit that starts from $x = 0$ in terms of the two constants A and B . Eventually, the particle is reflected at $x = \bar{x}$, and then continues until it returns to the $x = 0$ plane, but in general with different values of $y(0)$ and $\dot{y}(0)$. Importantly, the same equation (7) describes this return orbit, but now specified by different values of A and B , which we denote by $\hat{A}$ and $\hat{B}$. These new values can be related to A and B simply by imposing continuity on the particle orbit. Thus we demand that both orbits give rise to the same value for y when $x = \bar{x}$ and a value for $\dot{y}(t)$ equal in magnitude, but opposite in sign, corresponding to a reflection at $x = \bar{x}$. These conditions lead to a phase change in ϕ in the expression for y as given by (8) such that the new value $\hat{\phi} = 2\pi - 2h(x) - \phi$. Since the turning point of both orbits is the same, both orbits have the same value of μ . The adiabatic invariant μ does not change at the reflection point – only the phase does. Using (11) and (12), it is now possible to relate $(\hat{A}, \hat{B})$ and (A, B) , and we find

$$\frac{\pi}{4x} = A^2 + B^2 + \sqrt{2} AB = \hat{A}^2 + \hat{B}^2 + \sqrt{2} \hat{A} \hat{B} \quad (13)$$

and

$$\sqrt{2} \left(\frac{\hat{B}}{\hat{A}} + \frac{B}{A} \right) + (1+b) = -\frac{\hat{B}}{\hat{A}} \frac{B}{A} (1-b), \quad (14)$$

where $b = \tan(\frac{2}{3}\bar{x}^2)$.

The orbit illustrated in Fig. 2(a) has been followed numerically until it returned to the $x = 0$ plane. With initial conditions corresponding to $A = 0.2802$ and $B = 0$, the numerically obtained solution gave values of $\hat{A} = 0.3848$ and $\hat{B} = 0.1670$ as the particle passed through the $x = 0$ plane after one reflection at its mirror point $x = \bar{x}$ and of the order of 20 Larmor orbits. The corresponding values obtained using (9) are 0.3697 and 0.1604 respectively. This excellent agreement has been obtained without introducing any scaling parameters but merely demanding that both orbits have the same initial conditions. This same agreement has been found for a range of different orbits, showing that (9) gives an analytical expression which captures all the main features of the true orbit.

The particle now labelled by $(\hat{A}, \hat{B})$ continues to move, and enters the negative half x space – but this is exactly equivalent to entering the positive half x space with a simple change in the direction of the x component of the velocity. The particle may now be considered as moving to the right in positive x space, but labelled by $(-\hat{A}, \hat{B})$.

Thus we can specify the notion of a particle including many crossings of the $x = 0$ plane by a map relating A, B to $(-\hat{A}, \hat{B})$. This map is essentially given by (13) and (14). However, it can be simplified by considering particles moving through the $x = 0$ plane in the positive direction only. Thus if (A_n, B_n) labels the n th such crossing, we find

$$A_{n+1} = -\sqrt{2} A_n \cos h_n + B_n (\sin h_n - \cos h_n), \quad (15)$$

$$B_{n+1} = A_n (\cos h_n + \sin h_n) - \sqrt{2} B_n \cos h_n, \quad (16)$$

where $h_n = \frac{2}{3}\bar{x}_n^2$ and depends on (A_n, B_n) as expressed by (13). One can use these

equations together with (11) and (12) to calculate μ_{n+1} in terms of μ_n and the phase ϕ_n . To this extent, it is a generalization of the Chirikov map to the case where the particle passes in the vicinity of a null point in the magnetic field. However, it is limited to orbits that always remain in the 'x' arms (or, by symmetry, the 'y' arms) of the field. Such orbits are shown in Figs 1(a, b).

This map has been studied numerically, and shows chaotic behaviour for most initial conditions. It is found that the value of μ gradually increases owing to chaotic diffusion, which means that $\bar{x}$ decreases and the particle is reflected at a smaller value of x . This can lead to the particle moving into an arm of the cusp at right-angles to its present one. An example of such an orbit is shown in Fig. 1(c). However, once it is in the 'y' arm, it moves initially such that x is small and then into an adiabatic region with y large. Thus the analytical form of the orbit can still be described by (9), but with x and y interchanged. We specify this orbit by $(\tilde{A}, \tilde{B})$. Now a particle specified by (A, B) undergoing one reflection in an x arm, returns to the vicinity of the origin, and is specified by $(\tilde{A}, \tilde{B})$ as discussed above. For small x , one can use (9) to express this orbit in the form $y = \alpha_1 \tilde{B} + \alpha_2 \tilde{A}x$, where $\alpha_1 = \sqrt{2}/\Gamma(\frac{3}{4})$ and $\alpha_2 = 1/\sqrt{2}\Gamma(\frac{5}{4})$. This same orbit, expressed as one that eventually moves in a 'y' arm now takes the form $x = \alpha_1 \tilde{B} + \alpha_2 \tilde{A}y$. Simple algebra then shows that $\tilde{A} = 1/\alpha_2^2 A$ and $\tilde{B} = -\tilde{B}/\alpha_2 A$. Thus we can express the initial conditions for a particle moving into a 'y' arm, namely $(\tilde{A}, \tilde{B})$, in terms of (A, B) and hence (A, B) , which specify the previous excursion in a 'x' arm. Finally noting that a particle will only switch from an 'x' arm to a 'y' arm if the slope of the orbit near $x = 0$ is greater than unity, that is $\alpha_2 A > 1$, one can construct a map that generalizes that given above to allow for transfer between arms. This takes the form

$$(A_{n+1}, B_{n+1}) = \begin{cases} (-\tilde{A}_n, \tilde{B}_n) & \text{if } |\alpha_2 A_n| < 1, \\ \left(\frac{1}{\alpha_2^2 A_n}, -\frac{\tilde{B}_n}{\alpha_2 A_n} \right) & \text{if } |\alpha_2 A_n| > 1. \end{cases}$$

This map is a generalisation of the Chirikov map to allow a particle to move in regions where the magnetic field is small, and even zero, and the concept of an adiabatic invariant, such as μ , which can have small jumps $\Delta\mu$, is no longer valid.

A preliminary numerical investigation has been made using this map. It is found that particles are contained in a finite portion of the relevant phase space, that is with finite values of A_n and B_n , reflecting the boundedness of exact particle orbits due to energy conservation. The phase-space plots of typical orbits show little structure, implying that the particle motion is chaotic for most initial conditions.

Conclusions

An approximate but analytical expression has been obtained describing orbits of charged particles moving in the neighbourhood of a null point in a magnetic field. Comparison with direct numerical evaluation of particle orbits show that this analytical form is surprisingly accurate. Typical orbits have been followed over a time interval that includes many Larmor oscillations, with a resulting accuracy of a few percent.

This analytical form has been used to obtain a map that specifies an orbit in the vicinity of the magnetic field null in terms of two quantities (A_{n+1}, B_{n+1}) , and relates these to the previous values (A_n, B_n) when the particle was in the same region. This map can be used to study the long-time behaviour of particles (many crossings of the null region) more efficiently in computer time than a direct numerical integration of the particle-orbit equations. Preliminary results suggest that this map shows chaotic behaviour for all or most initial conditions.

Acknowledgement

We wish to thank the Development and Promotion of Science and Technology Talents Project (DPST) of the Thai government for financial support for one of us (K. Jaroensutasinee).

References

- Cohen, R., Rowlands, G. and Foote, J. H. 1978 *Phys. Fluids* **21**, 627.
Howard, J. E. 1971 *Phys. Fluids* **14**, 2373.
Jaroensutasinee, K. and Rowlands, G. 1994 *J. Phys.* **A27**, 1163.
Lichtenberg, A. J. and Lieberman, M. A. 1983 *Regular and Stochastic Motion*. Springer-Verlag, New York.
Nip, M. L. A., Tuszynski, J. A., Otwinowski, M. and Dixon, J. M. 1992 *J. Phys.* **A25**, 5553.
Rowlands, G. 1991 *J. Comput. Phys.* **97**, 235.

Visualization of Particle Motion in $(Q - xy)^2$ Potential

Krisanadej Jaroensutasinee¹ and George Rowlands²

¹Institute of Science, Walailak University, Tasala, NakhonSiThammarat 80160 Thailand
jkrisana@wu.ac.th, Tel. +6675 672005-6 Fax. +6675 672004

²Department of Physics, Warwick University, Coventry CV4 7AL ENGLAND

Abstract

The simple form of $(Q - xy)^2$ potential in classical Hamiltonian framework can surprisingly generate complicated particle behavior. In this work, various visualization techniques and theoretical simplification are combined to understand the dynamic of particle behavior. 3D graphic generation is employed to visualize particle trajectories and phase space. A cluster of PCs, whose OSs are both Windows and Linux, are parallelly utilized to generate Poincare surface of section and Fermi mapping which are in general required large computing time.

1 Introduction

Visualization is a very important tool when one comes across a classical system with high degree of non-linearity. In this work the simple form of potential $(Q - xy)^2$ in a classical Hamiltonian framework is selected for study. Due to non-linearity in the form of this potential, the motion of particle in this potential can surprisingly generate complicated particle behavior. Q appeared in the potential is a constant which origins from the constancy of the momentum in z direction and x and y are the other coordinate variables in the rectangular system. The Hamiltonian is

$$H = \frac{1}{2}(p^2 + q^2 + (Q - xy)^2), \quad (1)$$

where p and q are the corresponding momenta in x and y . The equations of motion are:

$$\begin{aligned} \frac{dx}{dt} &= p & \frac{dp}{dt} &= y(Q - xy) \\ \frac{dy}{dt} &= q & \frac{dq}{dt} &= x(Q - xy) \end{aligned} \quad (2)$$

In this work, various visualization techniques and theoretical simplification are combined to understand the dynamic of particle behavior. 3D graphic generation is employed to visualize particle trajectories and phase space. Computation has been done in 2 developing cluster environments, Java and Mathematica[1].

2 Normal Orbits and Periodic Orbits

As mentioned, this simple form of potential yields many classes of particle orbits. First, by analysis the stability of a special class of periodic orbits that is defined by the relation $x = \pm y$ in this potential, Jaroensutasinee and Rowlands [2] proved that there exist ranges of Q where these orbits are stable. However, this work shows that these orbits are not so useful in the

original problem in which the particle moves in a cusp magnetic field. 3D visualization shows that the periodic orbits are simply not periodic in the x - y - z space, but intricately weave into a net on the $x = \pm y$ plane. Nonetheless, mathematically these orbits are still useful in sense of the 2D potential of this class.

3 Multiple Scale Perturbation and Global Behaviour

In this system the particle is allowed to move very far away from the center. In other words, the particle space and, hence, the phase space are unbounded. The particle can go to infinity in both x and y direction. And in Plasma Physics research this is the situation where the loss from the cusp magnetic containment occurs.

There is another side effect. To generate Poincare surface of section of an unbounded system can become a very long computing time job. So, in order to obtain a kind of global map of particle behavior, one needs to resort to a mixture of analytical and numerical techniques.

By considering the fact that this system is derived from a charged particle motion in a non-uniform magnetic field, the guiding center approach is highly appropriate. In this system the guiding center system is simply the Hyperbolic coordinate system.

Once transformed into the Hyperbolic system, fast and slow time scales are made explicit by introducing multiple scale perturbation. In this perturbation scheme the first adiabatic invariance also comes out naturally and it is, of course, the magnetic moment. Adiabatic motion is the case when the magnetic moment is kept constant.

3.1 Nonadiabatic Motion

All above perturbation seems to work well when describing the motion until the particle crosses the so-called mid plane where the magnitude of magnetic field is smallest. The particle executes a magnetic moment jump when crossing the plane, and so the adiabatic approximation is no longer working. In other words, the particle executes a constant magnetic moment until it arrives near the point where B is minimum. Then, it makes a jump to a new value. This is known as “non-adiabatic” behavior. And for $Q > 1$ we can compute the jump analytically. That is the jump of magnetic moment can be calculated analytically from its time derivative with some helps from complex analysis[5]. It is found that the jump relates to the Larmor phase at the minimum B crossing.

If the change of the Larmor along the guiding center is approximated, then we can derive a form of mapping. This resulting map is very similar to the Fermi Map. In short, from the prediction of multiple scale perturbation study, we obtain Fermi-like map, which accounts both adiabatic and non-adiabatic behavior, hence global behavior of the particle. The mapping equations are:

$$\begin{aligned}\mu_{n+1} &= \mu_n + g(\mu_n)\cos\theta_n, \\ \theta_{n+1} &= \theta_n + d(\mu_n + 1),\end{aligned}\tag{3}$$

where

$$\begin{aligned}
g(\mu) &= -\frac{\pi\mu^{1/2}}{\Gamma(9/8)Q^{1/8}2^{3/8}}\exp\left(-\frac{1}{Q}\int_0^{\pi/2}\frac{\cos\theta d\theta}{\sqrt{1-\sqrt{2Q\mu}\cos^{1/2}\theta}}\right) \\
d(\mu) &= 2Q\int_0^{\sqrt{(\sqrt{2Q\mu})^4-1}}\frac{ds}{\sqrt{1-\sqrt{2Q\mu}(1+s^2)^{1/4}}}.
\end{aligned} \tag{4}$$

We tested the resonances with numerically integrated results from the full equations of motion in the original domain (x,y,p,q) and found very good agreements. It is worth noting that this map can be linearized around a resonance to give the standard or Chirikov (named after the first discoverer) mapping.

4 Visualization

4.1 Difficulties associate with this system

This system is unlike a chaotic system in which one can employ a personal PC to study it. It is highly chaotic. In fact it was long-time believed that this system for $Q = 0$ is ergodic, where there is no stable periodic solution at all, until a stable periodic orbit was found [4]. However, the main problems, that make it hard to tackle this system, are not only the chaotic property but also the unbounded property of the space. Both problems lead one to require extensive computing in both accuracy and speed when dealing with this system.

4.2 Crunching the numbers

As mentioned above, a cluster of PCs, which are both Windows and Linux based, are utilized parallelly to study this system. We create software in Java and Mathematica [1] environments. The main tasks are to generate Poincare surface of section and Fermi mapping which are in general required large computing time on a single computer. For Java, we create a small applet which, when run, will connects itself to a Java server program (which we call it mCenter). Once connected, mCenter will parse on the task to the applet via TCP/IP network. When the applet finishes the calculation, it will communicate the results back to mCenter. The connection can be extended to multiple computers on network and each connection will run on its own thread. Because the program is in the form of applet, we can use any available computers on the network as our workforce, no matter what operating system the machine is running. Java promises across platform compatibility, it is ready for numerical computing and its speed on certain platforms can out-perform some optimized C/C++ programs[6].

The other way of exploiting our cluster is to use Mathematica via Parallel Computing Toolkit. The front-end-and-kernel concept makes Mathematica immediately available on a cluster. Multiple kernels can run separately on each machine and all of the kernels connect to a master kernel, which, of course, also connects to a front-end. Parallel and distributed application can be developed right away with the unified concept of Mathematica programming[7]. We use this method to produce our Fermi-like map and some visualization of particle orbits. Mathematica version 4 has provided a new function of real time visualization of 3D graphic using OpenGL standard. This function is provided on both Windows and Unix platforms[8].

5 Conclusions

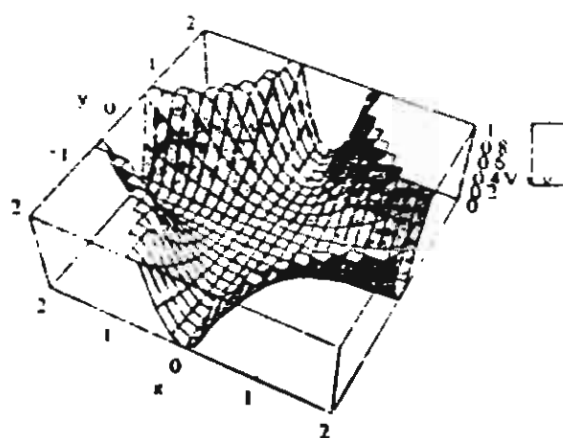
This work is a starting point of our research using a cluster of computer. Two methods have been developed and employed in the study of a selected chaotic system. We found that most methods are suitable to our research computing environment where linux is not only the option for our OS and each node is normally used in various forms of applications. Computation time is reduced as long as communication between nodes is kept minimum while computation load on each node is kept maximum. However, real time visualization, that we used here, still works on a single computer. Future development is needed so that cluster visualization of large numbers of component can be done in real time.

6 Acknowledgement

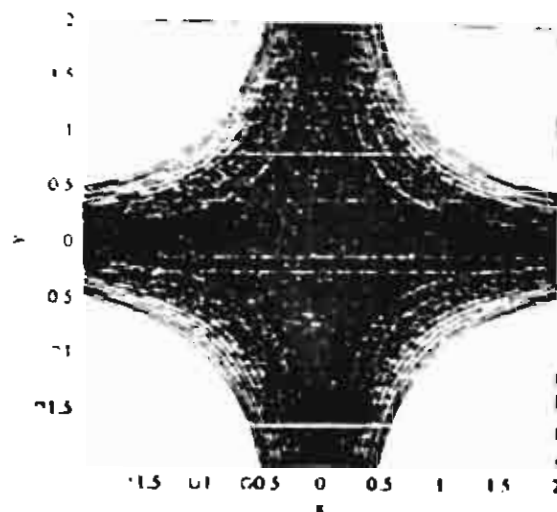
The authors would like to thank Thailand Research Fund, Post-Doctoral Grant (PDF/62/2540), for supporting this work.

7 References

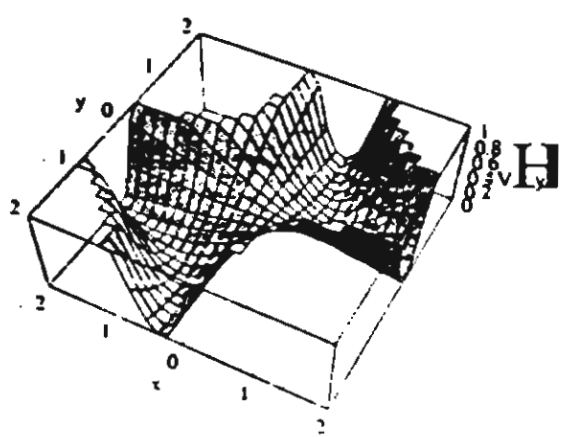
- [1] S Wolfram 1999 *The Mathematica Book* (CUP and Wolfram Media).
- [2] K Jaroensutasinee and G Rowlands 1994 *J. Phys. A* **27** p 1163.
- [3] K Jaroensutasinee and G Rowlands 2000 *J. Plasma Phys.* **63** (3) p 255.
- [4] P Dahlqvist and G Russberg 1990 *Phys. Rev. Lett.* **65** (23) p 2837.
- [5] J E Howard 1971 *Phys. Fluids* **14** p 2373.
- [6] RFB Boisvert, J Moreira, M Philippsen and R Pozo 2001 *Comp. Sci. Eng.* March/April 2001 p 18.
- [7] S Rujiwan and K Jaroensutasinee 2001 Paper Presented in ANSCSE5.
- [8] Wolfram Research Technical Staff 2001 *The Mathematical Jour.* **8** (1) p 63.



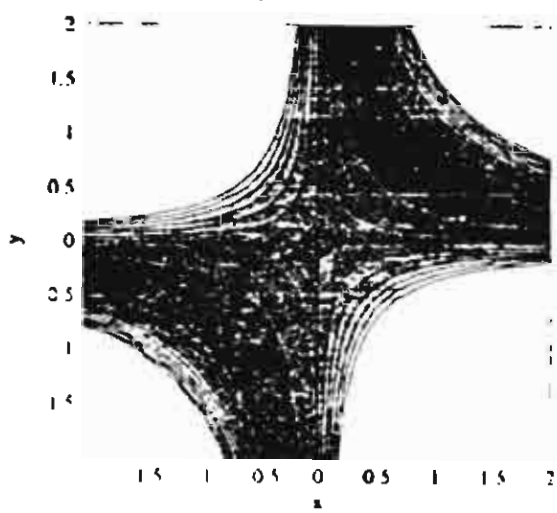
(a1)



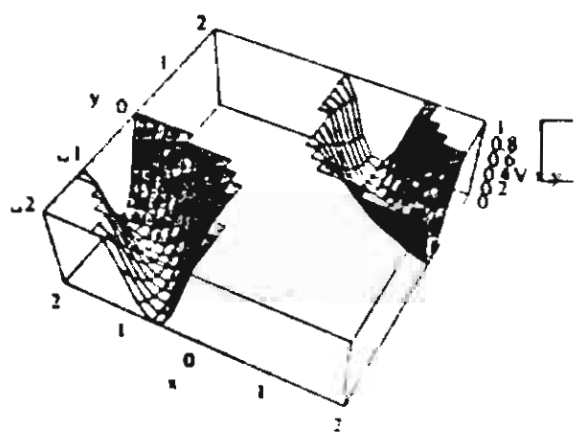
(a2)



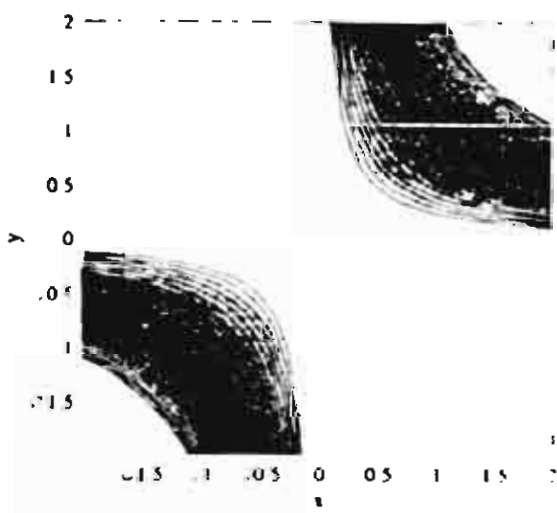
(b1)



(b2)



(c1)



(c2)

Figure 1. 3D views of the potential and their contours:
 $Q < 1$ for (a1) and (a2), $Q = 1$ for (b1) and (b2) and for (c1) and (c2) we set $Q > 1$

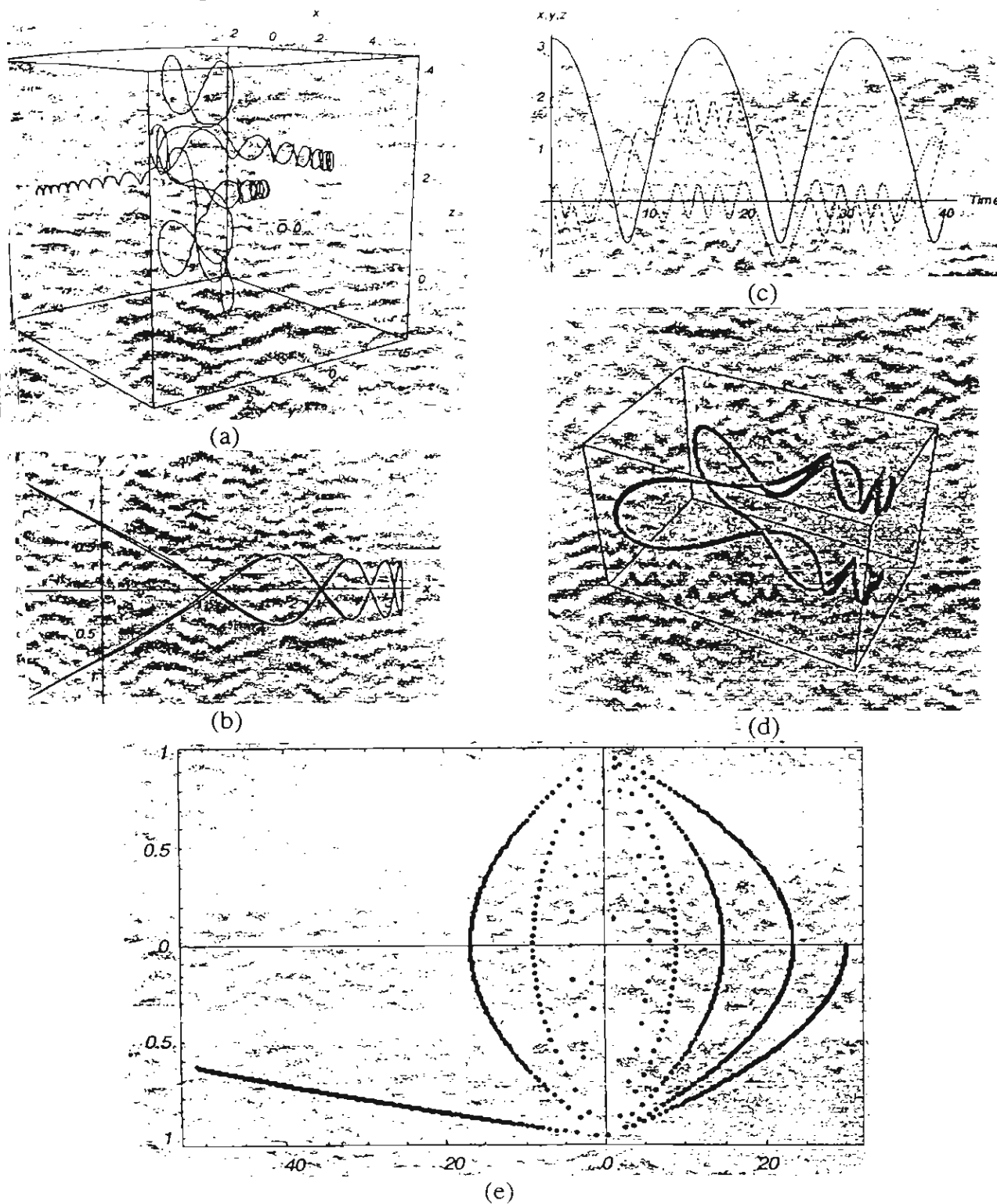
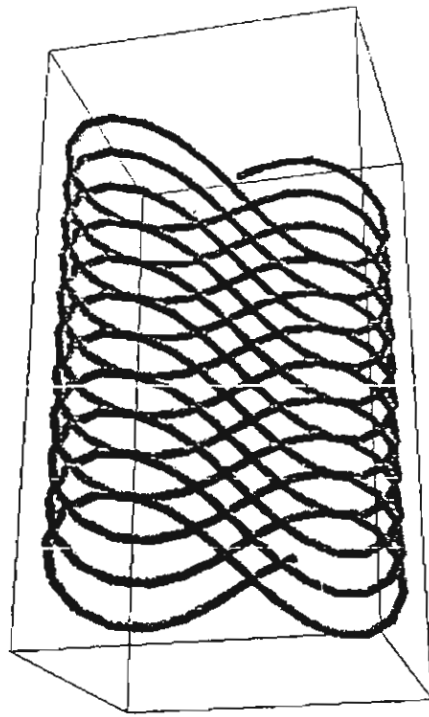
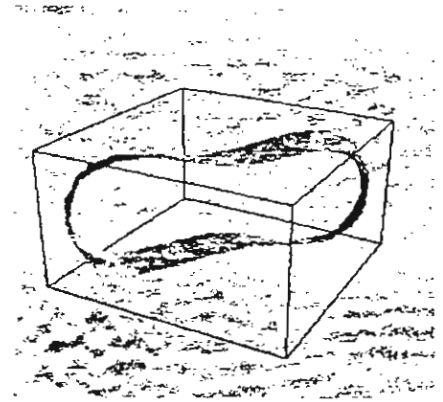


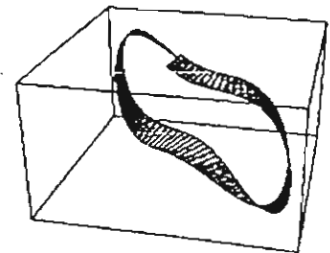
Figure 2. (a) 3D views of particle trajectory shows that channels, which the particle follows, do not necessarily lie down on the same plane. Instead these channels are similar to a number of circular rods sticking up and down horizontally to a pole. Figure (b) is the stable periodic orbit for $Q = 0$. Figure (c) shows the time series view for x , y , and z variable while (d) is the 3D construct of (c) which indicates true periodic nature of this orbit. Figure (e) is a typical phase space section for $Q = 0$ where the horizontal axis is x and the vertical axis is p . The section is taken at $y = 0$.



(a)



(b)



(c)

Figure 3. (a) 3D visualization in a rectangular coordinate of a particle trajectory for $Q = 0.6$. The plane where the trajectory lies is defined by $x = \pm y$. Figure (b) shows a torus construct of the same trajectory in phase space (x, y, p) and (c) shows a band construct of two nearby trajectories where the twist along the band can be clearly seen.

การศึกษาพฤติกรรม global ของอนุภาคในสนามศักย์ $(Q-xy)^2$ ที่ขึ้นกับค่า Q

Q DEPENDENCE OF GLOBAL PARTICLE BEHAVIOR IN $(Q-xy)^2$ POTENTIAL

กฤษณะ จารุณุตาสินี¹ และ จอร์จ โรว์แลนด์²

Krisanadej Jaroenutassinee¹ and George Rowlands²

¹Institute of Science, Walailak University, Tasala, NakhonSi Thammarat 80160, Thailand e-mail jkrisana@wvu.ac.th

²Department of Physics, University of Warwick, Coventry CV4 7AL, England

บทคัดย่อ: ให้ศึกษาพฤติกรรม global ของอนุภาคในสนามศักย์ที่อยู่ในรูป $(Q-xy)^2$ ที่ขึ้นกับค่า Q โดยใช้วิธีการวิเคราะห์ multiple scale perturbation เราพบว่าพฤติกรรมของระบบ global ขึ้นกับค่าใน Fermi map มากขึ้นเมื่อ Q น้อยลง

Abstract: How global particle behavior in $(Q-xy)^2$ potential depending on Q is investigated by the multiple scale perturbation analysis. We found that the global chaotic behavior occupy larger region in the Fermi map when Q gets smaller.

Methodology:

The Hamiltonian describing the motion in $(Q-xy)^2$ potential is $H(x, y, p, q) = 1/2(p^2 + q^2 + (Q-xy)^2)$. Once knowing the Hamiltonian, the equations of motion can be obtained easily and they are ordinary differential equations. However, the analytic solution to this set of differential equations is not as easy to obtain. Hence, it is difficult to understand the global behavior. Multiple Scale Perturbation analysis is applied to study this global behavior. To apply the multiple scale perturbation, the equations of motion are transformed to the hyperbolic coordinate system. And the global dynamic can be studied in terms of the first adiabatic invariant - μ magnetic moment. This dynamic is captured in this mapping equations $\mu_{n+1} = \mu_n + g(\mu_n) \sin \theta_n$ and $\theta_{n+1} = \theta_n + d(\mu_{n+1})$ where θ is the Larmor radius when the particle crosses the minimum magnetic field plane while g and d involve complicated integral related to motion along the guiding center. This guiding center lies on the hyperbolic curve $xy = Q$.

Results, Discussion and Conclusions:

Global behavior of the particle for $Q = 3, 4$ is depicted in Fig 1 and 2. This is typical behavior where the motion for high μ is regular (lines and islands). As μ decreases to normalized value of $K > 4$ where μ is called μ_{res} when the normalized map becomes the standard map, the behavior becomes completely chaotic and μ can drift down to a very small value which normally can take infinite time to trace the trajectory. In Fig 3 μ_{res} moves up to the value 0.5 when Q is approaching 1. This means that the global chaotic behavior occupy the larger area in the map. In other word, particle motion is more regular when Q is large.

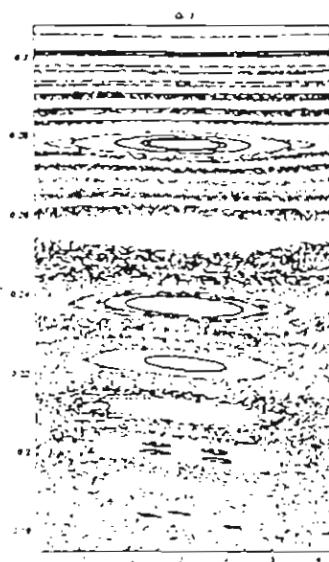


Fig 1 Global Behaviour of Particle Motion

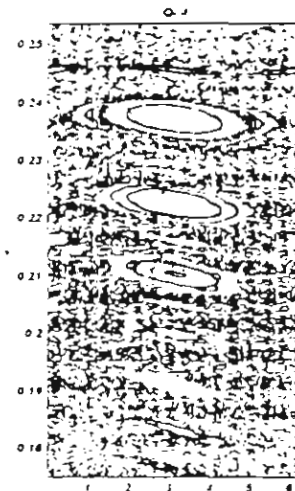


Fig 2 The Fermi map for $Q=4$

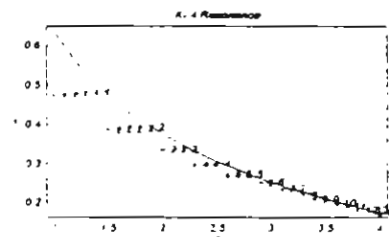


Fig 3 Resonance positions (Dots) vs Q . These dots are special because particle behaves chaotically for the value μ of less than this. The number labels beside the dots are the resonance number associated with each of the dots.

- References:** (1) A J Lichtenberg and M A Lieberman 1983 *Regular and Stochastic Motion* (New York: Springer-Verlag)
 (2) B V Chirikov 1979 *Phys. Reports* 52 p 265
 (3) J E Howard 1971 *Phys. Fluids* 14 p 2373
 (4) K Jaroenutassinee and G Rowlands 2000 *J. Plasma Phys.* 63 (3) p 253

Keywords: Fermi-map, Standard Map, Chaos, Multiple Scale Perturbation, Chaotic Behavior, Global Behavior

Cluster Computing with Mathematica

Krisanadej Jaroensutasinee

Institute of Science, Walailak University, Tasala, NakhonSiThammarat 80160 Thailand
jkrisana@wu.ac.th, Tel. +66075 672005-6 Fax. +66075 672004

Abstract

Parallel computing performance of Mathematica for a simple problem is tested on a cluster of 6 866 MHz Pentium III XEON workstations. Overhead time is computed and compared. The computing tasks are to numerically generate Standard Map and represents bincare Surface of Section of 2D chaotic system. It is found that Mathematica provides sufficient tools to perform parallel tasks such as managing parallel process and shared memory. Cluster computing is found to be faster but efficiency and speedup factors are reversed when the number of nodes is more than 2. The optimum number of nodes for this problem is 3.

1 Introduction

Most parallel computation carried on a cluster nowadays uses MPI which stands for Message Passing Interface, or PVM (Parallel Virtual Machines). MPI works on message passing basis (see [1] for example) and programmers are provided with a low-level communication library such as MPI_Bcast in order to communicate between nodes. However, MPI is not the only option for today parallel and cluster computation. Mathematica [2] offers an alternative.

Although Mathematica appears to be a commercial package, Mathematica is also both a programming language and a computing environment. Mathematica provides programmers with many programming styles such as the procedural style like C, the functional programming style like Lisp and also rule-based programming style like Prolog [3]. Nevertheless, fine-tuning to maximize Mathematica performance is a delicate task and programmers need to master the functional style of programming.

Mathematica operates on the idea of a front end and kernel basis where communication between front end and kernel rests upon a special communication protocol called MathLink. This protocol is a high level communication protocol for data exchange and can be used to connect multiple kernels of Mathematica running parallel on many machines or many processors on a single SMP machine to

fully take all power of available central processing units. Moreover, with its Parallel Computing Toolkit[4] where high level job distribution and queuing functions are provided, programmers can use Mathematica to perform parallel and cluster computation over SMP computers and a group of computers networked together. This work aims at testing the performance of the Mathematica parallel environment. The details of the hardware of our cluster will be presented in the following section.



Figure 1. Front end and kernel communication via MathLink in a Mathematica session.

2. The Designing Principle of Mathematica

Normally, Mathematica operates on the front end and kernel concept. This makes Mathematica immediately available on a cluster. Multiple kernels can run separately on each machine and all of the kernels connect to a master kernel, which, of course, also connects to a front end. Parallel and distributed application can be developed right away with the unified concept of Mathematica programming.

Moreover, Parallel Computing Toolkit instantly brings parallel computation to anybody having access to more than one computer on a network. It implements many parallel programming primitives, and includes high-level commands for parallel execution of operations such as animation, plotting and matrix manipulation. The toolkit also supports many current popular programming approaches such as parallel Monte Carlo simulation, visualization, searching and optimization. The implementations for all high-level commands in the Parallel Computing Toolkit are provided in Mathematica source form, and serve as templates for building additional parallel programs.

3. Platforms

The parallel computing toolkit on a cluster of 6 high performance workstation is tested. Each node is a Compaq model SP750 that has an Intel Pentium III XEON CPU, 66 MHz, Intel 840 chipset with 133-MHz front side bus, dual memory channels, dual-peer PCI buses, and 64-bit PCI with 18 GB Ultra SCSI-3 hard disk. It also has a memory of ECC PC800 RDRAM 256 MB with built-in Intel Pro 100+ network card. The master kernel runs on the node so there are 5 nodes left for parallelization. All these machines operate on Windows 2000 Professional. There is a plan to install Linux on these machines in the near future. All machines are linked up via 10/100 mbits LAN with a single normal 100 mbits hub. Connection is done via TCP protocol to link all the kernels with specific ports generated on-the-fly by starting Mathematica kernel in the Mathlink mode.

On the Windows platform, one can use a commercial remote shell daemon to start remote kernels automatically. Additionally, Mathematica allows the so-called passive mode to start the remote kernels. Once connected with the control kernel, these kernels are called "leaves". The parallel toolkit is loaded in the control kernel residing on the control machine of the cluster. The front end, which provides the cluster interface to the user, is connected to the control kernel locally.

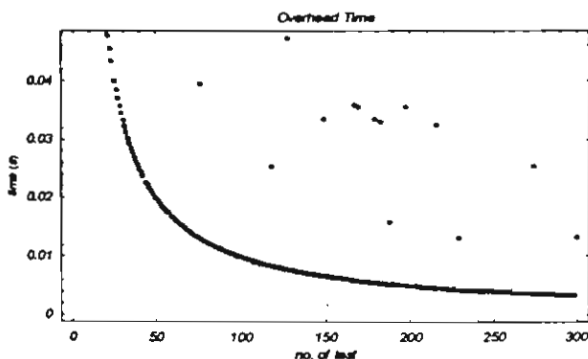


Figure 2. Variation of the overhead time of kernel to control kernel to service one trivial process. The overhead time approaches 0.003 s. The noisy data above the line is caused by connecting normal data resulted from the main computing process interrupted by background jobs.

4. Performance Measurement

Before testing our cluster with a real problem, a trivial task measuring the overhead time of the cluster is performed. Overhead time is defined as the time to service one trivial process. Listing 1 lists a section of Mathematica code that measures this overhead time. Results can be seen in Fig. 2 and it was found that the overhead time of the cluster was 0.003 seconds.

Listing 1. Mathematica code for computing overhead time of 200 number of tries.

```
With[{n=200,t=AbsoluteTime[]},
  ParallelMap[Identity,Range[n]]; (AbsoluteTime[]-
  t)/n
]
```

5. Test Problem

The application of cluster computers is generally expected to require intensive computing resources in terms of local memory, shared-memory and computing time, but in this research a linear test case is selected. The term "linear" in this case means the computing time increases linearly with the size of the problem. The main reason for choosing a linear case is that our cluster efficiency can be traced more accurately. Nonlinear cases can obscure non-linearity features of our cluster and hence key improvements cannot be specified clearly.

There are a number of linear problems that one can use. We choose the standard map:

$$\begin{aligned} p_{n+1} &= p_n + K \sin q_n \\ q_{n+1} &= (q_n + p_{n+1}) \bmod (2\pi) \end{aligned} \quad (1)$$

to test the performance of our cluster. p and q are normalized real variables related to the momentum and position accordingly. This map is a linearization around a resonance which provides a standard version of Poincare Surface Section generated by collecting cutting points of a trajectory in phase space with a specific plane. The trajectory is usually generated by solving nonlinear differential equations.

The domain for p and q , that is $[0, 2\pi] \times [0, 2\pi]$, is divided into the number of nodes and lets each node produce the result for the required iteration. K in the equation is a constant. Without a loss of generality, one can keep $K = 0.9$, which is the value for the map at the starting point of becoming globally chaotic. `ParallelMap[]` is the function to distribute our parallel computing. The function is one of the many functions provided by the parallel computing toolkit.

6. Results

It was found that the calculation time varies with the size of the problem linearly for all 5 cluster configurations. The slopes of the fitting lines to the data are important here because these slopes are used to calculate speedup and efficiency factors of our clusters. The speedup factor is the ratio of sequential time to total parallel execution time, while the efficiency factor is the ratio of the observed speedup to the number of processors. Numerical results are shown in Table 1.

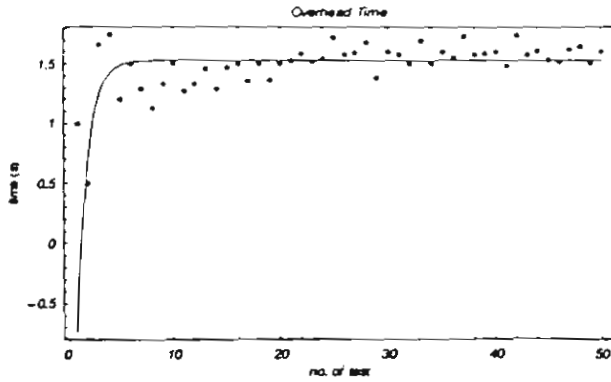


Figure 3. Variation of overhead time with a number of tests of a kernel-to-remote-kernel configuration. The overhead time approaches 1.53 seconds. The curve is described by the equation $y(x)=1.53-6.14 \text{ Exp}(-x)$.

7. Conclusions

It was found that although most cluster operating systems are Linux, Windows clusters do have their places in cluster computing. First of all, all these machines do not need to be "reserved" to be used only for cluster computing. Converting machines to Linux often reduces the size of the software pool that one can use. Secondly, Windows usually comes pre-installed, therefore, there is not much time needed to customize all hardware components before cluster computing can start. Thirdly, there is the availability issue. There are always a number of idle high performance computers such as Pentium III around. It is, therefore, fruitful if cluster approach is not too strict on one operating system and one limited environment. An open approach such as the Java environment is one of the most attractive cluster environments [6], while this work has shown that Mathematica is another one. Although Mathematica belongs to a commercial firm, its notebooks and files are open. One can easily open them in any text editors and can understand their structure very quickly. Mathematica link to Java is also available via Jlink[5].

Table 1. Speedup and efficiency factors of various cluster configurations.

Cluster configuration	Slope (s)	Speedup	Efficiency
1 node	0.000327	1	1
2 nodes	0.000195	1.67495	0.837476
3 nodes	0.000130	2.50847	0.836157
4 nodes	0.000162	2.02508	0.506269
5 nodes	0.000181	1.80862	0.361723

From Table 1, it can be concluded that cluster computing with Mathematica is an effective environment. It is generally faster even though the speedup is not at the factor of the number of nodes used. This is due to the fact that there is always time used in communication among nodes and with the master kernel. The allocation of jobs

for each node plays an important role here. As our results indicate, if the more nodes the task splits into, the more communication tasks are needed, and so the less efficient the cluster will be. Future work will include testing the virtual shared memory and packed array[5]. In addition, the number of nodes of our cluster will be increased and it will be possible to further explore different cluster models which are based on how all nodes are connected.

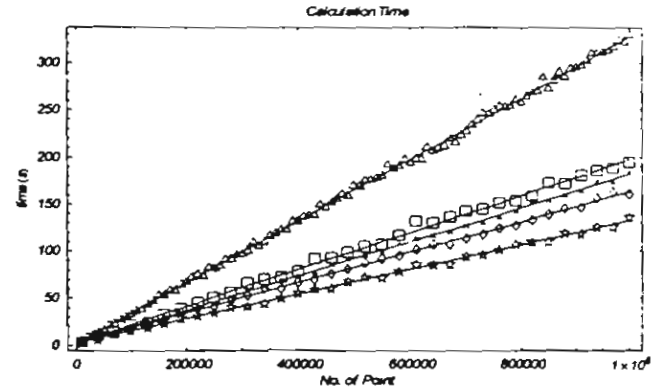


Figure 4. Computing time of the test problem for various cluster configurations. The triangular symbols are for 1 node, the box symbols are for 2 nodes, the diamond, star and cross symbols are for 3, 4 and 5 nodes respectively.

8. Acknowledgement

The author would like to thank the Thailand Research Fund, Post-Doctoral Grant (PDF/62/2540), for supporting this work.

9. References

- [1] J.S. Shang, M. Wagner, Y. Pan and D.C. Blake, *Comp. Sci. Eng.*, Jan/Feb 2000 p 11.
- [2] S. Wolfram, *The Mathematica Book*, CUP and Wolfram Media, 1999.
- [3] D.B. Wagner, *Power Programming with Mathematica*, Computing McGraw-Hill, 1996.
- [4] Wolfram Research Inc., *Parallel Computing Toolkit*, <http://www.wolfram.com/products/applications/parallel/>, 2001.
- [5] Wolfram Research Technical Staff, *The Mathematical Jour.* Vol. 8 No. 1, 2001, p 63.
- [6] R. Buyya, *High Performance Cluster Computing*, Vol. 1 and 2, Prentice Hall, 1999.

Functional programming implementation of the 4th order Runge-Kutta method

Asst.Prof.Dr. Krisanadej Jaroensutasinee¹ and Prof.Dr. George Rowlands²

¹Institute of Science, Walailak University,
Nakhon Si Thammarat 80160 Thailand
jkrisana@wu.ac.th

²Physics Department, Warwick University,
Coventry CV4 7AL England

Abstract

Mathematica computing environment allows a user to implement an algorithm in virtually all style of programming, however, only certain specific styles of programming can utilize the power of *Mathematica* fully. In this work, we implemented the 4th order Runge-Kutta method in *Mathematica* environment with various programming styles such as procedural, functional, and rule-based and tested their performance. We found that the functional programming implementation of the 4th order Runge-Kutta method worked best and the size of the code was the smallest. We examined these implementations with a non-linear dynamical system. These results are foundations for the following most efficient implementation of the Poincaré surface of section algorithm.

1 Introduction

1.1 *Mathematica* computing environment

Mathematica is a technical computing environment that is widely used today. *Mathematica* incorporates features from all major programming styles[1]. In a way, this is an advantage, however, *Mathematica* cannot provide its full computing power for all programming styles. Hence, choosing proper programming styles for the task is a very important issue.

Moreover, *Mathematica* is essentially an expert system. Failure to realize this fact always results in inefficient implementation of programs in this technical computing environment. For example, Procedural Programming style (PP) extensively uses assignment constructs. PP style assignment constructs involve fundamental data types e.g. integer, byte, etc. In a for loop, such as for $i = 1$ to 10, a simple counter, say i , can be just a register in the processor resulting in extremely fast access when updating its value in the assignment construct.

In contrast, due to the sophisticated nature of the requirements of an expert system, an assignment in *Mathematica* normally requires a great deal of computing resources because the assignment construct can possibly involve sophisticated abstractions e.g. symbols, symbolic

expressions, rule constructions, and/or infinite accuracy numbers. Therefore, resulting internal codes, that are as efficient as those compiled with PP languages, cannot always be guaranteed. Nevertheless, *Mathematica* provides a way to reach similar (at the same time produces more elegant and high level of abstraction source codes) internal codes through FP constructs such as Map[], Nest[], etc. Apart from the assignment problem, there is a number of widely used PP implementations that can significantly slow down *Mathematica* such as adding a record of data to a large list inside a loop construct[2].

1.2 Numerical solutions for ordinary differential equations (ODEs)

One of the most popular applications in *Mathematica* is solving ODEs numerically with various styles of programming (see [1] and [3] for example), however, discussion on the performance and practicality remains an issue.

Numerical ordinary differential equations integrators such as Runge-Kutta scheme or others are very important basis for higher level diagnostic tools, for example, Poincaré surface of section (PSS) method that is an important tool to classify motions in nonlinear dynamical systems, especially for 2D Hamiltonian systems. This study aimed at identifying the most efficient implementation of the 4th order Runge-Kutta method so that higher level functions could be derived from this most optimized resulting code.

Normally Runge-Kutta method is coded in the Procedural programming (PP) style. PP languages originate directly from assembly language. Procedural programming style (PP) can be noticed from extensive use of PP constructs such as assignments, loops, conditional statements, and subroutines. All of these PP constructs can be translated to assembly language easily. It is of important to note that FP is generally the least familiar programming style to most scientists, because FP often neglected in traditional Science education. As a result, most scientists normally use PP style to create their programs in *Mathematica* environment even though PP is inefficient in this environment.

Another different but more sophisticated approach in programming style is a so-called Functional programming (FP) style originated from LISP. LISP and FP languages are used mainly in the field of artificial intelligence and in symbolic computation. There is also another programming style that worth mentioning in this article. It is a declarative programming style such as Prolog. In *Mathematica*, this programming style is widely known as the rule-based programming style.

2 Types of Numerical Solutions for ODEs in *Mathematica*

2.1 Internal NDSolve[] Function

Mathematica internal function for ODEs NDSolve[] is a very powerful function. With the option Method is set to automatic, NDSolve[] can switch between a non-stiff Adams method and a stiff Gear method. Runge-Kutta-Fehlberg order 4–5 Runge-Kutta method can be employed for non-stiff equations. The code for NDSolve[] is very sophisticated, but efficient and according to *Mathematica* manual it is about 500 pages long.

2.2 Dynamic Programming Approach

In order to speed up calculation in *Mathematica*, one can use dynamic programming capability of the computing environment. Combinations of `Set[]` (symbol “=”) and `SetDelayed[]` (symbol “:=”) are used extensively and thus can be seen explicitly in the code such as in Table 1. [4]

Table 1 A section of an example of dynamic programming coding of the 4th Runge-Kutta method.

```
k1[n_] := k1[n] = f[t[n], x[n], y[n]];
k2[n_] := k2[n] = f[t[n] +  $\frac{h}{2}$ , x[n] +  $\frac{h k1[n]}{2}$ , y[n] +  $\frac{h m1[n]}{2}$ ];
k3[n_] := k3[n] = f[t[n] +  $\frac{h}{2}$ , x[n] +  $\frac{h k2[n]}{2}$ , y[n] +  $\frac{h m2[n]}{2}$ ];
k4[n_] := k4[n] = f[t[n] + h, x[n] + h k3[n], y[n] + h m3[n]];
m1[n_] := m1[n] = g[t[n], x[n], y[n]];
m2[n_] := m2[n] = g[t[n] +  $\frac{h}{2}$ , x[n] +  $\frac{h k1[n]}{2}$ , y[n] +  $\frac{h m1[n]}{2}$ ];
m3[n_] := m3[n] = g[t[n] +  $\frac{h}{2}$ , x[n] +  $\frac{h k2[n]}{2}$ , y[n] +  $\frac{h m2[n]}{2}$ ];
m4[n_] := m4[n] = g[t[n] + h, x[n] + h k3[n], y[n] + h m3[n];
```

This approach exchanges the speed with the extensive use of memory. This is, hence, not practical where a large list of result is required.

2.3 Runge-Kutta integrators in MathSource

There are examples of implementation of Runge-Kutta algorithm for *Mathematica* in the extensive knowledge base known as MathSource website (<http://www.mathsource.com/>). Runge-Kutta package[5] and Runge-Kutta-Nystrom integrator[6] are such examples. In this work we compared the computing time of Runge-Kutta package which is written by Maeder [5] with our implementations.

Table 2 An implementation of Runge-Kutta method from MathSource[5].

```

BeginPackage["RMPackages`RMRungeKutta`"]

RMRungeKutta::usage = "RungeKutta[{e1,e2,...}, {y1,y2,...}, {a1,a2,...}, {t1, dt}]
numerically integrates the ei as functions of the yi with initial values ai.
The integration proceeds in steps of dt from 0 to t1.
RungeKutta[{e1,e2,...}, {y1,y2,...}, {a1,a2,...}, {t, t0, t1, dt}] integrates
a time-dependent system from t0 to t1."

Begin["`Private`"]
RKStep[f_, y_, y0_, dt_] :=
Block[{ k1, k2, k3, k4 },
  k1 = dt N[ f /. Thread[y -> y0] ];
  k2 = dt N[ f /. Thread[y -> y0 + k1/2] ];
  k3 = dt N[ f /. Thread[y -> y0 + k2/2] ];
  k4 = dt N[ f /. Thread[y -> y0 + k3] ];
  y0 + (k1 + 2 k2 + 2 k3 + k4)/6
]
RMRungeKutta[f_List, y_List, y0_List, {t1_, dt_}] :=
NestList[ RKStep[f, y, #, N[dt]]&, N[y0], Round[N[t1/dt]] ] /;
Length[f] == Length[y] == Length[y0]

RMRungeKutta[f_List, y_List, y0_List, {t_, t0_, t1_, dt_}] :=
Block[{res},
  res = RMRungeKutta[ Append[f, 1], Append[y, t], Append[y0, t0], {t1 - t0, dt} ]
  (* Drop[#, -1]& /@ res - Do not drop the time from the list *)
] /; Length[f] == Length[y] == Length[y0]

End[]
Protect[RMRungeKutta]
EndPackage[]

```

3 Lorenz Attractor

Lorenz attractors can be constructed by solving the Lorenz equations.

$$\frac{dx}{dt} = -3(x - y), \quad \frac{dy}{dt} = -xz + \alpha x - y, \quad \frac{dz}{dt} = xy - z.$$

This model arises in a highly simplified model of a convecting fluid and was first introduced by Edward Lorenz. This model involves in modeling convection in the atmosphere and is used to demonstrate a very simple set of equations that yields a highly complicated behavior including chaos[7]. In this work we chose $\alpha = 26.5$.

In Table 3, Code #1 implemented the problem definition in terms of function f[] while rk[] implemented Runge-Kutta step (similar to RKStep[] in Table 1). Code #2 implemented the function to interpolate grid points and return InterpolatingFunction[]. Code #3 and #4 were examples of possible possible FP implementations of Runge-Kutta method. Loop construct was implemented via Table[] and NestList[] calls. Finally, Code #5 and Code #6 were PP constructs. The difference of these 2 codes was that Code #6 pre-allocated memory for output while Code #5 retained the flexibility of expanding the size of the list as needed.

Table 3 Various implementations of the 4th Runge-Kutta method. See text for detail.

Code #	Implementation
1	<pre> alpha = 26.5; t0=0.0; tmax=10.0; dt=0.0001;h=dt;h2=h/2.;h6=h/6.; f[{x_, y_, z_, t_}] := {-3 (x - y), -x z + alpha x - y, x y - z, 1}; rk[x_] := (k1 = f[x]; k2 = f[x + h2 k1]; k3 = f[x + h2 k2]; k4 = f[x + h k3]; x + h6 (k1 + 2. k2 + 2. k3 + k4)) </pre>
2	<pre> LorenzSolution[solPoints_] := Thread[Rule[{x, y, z}, Map[Interpolation, Transpose[Apply[{{#4, #1}, {#4, #2}, {#4, #3}} &, solPoints, {1}]]]]] </pre>
3	<pre> KJ1 := LorenzKJResult1 = Block[{x1}, Timing[x1 = x0; LorenzKJPoint1 = Prepend[Table[x1 = rk[x1], {Round[N[(tmax - t0) / dt]]}], x0]; LorenzKJSolution1 = Thread[Rule[{x, y, z}, Map[Interpolation, Transpose[Apply[{{#4, #1}, {#4, #2}, {#4, #3}} &, LorenzKJPoint1, {1}]]]]]]] </pre>
4	<pre> KJ2 := LorenzKJResult2 = Block[{}, Timing[LorenzKJPoint2 = NestList[rk, x0, Round[N[(tmax - t0) / dt]]]; LorenzKJSolution2 = LorenzSolution[LorenzKJPoint2]]] </pre>
5	<pre> PP1[tmax_] := LorenzPPResult1 = Block[{xs, i, iMax}, Timing[iMax = Round[N[(tmax - t0) / dt]]; xs = x0; LorenzPPPpoints1 = {xs}; For[i = 1, i ≤ iMax, i++, xs = rk[xs]; AppendTo[LorenzPPPpoints1, xs];]; LorenzPPSolution1 = LorenzSolution[LorenzPPPpoints1]]] </pre>

```

6  PP2[tmax_] := LorenzPPResult2 = Block[{xs, i, iMax},
    Timing[
      iMax = Round[N[(tmax - t0) / dt]];
      xs = x0;
      LorenzPPPoints2 = Table[xs, {iMax + 1}];
      LorenzPPPoints2[[1]] = xs;
      For[i = 1, i ≤ iMax, i++,
        xs = rk[xs];
        LorenzPPPoints2[[i + 1]] = xs;
      ];
      LorenzPPSolution2 = LorenzSolution[LorenzPPPoints2]
    ]]
```

4 Results

We tested these implementations on a notebook computer with Pentium III processor running at 866 MHz. The Front side bus speed was 100 MHz with SDRAM of 384 MB. We used *Mathematica* professional version 4.1 on Windows 2000 Professional operating system.

We needed to obtain the solutions not only in terms of grid points of the time interval of 0.0001 from $t = 0.0$ to $t = 10.0$ but also in terms of *Mathematica* `InterpolatingFunction[]` in order to arrive at equivalent quality and quantity of the outputs. This is because `NDSolve[]` always produces `InterpolatingFunction[]` as its output while the other methods normally produce grid points of equal interval. Table 4 shows the result of our test. For each implementation, we ran 3 tests to ensure that background jobs did not significantly interfere with our tests.

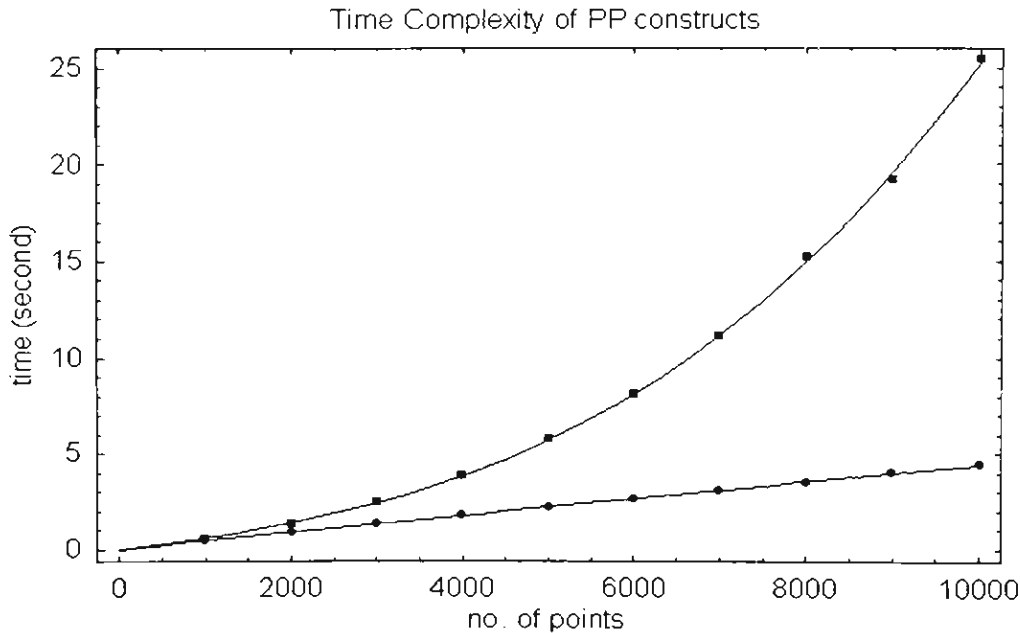


Figure 1 Different time complexity of PP constructs. Squares and Dots are correspond to Code #5 and Code #6 of Table 3 respectively.

Table 4 Computing time (for 3 runs) of each implementations.

Method used	Time used (second)
NDSolve[]	8.712, 8.812, 8.852
RM Runge-Kutta Package	74.096, 74.277, 74.507
Our implementation 1 (Table) Code #3 in Table 3	41.690, 41.569, 41.700
Our implementation 2 (NestList) Code #4 in Table 3	41.250, 41.319, 41.320
Our implementation 3 (PP) Code #5 in Table 3	N/A
Our implementation 4 (PP) Code #6 in Table 3	44.204, 44.263, 44.244

5 Conclusion

Table 4 indicates that internal function call to NDSolve[] was clearly the most efficient way to obtain ODEs solutions. Whenever any internal function call was not desirable, FP constructs worked best.

The PP construct (Code #5 in Table 3) took too long when called to computing 100001 grid points. Interpolation of computing time vs grid points with a polynomial function of the order up to x^3 suggested that it would take 16,154.000 second to complete 100,001 grid points. And this was obviously unpractical. However, one can fine tune the PP construct to produce speed of computing at the same level of Table[] and NestList[] constructs. This was achieved by the implementation that takes the advantage of pre-allocated data and the make use of PackedArray[] [10]. PackedArray is an add-on feature to accelerate numerical computing in *Mathematica* by forcing machine precision to numerical objects. Code #6 in Table 3 implements these features, however, with exchange to speed, we lose generality by pre-allocating the data beforehand, hence the size must be known in advance and there will be unnecessary computing time to initialize these pre-allocated data, while the other two FP implementations do not. Moreover, PP construct also has more lines of code than FP ones.

In conclusion, this article encourages the implementation that makes use of internal function calls whenever possible, in this case, NDSolve[]. FP approach then comes as a second choice while PP should be the last way to implement any numerical algorithm in *Mathematica*.

Acknowledgement

This work is supported by TRF grant PDF/62/2540. One of the authors (KJ) would like to thank for the support from the Thailand Research Fund (TRF).

References

- [1] Maeder R. E., Computer Science with *Mathematica*, CUP, 2000.
- [2] Wagner D.B., Power Programming with *Mathematica*, McGraw-Hill, 1996.
- [3] Gray J., Mastering *Mathematica*, 2nd ed., Academic Press, 1997.
- [4] Abell, M.L. and Braselton J.P., Differential equations with *Mathematica*, AP Professional, 1993.
- [5] Maeder R. E., "Runge-Kutta Method", MathSource2.2, <http://www.mathsource.com/Content22/Enhancements/Numerical/0201-362/>.

- [6] Paternoster B. and Stecca A., "RUNGE-KUTTA-NYSTROM integrator", MathSource, <http://www.mathsource.com/Content/Applications/Mathematics/0207-458>.
- [7] Hilborn R.C., Chaos and Nonlinear Dynamics, OUP, 1994.
- [8] Gray A., Mezzino M. and Pinsky M., Introduction to Ordinary Differential Equations with *Mathematica*, Springer-Verlag, TELOS, 1997.
- [9] Jaroensutasinee K. and Rowlands G., "Poincaré Surface of Section Construction by Functional Programming Approach", Proceedings of Computational Mathematics and Modeling, Mahidol University, Thailand, 2002, accepted for publication.
- [10] Knapp R., "Packed Arrays", *Math. Jour.* 2000, 7 (4), pp 542-558.

Poincaré Surface of Section Construction by Functional Programming Approach

K. Jaroensutasinee

*Institute of Science, Walailak University, NakhonSiThammarat 80160 Thailand
and*

G. Rowlands

Physics Department, University of Warwick, Coventry CV4 7AL England

ABSTRACT

Poincaré Surface of Section (PSS) is a very important tool for studying nonlinear behavior of a 2D Hamiltonian system. In general, it is constructed by numerical integration of the corresponding differential equations that describe the motion in the phase space. The numerical integration is usually coded in a Procedural Programming (PP) approach in a normal programming environment such as C++ or Java. In this work, we approach the problem in a different way by using Functional Programming approach (FP). FP is intrinsically a must in a powerful computing environment such as Lisp, *Mathematica* and Prolog. Constructing PSS by using PP is extremely popular but highly impractical in such an environment, while FP is much more appropriate and at the same time keeps the speed of computing as fast as possible. Our results indicate that FP shows a very significant speed improvement over PP.

KEY WORDS – Poincaré Surface of Section Calculation, Functional Programming, *Mathematica*

1. Introduction

Poincaré surface of section (PSS) method is an important tool to classify motions in nonlinear dynamical systems, especially for 2D Hamiltonian systems. Generally PSS method requires extensive use of numerical ordinary differential equations integrators such as Runge-Kutta scheme or others. Choosing the right implementation of the integrator is the key to obtain quality PSS as of equal importance is to obtain the right implementation of the PSS method itself. It is the purpose of this study to identify the right implementation of PSS method which can be used to study nonlinear dynamical systems in *Mathematica* environment so that *Mathematica* parallel computation and powerful visualization capabilities can be applied promptly.

Normally PSS method is coded in the Procedural programming (PP) style. PP languages originate directly from assembly language. Procedural programming style (PP) can be noticed from extensive use of PP constructs such as assignments, loops, conditional statements, and subroutines. All of these PP constructs can be translated to assembly language easily. Another different but more sophisticated approach in programming style is a so-called Functional programming (FP) style originated from LISP. LISP and FP languages are used mainly in the field of artificial intelligence and in symbolic computation. There is also another programming style that worth mentioning in this article. It is a declarative programming style such as Prolog. In *Mathematica*, this programming style is widely known as the rule-based programming style.

Mathematica is a technical computing environment that is widely used today. *Mathematica* incorporates features from all major programming styles (see for example Maeder 2000 and Gray 1997). In a way, this is an advantage, however, *Mathematica* cannot provide its full computing power for all programming style. Hence, choosing proper programming styles for the task is an important issue. It is of important to note that PP is generally the most familiar programming style to most scientists because FP often neglected in traditional Science education. As a result, most scientists program with PP style in *Mathematica* environment even though PP is inefficient in this environment. This work will show the most efficiency FP algorithms for PSS method in *Mathematica* environment.

2. Poincaré Surface of Section Method via Functional Programming

Mathematica is essentially an expert system. Failure to realize this fact always results in inefficient implementation of programs in this technical computing environment. For example, PP style extensively uses assignment constructs. In PP languages, PP style assignment constructs involve fundamental data types e.g. integer, byte, etc. For example, in a for loop, such as for $i = 1$ to 10, a simple counter, say i , can be just a register in the processor resulting in extremely fast access when updating its value in the assignment construct.

In contrast, due to the sophisticated expert system nature, an assignment in *Mathematica* normally requires a great deal of computing resource because the assignment construct can involve sophisticated abstractions e.g. symbols, symbolic expressions, rule constructions, and/or infinite accuracy numbers. Therefore, resulting internal codes, that are as efficient as those compiled with PP languages, cannot be guaranteed. Nevertheless, *Mathematica* provides a way to reach similar (at the same time produces more elegant and high level of abstraction source codes) internal codes through FP constructs such as Map, Nest, etc.

Apart from the previous example, there is a number of inefficient PP that can significantly slow down *Mathematica* such as adding a record of data to a large list inside a loop construct (see for example Wagner 1996). There are also a number of work done in applying *Mathematica* to solve ODEs numerically with various styles of programming (see for example Abell and Braselton 1993 and Gray, Mezzino and Pinsky 1997), however, discussion on the performance and practicality remains an issue. Jaroensutasinee and Rowlands (2002) shows that PP constructs of Runge-Kutta integrator is slowest while the use of internal ODE integrator, `NDSolve[]`, is the fastest and a simple combination of rule-based and functional construct falls between. In this work, after ODEs are solved simultaneously, PSS methods are implemented in FP constructs. `NestWhile[]` is used instead of `While[]` loop and the conditions for cutting the section are implemented through rules construction (`PSScond[]` rules in the following examples) instead of using a side effect via inefficient variable assignments and calling `If[]`.

2.1 The First Example: The Lorenz Attractor

Lorenz attractors can be constructed by solving the Lorenz equations

$$\frac{dx}{dt} = -3(x - y), \quad \frac{dy}{dt} = -xz + \alpha x - y, \quad \frac{dz}{dt} = xy - z.$$

In this work we chose $\alpha = 26.5$ and the section was chosen at $y \equiv 0$. Table 1 shows the source codes for this system.

Table 1. FP style codes for PSS calculation for Lorenz system.

Code #	Implementation
1	<pre>f[{x_, y_, z_, t_}] := {-3 (x - y), -x z + alpha x - y, x y - z, 1}; rk[{_, x_}] := {x, (k1 = f[x]; k2 = f[x + h2 k1]; k3 = f[x + h2 k2]; k4 = f[x + h k3]; x + h6 (k1 + 2. k2 + 2. k3 + k4))}</pre>
2	<pre>Remove[PSScond]; PSScond[{(x1_, y1_, z1_, t1_), (x2_, y2_, z2_, t2_)}] := (True) /; ((y1 > y2) && ((y1 y2) < 0.0)); PSScond[{t_, _}] := False;</pre>
3	<pre>pssFunc[x1_] := NestWhile[rk, x1, (! PSScond[#2]) &, 2, 100000]</pre>
4	<pre>Timing[pss1 = NestWhileList[pssFunc[#] &, {x0, x0}, (#[[2]][[4]] < tmax) &, 1, 20];]</pre>
5	<pre>LorenzNDSolveResult = Timing[LorenzNDSolveSolution = NDSolve[{x'[t] == -3.0 (x[t] - y[t]), y'[t] == -x[t] z[t] + alpha x[t] - y[t], z'[t] == x[t] y[t] - z[t], x[0] == 0.0, y[0] == 1.0, z[0] == 0.0}, {x, y, z}, {t, t0, tmax}, MaxSteps -> Infinity, MaxStepSize -> dt, Method -> RungeKutta][[1]]]</pre>

6	<code>sol = y /. LorenzNDSolveSolution</code>
7	<code>fullSol = Append#[[2]] [t] & /@ Flatten[LorenzNDSolveSolution], t]</code>
	<code>pssFuncND[t_] := NestWhile[#+ dt) &, t, (! ((sol[#1] > sol[#2]) && (sol[#1] sol[#2] < 0.0)) && (#2 <= tmax - dt)) &, 2, 100000]</code>

2.2 The Second Example: The CUSP magnetic field

The motion of a charged particle in a CUSP magnetic field can be generated by the following equations of motion:

$$\begin{aligned}\frac{dx}{dt} &= p, & \frac{dy}{dt} &= q, \\ \frac{dp}{dt} &= y(Q - xy), & \frac{dq}{dt} &= x(Q - xy),\end{aligned}$$

where (x, y) is the position of the particle in the rectangular coordinate and (p, q) are the corresponding momentum in both directions. Q is a constant related to the z direction momentum which is conserved during the motion. In this work we chose $Q = 0$ and the section was chosen at $y \equiv 0$. Table 2 shows the source codes for this system.

Table 2. FP style codes for PSS calculation for CUSP system.

Code #	Implementation
1	<code>NDSolveResult = Timing[NDSolveSolution = NDSolve[{ x'[t] == p[t], p'[t] == y[t] (Q - x[t] y[t]), y'[t] == q[t], q'[t] == x[t] (Q - x[t] y[t]), x[0.0] == xIn, y[0.0] == yIn, p[0.0] == pIn, q[0.0] == qIn}, {x, y, p, q}, {t, t0, tmax}, MaxSteps -> Infinity, MaxStepSize -> dt, Method -> RungeKutta] [[1]]];</code>
2	<code>Remove[PSScond]; PSScond[{{x1_, y1_, p1_, q1_, t1_}, {x2_, y2_, p2_, q2_, t2_}}] := (True) /; ((y1 > y2) && ((y1 y2) < 0.0)); PSScond[{t_, _}] := False;</code>
3	<code>sol = y /. NDSolveSolution; fullSol = Append#[[2]] [t] & /@ Flatten[NDSolveSolution], t]; pssFuncND[t_] := NestWhile[#+ dt) &, t, (! ((sol[#1] > sol[#2]) && (sol[#1] sol[#2] < 0.0)) && (#2 <= tmax - dt)) &, 2, 100000];</code>
4	<code>pssND = Drop[Drop[NestWhileList[(pssFuncND[#]) &, t0, (# <= tmax - dt) &, 1, 1000000], -1], 1]; pssND = Transpose[fullSol /. t -> pssND];</code>

3. Results

We tested the implementations on a notebook computer with Pentium III processor running at 866 MHz. The Front side bus speed was 100 MHz with 384 MB. We used *Mathematica* professional version 4.1 on Windows 2000 Professional operating system.

For Lorenz system, the computing time used to calculate PSS for t starts from 0 to 10.0 was 47.007 second for the integrator proposed in Jaroensutasinee and Rowlands (2002), that is FP constructs for Runge-

Kutta method (rk[_,_]) in Table 1). Results for NDSolve[] technique were more impressive. It takes only 6.029 second which was the total time of 3.395 second for PSS search plus 2.634 second for NDSolve[] itself.

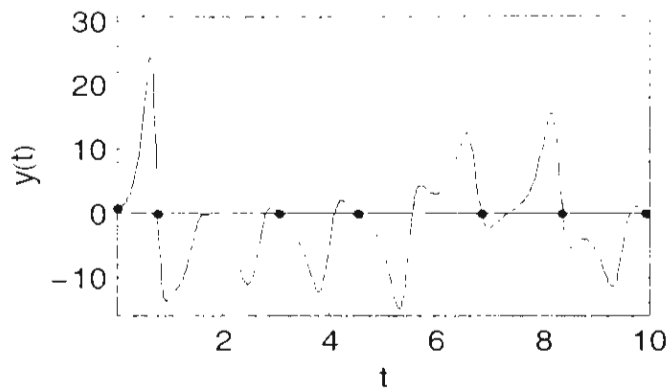


Figure 1. Time series of $y(t)$ obtained from solving Lorenz system with cuts (red lines and dots) from PSS computation.

And for CUSP magnetic field case, time used to calculate PSS for t starts from 0 to 100.0 was 1.623 second. More interesting results were (1) the computing time was 385.655 second for 7316 cuts and (2) the time was 674.189 second for 12249 cuts. The last two examples signify that this implementation is of practical use, however, the methods shown in Table 2 were modified to the practical level by dividing the maximum time into smaller chunks for better accuracy in NDSolve[].

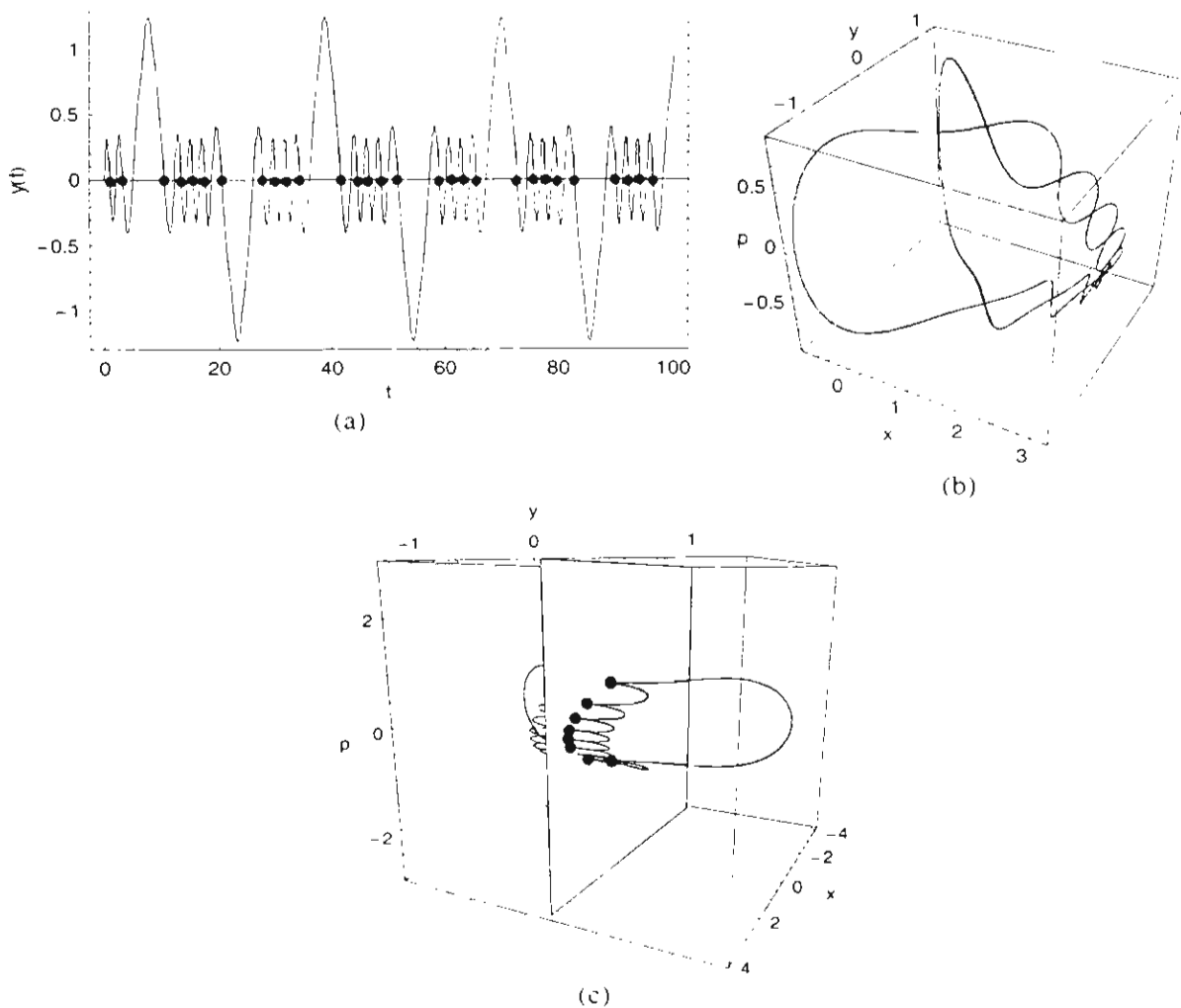


Figure 2. (a) Time series $y(t)$ obtained from solving CUSP system with cuts (red lines and dots) from PSS computation. (b) 3D visualization of the trajectory in (x, y, p) showing its periodicity. (c) 3D visualization showing PSS cuts, the PSS plane and the trajectory.

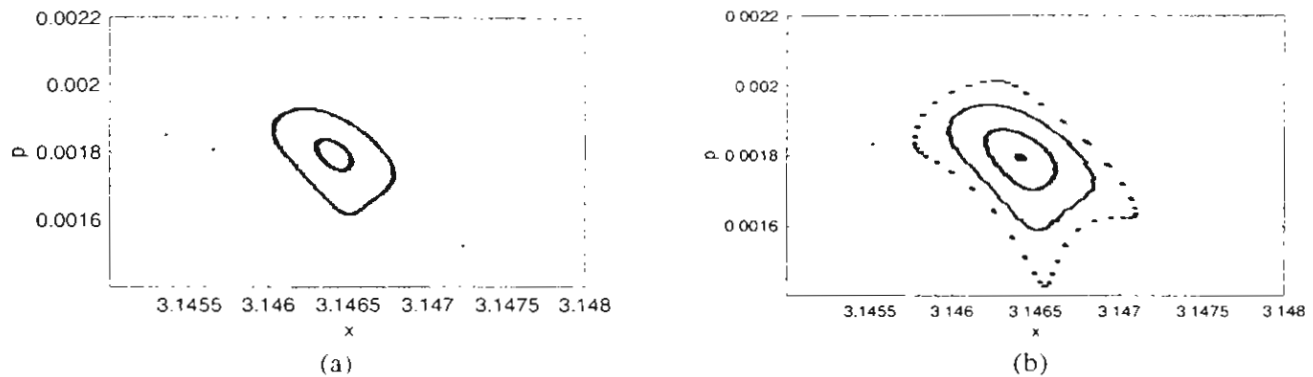


Figure 3 Poicaré Surface of Section. (a) and (b) illustrate quasi-periodicity of the trajectory for different initial conditions. This quasi-periodicity is only revealed by computing PSS cuts many hundred times to form the island structure (each island corresponds to one initial conditions).

4. Conclusion

Combination of a series of internal function call to `NDSolve[]` with FP style coding to collect PSS points can generate PSS results of practical use. Only slight modification of the codes can bring them to parallel computing environment via Parallel Computing Toolkit and visualization of PSS cuts can be accomplished promptly as shown in Figure 2 and 3.

Acknowledgement

This work is supported by TRF grant PDF/62/2540. One of the authors (KJ) would like to thank for the support from the Thailand Research Fund (TRF).

References

- [1] Abell, M.L. and Braselton J.P., Differential equations with *Mathematica*, AP Professional, 1993.
- [2] Gray A., Mezzino M. and Pinsky M., Introduction to Ordinary Differential Equations with *Mathematica*, Springer-Verlag, TELOS, 1997.
- [3] Gray J., Mastering *Mathematica*, 2nd ed., Academic Press, 1997.
- [4] Jaroensutasinee K. and Rowlands G., *Functional programming implementation of the 4th order Runge-Kutta method*, Proceedings of the 6th Annual Symposium on Computational Science and Engineering, Walailak University, Thailand, 3-5 April 2002, accepted for publication.
- [5] Maeder R. E., Computer Science with *Mathematica*, CUP, 2000.
- [6] Wagner D.B., Power Programming with *Mathematica*, McGraw-Hill, 1996.

Parallel Construction of Poincaré Surface of Section Method on WAC16P4 Cluster

Abstract

Parallel computing performance of Mathematica on Walailak University Cluster of 16 nodes Pentium 4 (WAC16P4) is measured. Cluster Operating System is NPACI ROCK 2.21. Overhead time is computed and compared. The computing tasks are to generate Poincaré Surface of Sections for a highly nonlinear system by a functional programming code. Speed-up and efficiency of the cluster (WAC16P4) is computed and discussed.

1. Introduction

In previous work [1] parallel computation in a cluster of personal computer (PC) can be achieved in *Mathematica* computing environment [2] if it is operating with an access to more than one computer on a network. *Mathematica* provides both a programming language and a computing environment with many programming styles such as the procedural style like C, the functional programming style like Lisp and also rule-based programming style like Prolog [3]. Fine-tuning to maximize *Mathematica* performance is a delicate task and programmers need to master the functional style of programming.

Parallel Computing Toolkit is a tool designed to support this functional programming style with many high-level commands for parallel execution of operations such as `ParallelMap[]` using the concept of Frontend and Kernel design philosophy of *Mathematica* with MathLink as the standard protocol. Parallel programming primitives including animation, plotting and matrix manipulation functions are also provided. Currently, the toolkit supports many current popular programming approaches such as parallel Monte Carlo simulation, visualization, searching and optimization. The implementations for all high-level commands in the Parallel Computing Toolkit are provided in *Mathematica* source form, and serve as templates for building additional parallel programs [4].

As a summary, *Mathematica* computing environment with the Parallel Computing Toolkit is based on the functional programming style and, in fact, functional model is one the most powerful model of parallel computation [5].

This work furthers the performance testing of the Parallel Computing Toolkit that is done in our previous work [1] with a different cluster configuration and with a different application. Our new cluster configuration is expanded to 16 nodes, so called Walailak University P4 Cluster (WAC16P4),

and the test problem is also closer to real research problems. That is Poincaré Surface of Section calculation.

2. Poincaré Surface of Section (PSS) calculation via Functional Programming

Poincaré surface of section (PSS) method is an important tool to classify motions in nonlinear dynamical systems, especially for 2D Hamiltonian systems. The detail of this method is clarified with our test problem in the next section. Generally PSS method requires extensive use of numerical ordinary differential equations integrators such as Runge-Kutta scheme or others [6]. Choosing the right implementation of the integrator is the key to obtain quality PSS as of equal importance is to obtain the right implementation of the PSS method itself. The practical implementation of PSS method, which can be used to study nonlinear dynamical systems in *Mathematica* environment, was identified in [7]. The implementation is in Functional Programming style ([8], [9], and [10]), so it is ready to utilize *Mathematica* parallel computation and powerful visualization capabilities.

3. Test problem

The motion of a charged particle in a CUSP magnetic field can be generated by the following ordinary differential equations which is also called the equations of motion:

$$\begin{aligned}\frac{dx}{dt} &= p, & \frac{dy}{dt} &= q, \\ \frac{dp}{dt} &= y(Q - xy), & \frac{dq}{dt} &= x(Q - xy),\end{aligned}$$

where (x, y) is the position of the particle in the rectangular coordinate and (p, q) are the corresponding momentum in both directions. Q is a constant related to the z direction momentum which is conserved during the motion. All these variables are in the normalized form. It is vital to point out that each variable needs not be identified physically the same as above. This problem can be treated and studied as a class of ODE and the same ODEs can be found in many other applications.

In this system we have four independent variables with four differential equations. Nevertheless, it is easy to clarify that the quantity $H = p^2 + q^2 + (Q - xy)^2$ is always a constant. This signifies that the system only has three independent variables, hence the motion is essentially 3 dimensional. If one variable is fixed as in the form of a plane in 3D, then one can study the system with only two variables. In essence, each time the trajectory in 3D cuts the plane, the program records the cutting position and collects all these cutting points (see Figure 1(b) and (c) for illustration). This algorithm can be viewed as a form of mapping and can be used to classify motions e.g. a finite set of point indicates a periodic trajectory, a line indicates a quasi-periodic one, while a messy output indicates a chaotic trajectory. This is a powerful method and Poincaré was the first to use this method, hence the method was named after him. In this work we choose $Q = 0$ and the section is chosen at $y = 0$. Detail implementation of this PSS method can be found in [7]. Parallelization can be achieved simply by replacing `Map[]` with `ParallelMap[]`, which is the function to distribute our computational tasks. The function is one of the many high level functions provided by Parallel Computing Toolkit. Figure 1(a) shows the time series of y variable. It also illustrates the positions (dots on the axis) where PSS method collects the data.

4. Test Platform

WAC16P4 comprises of 16 identical personal computers. Each has a 1.3 GHz Pentium 4 Processor for 400 MHz system bus with 256 KB L2 cache, a Gigabyte Dual Bios GA-8ITX3 for Pentium 4 Processor mainboard with 400 MHz system bus, a total RAM memory of 512 MB RIMM RDRAM 400 MHz system bus, two Intel Pro 100+ network cards (currently using only 1 card except the front-end node), an ASUS ATA-100 supported DVD ROM drive and a Western Digital WD200 20 GB ATA-100 harddisk. The cluster is networked through a single 24-port Accton CheetahSwitch Workgroup-3526L 100 Mbits switch.

NPACI Rocks Linux [11] release 2.21 is our operating system for cluster installation. For the computation, the Parallel Computing Toolkit is installed on the front-end node while *Mathematica* 4.1 Professional for Linux is installed. The master kernel needs to run on one node, hence, there are 15 nodes left for parallelization. The parallel computing toolkit utilizes the connection via TCP protocol to link all the kernels with specific ports generated on-the-fly by starting *Mathematica* kernel automatically in the Mathlink mode on each node with ssh command. Once connected with the control kernel, these kernels are called “slaves”. The parallel toolkit is loaded in the control kernel residing on the front-end machine of the cluster. The front end, which provides the cluster interface to the user, is connected to the control kernel locally.

Table 1 Network configuration for the front-end machine obtained on the front-end machine with ifconfig command.

eth0	Link encap:Ethernet Hwaddr 00:02:B3:95:58:69 inet addr:10.1.1.1 Bcast:10.255.255.255 Mask:255.0.0.0 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:4203200 errors:0 dropped:0 overruns:0 frame:0 TX packets:4130192 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:100 RX bytes:346449607 (330.4 Mb) TX bytes:31620216 (30.1 Mb) Interrupt:21 Base address:0xc000
eth1	Link encap:Ethernet Hwaddr 00:02:B3:95:59:E3 inet addr:202.28.68.175 Bcast:202.28.68.191 Mask:255.255.255.192 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:5143189 errors:0 dropped:0 overruns:0 frame:1 TX packets:1502483 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:100 RX bytes:1460362238 (1392.7 Mb) TX bytes:150683776 (143.7 Mb) Interrupt:22 Base address:0xc000
lo	Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 UP LOOPBACK RUNNING MTU:16436 Metric:1 RX packets:1529436 errors:0 dropped:0 overruns:0 frame:0 TX packets:1529436 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:1293172099 (1233.2 Mb) TX bytes:1293172099 (1233.2 Mb)

The front-end machine has 2 network cards, one for the connection with the internet (IP 202.28.68.175) and the other for cluster connection. IP numbers for the computing nodes are ranging from 10.255.255.240 to 10.255.255.254. Table 1 gives detail information of network configuration of the front-end machine using ifconfig UNIX command.

5. Performance Measurement

Overhead time (see the definition and *Mathematica* code in [1]) for local kernel for WAC16P4 is computed. Then, for the test problem, we set up two different configurations for parallel task distribution. The first should give results that are linear time complexity, while the other should give results that are quadratic time complexity.

6. Results

For overhead measurement, Figure 2 shows that the overhead time is around 1.9 seconds and slowly increasing but not at a significant level. It is important to note that this overhead time is significantly higher than 0.03 seconds which is the result for the local kernel test reported in [1]. In case of remote kernel, the overhead time approaches 0.41 seconds for 4 nodes configuration and approaches 0.017 for 8 nodes configuration. Comparison with the overhead time of the 5 nodes configuration [1], which was 1.53 seconds, indicates that our new cluster has better connection speed over the old one.

For the test problem results are summarized in Table 2 and Table 3 together with Figure 3 and Figure 5. Grouping of speed up is very obvious because all of the computing nodes have exactly the same configuration. In other words, each node carry the same load capacity meaning load balancing can be accomplished.

Table 2. Speedup and efficiency factors of various cluster configurations. Efficiency is calculated using the mean value of speed up.

#node	Speedup			Efficiency
	Min	Max	Mean	
2	1.849056604	1.913043478	1.880853658	0.940426829
3	2.925373134	3.142857143	3.024968389	1.008322796
4	3.563636364	4	3.760440687	0.940110172
5	4.84	5.5	5.028645161	1.005729032
6	4.780487805	5.5	5.004742722	0.834123787
7	4.75	5.5	4.974097561	0.710585366
8	6.75862069	7.5625	7.278509852	0.909813732
9	6.909090909	8.8	7.554318182	0.839368687
10	7	8.8	7.531148459	0.753114846
11	7.117647059	8.8	7.583000311	0.689363665
12	7.117647059	8.8	7.583000311	0.631916693
13	7	8.8	7.531148459	0.579319112
14	6.75862069	8.8	7.482872597	0.5344909
15	12.66666667	22	15.23555556	1.015703704

Table 3. Maximum speed up and efficiency of polynomial for the case with polynomial time complexity.

#node	Maximum Speedup	Efficiency
2	2.039783002	1.019891501
3	3.107438017	1.035812672
4	4.147058824	1.036764706
5	4.925764192	0.985152838
6	6.197802198	1.032967033
7	6.635294118	0.94789916
8	7.726027397	0.965753425
9	8.952380952	0.994708995

10	9.245901639	0.924590164
11	9.170731707	0.833702882
12	11.87368421	0.989473684
13	12	0.923076923
14	12.3956044	0.885400314
15	12.6741573	0.84494382

7. Conclusions

Figure 3 shows two different time complexities set up for this problem. For the linear time complexity, further results are shown in Figure 5(a) which indicates that when supply enough tasks to be distributed over the nodes, speed up near ideal case can be obtained. Therefore, in this set up speed up and efficiency are nearly independent with the size of the problem. This is shown by almost flat lines in the figure. In fact the speed up can be actually decreased when the problem size is increased. This is due to the linear time complexity of this problem. The increase in time needed for computing in each node cannot compete with the increase in the network communication between the front-end and the slave nodes to send and receive the results. It is important to note that since our parallelization is based on sending a number of tasks, in which each of them has a different set of initial conditions, to operating node (so called Task-Farming [5]), grouping of speed up (e.g. for 8-14 node) can occur when the number of task is indivisible with the numbers of node.

For the quadratic case, Figure 5(b) shows the results. It is obvious in the figure that speed up can be increased with the size of the problem. This is the normal case observed by many researchers. Grouping of speed up is also noticeable in this case. Figure 5 (a) and (b) are different visualizations of Figure 4 (a) and (b) accordingly.

In conclusion, WAC16P4 operates with higher efficiency than the cluster we tested in the previous work where efficiency was significantly decreased after 3 nodes [1]. Additionally, our results indicate WAC16P4 behaves like an ideal Task-Farming parallelism. Therefore, we can expect the bottleneck problem in the future. The solution to this problem is to extend the single master to a set of masters, each of them controlling a different group of process slaves [5]. Further test with this idea with the Parallel Computing Toolkit with *Mathematica* is needed.

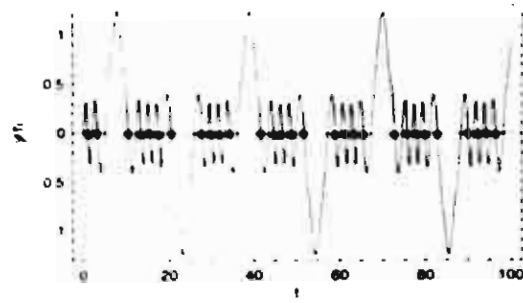
8. Acknowledgement

One of the authors (KJ) would like to thank the Thailand Research Fund, Post-Doctoral Grant (PDF/62/2540). We also thank Walailak University for partial financial support under National Research Council of Thailand recommendation, contract no. ๓๗ 1/2545. In addition, the authors would like to thank Pitoon Musik for his technical support with our cluster, WAC16P4.

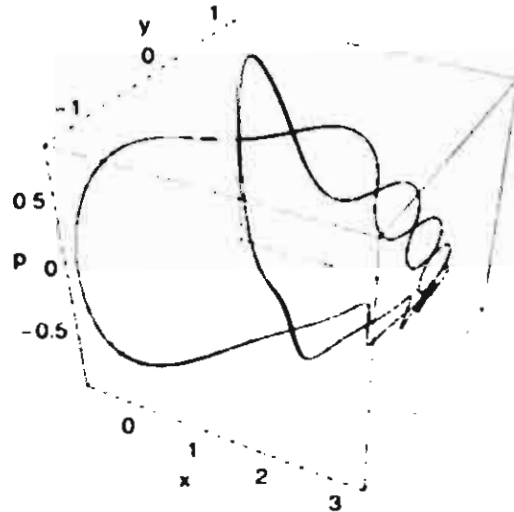
9. References

- 1] K. Jaroensutasinee, Proceedings of The 5th National Computer Science and Engineering Conference, Nov.7-9, 2001, Chiang Mai, Thailand, p 429.
- 2] S. Wolfram, The Mathematica Book, CUP and Wolfram Media, 1999.
- 3] D.B. Wagner, Power Programming with Mathematica, Computing McGraw-Hill, 1996.
- 4] Wolfram Research Inc., Parallel Computing Toolkit , <http://www.wolfram.com/products/applications/parallel/>, 2001.

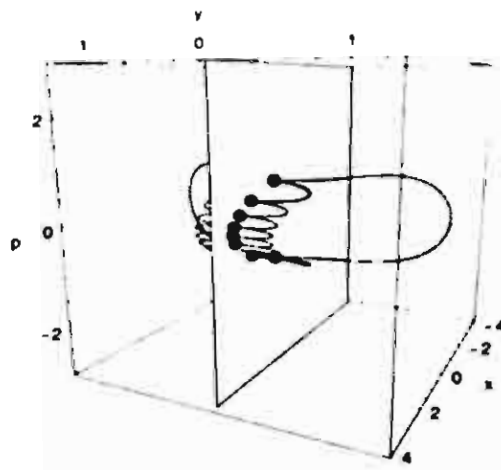
- [5] R. Buyya, High Performance Cluster Computing, Vol. 1 and 2, Prentice Hall, 1999.
- [6] K. Jaroensutasinee and G. Rowlands, to be appeared in the proceedings of the 6th Annual National Symposium on Computational Science and Engineering, Apr.3-5, 2002, Walailak University, Nakhon Si Thammarat, Thailand.
- [7] K. Jaroensutasinee and G. Rowlands, to be appeared in the proceedings of International Conference on Computational Mathematics and Modeling, May 22-24, 2002, Bangkok, Thailand.
- [8] J. Gray, Mastering Mathematica, 2nd ed., Academic Press, 1997.
- [9] R. E. Maeder, Computer Science with Mathematica, CUP, 2000.
- [10] D. B. Wagner, Power Programming with Mathematica, McGraw-Hill, 1996.
- [11] <http://rocks.npaci.edu/>
- [12] C. Kruengkai and C. Jaruskulchai, Proceedings of The 5th National Computer Science and Engineering Conference, Nov.7-9, 2001, Chiang Mai, Thailand, p 325.
- [13] M. L. Abell and J. P. Braselton, Differential equations with Mathematica, AP Professional, 1993.
- [14] A. Gray, M. Mezzino and M. Pinsky, Introduction to Ordinary Differential Equations with Mathematica, Springer-Verlag, TELOS, 1997.



(a)



(b)



(c)

Figure 1 (a) Time series $y(t)$ obtained from solving CUSP system with cuts (vertical lines and dots) from PSS computation. (b) 3D visualization of the trajectory in (x, y, p) showing its periodicity. (c) 3D visualization showing PSS cuts, the PSS plane and the trajectory.

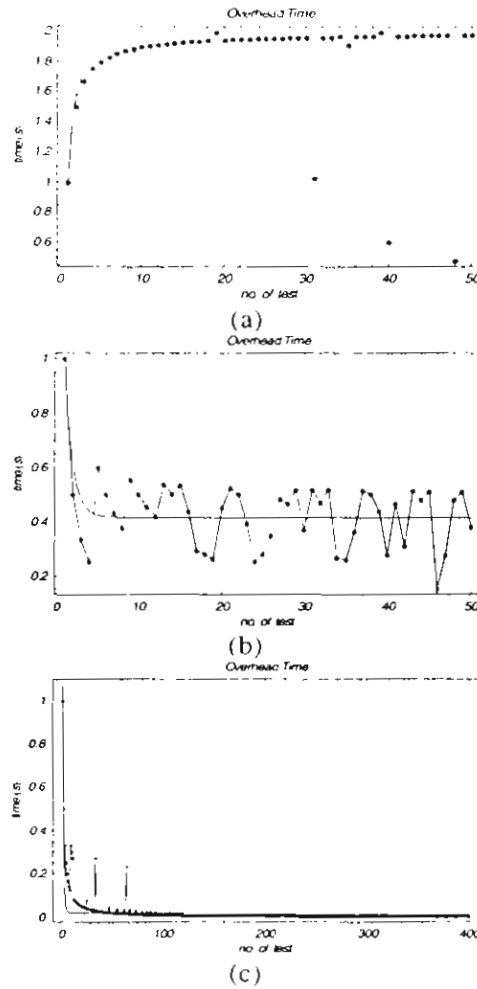
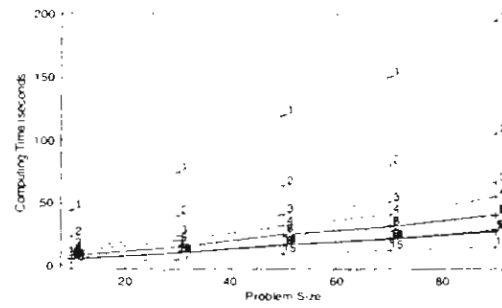
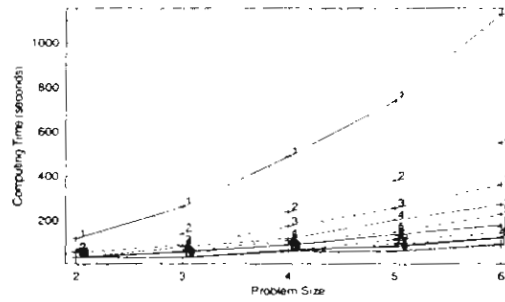


Figure 2 Variation of the overhead time of kernel to local kernel to service one trivial process. (a) local kernel. The overhead time approaches 1.94s. The dotted line is calculated with noisy data removed. (b) 4 remote kernels. (c) 8 remote kernels. The noisy data above the obvious line, formed by connecting normal data, resulted from interruptions of the main computing process by background jobs.

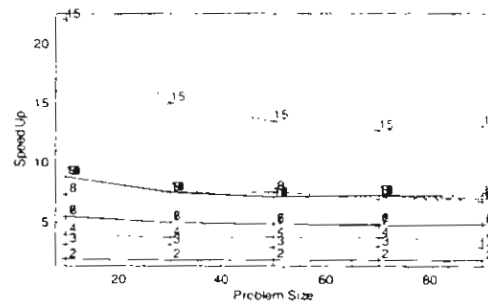


(a)

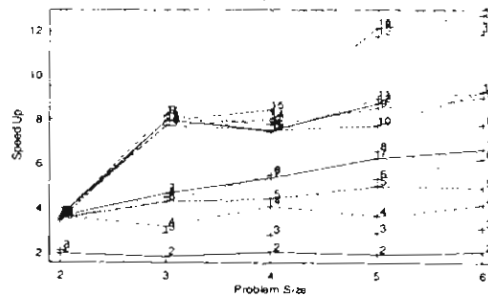


(b)

Figure 3 Computing time of the test problem for various cluster configurations. Marking numbers above the plus sign signify the number of nodes used in computation. (a) for linear time complexity. (b) for polynomial time complexity.



(a)



(b)

Figure 4. Speed up variation with the problem size. Marking numbers above the plus sign signify the number of nodes used in computation.

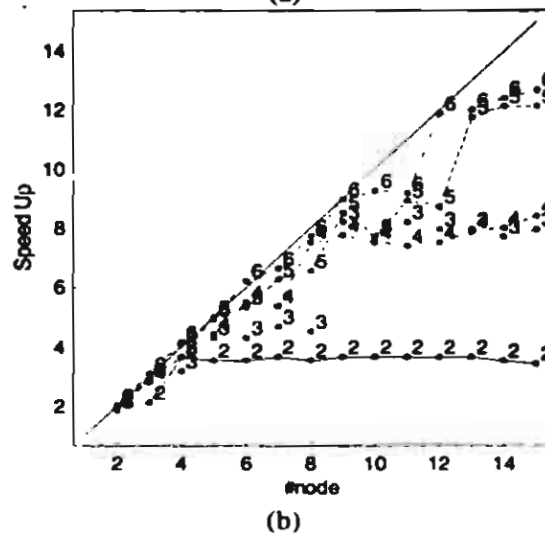
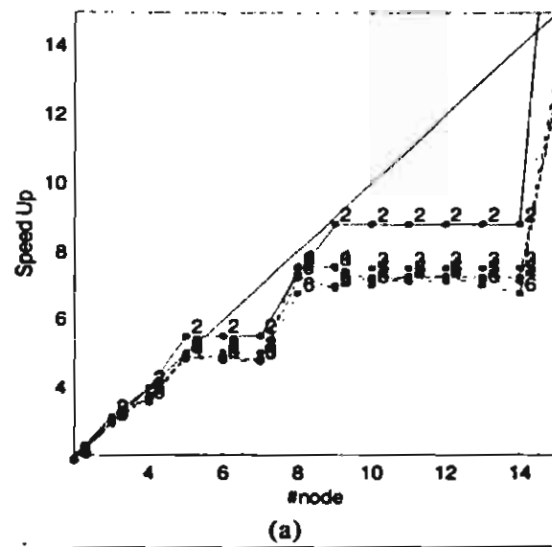


Figure 5. Speed up variation with various numbers of cluster configuration. Marking numbers above the plus sign signify the problem size.